Consistent Anticipatory Route Guidance

by

Jon Alan Bottom

Bachelor of Science in Mathematics, Massachusetts Institute of Technology (1976)
Master of Science, Massachusetts Institute of Technology (1976)

submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Transportation Systems

at the

Massachusetts Institute of Technology
September 2000

Signature of Author:_____

Department of Civil and Environmental Engineering
August 11, 2000

Certified by:_____

Moshe E. Ben-Akiva
Edmund K. Turner Professor of Civil and Environmental Engineering
Thesis Co-Supervisor

Certified by:_____

Ismail Chabini
Assistant Professor of Civil and Environmental Engineering
Thesis Co-Supervisor

Accepted by:_____

Daniele Veneziano
Professor of Civil and Environmental Engineering
Chair, Department Committee on Graduate Studies

Consistent Anticipatory Route Guidance

by

Jon Alan Bottom

submitted to the Department of Civil and Environmental Engineering
on August 11, 2000 in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Transportation Systems

Abstract

Anticipatory route guidance consists of messages, based on traffic network forecasts, that assist drivers' path choice decisions. Guidance is consistent when the forecasts on which it is based are verified after drivers react to it. This thesis addresses the formulation and development of solution algorithms for the consistent anticipatory route guidance generation (RGG) problem.

The thesis proposes a framework for the problem, involving a set of time-dependent variables and their relationships. Variables are network conditions, path splits and guidance messages. Relationships are the network loading map, transforming path splits into network conditions; the guidance map, transforming network conditions into guidance messages; and the routing map, transforming guidance messages into path splits. The basic relationships can be combined into three alternative composite maps that model a guidance problem. Consistent guidance corresponds to a fixed point of a composite map. With stochastic maps, RGG model outputs are stochastic process realizations. In this case, the consistency fixed point corresponds to stationarity of the RGG solution process.

Numerical methods for fixed point computation were examined, focusing on approaches that are rigorous and applicable to large-scale problems. Methods included Gibbs sampling for highly stochastic maps; generalizations of functional iteration for deterministic maps; and the MSA and Polyak iterate averaging method for "noisy" (deterministic plus disturbance) maps.

A guidance-oriented dynamic traffic simulator was developed to experiment with RGG solution methods. Computational tests using the simulator investigated the use of Gibbs sampling to compute general stochastic process outputs; and examined the performance of the averaging methods under different model formulations, problem settings and degrees of stochasticity.

Gibbs sampling successfully generated realizations from the stationary solution process of a fully stochastic model, but entails considerable computational effort. For noisy problems, the MSA found fixed points in all cases considered. Polyak averaging converged between two and four times faster than the MSA in low or moderate stochasticity problems, and performed comparably to the MSA in other problems. Formulations involving path-level variables converged more quickly than those involving link-level variables.

Thesis Co-Supervisor: Moshe E. Ben-Akiva
Title: Edmund K. Turner Professor of Civil and Environmental Engineering

Thesis Co-Supervisor: Ismail Chabini
Title: Assistant Professor of Civil and Environmental Engineering

Acknowledgements

Funding for much of this work was provided by an Eisenhower graduate fellowship from the USDOT, which I acknowledge with thanks.

I would like to thank the following people, who have directly contributed to this effort in a variety of significant ways; it has been a privilege to interact with them:

Moshe Ben-Akiva and Ismail Chabini, my thesis co-supervisors;

Michel Bierlaire, Ennio Cascetta and Markos Papageorgiou, members of my thesis committee;

David Bernstein, Piet Bovy, Giulio Cantarella, Leonid Engelson, Haris Koutsopoulos, Rabi Mishalani, Kai Nagel, André de Palma, Georgia Perakis and Jan van Schuppen, who generously gave time, insights and feedback.

Nigel Wilson was not directly associated with this research yet on many occasions he provided valuable advice and support, for which I am very grateful.

During my stay at MIT I have shared offices with many good people, but I would particularly like to thank Joan Walker and Scott Ramming for their friendship and contributions to the office environment.

I thank my mother and honor the memory of my father for the affection and encouragement they unfailingly gave me.

I would not have been able to complete this work without the love and patience of my wife Fotini and my sons Michael and Paul. I dedicate it to them with all my gratitude and love.

Contents

1 Introduction 15
- 1.1 Congestion and its treatment . . . 15
- 1.2 Scope of the thesis . . . 17
- 1.3 Summary of research and findings . . . 19
- 1.4 Contributions . . . 23
- 1.5 Thesis organization . . . 23

2 Approaches to Route Guidance 25
- 2.1 Static guidance . . . 26
- 2.2 Reactive guidance . . . 26
  - 2.2.1 Limited information approaches . . . 27
  - 2.2.2 Full information approaches . . . 29
- 2.3 Anticipatory guidance . . . 30
  - 2.3.1 Link-level extrapolations . . . 31
  - 2.3.2 Incident-only predictions . . . 32
  - 2.3.3 Path-level predictions . . . 32
  - 2.3.4 Network-level models . . . 33
  - 2.3.5 Guidance characteristics have to be modeled . . . 35
  - 2.3.6 Anticipatory guidance system implementation issues . . . 37

3 Consistent Anticipatory Route Guidance 39
- 3.1 Overview . . . 39
- 3.2 Framework for the RGG problem: deterministic case . . . 40
  - 3.2.1 Components of the analysis framework . . . 40
    - 3.2.1.1 Exogenous elements . . . 41
    - 3.2.1.2 Framework variables . . . 43
    - 3.2.1.3 Framework maps . . . 44
  - 3.2.2 Composite map formulations of guidance consistency . . . 47
  - 3.2.3 Relationship to full information models . . . 49
    - 3.2.3.1 Model assumptions and structure . . . 49
    - 3.2.3.2 Interpretation of fixed point conditions . . . 50
  - 3.2.4 Discussion of framework and alternative approaches . . . 51
    - 3.2.4.1 Generality of the overall framework . . . 51
    - 3.2.4.2 Paths and subpaths . . . 52
    - 3.2.4.3 Flow variables . . . 53

3.2.4.4 Condition variables . . . 54
3.2.4.5 Message variables . . . 55
3.2.4.6 Routing map . . . 55
3.2.4.7 Conclusions . . . 56
3.3 Framework for the RGG problem: stochastic case . . . 57
3.3.1 Nature of model stochasticity . . . 57
3.3.2 Approaches to guidance generation in the presence of stochasticity . . . 57
3.3.3 Guidance consistency in the presence of stochasticity . . . 59
3.4 Review of related work . . . 60
3.4.1 Fixed point approaches for network equilibrium models . . . 61
3.4.2 Stochastic process and related network models . . . 63
3.4.3 Self-fulfilling and recursive forecasts . . . 65
3.4.4 Bovy's and van der Zijpp's analysis . . . 66
3.4.5 Rational expectations models . . . 67
3.4.6 Game theory approach to route guidance . . . 68
3.4.6.1 Background . . . 68
3.4.6.2 Static network case . . . 69
3.4.6.3 Game theory and route guidance . . . 69
3.4.7 Kaysi's analysis . . . 73
3.4.8 Advantages of a fixed point approach . . . 75

4 Fixed Point Algorithms for Guidance Generation 77
4.1 Gibbs sampling and related methods . . . 78
4.1.1 Gibbs sampling . . . 78
4.1.1.1 Mathematical background . . . 78
4.1.1.2 Applications . . . 81
4.1.2 The TRANSIMS rerouting algorithm . . . 83
4.1.3 Fictitious play methods . . . 84
4.2 Stochastic approximation methods . . . 86
4.2.1 Introduction . . . 86
4.2.2 Recursive averaging . . . 86
4.2.2.1 Mathematical background . . . 86
4.2.2.2 Applications . . . 88
4.2.3 Iterate averaging . . . 89
4.2.3.1 Mathematical background . . . 89
4.2.3.2 Applications of iterate averaging . . . 90
4.3 Deterministic methods . . . 90
4.3.1 Functional iteration and related methods . . . 91
4.3.1.1 Mathematical background . . . 91
4.3.1.2 Applications . . . 93
4.3.2 Triangulation methods . . . 95
4.3.2.1 Mathematical background . . . 95
4.3.2.2 Applications . . . 96
4.4 Conclusions . . . 97

5 Computational Tests 99
5.1 Introduction . . . 99
5.2 Description of the simple simulator . . . 100
5.2.1 Network loading map . . . 100
5.2.2 Guidance map . . . 103
5.2.3 Routing map . . . 104
5.2.4 Sources of stochasticity in the simulator . . . 107
5.2.4.1 Rounding issues . . . 107
5.2.4.2 Individual sources of stochasticity . . . 110
5.2.5 Implementation issues . . . 112
5.2.5.1 The composite link condition map . . . 113
5.2.5.2 The composite path split map . . . 115
5.2.5.3 The composite message map . . . 115
5.2.5.4 Simulator implementation decisions . . . 118
5.3 Description of the test network . . . 118
5.4 Investigation of simulator properties . . . 118
5.4.1 Stochasticity in the network loading map . . . 119
5.4.2 Stochasticity in the composite routing/loading map . . . 123
5.5 Gibbs sampling solution of a stochastic D ∘ G ∘ S model . . . 126
5.6 Computational tests of guidance generation algorithms . . . 128
5.6.1 Composite network condition formulation . . . 129
5.6.1.1 MSA algorithm . . . 130
5.6.1.2 Polyak iterate averaging algorithm . . . 136
5.6.2 Composite path split formulation . . . 139
5.6.2.1 MSA algorithm . . . 139
5.6.2.2 Polyak algorithm . . . 143
5.6.3 Composite message formulation - continuous case . . . 144
5.6.3.1 MSA algorithm . . . 145
5.6.3.2 Polyak algorithm . . . 148
5.7 Conclusions . . . 149
5.8 Graphic outputs . . . 150

6 Conclusions and Directions for Further Work 233
6.1 Recapitulation of the work and conclusions . . . 233
6.2 Directions for further work . . . 237

Bibliography 241

List of Tables

5.1 Computational tests of the network loading map 120
5.2 Computational tests of the combined routing/loading map 124
5.3 Computational tests of the MSA applied to $S \circ D \circ G : C \mapsto C$ 131
5.4 Computational tests of Polyak averaging applied to $S \circ D \circ G : C \mapsto C$ 137
5.5 Computational tests of the MSA applied to $D \circ G \circ S : P \mapsto P$ 141
5.6 Computational tests of Polyak averaging applied to $D \circ G \circ S : P \mapsto P$ 143
5.7 Computational tests of the MSA applied to $G \circ S \circ D : M \mapsto M$ 147
5.8 Computational tests of Polyak averaging applied to $G \circ S \circ D : M \mapsto M$ 149
5.9 Computational tests of individual and combined maps 151
5.10 Computational tests of formulations and solution methods 151

List of Figures

1.1 Dynamic Traffic Management Overview [OECD, 1987] 16

2.1 Simple guidance example network 36

3.1 A framework for route guidance generation 41
3.2 Approximate computation of Nash equilibrium routing directives [van Schuppen, 1997] 71

5.1 All-at-once implementation of the S ∘ D ∘ G map 114
5.2 On-the-fly implementation of the S ∘ D ∘ G map 114
5.3 All-at-once implementation of the D ∘ G ∘ S map 116
5.4 On-the-fly implementation of the D ∘ G ∘ S map 116
5.5 All-at-once implementation of the G ∘ S ∘ D map 117
5.6 On-the-fly implementation of the G ∘ S ∘ D map 117
5.7 Test network 119
5.8 Run NLM-1 (part 1/2) 152
5.8 Run NLM-1 (part 2/2) 153
5.9 Run NLM-2 (part 1/2) 154
5.9 Run NLM-2 (part 2/2) 155
5.10 Run NLM-3 155
5.11 Run SD-1 (part 1/2) 156
5.11 Run SD-1 (part 2/2) 157
5.12 Run SD-2 (part 1/2) 158
5.12 Run SD-2 (part 2/2) 159
5.13 Run SD-3 160
5.14 Run SD-5 161
5.15 Selected results from Gibbs sampling of D ∘ G ∘ S map 162
5.16 IC_C norms evaluated from 100 replications of the S ∘ D ∘ G map 163
5.17 Run SDG-1 (part 1/3) 164
5.17 Run SDG-1 (part 2/3) 165
5.17 Run SDG-1 (part 3/3) 166
5.18 Run SDG-2 (part 1/3) 167
5.18 Run SDG-2 (part 2/3) 168
5.18 Run SDG-2 (part 3/3) 169
5.19 Run SDG-3 (part 1/3) 170
5.19 Run SDG-3 (part 2/3) 171
5.19 Run SDG-3 (part 3/3) 172

5.20 Run SDG-4 (part 1/3) . . . 173
5.20 Run SDG-4 (part 2/3) . . . 174
5.20 Run SDG-4 (part 3/3) . . . 175
5.21 Run SDG-5 (part 1/3) . . . 176
5.21 Run SDG-5 (part 2/3) . . . 177
5.21 Run SDG-5 (part 3/3) . . . 178
5.22 Run SDG-6 (part 1/3) . . . 179
5.22 Run SDG-6 (part 2/3) . . . 180
5.22 Run SDG-6 (part 3/3) . . . 181
5.23 Run SDG-3-pol . . . 182
5.24 Run SDG-4-pol . . . 183
5.25 Run SDG-5-pol . . . 184
5.26 Run DGS-1 (part 1/4) . . . 185
5.26 Run DGS-1 (part 2/4) . . . 186
5.26 Run DGS-1 (part 3/4) . . . 187
5.26 Run DGS-1 (part 4/4) . . . 188
5.27 Run DGS-2 (part 1/4) . . . 189
5.27 Run DGS-2 (part 2/4) . . . 190
5.27 Run DGS-2 (part 3/4) . . . 191
5.27 Run DGS-2 (part 4/4) . . . 192
5.28 Run DGS-3 (part 1/4) . . . 193
5.28 Run DGS-3 (part 2/4) . . . 194
5.28 Run DGS-3 (part 3/4) . . . 195
5.28 Run DGS-3 (part 4/4) . . . 196
5.29 Run DGS-4 (part 1/4) . . . 197
5.29 Run DGS-4 (part 2/4) . . . 198
5.29 Run DGS-4 (part 3/4) . . . 199
5.29 Run DGS-4 (part 4/4) . . . 200
5.30 Run DGS-5 (part 1/4) . . . 201
5.30 Run DGS-5 (part 2/4) . . . 202
5.30 Run DGS-5 (part 3/4) . . . 203
5.30 Run DGS-5 (part 4/4) . . . 204
5.31 Run DGS-6 (part 1/4) . . . 205
5.31 Run DGS-6 (part 2/4) . . . 206
5.31 Run DGS-6 (part 3/4) . . . 207
5.31 Run DGS-6 (part 4/4) . . . 208
5.32 Run DGS-3-pol . . . 209
5.33 Run DGS-4-pol . . . 210
5.34 Run DGS-5-pol . . . 211
5.35 Run GSD-1 (part 1/3) . . . 212
5.35 Run GSD-1 (part 2/3) . . . 213
5.35 Run GSD-1 (part 3/3) . . . 214
5.36 Run GSD-2 (part 1/3) . . . 215
5.36 Run GSD-2 (part 2/3) . . . 216

5.36 Run GSD-2 (part 3/3) 217
5.37 Run GSD-3 (part 1/3) 218
5.37 Run GSD-3 (part 2/3) 219
5.37 Run GSD-3 (part 3/3) 220
5.38 Run GSD-4 (part 1/3) 221
5.38 Run GSD-4 (part 2/3) 222
5.38 Run GSD-4 (part 3/3) 223
5.39 Run GSD-5 (part 1/3) 224
5.39 Run GSD-5 (part 2/3) 225
5.39 Run GSD-5 (part 3/3) 226
5.40 Run GSD-6 (part 1/3) 227
5.40 Run GSD-6 (part 2/3) 228
5.40 Run GSD-6 (part 3/3) 229
5.41 Run GSD-3-pol 230
5.42 Run GSD-4-pol 231
5.43 Run GSD-5-pol 232

Chapter 1

Introduction

1.1 Congestion and its treatment

Congestion on road networks manifests itself in the form of travel delays compared to free-flow conditions. Less obviously, perhaps, congestion is the cause of pervasive economic inefficiencies as individuals, households and firms adjust their activities to compensate for time lost in traveling and to hedge against the possibility that trips may take longer than expected. An important characteristic of congestion is in fact its randomness: for example, it has been estimated that roughly 60% of congestion delays on urban freeways in the U.S. are non-recurrent, i.e., due to specific random events such as accidents, vehicle breakdowns and the like [Lindley, 1987].[1] Even recurrent congestion, resulting from traffic levels that are systematically high relative to available roadway capacity over a particular period of time, has a random component that derives from variability in travel demand patterns and in network performance. Because of this randomness, drivers' experience is an unreliable basis for predicting the conditions associated with their various travel options, and so for making optimal travel choices.

In the past, the typical response of governments to serious congestion has been to build or upgrade roads. This increases the capacity available for usual traffic levels and also serves as a buffer against random demand surges or point capacity reductions (e.g. incidents). For a variety of reasons, however, this approach is becoming increasingly less feasible.

Advances in sensor, data processing and communications technologies have made possible a different kind of response to traffic congestion. These technologies provide opportunities to improve the performance of existing roads by monitoring and controlling in real time the movements of vehicles on them. Approaches based on this idea are generally known as advanced traffic management systems (ATMS).

A complementary approach, which builds on many of the same technologies, is to provide tripmakers with data that help them make better travel decisions before and during their trips. Such systems are generally called advanced traveler information systems (ATIS). The data provided by such systems may potentially relate to any tripmaking-related choice: the decision to travel or not, what destination or destinations to go to, what time to travel, what mode to take, and by what route or transit line.

[1] The empirical basis for this estimate has been called into question [Todd Olmstead, unpublished note, 1995]. Even if the non-recurrent delays are only half of what is claimed, however, it is nonetheless clear that a significant amount of congestion is random.

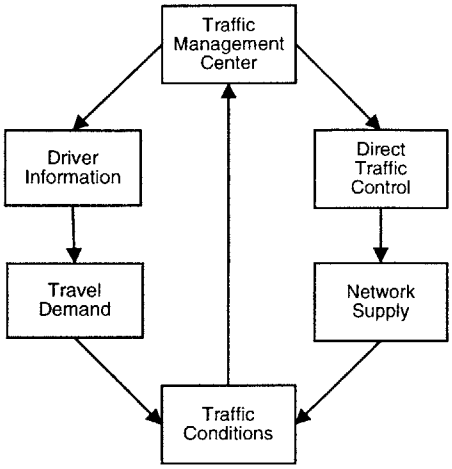


Figure 1.1: Dynamic Traffic Management Overview [OECD, 1987]

Figure 1.1, from an early report by the Organization for Economic Cooperation and Development (OECD) on dynamic traffic network management, illustrates the respective roles of ATMS and ATIS in the overall management of a road network and its traffic. The two are quite different: ATMS acts on network supply and drivers' compliance with it is as obligatory as with other traffic controls; whereas ATIS acts on travel demand and drivers may react to messages in any way that they like.

Route guidance and information systems (RGIS) are a particular type of ATIS specifically intended to assist drivers in making route choice decisions. In some discussions of RGIS, the term *route guidance* refers exclusively to prescriptive data such as route recommendations while the term *route information* is reserved to connote descriptive data on traffic conditions such as link volumes and travel times, incident locations, and the like. This fine distinction will not usually be germane to the discussion here so, unless otherwise noted, the words guidance and information will be used interchangeably. Any particular set of guidance data disseminated to drivers will be called a *message*. Messages may have a variety of forms (i.e., spoken or displayed text messages, graphical displays, etc.)

Some RGIS technologies broadcast guidance messages so that they can be received by any suitably-equipped vehicle located in a relatively large range. Radio-based systems such as highway advisory radio (HAR) or traffic message channels (TMC) are examples of this. Other technologies are short-range: the guidance messages can only be received by vehicles that are in close proximity (within a few hundred meters at most) to the guidance "transmitter". Variable message signs (VMS) or short-range infrared or microwave beacons are examples of this. Other guidance dissemination means are also possible (e.g. by telephone). Some technologies may restrict reception of guidance messages to certain tripmakers (those with special receivers, or who have paid for the service), while other technologies may make the guidance available to all.

Thus, depending on the RGIS technology, some or all tripmakers may be able to receive messages at their trip origins, or at intermediate trip locations, or both. Guidance received at the origin is referred to as *pre-trip*; such guidance assists tripmakers in their trip planning and in the initial choice of route to follow. Guidance received at one or more intermediate locations during a trip is referred to as *en route* guidance, and it assists tripmakers in deciding whether to switch to a new

route to follow towards their destination.

It is common to distinguish between static and dynamic RGIS. *Static* systems disseminate fixed or infrequently-updated data that can be helpful in trip planning or wayfinding, but that by definition do not reflect actual traffic conditions. *Dynamic* systems, on the other hand, provide messages based directly or indirectly on traffic conditions measured around the time of a trip. In many cases, the data provided by dynamic route guidance systems are intended to address decision-making uncertainties induced by randomness in traffic conditions. For example, a driver might be informed that a habitually chosen route has been blocked by an accident and shown an alternative that offers shorter travel time.

Dynamic RGIS can be further classified into *non-predictive* and *predictive* (also called *anticipatory*) systems. The former base the messages disseminated to tripmakers on estimates of the prevailing network state while the latter derive messages from forecasts of future network states.[2] While both types of system base their guidance messages on real-time traffic measurements (and other data), the subsequent data processing to generate guidance is very different: it is much more complex in the case of predictive systems. The merits of the two system types are the subject of active debate, and it is fair to say that no definitive conclusion is currently available regarding their relative worth, in part because actual operational experience with full-scale systems of each type is lacking.

In a review of opportunities for application of predictive control in a variety of traffic management contexts, van Toorenburg and van der Linden [1996] concluded that predictive route guidance could be very effective in specific cases such as incident situations. Simple examples in which anticipatory guidance is required to avoid counter-productive guidance effects can readily be constructed [Ben-Akiva et al., 1996]. Conversely, simulation studies have shown that, in certain networks, simple non-predictive guidance can approach or equal the effectiveness of predictive guidance [Pavlis and Papageorgiou, 1999]. Even in cases where predictive guidance can be shown to be theoretically more effective, there remain unresolved questions concerning its timeliness and accuracy in operational settings, since generating anticipatory guidance requires considerable amounts of data processing and can be sensitive to modeling errors and computational approximations [Papageorgiou and Messmer, 1991].

Chapter 2 presents a more detailed discussion of the various approaches to route guidance that have been briefly mentioned here.

1.2 Scope of the thesis

This thesis focuses on anticipatory route guidance. As explained above, in this form of guidance real-time traffic measurements and other data are processed into short- to medium-term traffic condition forecasts, and these forecasts are used to generate guidance messages that are disseminated to drivers. The messages may inform drivers about the predicted future conditions, or make route recommendations based on the predictions, or some combination of the two.

The basic problem in generating anticipatory guidance is the following: when messages based on traffic condition forecasts are disseminated, drivers' reactions to the messages may invalidate the

[2] It could be said that, in the absence of predictive capabilities, an estimate of the prevailing network state is the "best" forecast of future network state. Another interpretation is that drivers have internal models that they use to translate prevailing condition information into their own condition forecasts.

forecasts and render the guidance irrelevant or worse. Messages based on forecasts of impending congestion in one corridor, for example, may cause drivers to switch to another corridor, leaving the original one relatively uncongested. The basic problem in generating anticipatory guidance, in other words, is to ensure that the guidance messages do not become self-defeating prophecies.

Although predictive models based on extrapolations of prevailing network conditions have sometimes been proposed for generating anticipatory guidance, such simple models are not generally capable of accounting for drivers' reactions to the generated messages. In the general case some form of network-level traffic prediction model is required to carry out the forecasts used in anticipatory guidance. For this purpose, researchers have sometimes employed what might be termed within-day full information dynamic traffic assignment models. There can be many variations in the details of such models. However, a "generic" model of this type considers an analysis time period within a single day, ignoring day-to-day effects that result from drivers' reactions to prior days' travel experiences. The model assumes that drivers departing from their origin are fully informed about travel conditions, both at the time of departure and throughout the trip, on the alternative possible paths. Drivers' information might be, for example, the travel time required to reach the destination along each available path. The information is not necessarily completely precise or accurate; however, no better information will become available during the course of a trip. The information is used in a path choice model that determines path flows, and these flows are propagated along the paths to their destinations, thus determining link flows and conditions. All variables and relationships involved in the model are explicitly time-dependent. The model logic attempts to ensure that the "full information" on network conditions that is the basis of traveler route choices is verified after the trips are assigned—in other words, it attempts to compute a dynamic user optimal flow pattern. However, as will be seen repeatedly throughout this dissertation, this model (which for the sake of brevity will be called the *full information* model) is not well-suited to the analysis of the general route guidance problem.

In the framework of a particular predictive model, anticipatory guidance will be said to be *consistent* if the assumptions on which it is based prove to be realized after the guidance is disseminated and drivers react to it. Consistency is a model-based concept: it is simply the requirement that model inputs and outputs not be mutually contradictory.

Consistent anticipatory route guidance has not often been considered in its full generality in the literature. Individual research efforts have usually examined special cases of the problem. A common approach is to consider that RGIS converts a situation in which tripmakers choose routes based on some default condition assumptions into a full information situation. However, this approach neglects the fact that an RGIS will rarely provide full information. More typically, guidance messages will be a condensed summary of condition forecasts. Messages might only be available to a subset of drivers at a limited number of network locations, and might only be updated at intervals that are long compared to the time scale of traffic dynamics. Guidance provided in this way can hardly be considered full information on traffic conditions.

It follows that guidance system modeling requires driver information itself to be a component of the model relationships. This leads directly to a more complex model structure than full information models, for which questions of information format, content, accuracy, and temporal or geographic availability do not arise. In guidance system modeling, the guidance messages must be explicitly represented, driver response to different possible messages has to be modeled, the traffic consequences of driver responses (including possible en route path switches) need to be predicted,

and guidance consistency must be enforced.

These various requirements are sufficiently different from those of full information models that it is fair to claim that anticipatory guidance generation, in its general form, is a distinct problem. A few research efforts have investigated both theoretical and software aspects of the general problem over the past decade.

Guidance-related software development efforts in the United States have led to the creation of DYNASMART by Mahmassani and co-workers [Mahmassani et al., 1994]; and of DynaMIT by Ben-Akiva and co-workers [Ben-Akiva et al., 1997]. Both of these traffic simulation systems have anticipatory route guidance generation capabilities and are ultimately intended to be deployable in an operational traffic information center.

Theoretical issues associated with anticipatory guidance have been investigated by a number of researchers. Early work by Kaysi [1992] (see also Ben-Akiva et al. [1991]) identified and explored key aspects of the anticipatory guidance generation problem. Chen and Underwood [1991] was another early effort that identified many aspects of the anticipatory guidance problem and sketched out a research program to examine relevant issues. Noteworthy analyses carried out following these pioneering works include those by Kaufman et al. [1991], Engelson [1997], van Schuppen [1997] and Bovy and van der Zijpp [1999]. (These investigations and others are discussed in detail in Chapter 3.)

Despite the ground-breaking nature and the quality of these efforts, anticipatory route guidance cannot yet be considered a mature field. The theoretical investigations have resulted in many insights into the problem, but have not yet provided a general and unifying understanding of anticipatory guidance generation as a distinct and well-defined problem. The software development efforts have generally built on earlier work with traffic simulators, without the benefit of a good theoretical understanding of the problem and without systematically considering alternative guidance modeling and generation approaches.

The objective of the research presented in this dissertation is to formalize, generalize and extend current understanding of anticipatory route guidance in a within-day context (day-to-day effects are not considered). To this end, the dissertation describes the major features of the problem, proposes a framework for analyzing it, surveys algorithmic methods applicable to guidance generation, investigates computational tools (traffic simulators), and tests alternative problem formulations and solution methods under a variety of problem scenarios. Where appropriate, the discussion relates the analysis of the guidance problem to other research efforts in transportation and related fields. In the course of this exploration a number of questions are identified as topics for future research.

1.3 Summary of research and findings

A very natural way of defining consistency in anticipatory route guidance is as a fixed point of a map that represents the network-level traffic prediction model. A model system useful for guidance generation purposes must incorporate three fundamental relationships:

- the dynamic network loading map, which predicts the time-dependent network conditions that result from a set of time-dependent path choices by the tripmakers;
- the guidance message generation map, which converts predicted time-dependent network conditions into time-dependent messages that are disseminated to tripmakers at specific locations

on the network;

- the routing (or path choice) map, which predicts tripmakers' path choices resulting from their reactions to the disseminated guidance messages.

These relationships are both similar to and different from relationships embodied in standard full-information dynamic network models. For example, in full information models path choice is made at the origin, whereas in route guidance the routing map must be able to predict path switches based on guidance messages at en route locations as well. In conventional models, the network loader need only propagate flows along paths that are chosen at the origin, where in route guidance the loading map must be able predict the network consequences of tripmakers' en route path switches. There is no general analog, in the conventional model, of the guidance map.

In this research, the individual relationships are treated essentially as "black boxes" subject only to minimal restrictions. The development of accurate models of these relationships is a major research task in itself, and is left for future work. The focus of the work is rather on the definition and computation of consistent anticipatory guidance, assuming that satisfactory models of the basic relationships are available. In practice, the individual models may well be stochastic, either in an attempt to replicate the randomness of reality itself, or as a device to facilitate the simulation processing (for example, processing links in random order to avoid biases). The research considers models that are fully deterministic, fully stochastic and quasi-deterministic (i.e., with outputs affected by noise). Furthermore, the relationships may be evaluated either through analytical forms or via simulation.

The individual maps can be combined in three distinct orders to form composite maps representing a predictive model. For example, one such composite map starts with assumed time-dependent network conditions, determines guidance messages based on these, predicts the path splits that result from tripmakers' responses to the messages, and carries out a dynamic network loading to obtain a new set of time-dependent network conditions. Other possible maps start with time-dependent path splits or guidance messages, and are obtained by permuting the invocation order of the basic models. The correspondence of predictive model inputs and outputs at consistency can be expressed by saying that they are a fixed point of the composite map.

Although it is a relatively straightforward translation of the notion of consistency, this fixed point approach has a number of advantages. It immediately suggests alternative formulations of the route guidance generation problem (corresponding to the three composite maps). It is robust under different degrees of stochasticity in the involved maps and under analytical or simulation-based model formulations. It generalizes full information within-day dynamic traffic models, that are seen to be a special case in which the guidance map is an identity. Finally, it focuses attention on standard fixed point solution methods as a basis for developing algorithms to compute consistent anticipatory guidance.

Accordingly, the research carried out a review of fixed point solution techniques, emphasizing methods that are both mathematically correct and computationally feasible in application to the guidance generation problem.

For fully stochastic problems, Gibbs sampling was identified as an appropriate (although computationally expensive) solution method. It computes sample trajectories from the stochastic process outputs of the model in accordance with the stationary probabilities. Key to increasing the efficiency of Gibbs sampling (as well as a number of other iterative methods for dynamic traffic

network problems) is a method for reoptimizing the dynamic network loading procedure. (Reoptimization refers to any method that utilizes an available solution to one set of problem data to facilitate the solution determination for another set of problem data.) This problem does not appear to have been recognized in the literature, and was identified as a subject for future research.

For quasi-deterministic problems, the family of stochastic approximation methods was surveyed. The method of successive averages (MSA) is, of course, the prototype of such methods and has been widely applied to solve transportation network problems; however, it is known to suffer from slow convergence. The survey identified a number of other interesting and potentially applicable methods that have been developed since the MSA. Iterate averaging methods, which are computationally simple yet have asymptotically optimal convergence properties among the class of averaging methods, were identified as particularly promising; they do not appear to be widely known in the transportation community.

Methods for fully deterministic problems were also reviewed, but no generally applicable method was singled out because in practice guidance generation models are likely to incorporate some degree of stochasticity. Furthermore, there are practical difficulties with methods that use the map's gradient or Hessian since in general these would have to be numerically rather than analytically evaluated, and this would be computationally prohibitive. Methods based on the Lipschitz properties of the composite map (functional iteration for contractive maps, a method due to Baillon for non-expansive maps, and a family of methods due to Magnanti and Perakis for somewhat more general maps) cannot be rigorously applied since the Lipschitz properties of the composite maps used in guidance generation are not usually known. Furthermore, the methods of Magnanti and Perakis require a line search in each iteration, which may also prove to be computationally prohibitive.

A simple traffic simulator was designed and programmed to test the computational performance of the different composite formulations and solution methods. The simulator is basically a means to evaluate the network loading, routing and guidance maps identified above. It incorporates intentionally simple versions of these maps, and provides various options for controlling their degree of stochasticity. The individual maps can be easily combined in different orders to evaluate the various composite maps that can be used to represent guidance models. Additional logic can be wrapped around the simple simulator to compute guidance fixed points in alternative ways.

Initial computational tests focused on ascertaining relevant properties of the simulator: in particular, the factors that contribute to the stochasticity of its outputs. Not unexpectedly, it was found that the major factor influencing the degree of stochasticity in simulator outputs was the way in which path splits are applied to tripmakers—either individually or as a group. The way in which non-integer values are rounded to adjacent integers also has a noticeable but relatively less important effect.

Independent application of path splits to individual tripmakers results in multinomially distributed path flows, whereas collective application of splits to a homogeneous group of tripmakers (i.e., a group departing in the same time step from the same location to the same destination) results in approximately deterministic flows as the number of tripmakers increases and the law of large numbers becomes more applicable. Even with collective application of path splits, however, it not infrequently happens that the size of a homogeneous group is so small that, after rounding, the path flows are again essentially random. In these cases, it was found that the device of "dividing" each individual vehicle into n independently assignable vehicle fractions populates the simulator with a sufficient number of flow elements to validate the large number laws; essentially

deterministic simulator outputs can be obtained in this way. Thus, it is possible to control both the degree and the sources of stochasticity in the simulator. Many of the methods used in the simple simulator to affect the stochasticity in its outputs should be applicable to other, more complex, traffic simulators as well. It is a matter for future empirical research to determine the corresponding degrees of stochasticity in actual path flows and vehicle propagation through a network, and to match simulator behavior to these.

Investigation of the computational properties of different problem formulations and solution algorithms used a simple test network and examined a variety of factors. The experimental design involved three different problem scenarios, the three composite map formulations, solution using the MSA and an iterate averaging method due to Polyak, and models with low and high degrees of stochasticity; all meaningful factor combinations were examined.

Problem scenarios included a full information situation without incident; a full information situation with an incident; and a general guidance situation with an incident, where only drivers who pass a particular location (a "VMS") on the network receive guidance. The low and high degrees of stochasticity were obtained by appropriate adjustment of simulator parameters as explained above, and were chosen to illustrate the range of possible output properties of quasi-deterministic maps. No attempt was made to "calibrate" the models to reality. Calibration of guidance models, including adequate replication of the actual nature and extent of stochasticity in traffic phenomena, is an area for future research; in any case, very little detailed data on the stochastic properties of path choices and vehicle propagation are currently available with which to attempt such an exercise.

For a given problem map, different test situations were compared using an "inconsistency norm" that measured the vector 2-norm distance between the input and output time trajectories of the map's variable. Because the simulator code was written to be easily modifiable and to monitor computational details, rather than to optimize computational efficiency, there was no attempt to measure or compare the execution times or memory requirements of the different test situations.

Referring first to the quasi-deterministic problem maps, it was found that the MSA could successfully solve the required fixed point problems but that it suffered from a considerable "tail" effect; this is a known property of the method. Iterate averaging (in particular, a method due to Polyak) outperformed the MSA by factors ranging from two to over four as measured by the value of the inconsistency norm in different iterations, at very minor additional computational cost (a single averaging operation per iteration).

It was not possible to directly compare the convergence properties of alternative formulations using a single norm because of stochasticity inherent in the norm evaluations. However, by examining the progress of the solution algorithms in reducing the inconsistency norm for different formulations, it was seen that path-based formulations (composite maps involving path splits or path time messages) generally converged in fewer iterations than link-based formulations.

For fully stochastic problem maps, both the MSA and Polyak's method were successful in computing a stationary distribution of the stochastic variables. However, there was so much variability in the solution output values that they are unlikely to form a reliable basis for route guidance. In such cases, either adaptive guidance or a multi-level guidance and traffic control system is likely to be required in order to respond quickly to local variations in measured traffic conditions while pursuing guidance consistency as an objective. This is identified as a subject for future research.

1.4 Contributions

Following are major contributions of this research:

- the development of a formal framework for the analysis of the general within-day RGG problem, including three fixed point formulations of guidance consistency and a unified treatment of deterministic, quasi-deterministic (noisy) and fully stochastic models. This framework opens the RGG problem to systematic investigation of solution methods;
- the identification of a class of stochastic approximation methods (iterate averaging methods) that are directly applicable to guidance generation and related (e.g., dynamic traffic assignment) problems. These simple algorithms can be rigorously applied whenever the MSA can be and, unlike the MSA, they have asymptotically optimal convergence properties. They appear to be relatively unknown to the transportation community;
- a detailed investigation of the causes, consequences and ways of controlling stochasticity in a commonly-used form of traffic simulation model. The conclusions of this investigation should extend to most simulators of the same type;
- the first known application of Gibbs sampling to determine the stochastic process output properties of a stochastic dynamic traffic model;
- a systematic computational investigation of alternative RGG problem formulations and solution methods, including iterate averaging. This is believed to be the first application of these methods to a traffic prediction-type problem, and shows great promise.

1.5 Thesis organization

Chapter 2 provides an overview of route guidance systems, examining them from the angle of what information they base their messages on. This chapter situates anticipatory route guidance systems within the range of guidance system types, and introduces the problem of generating consistent anticipatory route guidance (the RGG problem).

Chapter 3 proposes a framework for formulating the RGG problem. The framework allows the problem to be expressed in different ways as fixed point problems. Implications of stochasticity in problem formulation and solution are noted. Chapter 3 also discusses connections between the framework and related research that has appeared in the literature on traffic models.

Chapter 4 is a review and synthesis of methods for solving fixed point problems, particularly examining methods that are mathematically rigorous and computationally feasible in a route guidance setting. Solution methods suitable for fully deterministic, quasi-deterministic (noisy) and fully stochastic models are discussed.

Chapter 5 carries out computational tests of combinations of RGG problem formulations and fixed point solution methods using the simple traffic simulator mentioned above. The chapter presents the design of the simple simulator, describes the results of a number of test runs that were performed to ascertain basic simulator properties, then presents the results of the computational tests of guidance generation algorithms.

Chapter 6 presents the conclusions of the research and indicates a number of directions for future work.

Chapter 2

Approaches to Route Guidance

A route guidance system must perform some or all of the following tasks:

- obtaining, storing and retrieving background data about the road network, activity system and traffic flows and conditions in the guidance area;
- collecting real-time data on prevailing traffic flows, conditions and factors that affect these;
- processing the available data to generate guidance information;
- disseminating guidance information and data to system users and operators;
- transmitting data and guidance information between the various system components.

Each of these tasks can be approached in a number of ways. A variety of technological options is available to carry out each such approach, and the individual options can be combined in many feasible ways to form a very diverse range of integrated hardware and software systems for route guidance. Of course, actual route guidance systems are created and operated within particular institutional contexts, and this adds yet another dimension to the possible variety of systems.

It is not the intent of this chapter to develop a complete taxonomy of possible guidance system designs. Rather, the chapter surveys guidance systems from one point of view only: *what is the basis of the guidance information provided to drivers?*

In this dissertation, the terms *guidance*, *information* and *guidance information* will be used interchangeably to refer to any *message* that is conveyed to drivers by a route guidance system with the objective of affecting their route choice. These terms are intended to encompass any kind of message relevant to route choice decision, so the finer distinction that is sometimes made between prescriptive *guidance* (i.e., route recommendations) and descriptive *information* (i.e., information about traffic conditions) is not usually relevant to the discussion here. Indeed, there is some indication from tests of drivers in car simulators that the most effective guidance messages contain both prescriptive and descriptive elements ("30 minute delays ahead—take alternative route XYZ").

Using the type of underlying data as the basis for classification, guidance systems can be distinguished into those based on static and those based on dynamic data; within the class of dynamic systems, reactive and anticipatory systems can be further distinguished. Sections 2.1, 2.2 and 2.3 of this chapter discuss static, reactive and anticipatory systems, respectively.

The purpose of this selective survey is to illustrate the range of ways that have been proposed to use information for guidance; to situate anticipatory route guidance systems with respect to other guidance system types; and to introduce the problem of generating consistent anticipatory route guidance.

2.1 Static guidance

Static guidance refers to guidance that is independent of specific traffic conditions. It is derived from an infrequently-updated database of background information about the road network, the activity system that it serves and typical traffic conditions that result from their interaction, but it does not incorporate information on actual traffic conditions at any particular time.

Examples of static guidance include guidebook-type information (e.g., a listing of sites of interest in an area together with their locations and characteristics) and point-to-point driving directions based on "average" network conditions. Static guidance may be of use in tasks such as wayfinding or preliminary trip planning.

Static guidance is probably most useful to drivers who are relatively unfamiliar with an area. Guidance provides these drivers with planning and wayfinding capabilities that can save them time and effort before and during a trip. In recognition of this, car rental agencies sometimes offer vehicles equipped with navigation devices that display static digital map data from a CD-ROM. However, unless such drivers make up a sizeable portion of the total traffic on a network (such might be the case at major tourist destinations), and the static guidance system enables these drivers to avoid inefficient paths and unnecessary searches, the system would probably not have a significant impact on traffic conditions.

There is evidence that even drivers who feel that they know a network well may not be fully aware of the route options and so may choose from an unnecessarily restricted set of alternatives [Stern and Leiser, 1988]. For these drivers, static guidance systems may provide greater awareness of route alternatives, just as even experienced drivers can sometimes benefit from studying a local road map.

Because static guidance does not take account of actual traffic conditions, it has limited potential for affecting them in a controlled way. For this reason, static guidance will not be considered further in this dissertation.

2.2 Reactive guidance

Reactive guidance is based on data about the network conditions that prevailed at or around the time that the guidance is generated; these are called *instantaneous* conditions (i.e., a "snapshot" of conditions at single point in time). The guidance might be a summary description of traffic conditions at a particular time, or routing recommendations based on these conditions. The defining characteristic of reactive guidance is that the guidance messages reflect instantaneous conditions only; there is no attempt to predict future conditions as they may evolve from the traffic data measurements.

To the extent that the instantaneous conditions accurately reflect the conditions that a driver would experience in following an available path through the network, then route choice decisions made using reactive guidance will be well-founded. Conversely, the potential limitation of such

guidance is that network conditions may change significantly during a trip and so invalidate a decision based on the particular situation prevailing around its departure time.

One of the advantages claimed for reactive guidance systems is that, since they do not need to perform forecasts, they can respond quickly to changing traffic situations, and it is true that they generally carry out less processing than do the more complex anticipatory systems that are discussed below. However, this argument clearly only has force if reactive guidance is beneficial according to some evaluation criterion. The benefits of the guidance can be determined directly, through measurements of actual guidance systems or specific simulations; or indirectly, by comparing guidance generated on the basis of instantaneous and experienced conditions.

The question of when guidance based on instantaneous conditions would coincide with guidance based on experienced conditions has been investigated, usually for specific example networks, both analytically and by simulation. Examples of the former include Papageorgiou [1990] and Ben-Akiva et al. [1996]; examples of the latter include Pavlis and Papageorgiou [1999]. It is fair to say that conclusive, broadly-applicable answers to the question have not yet been rigorously derived. However, investigations suggest that reactive guidance may be effective for a particular trip when

- traffic conditions do not change radically during the trip;
- the duration of the trip is less than or comparable to the time scale of the dynamics of significant traffic condition changes; and
- the network provides ample opportunities for changing routes (e.g., there are frequent intersections providing access to alternate routes), and the tripmaker receives reactive guidance messages often enough to be able to take advantage of these route switching opportunities.

Reactive guidance approaches can be distinguished based on the amount of information that they require. Limited information approaches process data on traffic conditions in a portion of the network and they generate guidance for traffic in that area only. Full information approaches, on the other hand, require data on conditions throughout the entire network and they can output guidance for traffic anywhere on it. Representative limited and full information reactive systems are discussed in the next two sections.

2.2.1 Limited information approaches

Hawas and Mahmassani [1995] proposed a simple decentralized guidance approach based on measurements of prevailing link conditions in a local area, and estimates of conditions in the non-local area beyond. They assumed a network in which every node has an independent guidance controller that adjusts link splitting rates of traffic at the node.[1] Each controller receives real-time data about conditions on all links within a "radius" of k links (i.e., on subpaths k links long that start at the controller). Beyond the k links, the controller estimates the remaining distance to a destination using a combination of Euclidean and Manhattan metrics, and from that distance then estimates the travel time to the destination. (The non-local distance and time estimates are necessarily

[1] Link splitting rates $\lambda_{ij}^{d}(t)$ are the fractions of the flow exiting node i at time t for destination d via each of i's outgoing links (i,j). The briefer expression *splitting rate* is frequently used, but the more exact expression will generally be preferred in this dissertation to make clear the distinction between these fractions and *path splits*, which are introduced in Chapter 3 and used extensively thereafter.

path-independent since the controller has no knowledge of the network beyond the local area of radius k.) The sum of the known travel time on a k-link subpath from a controller node plus the estimated travel time from the end of the subpath to the destination provides an assessment of the subpath. (In some ways, this procedure resembles the A* minimum path algorithm [Hart et al., 1968], which computes the node labels needed for path finding using an approximate rather than an exact method.)

Once the various subpaths from a node to a destination have been assessed, a control rule is applied to determine guidance (in the form of link splitting rates). Drivers are assumed to comply with the guidance and to follow the k-link subpath to which they are allocated by application of the splitting rates. When they reach the end of one subpath, they receive from the controller located there new guidance for the next leg of their trip, continuing in this way until they arrive at the destination.

Hawas and Mahmassani considered three different control rules for setting the link splitting rates at nodes. The first two of these were based directly on the estimated travel time on different paths from the node to the destination. The first rule routed all traffic for a destination to the path having minimum estimated time. The second rule considered paths whose times are within a specified fraction of the minimum, and split traffic among these according to probabilities output by a logit function based on the estimated path times. The third rule used a more complex path assessment criterion, involving the average vehicle concentration on the initial subpath, together with distances and travel times on both the local subpath and the non-local remainder of the trip to the destination. It used this indicator in a logit function to split traffic among feasible paths.

Hawas and Mahmassani applied the DYNASMART mesoscopic traffic simulator to investigate the performance of the different routing schemes, using for this purpose a system optimal total time minimization criterion. They compared the total travel times of the flow patterns produced by the various control rules to the total time attained using DYNASMART's full-information system-optimal dynamic simulation-assignment algorithm.[2] The third method was found to have the best performance, achieving total times between 10% (for k=5 and light traffic) and 42% (for k=1 and heavy traffic) higher than DYNASMART's full information system optimal approximation.

Bolelli et al. [1991] also proposed a limited-information decentralized approach for reactive route guidance. The method requires the *a priori* specification of (fixed) nominal conditions for nodes (link splitting rates $\bar{\lambda}_{ij}^{d}$) and links (density, travel time, volume, or other measurable variables $\bar{d}_{ij}$). It adjusts the link splitting rates at every node in response to real-time measurements of link splitting rates and traffic conditions, attempting to drive the measured conditions to their nominal values.

Bolelli et al. considered two control methods for this problem. The first is fully decentralized feedback regulation, in which node-based regulators using a proportional control law react to the discrepancy between nominal and actual values of variables measured at the node and its outgoing links. The disadvantage of this method is that conditions elsewhere in the network—for example, congestion that is propagating towards but that has not yet arrived at a node—are not taken into account in determining the link splitting rates. The authors therefore proposed a second method which is intermediate between the local-level and full information approaches.

In this second method, an "information wave" of summary data about upstream traffic con-

[2]In fact DYNASMART actually calculates an approximate system-optimal traffic pattern because it approximates the first-order link time effects and it neglects the second- and higher-order effects of path flow changes.

ditions propagates from one controller to the next, in advance of and along the same path as the "congestion wave". A weighted combination of data from the information wave and local measurements is used as input to a feedback rule similar to that described above. This method is, in some respects, similar to the Gallager [1977] algorithm for distributed message routing in communications networks. Although all data used by the method are either current or older, the method allows link splitting rates to be set proactively, for example allowing a node to divert traffic away from a congested route before congestion actually reaches it.

The approaches of Bolelli et al. [1991] are well suited to a multilevel guidance system in which a higher level center computes set points (the nominal values of the control variables) which lower level controllers attempt to enforce through local adjustments. A sophisticated ATMS/ATIS system based on the second proposed method was in fact implemented in Torino, Italy [Mauro, 1998].

2.2.2 Full information approaches

Full information reactive guidance assumes that data on instantaneous flows or conditions are available for all paths in the network.

Obtaining such data requires more extensive sensor, communications and computation capabilities than did the limited information approaches discussed in the preceding section. In principle, a sufficiently extensive system of traffic detectors could provide the needed data using technologies similar to those deployed in a modern traffic control center. With more limited coverage of the network by traffic detectors, available measurements might need to be processed using a traffic model to obtain a full image of flows and conditions on the network. Methods for collecting and processing real-time traffic measurements to estimate prevailing traffic and network conditions constitute in themselves a major research area. Consideration of such methods is beyond the scope of this research; it is simply assumed that the measurements have been made and processed, and that the necessary data are available.

Given such data, generating reactive guidance is relatively straightforward: instantaneous path times might be disseminated without change (this would correspond to complete descriptive guidance) or they might be transformed into some other form such as path time synopses or path recommendations.

In the latter case, attainment of instantaneous user-optimality conditions could be simply achieved by computing the instantaneous path times and recommending the fastest. This is a form of "bang-bang" control, characterized by abrupt and possibly frequent changes in the guidance recommendations. Other forms of control law mitigate these undesirable features while still driving the network flows and conditions towards instantaneous user-optimality.

Papageorgiou and his co-workers have extensively investigated such approaches, using for this purpose a general state space model of flow propagation through a traffic network based on macroscopic traffic flow relationships [Papageorgiou, 1990]. The model's control variables are link splitting rates at all network nodes. OD demand rates and the amount of driver compliance with guidance link splitting rates are treated as the (exogenous and unknown) disturbances that drive the model. A computer program called METANET [Messmer and Papageorgiou, 1990] was developed to carry out the requisite calculations.

Papageorgiou investigated the effectiveness of feedback regulation in solving the instantaneous user optimal guidance problem. It was assumed that complete information on instantaneous travel

times was available at all node controllers. In one set of experiments, a bang-bang controller used the travel times to adjust link splitting rates. In another set the travel times were input to a multiple-input multiple-output (MIMO) feedback regulator with integral parts, where the proportional and integral gain matrices were estimated by applying linear-quadratic (LQ) optimal control methods to a linearization of the flow propagation model around a nominal steady state. The experiments showed that the considered MIMO feedback regulator is effective in achieving instantaneous user-optimal conditions, and is robust under changing demand and user compliance scenarios. This work was described in Papageorgiou and Messmer [1991] and in Messmer and Papageorgiou [1994].

Along similar lines, Pavlis and Papageorgiou [1999] investigated by simulation the effectiveness of fully decentralized single-input single-output (SISO) bang-bang, proportional (P) and proportional-integral (PI) feedback regulators for achieving instantaneous user optimal conditions in densely meshed corridor networks[3]. They did this by controlling the link splitting rates at a subset of nodes, again assuming the availability of instantaneous path times from the nodes to the destinations. All controllers were of the same type and used identical gain parameters, which were determined through manual (i.e., trial-and-error) tuning. Pavlis and Papageorgiou showed that SISO feedback regulation control can be an effective means of attaining instantaneous user-optimality in such networks. They further showed that, in the networks they considered, user optimal flow patterns based on guidance derived from prevailing and from actual times were essentially identical. In these networks, in other words, fully decentralized SISO regulators reacting to instantaneous times were able to attain user optimal conditions based on experienced times. A VMS system implemented in the Aalborg (Denmark) urban area as part of the QUO VADIS project[4] disseminated reactive guidance obtained using a similar decentralized proportional control approach [Mammar et al., 1996].

2.3 Anticipatory guidance

In anticipatory guidance, the guidance information that drivers receive is based on the conditions that they are likely to experience throughout the duration of their trip: information on what traffic conditions will be at some location *at the time drivers actually arrive there*, for example, or path recommendations that take account of the changing performance of network elements over time because of changing traffic loads.

Clearly, anticipatory guidance must be based on predictions of future traffic network variables. In general, these predictions will be derived from a combination of real-time traffic measurements and historical data on traffic patterns and conditions. These data sources are inputs to a forecasting model, which carries out the required predictions.[5] In some cases, the predictions themselves might be disseminated as (descriptive) guidance; in others, they will be transformed in some way with the results of the transformation constituting the guidance.

[3]These were corridor-type networks with two parallel main routes offering numerous possibilities for switching between them.

[4]QUO VADIS was a project in the European Community's DRIVE II program. Its objective was to investigate the potential of VMS for the provision of high quality route guidance and the improvement of network capacity.

[5]Again, the appropriate processing of real-time data for traffic condition forecasting is a major research topic in itself, and is not considered in this research. It is simply assumed here that data have somehow been collected and processed into the inputs required by a traffic prediction model.

The basic problem of anticipatory guidance generation is this: when drivers receive guidance based on predictions of future traffic conditions, their reactions to the guidance (whatever those reactions might be) will affect future traffic conditions, possibly invalidating the predictions and rendering the guidance ineffective or worse.

Of course, if the number of drivers who receive and react to the guidance is so small that their reactions have negligible effect on network conditions, then these effects can be ignored when predicting traffic and generating guidance: forecasting could be done using some conventional dynamic traffic model, with guidance directly generated from model outputs. Predicted path times or dynamic shortest paths, for example, could be calculated from the model outputs and disseminated as guidance. Such guidance might be useful to the drivers who receive it but, by definition, would not have system-level benefits.

If, on the other hand, the combined reactions of individual drivers to guidance are significant enough to affect network conditions, then these effects must themselves be taken into account: the traffic prediction model must be able to forecast the conditions that will result from the dissemination of guidance; and guidance generation must consider its own impacts on future conditions when determining what messages to disseminate. These effects make themselves felt over a time span starting at the time of dissemination and extending into the future. Because of this, the generation of reactive guidance from prevailing conditions data does not need to explicitly consider the effects of the guidance.

The notion of guidance *consistency* arises naturally when it is recognized that guidance generation must take into account guidance effects on future traffic conditions. Guidance is said to be consistent when the assumed future conditions from which guidance is derived turn out to be realized, based on the predictive model logic, after drivers receive the guidance and react to it.

Anticipatory guidance has been less investigated than reactive or static guidance. The remainder of this chapter first describes a number of approaches described in the literature for handling special instances of the problem, then turns to the anticipatory route guidance problem in its full generality. It discusses modeling and solution method requirements posed by anticipatory guidance, comparing these with dynamic network models based on an assumption of full information availability. It provides a simple example intended to demonstrate that such full information approaches are not a correct basis for guidance generation. It concludes with a brief discussion of issues related to the design and operation of a traffic information center; this is useful as a background to portions of later chapters.

2.3.1 Link-level extrapolations

One approach to predictive guidance measures prevailing link conditions, extrapolates these into the future, and generates guidance based on the extrapolated conditions. The ALI-SCOUT route guidance system [von Tomkewitsch, 1986], for example, was based on this idea.

Link condition extrapolations are generally based on typically observed variations in link conditions by time of day (these are called travel time "development curves" in the ALI-SCOUT literature). Measured prevailing conditions are related to the corresponding typical conditions depicted on the curves and extrapolated. The extrapolations are thus independent of the generated guidance: they do not take account of drivers' reactions to guidance that might be generated based on them.

For this reason, such systems cannot be considered anticipatory, and will not be considered further in this dissertation.

2.3.2 Incident-only predictions

Wang et al. [1992] briefly outlined a novel approach that could potentially provide some of the benefits of fully-general anticipatory guidance while avoiding some of its complexities. They call their method *partially-predictive* guidance. The idea is simple:

- in normal traffic situations, base guidance on prevailing conditions, as in the reactive guidance approaches discussed above;
- when an incident significantly affects the capacity of a link, make a prediction of the time evolution of that link's conditions based on expected demand and on an estimate of the reduced capacity. In generating guidance, use prevailing conditions for unaffected links and predicted conditions for affected links.

The authors do not suggest specific techniques for making the required link condition predictions.

In support of this approach, Wang et al. acknowledge the theoretical advantages of guidance based on instantaneous rather than experienced conditions, but they also point out the difficulty of generating anticipatory guidance with sufficient timeliness and accuracy to be of benefit to drivers. The question of timeliness is particularly *a propos* in incident situations, when there may be a significant lag between the occurrence of an incident on the one hand, and its incorporation in full-information guidance generation on the other. Reasons for such a lag could include delays in (i) detecting the incident, (ii) quantifying its characteristics (amount of capacity reduction, expected duration, etc.) with the precision needed for reliable forecasts, (iii) re-running the prediction and guidance generation model, or (iv) disseminating the new guidance.

Nonetheless, the approach is clearly *ad hoc*. Incident situation forecasts made in this way are likely to be approximate because they do not account for possible network-wide effects such as spillback, and do not consider the effects of the guidance itself. However, the authors argue that in incident situations the rapidity of a guidance response can be more important than its accuracy because a rapid response may significantly decrease the size of the resulting queue, hasten queue discharge, and reduce overall delay. This argument is not without merit, but it is possible to imagine situations where incorrect guidance, rapidly disseminated, quickly worsens rather than improves an incident situation. In any case, simulation or operational tests of this approach do not appear to have been reported in the literature.

2.3.3 Path-level predictions

Messmer et al. [1998] discuss a different guidance generation method that avoids dealing with the full complexities of network-level guidance generation. The method was used to generate VMS messages in a QUO VADIS test project involving the Scottish interurban motorway network.

The project network is sparse, and is characterized by relatively long links and relatively few opportunities to switch from one path to another. As indicated above, reactive guidance is unlikely to be effective in a network with this structure. On the other hand, because of the relatively simple network structure, if accurate models of traffic flow and queue dynamics are available, then

traffic measurements made at the beginning of a link are likely to allow good predictions of link conditions at downstream locations at some later time. Messmer et al. took advantage of this characteristic to develop a Smith predictor regulator for the control of variable message signs in the network. Smith predictors are standard control law forms that combine both feedback and feedforward terms: the feedforward term uses measurements of the real-time system disturbances (traffic entering each link) to predict the future system states (link delays and spare capacities), and the feedback term then determines the appropriate control response to the predicted states. In general, Smith predictors can be effective in situations with high latency (lag between input disturbances and output response), provided that it is possible to accurately measure the input disturbances and predict the system outputs.

The authors performed off-line testing of the approach under different incident scenarios using the METANET macroscopic traffic simulation model and found that, in most cases, benefits (in the form of avoided travel delays) were significant. The approach was then implemented by interfacing the control logic with an existing traffic control system covering the project network. Initial "dummy control" operations (in which the system performed all tasks using real-time data, but the computed messages were not displayed on the network's VMS) were followed by a full implementation phase of automatic control. Although the QUO VADIS schedule only allowed a relatively short period of operation of this system, Messmer et al. report that public response to it was favorable.

2.3.4 Network-level models

In the general case, guidance generation needs to consider traffic flow and conditions at the level of an entire network. The approaches discussed in the preceding sections, while effective in particular situations, do not readily scale to more general network-level guidance problems. To handle such problems requires a model that can predict traffic flows and conditions over an entire network, and that can take account of the effects of guidance itself when generating guidance.

There is a wide variety of analytical and simulation-based dynamic traffic modeling systems currently in research or production use, but very few of these provide capabilities for realistically representing guidance information and driver response to it. In later discussions, it will be convenient to refer to a "typical" dynamic traffic model that exemplifies a conventional approach to dynamic network modeling and includes a set of features present in some form in many (although by no means all) currently-used traffic modeling systems; this typical model will be contrasted with the specific modeling requirements of the route guidance problem.

Such a model considers an analysis period typically between a fraction of an hour and several hours, focusing exclusively on relationships whose effects are seen during this time span (a *within-day* model). It is initialized with a set of flows on each link (an initially empty network is a common choice). The analysis period is discretized into a sequence of time steps. A schedule of trip departures per time step, by origin, destination and tripmaker characteristics is exogenously provided. In each time step, trips due to depart from an origin are associated with specific paths to their destination according to some routing principle. Typical routing assumptions are that trips follow a minimum time path, or that they select one from among the available paths according to a probabilistic choice model. A key assumption is that routing decisions are made on the basis of full information about path options and characteristics. Because the information might not be perfect,

or because drivers might integrate other considerations or perceptions (which available data cannot measure and a model cannot capture) in their route choice decision, probabilistic model forms are sometimes used; this does not change the basic assumption that drivers make path choice decisions at the origin based on full information about path alternatives. For this reason, these models will be called *full information models* in this dissertation.

When the trips departing from their origins have determined their paths, all trips—those just beginning their journey as well as those already on the network—are advanced along their paths towards their destination. Full information models generally assume that once a trip has been associated with a path, it follows that path unswervingly to its destination. Network conditions determine the amount of advance per time step on each link and from link to link. Conversely, network conditions are themselves determined by the dynamics of traffic propagating along paths. The relationship between time-dependent trips by path and the resulting time-dependent network conditions is worked out by a procedure called *dynamic network loading.*

The primary task of the model is to ensure that the network conditions that were the basis for associating trips with paths coincide with the conditions actually experienced by trips as they travel over the network. If minimum time path routing were assumed, for example, the model would attempt to ensure that a path that was thought to be minimum before loading in fact proved to be so after loading. A full information model computes a dynamic network supply-demand equilibrium in the sense that, according to the routing assumptions incorporated in the model, no trip has an incentive at any time to change from the path it is following to some other path.

Route guidance research is sometimes carried out with full information dynamic network models. They might seem to be a natural choice for such work because they provide a network-level prediction capability that generalizes the incident-specific and path level approaches discussed earlier. In such applications, a full information model is provided with an initial network state and a time trajectory of dynamic OD demand over a specified horizon. The model is run, and guidance is derived from the time-dependent network conditions that it outputs. However, the full information assumption is inappropriate for guidance modeling because it provides limited ability to represent guidance or drivers' reactions to it.

Realistic modeling of route guidance issues must explicitly recognize the characteristics of the guidance information available to drivers. Many of the limitations derive from the nature of the technology used to compute and disseminate guidance messages. Such limitations can take various forms. For example, guidance might not be available to all network users because its reception may require special equipment. Even to vehicles able to receive it, guidance might not be ubiquitous (available everywhere on the network) because its reception range may be restricted to a relatively short distance (a few hundred meters) around specific dissemination infrastructure such as VMS or infrared beacons. Limited communications bandwidth and human information processing ability might constrain the detail and accuracy of the information that can be conveyed in guidance messages. Computation and communication delays might leave drivers with out-of-date guidance. The guidance itself might be inaccurately computed or perhaps corrupted during transmission.

Guidance is an effort to improve on the imperfect information that is the basis for most drivers' route choice decisions. But, as the above discussion indicates, guidance information itself has limitations and is not perfect. It is readily conceivable that, for a given network and traffic pattern, two guidance system designs, differing in one or more of the above aspects, may have very different impacts on network traffic conditions. It follows that realistic guidance modeling must be able to

represent specific characteristics of guidance system design.

Having represented the characteristics of guidance information, realistic route guidance modeling must also accurately capture the diversity of possible driver responses to guidance messages. Some drivers may rely heavily on guidance information; others may choose to do the *opposite* of what the guidance suggests in an attempt to "avoid the crowd"; yet others may ignore it completely and follow their habitual route choices. Driver behavior models sufficiently sophisticated to faithfully replicate this range of reactions are the subject of active research, but are still in their beginning stages.

There are various ways in which a full information model might be enhanced to meet the requirements of traffic forecasting in the presence of guidance information. In general terms, the simulator would require the addition of some means of representing guidance messages and their characteristics. Its routing component, which determines how OD demand is allocated among available paths, would need to be modified to incorporate the effect of these messages on path allocation; a particular issue is driver response to en route guidance. Finally, the dynamic network loader would have be changed to correctly handle the subpath and link flows that ensue from en route path switches. These enhancements would allow a traffic simulation model to carry out dynamic traffic assignment in the presence of pre-trip and en route guidance information.

More specifically, adequate incorporation of guidance effects in forecasting requires a dynamic traffic model that can:

- represent guidance: its content, time and duration of dissemination, reception area, etc.;
- predict the effects of guidance on driver behavior, in the form of pre-trip decisions to choose a path and en route decisions to switch from one path to another; and
- predict the traffic flows and conditions that ensue as a result of drivers' responses to guidance.

A model with the above functionality would, in principle, be able to predict the consequences of disseminating a given set of guidance messages.

A traffic model that incorporates these features will be referred to as an *enhanced* traffic model. Although implementation of such enhancements clearly involves many decisions about both major issues and details, there is no fundamental reason why most standard full information models could not be modified to accommodate them within their existing structure.

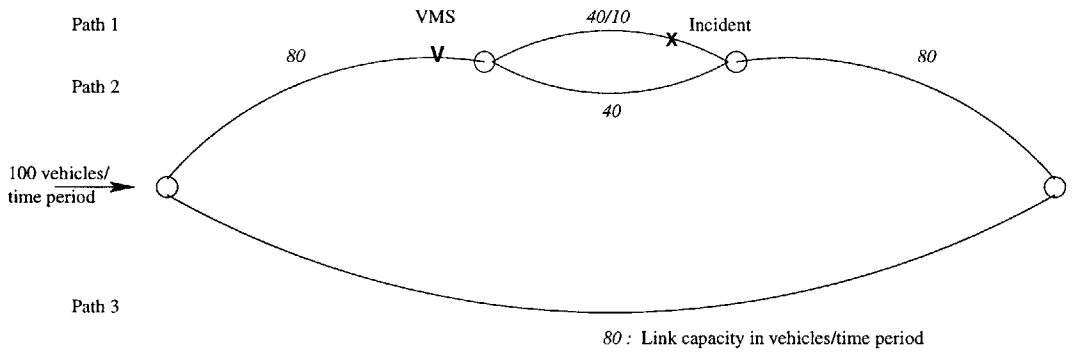
Yet even if a full information model were so enhanced, it is still not clear how it would be applied to generate consistent anticipatory route guidance. For example, if we somehow guessed that a particular set of guidance messages was appropriate and ran an enhanced traffic model to predict the network conditions that would result from this guidance, we might conclude, after examining the predictions, that a different set of messages should in fact have been disseminated. This indicates that the initial set of guidance messages was not consistent, but does not indicate how they should be modified to achieve consistency.

The problem of generating consistent anticipatory guidance under general and realistic assumptions about the guidance system is the primary focus of the research presented in this dissertation.

2.3.5 Guidance characteristics have to be modeled

This section presents a simple example intended to emphasize the point that, in route guidance applications, the detailed features of guidance need to be explicitly modeled. It considers an

Figure 1



Path flows (vehicles/time period)

Path	Non-incident	Incident DTA	VMS with DTA	VMS with RGG
1	25	10	13	10
2	25	30	37	40
3	50	60	50	50

Figure 2.1: Simple guidance example network

incident situation where some trips pass by a VMS and receive guidance. Two ways of generating guidance for the VMS are considered. In one, the actual features of the guidance system (limited range of the VMS) are explicitly considered when generating guidance; the other implicitly assumes that guidance provides full information.

Consider the network depicted in Figure 2.1. There are three OD paths, labeled 1 through 3 from top to bottom. Each link allows flow to traverse in fixed time until its capacity is reached, at which point queuing delays occur. The capacity is 80 vehicles/time period on the long links and 40 vehicles/time period on the short links. Uncongested travel times are the same on all paths. The OD flow rate is constant over time at 100 vehicles/time period. Drivers choose minimum time paths. In this situation, a full information dynamic traffic model might predict flow rates of 25-25-50 vehicles/time period on the three paths. Now suppose that an incident occurs at the location indicated by an "X" on the top path, partially blocking the link and reducing its capacity to 10 vehicles/time period. Suppose that, after a period of transient adjustments, the traffic model now predicts path flow rates of 10-30-60 vehicles per time period.

But how do drivers know which paths have minimum time in the incident situation? Their experience may have taught them the characteristics of the different paths in non-incident conditions but, when an incident occurs, their prior knowledge most likely becomes invalid. If drivers are unaware of this, they will probably maintain their habitual behavior until otherwise informed. Suppose that the only source of route guidance is a VMS indicated by a "V" at the bifurcation of paths 1 and 2. The traffic management center (TMC) can influence the fraction of flows choosing

each path at the bifurcation by varying the amount of time that different route recommendation messages are displayed on the VMS.

If the TMC were to base its guidance on the full information model forecasts for the incident situation, it would assume an arrival flow rate of 40 vehicles/time period at the VMS, and would attempt to route 75% of the drivers passing the VMS away from the incident. Drivers leaving the origin, however, are unaware of the incident and have no reason to change their habitual behavior. More realistically, therefore, the flow rate arriving at the VMS would be expected to be at its usual value of 50 vehicles/time period. Applying the full information model guidance would send 13 vehicles/time period to a link with a capacity of 10 vehicles/time period. The extra traffic routed towards the incident would eventually spill back, leading to blockage of paths 1 and 2, diversion of all traffic towards path 3, and gridlock. If the flow prediction model had more carefully recognized the specific information available to drivers at different locations, the TMC could have routed 80% of drivers at the VMS away from the incident, and could have indefinitely sustained equilibrium path flows of 10-40-50 vehicles/time period.

What went wrong with using a full information model to generate guidance in this example? Clearly the problem is that the model does not take appropriate account of the information-related aspects of the problem. In this particular case, the model did not recognize the VMS and its purely local effects. More generally, there is no representation in a full information model of how or in what form dynamic network condition information is made available at the time of drivers' initial path choice decisions, no possibility of providing additional information en route, no model of how drivers might react to such information updates when they are received, and no provision for en route path switches that might result from drivers' reactions. In fact the full information model implicitly assumes that drivers' pre-trip perceptions of network conditions are fully accurate. Of course, if this assumption were correct, there would be little need for route guidance.

2.3.6 Anticipatory guidance system implementation issues

The focus of this research is on the formulation and solution of the anticipatory route guidance generation problem, rather than on the design or implementation of an operational route guidance center for some network. Nonetheless, it is appropriate to describe briefly here how such a center might apply a guidance generation system in practice, since this is background to some of the discussion in later chapters.

A reasonable implementation approach would be to adopt a rolling horizon method. In this method, traffic condition and flow data are collected in real time from sensors distributed over the network. In each roll stage (every 15 minutes, for example), the current network state is estimated from the accumulated data. Then, with the estimated current state as a starting point, mutually consistent network flow and condition forecasts and guidance are determined over a fixed duration guidance horizon (say one hour into the future).

Guidance consists of a set of time-dependent messages to be disseminated to drivers via the available communications system. The time discretization used for these messages defines the guidance update interval (one minute, for example); at each guidance update, the set of messages corresponding to that time span is disseminated. Guidance messages are considered to be fixed over the duration of an update interval.

The guidance computation needs to consider a horizon beyond the next roll stage since traffic

conditions predicted to occur at later times may affect the guidance provided to currently-traveling vehicles having trip durations longer than one stage. However, only the set of messages for times up to the next roll stage are actually disseminated. Then data collection and guidance processing begin anew after moving forward in time by one roll stage.

Implementation of this rolling horizon framework requires decisions about the different parameters identified above. By varying these parameters, it is possible to influence the computation speed and the accuracy of the computed guidance solution. This is an important general relationship that is central to the design of an operational system with real-time response capability: by taking more computer time, a more consistent guidance solution can in principle be computed; on the other hand, a longer computation time reduces the timeliness and relevance of the guidance once it is determined and disseminated.

Following are some specific rolling horizon implementation parameters and the speed/quality tradeoffs they affect:

- the length of the roll stage: a longer stage requires less frequent guidance recomputation and allows more time for guidance generation and dissemination; on the other hand, it involves forecasts over a longer time period, which are a less accurate basis for guidance;
- the length of the guidance update interval (or equivalently, the number of guidance changes in a roll stage): a longer interval between guidance changes requires less computation but makes for less precise guidance (because messages are less able to track predicted changes in conditions) and may lead to situations in which significant numbers of drivers all react in similar ways, shifting the location and possibly exacerbating the level of congestion;
- the tolerance used (or the number of iterations allowed) for the determination of consistent guidance: a less stringent tolerance or fewer number of iterations speeds termination of the algorithm but reduces the accuracy of the computed guidance.

Bottom et al. [1999] examine and report on some of these tradeoffs, using the DynaMIT traffic simulation software system. However, it is fair to say that understanding of these issues is still in a very preliminary stage.

Chapter 3

Consistent Anticipatory Route Guidance

3.1 Overview

The preceding chapter identified a number of characteristic requirements of the consistent anticipatory route guidance generation (RGG) problem:

- guidance generation must recognize the specific properties (content, format, precision, accuracy, temporal and geographic availability, etc.) of the guidance messages that are provided to drivers;
- because a guidance generation model provides an explicit representation of guidance, it must also be able to predict how drivers will react to the different types of guidance messages that can be represented;
- in general route guidance modeling, a trip may decide to switch from one path to another at any en route location where guidance information is received, rather than follow unswervingly a path chosen at the origin;
- a guidance generation model must achieve consistency between the future conditions that are assumed as the basis for guidance, and the conditions that are predicted to result after guidance is disseminated.

Based on the above, RGG is seen to be a more general problem than the full information dynamic network problem discussed in Chapter 2; in fact, it will be argued below that the full information problem is a special case of the RGG problem. Because of these differences, results obtained from analyses of the full information problem do not necessarily apply to the RGG problem. Indeed, it may not be immediately obvious how to formulate and solve the RGG problem.

To this end, this chapter proposes a framework that allows the formulation and analysis of the RGG problem. By *framework* is meant here the identification of the principal elements important to the problem, and of the significant relationships between them. A framework displays the logical structure of the problem without necessarily being mathematically rigorous. It represents the relationships between the component models of a route guidance model system rather than the specifics of the individual models themselves. It is abstract and lacks operational detail, but is sufficiently clear that existing model systems can be represented within it and new model systems

can be derived as instances of it. Finally, it has enough content to suggest fruitful solution approaches, although development of efficient methods for particular models may depend on specific model details.

The chapter is organized as follows. Section 3.2 describes the variables and component maps that make up the RGG framework. It then proposes three different formulations of the notion of consistency in anticipatory route guidance, by defining alternative compositions of the components. In each case, consistency corresponds to a fixed point of the composite map. The discussion in Section 3.2 treats deterministic maps only.

In practice the various mappings involved in route guidance generation are often evaluated using simulation, and many simulators used for this purpose incorporate random elements in their logic, either to replicate the randomness of reality itself, or as a computational device to facilitate the required processing (for example, by randomizing the processing order of links in order to avoid biases that would occur if vehicles on one link were systematically processed before those on another). The resulting stochastic models require an analysis considerably different from that of deterministic models. For this reason, deterministic and stochastic formulations are considered separately below, in sections 3.2 and 3.3 respectively. It is an indication of the robustness of the framework that, despite the very different issues of detail, consistency in both deterministic and stochastic anticipatory guidance models can be analyzed using fixed point concepts.

The final section of this chapter, Section 3.4, relates to the framework a number of published research efforts in route guidance and similar areas. This illustrates the framework's usefulness as a structure for analyzing the RGG problem, and also highlights the differences between these efforts and the approach adopted here.

In Chapter 4, the elaboration of the framework is pursued by considering techniques for generating consistent anticipatory route guidance; from the conclusions of the present chapter, it follows that attention can be restricted to fixed point computation methods. Solution methods are considered for fully deterministic, fully stochastic and quasi-deterministic problems. Following the review of possible solution methods, Chapter 5 describes the features of a software system that implements a simple instance of this chapter's framework, and applies the software empirically to investigate the properties of different fixed point consistency formulations and solution approaches.

3.2 Framework for the RGG problem: deterministic case

3.2.1 Components of the analysis framework

A basic framework for analyzing anticipatory route guidance is defined by three time-dependent variables and three maps that interrelate them. The variables of interest are path splits, network conditions and guidance messages. The maps of interest are the network loading map, the guidance map and the routing map. These maps can be combined in a variety of ways to obtain composite relationships needed to define and compute consistent anticipatory guidance.

The discussion below presents all these modeling constructs, beginning with an identification of the framework's exogenous elements in subsection 3.2.1.1. Framework variables are discussed in subsection 3.2.1.2, the framework maps are discussed in subsection 3.2.1.3, and the composite maps are discussed in subsection 3.2.2. Where appropriate, the discussion also points out how these constructs differ between route guidance and full information models. Following the presentation

of the framework, subsection 3.2.4 provides a number of comments on the framework as a whole, as well as some explanation of why it was defined in the particular way it was, and a discussion of alternatives that were considered.

For ease of reference, Figure 3.1 diagrams the route guidance generation framework that is presented in this section.

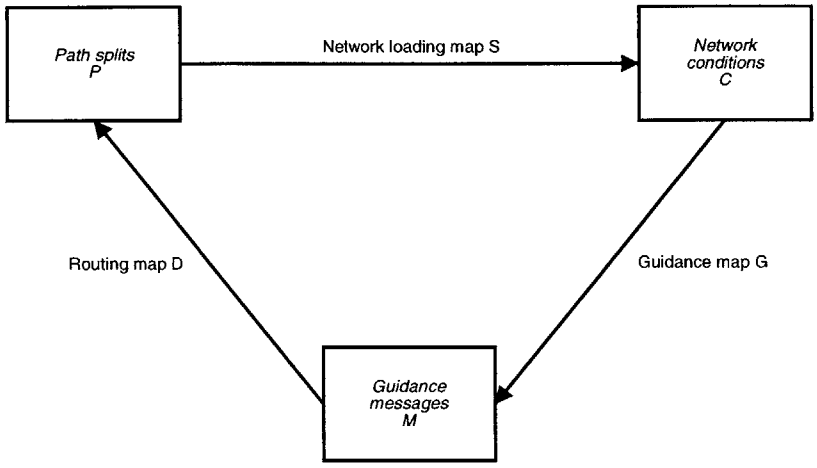


Figure 3.1: A framework for route guidance generation

3.2.1.1 Exogenous elements

Network topology Assume a standard representation of a traffic network as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ consisting of sets of nodes $\mathcal{N}$ and directed links $\mathcal{L} \subseteq \mathcal{N}\mathrm{x}\mathcal{N}$, together with their associated attributes (length, capacity, etc.) Subsets $\mathcal{O}$ and $\mathcal{D}$ of the nodes are distinguished as origin and destination centroids, representing locations at which trips enter and leave the network, respectively.

Time frame Traffic dynamics evolve over a fixed analysis time horizon that is discretized into a finite number T of fixed duration time steps indexed by $\{1, \ldots, T\} = \mathcal{T}$. The sequence of values of a time-dependent variable at each of the successive time steps in $\mathcal{T}$ will be referred to as the *time trajectory* (or simply *trajectory*) of the variable.[1]

Demand Trips traveling on the network are characterized by their user class $u \in \mathcal{U}$. A trip's user class determines its origin $u_o \in \mathcal{O}$ and destination $u_d \in \mathcal{D}$; parameters $u_{z_1} \ldots$ related to its path choice behavior (trip purpose, value of time, etc.); and the types $u_{g_1} \ldots \in \mathcal{G}$ of guidance information that it can receive. The level of OD trip demand (either trip rates or absolute numbers of trips) by user class u can vary from one time step to the next; however, the time trajectory $\mathcal{X}^u(t)$ (schedule of demand as a function of the time step) is exogenous and fixed. This latter assumption distinguishes the route guidance generation problem from more general problems in

[1] Time could equally well be treated as continuous. Discretized time has been adopted to facilitate the exposition and because a discrete time simulator (described in Chapter 5) has been used to illustrate the framework described in this chapter.

which the decision to travel, the departure time and mode choice may also be influenced by guidance information. The more general guidance problems can be handled by a straightforward extension of the approach described here, but it was preferred not to do this in order to keep the discussion focused. Treatment of more general guidance problems is discussed in Chapter 6.

Paths A path h is a sequence of adjacent directed links, all distinct, that begins at an origin centroid o and ends at a destination centroid d. Let $\mathcal{H}^{od}$ be the set of paths joining o to d. It is assumed that the set of feasible paths in the network is explicitly enumerated and that a path's origin h_o and destination h_d can be determined from the path's identification. A trip of user class u is assumed to be following a path $h \in \mathcal{H}^{u_o u_d}$ at all times.

Decision points Many full information traffic models assume that path choice is a pre-trip decision: prior to setting out from the origin, a tripmaker identifies and assesses the set of available paths, then chooses one path from the set and follows it unswervingly to the destination. This approach does not recognize that travelers may decide to change paths en route because of conditions that they experience or guidance information that they receive while traveling, and so is ill-suited to guidance modeling.

To address the modeling of en route guidance acquisition and path switching, a subset $\mathcal{Q} \subseteq \mathcal{N}$ of nodes is distinguished as *decision points*. At these nodes, trips may change from one path to a different one continuing on to the same destination; such changes are not otherwise possible. The set of decision points encountered on paths going from origin o to destination d is designated $\mathcal{Q}^{od}$.

At each decision point, one or more types of guidance information is available. The guidance provided at decision node q is accessible by trips on links in the decision node's backward star $B(q) = \{(i,q) \in \mathcal{L}\}$. If a trip on such a link is of a user class that corresponds to one of the guidance types provided at the decision point, it receives the corresponding guidance.

Origin centroids are always considered to be decision points. Trips entering the network receive information there. If these are the only decision points, then only pre-trip path choice is allowed. Alternatively, if all nodes are decision points, then path switches are allowed anywhere. More generally, the set of decision points will be the union of the origin centroids with an arbitrary set of other nodes; this allows modeling of pre-trip path choice as well as en route path switching at short-range guidance sources such as VMS or infrared transmitters.

It will sometimes be said that the guidance at a decision node is provided by a *beacon*. Strictly speaking, this term applies only to short-range infrared or microwave transmitters that some guidance systems use to disseminate guidance to vehicles equipped with special receiver and display devices. For convenience of expression, the meaning will be extended in this dissertation to include any source of guidance information, considered as originating at the decision node; thus, a wide-area radio service that blankets a network with guidance information is considered to broadcast its messages via beacons at all nodes.

Subpaths Decision points induce a natural decomposition of paths into subpaths: an arbitrary sequence of adjacent directed links that begins at one decision point and ends at another. A complete origin-destination path can be seen as a sequence of adjacent subpaths connected at decision points. Given a decision point $q \in \mathcal{Q}^{od}$, $\mathcal{H}_q^{od}$ designates the set of complete paths from o to d that go through q, $\mathcal{H}_q^{o\cdot}$ designates the set of subpaths going from o to q and $\mathcal{H}_q^{\cdot d}$ designates

the set of all subpaths going from q to d. The set of feasible subpaths is automatically determined by the set of paths and decision points. A subpath's beginning and end nodes are implied by its identification.

3.2.1.2 Framework variables

This section presents the framework's endogenous variables. All of these variables are time-dependent and defined on the set $\mathcal{T}$ of time steps.

Path splits Given a decision point $q \in \mathcal{Q}$ and a destination $d \in \mathcal{D}$, a path split $p_h^u(t)$ is the probability that a trip by user class u that departs in time step t will follow subpath $h \in \mathcal{H}_q^d$ from its beginning node q to the destination d. A path split is considered to be defined even if there is no traffic of class u leaving decision node q at time t. When there is such traffic, the path splits may be applied in either an aggregate or a disaggregate fashion. Aggregate application means that the path splits are treated as fractions: the total traffic of class u leaving node q at time t is multiplied by the various fractions to obtain the respective (deterministic) subpath flows. Disaggregate application means that the splits are treated as subpath choice probabilities. Each flow unit independently selects a subpath in accordance with these probabilities and it follows that, in this case, the subpath flows are multinomially distributed. In either case, conservation of flow at the decision point requires that

$$\sum_{h \in \mathcal{H}_q^d} p_h^u(t) = 1 \ \forall t.$$

$P(\cdot) = \{p_h^u(t)\} \ \forall u, \forall h, \forall t$ refers to the entire set of such time-dependent fractions over the entire analysis period.

Network conditions In general, network conditions represent any indicators of the performance of a network in facilitating the movement of traffic. These indicators are considered to be objective ("engineering") values that are constant within each time step of the analysis period. There is clearly a very wide range of possible measures of network conditions; examples might include the presence or not of incident conditions at some downstream location, the length of the queue at a particular node, the time to traverse a link, or the total cost under a congestion pricing scheme of following a particular path to a destination. For simplicity, it is assumed that conditions do not need to be distinguished by vehicle type. A model may include several different condition variables.

Although the framework can handle arbitrary definitions of network conditions, in the interest of definiteness conditions will generally be taken in this dissertation to be link-level impedance variables; this is a standard choice. Note that link traversal time must always be a defined condition variable, even if model relationships involve other impedance variables, because of its role in establishing traffic dynamics. The entire set of condition variable values over time horizon $\mathcal{T}$ will be designated by $C(\cdot)$; $c_\ell(t)$ is link ℓ's condition in time step t. This is defined as the average impedance experienced during its traversal by trips that *entered* the link during the time step.

Because of the use of average values to define link condition data, these will not in general provide information on the impedance experienced by individual trips. The duration of a time step could be made smaller, in an attempt to reflect more accurately the impedance encountered by individual trips. However, this would increase the number of time steps during which no

trips enter particular links, increasing the variability of the data or requiring use of interpolation or filtering to estimate missing values. Conversely, the duration of the time step must be less than the minimum possible link traversal time (the minimum of all link free flow travel times), since otherwise trips would be able to "jump over" links and this would lead to model errors and numerical instabilities since flow conservation relationships would be violated.

Guidance messages Guidance information is disseminated in the form of discrete units that will be called *messages*. A guidance message $m_q^g(t)$ can be represented as a quadruple involving guidance type $g \in \mathcal{G}$, decision point $q \in \mathcal{Q}$, time $t \in \mathcal{T}$, and content. The guidance type (a categorical variable) indicates the characteristics of the message, such as the technology involved in its dissemination and the nature of its content, and determines what user classes can receive it; it is also convenient to define a universal message type that can be received by any user class. A message's time t is the time step in which it is disseminated; the message is maintained during the entire time step, but must be re-disseminated in subsequent steps if it is to be continued for a longer period. It is supposed that a message is accessible to any trip of a user class that is capable of receiving messages of its guidance type, and that is on a link in the backward star of decision point (node) q during the time step t of dissemination. If a trip is on a link for multiple time steps and receives different messages during that period, it is assumed that only the last message received before it leaves the link will be used for path choice decisions. The set of all messages disseminated over the time horizon $\mathcal{T}$ is designated $M(\cdot)$.

Content is the most loosely defined of the message components. Depending on the guidance technology being represented, the content might be an element of a finite set (e.g., prescriptive guidance in the form of a particular path recommendation), a matrix of real numbers (e.g., descriptive information in the form of a table of predicted network conditions by time period), or something else. (Note that it is possible and reasonable for a message to have both prescriptive and descriptive components: "30 minute delays ahead, take alternate route XYZ".) In all cases messages may have null content, meaning that no guidance of that particular type is provided at the given time and location. Of course, allowing this degree of flexibility in the definition of guidance message content means that elsewhere within the framework there must be a correspondingly general capability to interpret the messages and to predict their impacts on traffic patterns; this will be seen in the discussion of the routing map below.

3.2.1.3 Framework maps

Framework maps express the direct causal relationships between pairs of the framework variables defined above.

Network loading map The network loading map $S : P \mapsto C$ determines the dynamic network conditions C that result from the movement of the exogenous time-dependent OD demands $\mathcal{X}$ over the network in accordance with the path splits P. This is the most standard part of the RGG framework, and corresponds fairly closely to the dynamic network loading component of full information dynamic models. However, as discussed above, most conventional dynamic network loading models assume that path choices are determined at the origin, and do not allow path switches at en route locations. Furthermore, the conventional dynamic network loading map is usually considered to

output link volume and/or traversal time trajectories, whereas the network loading map considered here allows the output of arbitrary network condition variable trajectories.

A network loading model must be capable of determining any network condition measures that have been defined as framework variables. In simulation models, link traversal times are computed by tracking the progress of individual trips through the network. The simulator notes a trip's time of entry to and exit from each link on its path. The difference between the two values is the trip's link traversal time, which is associated with its time of link entry. After all trips that entered a link at a particular time have traversed it and exited, the average (or otherwise filtered) traversal time value of the trips is computed. Computation of other link-to-path additive condition variables can be done in a similar way, i.e., by comparing the cumulative values of simulation-determined condition measures at the beginning and end of a link. If no trip entered a particular link in a particular time step, the corresponding link condition values have to be determined in some other way.

Guidance map The guidance map $G : C \mapsto M$ represents the response of the traffic information system, in the form of disseminated guidance messages, to a given set of network conditions predicted over the analysis time grid $\mathcal{T}$. Although the network condition predictions output by the network loading map are complete (i.e., they cover all network locations and all times), there is no requirement that the generated guidance messages convey guidance information or recommendations having a comparable level of detail or precision. A set of detailed network condition predictions from the network loading map might be summarized, for example, in a message such as "expect congestion ahead for next 30 minutes"; a complete time-dependent minimum path calculation might be summarized as "turn left at next traffic signal" with further recommendations provided at subsequent decision points.

Of course, a guidance provider will be pursuing some objective in generating messages from network conditions. Traditional objectives such as individual- or system-level time minimization are examples, but the range of possible objectives, including pursuit of traffic management and environmental goals, is clearly very wide. No assumption will be made about the specific objective underlying the guidance messages; it will simply be assumed that the chosen objective is accurately reflected in the specific form of the guidance map. Section 3.2.4.5 provides further discussion of the choice and representation of guidance objectives.

Note that the messages output by the guidance map for a given set of input network condition predictions are not necessarily consistent, since driver reaction to these messages may produce a change in the predicted network conditions. As a modeling construct, the guidance map is unique to the RGG problem, and has no general analog in the conventional full information problem.

However, a particular instance of the guidance map is the "identity" map $\mathcal{I}$, which copies without change the entire set of network condition predictions to decision points at all nodes and in all time steps. This can be considered to provide perfect descriptive guidance, and so could be thought of as implicitly representing guidance in conventional full information models; Section 3.2.3 discusses this further.

More general guidance maps can account in a relatively straightforward way for processing and communication delays, limited-range guidance dissemination technologies, and other technological or operational limitations of realistic guidance systems; they do this by outputting appropriate location- and time-specific messages.

For example, a VMS display could be modeled as providing messages having the universal type, located on the particular link where the VMS is installed, disseminated during some given stretch of time, and with a content typically drawn from a "dictionary" of possible messages (i.e., a finite set; see for example Bolelli and Rutley [1991]). A guidance map from the domain of network conditions to this range of messages would then represent a particular VMS response plan to different possible network conditions.

As another example, information about forecast conditions on selected network links transmitted over a private medium-range digital radio system might be represented by messages of a particular (non-universal) type, available at the time of broadcast on all links in the range of the transmitter. The message content would be derived from network conditions but might not be an exact copy because of transmission inaccuracies or because of an intentional attempt to reduce the quantity of transmitted data due to bandwidth constraints.

Routing map The routing map $D : M \mapsto P$ predicts the path splits that result from drivers' reactions to a given set of guidance messages. This map is a generalization of the route choice component of demand models used in conventional full information traffic assignment models. In full information models, the demand model relates path splits at trip origins (or, equivalently, OD path flows) to path attributes, which in turn are directly derived from link conditions. In guidance models, on the other hand, path splits at decision points (trip origins or otherwise) are related to the guidance messages available there, and these are indirectly derived from network conditions via the guidance map.

As indicated above, path splits may be applied in an aggregate or a disaggregate fashion to obtain deterministic or multinomially distributed path flows, respectively. The routing map itself may be based on some form of aggregate model (such as the diversion curves sometimes applied in highway bypass studies), or on a disaggregate path choice model. Applying disaggregate path choice models to predict aggregate path splits is a useful approach because it builds on the considerable effort that has been devoted to the development of theoretically sound disaggregate behavioral models over the past several decades.

It will generally be assumed here that the routing map is one-to-one: a given message trajectory results in a unique path split trajectory. This assumption is true for the (large) family of path choice models based on random utility theory, and it simplifies some of the later developments. Adoption of traditional deterministic user optimality assumptions (e.g., strict minimum time path choice) would lead to a one-to-many routing correspondence, which is less convenient to work with. This issue is discussed further in subsection 3.2.4.6.

In terms of required path choice model characteristics, the major differences between full information and route guidance problems are that the latter must:

- explicitly predict drivers' responses to each type of guidance message that can be represented by the message variables. Predicting driver response to prescriptive route recommendations and to descriptive information about traffic conditions, for example, obviously calls for different model forms. A modeling effort needs jointly to develop both the set of guidance messages as well as the routing map that interprets them. As a particular case, the map must be able to predict path splits at a decision point that has no guidance available at a particular time (the case of null guidance messages mentioned above).

- distinguish between pre-trip and en route decisions. A pre-trip path choice is a commitment without immediate antecedent, whereas an en route path switch may involve overcoming a reluctance to abandon a route that was selected earlier in the trip. For this reason, the en route switching decision is likely to exhibit hysteresis. This effect has been modeled as a form of bounded rationality in Mahmassani and Jayakrishnan [1991], and also by using switching penalties (see subsection 5.2.3 for an example of this).

The emphasis here is on the effect of guidance messages on path splits. In reality, of course, drivers may base their route choice decisions on a wide variety of other factors including their prior knowledge of traffic conditions, their prior experience with the guidance system (if any), and different informal and formal information sources. It is not intended here to neglect these other influences; rather, it is assumed that they have been fully subsumed in the routing map, leaving a direct functional dependence of path splits on the guidance messages only. As a particular matter, the effects of prior days' experiences on the current day's behavior are assumed to be adequately captured in the routing map. The analysis presented here is of within-day driver behavior. A more general analysis would be based on a model of decision making that integrates the effects of prior days' experience, current experience up to the time of decision, and guidance information. This is a topic for future research.

3.2.2 Composite map formulations of guidance consistency

The above considerations lead naturally to the definition of composite maps that combine the dynamic network loading, routing and guidance maps in different sequences. Each composite map transforms an input element (i.e., time trajectory) of one framework variable into an output element of the same variable. In fact there are three such composite maps (the symbol $\circ$ below denotes functional composition, which is read from right to left):

- a composite map $D \circ G \circ S : P \mapsto P$ from the domain of path splits into itself, which starts with a set of time-dependent path splits, forecasts the corresponding network conditions, determines an appropriate set of guidance messages, which are disseminated to drivers and cause them to react in some way, leading to a new set of path splits.

- a composite map $S \circ D \circ G : C \mapsto C$ from the domain of network conditions into itself. This map begins with a set of time-varying network conditions and determines the messages with which the guidance system responds; these are communicated to drivers and affect the path splits, thus resulting in a new set of network conditions.

- a composite map $G \circ S \circ D : M \mapsto M$ from the domain of guidance messages into itself. Here the map begins with a set of time-dependent messages, predicts the resulting path splits, forecasts the network conditions which ensue from these, then determines a new set of guidance messages.

Any of these maps will be referred to as the composite framework map corresponding to the involved input and output framework variable.

Unless otherwise noted, all component maps will be assumed to be one-to-one, so that the composite framework maps are one-to-one as well. As noted above, this assumption excludes deterministic minimum path routing, which leads to a one-to-many routing correspondence. The

complications of one-to-many maps would merely distract us from the issues of primary concern here, so it is preferable to circumvent them by focusing on one-to-one maps.

In operational terms, evaluating one of these composite framework maps corresponds to executing a one-pass forecasting model that invokes the component maps in the indicated order of composition. Input to the model is an assumption about the time trajectory of one of the framework variables (splits, conditions or messages); its output is a prediction of a possibly different trajectory of the same variable. This can be written as:

$$\text{model(assumptions)} = \text{predictions}$$

Recall that the guidance generated by a model is said to be consistent when the assumptions used as the basis for generating it prove to be verified, within the logic of the predictive model, after drivers receive the guidance and react to it. In terms of the composite framework maps, consistency means that a map's predicted time-dependent outputs coincide with its assumed time-dependent inputs. Again, this can be written as:

$$\text{model(assumptions)} = \text{predictions} = \text{assumptions}$$

For the composite path split map, guidance is consistent if the forecast path splits coincide with the splits that were assumed at the start. For the composite network condition map, guidance is consistent if the initial network conditions used for the guidance determination coincide with those that result after the guidance is disseminated. For the composite message map, guidance is consistent if the resulting messages coincide with the initially-assumed set of messages.

The coincidence of the assumed input value (i.e., time trajectory of a framework variable) with the corresponding predicted output value of a composite framework map can be expressed by saying that the value is a fixed point of the map. (If $T : X \mapsto X$ is a one-to-one map, $x^* \in X$ is a *fixed point* of T iff $x^* = T(x^*)$.) Based on the above, it can be seen that consistency in the context of anticipatory route guidance can be expressed and studied in terms of fixed points of the composite framework maps identified above. Solving a RGG problem to obtain consistent guidance can be accomplished by finding a fixed point of the composite map that is chosen to represent the problem. The variables of ultimate interest—guidance messages—are the direct output of the composite message map only; in the other formulations, guidance messages are obtained as an intermediate result of the evaluation of the composite map at a fixed point.

Note that the three composite maps are equivalent with respect to the existence of a fixed point: if one has a fixed point then they all do, and if one does not then none does. Consider, for example, the composite message map and let m' be a fixed point of it: $m' = G \circ S \circ D(m')$. Then $p' = D(m') = D \circ G \circ S \circ D(m') = D \circ G \circ S(p')$, so p' is a fixed point of the composite path flow map. The argument can be permuted for the other composite maps. Furthermore, an argument by contradiction establishes the contrapositive. Hence, the property of guidance consistency depends on the RGG problem itself (which is completely specified by the specification of its component maps), and not on the particular composite formulation chosen to represent it.

Of course, the properties of the fixed point problem depend on the composite map and on its domain of definition. Indeed, there is no guarantee that such fixed points exist. Standard theorems such as those of Brouwer or Kakutani can be used to prove the existence of a fixed point when the composite maps satisfy specific properties (e.g., when they are continuous or upper semi-continuous maps on convex, compact sets). Stronger properties are required to assert uniqueness. It is not difficult to think of realistic guidance system designs that violate these properties, however: for example, any guidance system with a discrete message set has a discontinuous guidance map.

Continuity properties are sufficient but not necessary to establish fixed point existence; a number of research efforts have examined the existence of fixed points and equilibria in models with discontinuous maps. These include Dasgupta and Maskin [1986a,b] for Nash equilibria; and Bernstein and Smith [1994] and de Luca [1995] for traffic equilibria with discontinuous cost functions. A very general theorem due to Tarski [1955] asserts the existence of fixed points of a monotone (but not necessarily continuous) function defined on a lattice. However, these results have not yet been applied to the route guidance generation problem with its discontinuities.

It would be desirable to be able to analytically quantify the consequences of non-attainment of a fixed point, either because a fixed point does not exist, or because it is not accurately computed by a particular solution method. Such consequences might include flip-flopping of traffic between alternate paths, and cycling of predicted network conditions and guidance messages in successive iterations. With the ability to quantify such effects, it might then also be possible to evaluate them and to decide on an objective basis if a feasible but inconsistent solution were nonetheless acceptable as a basis for guidance generation.

Unfortunately, the state of the art does not yet allow this kind of perturbation or sensitivity analysis in dynamic traffic models of the composite form considered here. For this reason, the approach adopted in this research is essentially pragmatic: attempt to find fixed points of the composite maps *as if* the fixed points were assured to exist, but be alert to the possibility of non-existence and its consequences in terms of inappropriate guidance and resultant traffic instabilities. Difficulties encountered while working with guidance generation in this way will identify particular areas requiring deeper and more rigorous mathematical investigation in subsequent research.

3.2.3 Relationship to full information models

Section 3.1 contrasted the assumptions of conventional full information models with the requirements of realistic guidance modeling. The relationships between the two approaches can now be articulated more clearly. The most characteristic differences concern the assumptions embodied in each model as reflected in its structure, and the interpretation of the fixed point conditions. Each of these is discussed below.

3.2.3.1 Model assumptions and structure

A conventional full information model assumes that the path choice decision is made upon departure from the origin and is not reconsidered en route. Drivers are assumed to have accurate perceptions of the attributes of alternative paths and to choose a path that maximizes their perceived utility.[2] In terms of the basic maps introduced here, the guidance map (from time-dependent network conditions to guidance messages) for the full information model is simply the guidance identity $\mathcal{I}$. The routing model applies a standard path choice model based on "messages" that convey the exact network conditions. In this situation the composite network condition map $S \circ D \circ (\mathcal{I}) : C \mapsto C$ and the composite message map $(\mathcal{I}) \circ S \circ D : M \mapsto M$ become equivalent. Only two maps remain: the composite path split map $D \circ (\mathcal{I}) \circ S : P \mapsto P$ and (for example) the composite network condition map $S \circ D \circ (\mathcal{I}) : C \mapsto C$. (The composite message map could equally well have been chosen as the second map but the composite condition map was preferred because it can be discussed using

[2]This choice may nonetheless seem random to an observer who is not fully informed about the driver's decision situation.

standard terminology.) In a situation with fixed OD demand trajectories and pre-trip path choice only, path flows and path splits are fully equivalent. Consistency, in the context of a conventional full information model, simply means that the model's routing principles hold at each time step of the computed time-dependent solution. This is what was called in Chapter 2 a dynamic supply-demand network equilibrium. Hence it can be seen here that these correspond naturally to fixed points of the (simpler) composite maps.

It is also true that the RGG problem can be formally recast as a full information problem through suitable redefinition of the supply (network loading) or demand (routing) maps. For example, if network loading is considered to result in guidance messages rather than network conditions, then a map of the form $D \circ (G \circ S) : P \mapsto P$ is obtained. This has the structure of the full information problem D∘S formulation, with (G∘S) playing the role of a modified "supply" map. Similarly, if tripmakers are considered to generate their own messages from network conditions and to react to these, then a map of the form $S \circ (D \circ G) : C \mapsto C$ is obtained, having the structure of the $S \circ D$ full information problem, with $(D \circ G)$ as the modified "demand" map.

These redefinitions are, however, formal manipulations only. They blur important distinctions between the network, tripmakers and guidance information provider, and they do not provide useful insight into the RGG problem. Furthermore, standard solution algorithms for the full information problem are unlikely to be effective for the redefined problems because their modified "supply" and "demand" maps ($G \circ S$ and $D \circ G$, respectively) will generally have properties very different from those of true S and D maps typically used in full information models.

To summarize the above discussion, the conventional full information problem can be viewed as a special case of the RGG problem, with the general guidance map replaced by the guidance identity. Moreover, if guidance is not available everywhere, if it is not accurate or up to date—in other words, if the guidance map is not an identity—then the RGG problem is a true generalization of the full information problem and the flow patterns and network conditions that solve the two problems will not be the same (this was the point of the example provided at the end of Chapter 2). In this case, route guidance messages derived from conventional full information model outputs would not be correct.

3.2.3.2 Interpretation of fixed point conditions

The conventional interpretation of the solution to a full information model, particularly when the model embodies one of the standard user-optimal path choice assumptions, is that it represents some sort of dynamic equilibrium condition established between the travel choices of network users and the network's response to these choices. The fixed point formulation of supply-demand equilibrium is a mathematical translation of this consistency between, on the one hand, user behavior based on the anticipated network performance and, on the other hand, the actual network response to that behavior.

A plausibility argument often advanced to justify the equilibrium assumption is that users have learned over time to correctly anticipate network conditions, have experienced and assessed the consequences of the different choices they might make, and have eventually settled upon a choice (or set of choices) that gives them no incentive to change their behavior. As Cascetta [1989] and Cantarella and Cascetta [1995] have shown, however, the outcome of a day-to-day learning process depends heavily on the way in which prior days' experiences and expectations are filtered into the current day's decisions. A fixed point equilibrium situation results from particular filter forms, but

other limiting behaviors of the dynamic process (such as periodically repeating or chaotic travel patterns from day to day) are also possible.

In any case, such an interpretation is difficult to maintain in a route guidance context. Incidents, for example, occur randomly and relatively infrequently. Consequently, users may not have sufficient opportunity to learn about and optimize their response to specific incident situations. Instead, they may have a reflex, knee-jerk reaction when they become aware, through route guidance or otherwise, of an incident on their route. Further, the exact circumstances of the incident situation might never be repeated. It is difficult to speak of equilibrium in such a context. (This is essentially the argument advanced by Engelson [1997] against the rational expectations approach to guidance generation proposed by Kobayashi [1994]. See Section 3.4.5 below for further discussion of this point.) How, then, should the guidance consistency fixed point condition be interpreted?

Based on the discussion earlier in this chapter, it is clear that the key difference between the RGG problem and conventional dynamic network problems is the presence of a guidance provider that generates and disseminates messages. Whereas in the conventional model equilibrium is attained when there is no incentive for users to behave in a way that changes the path splits, in the RGG problem consistency is attained when the guidance provider has no reason to modify the guidance messages. This condition does not depend on the nature of users' reactions to the messages (knee-jerk or other), but does assume that these reactions can be modeled. The implicit learning process that ultimately results in dynamic equilibrium under full information model assumptions is replaced, in the route guidance generation problem, by an iterative algorithm that explicitly computes the consistency fixed point.

3.2.4 Discussion of framework and alternative approaches

This section provides some general comments on the overall framework presented above, and on alternative approaches that were considered but not adopted.

3.2.4.1 Generality of the overall framework

It is convenient in discussions to use link traversal times as an example of a network condition variable, and information about these times as an example of guidance messages. However, these examples should not obscure the greater generality that the framework allows in the choice of network condition variables and messages, and so also in the variety of guidance systems that can be represented and analyzed.

The choice of condition variables is arbitrary. What is important is that, given path splits, the network loading map must be able to compute whatever particular condition variables have been chosen. Similarly, the format and content of guidance messages are arbitrary. Again, what is important is that the guidance map must be able to generate the appropriate messages for any possible values of the chosen condition variable. Finally, the routing map must be able to predict the path splits that result from each possible message that the guidance map may produce.

To illustrate this point, imagine an anticipatory route guidance system based entirely on a binary variable indicating whether or not an incident situation is present. The condition variable is simply the time trajectory of these 0-1 values for each link. This variable is not continuous, it is not link-to-path additive, it does not verify any kind of symmetry property, and it would probably be considered a very inconvenient variable to work with in most network model systems. In the

RGG framework, the network loading map has the task of establishing that an incident is present on a link (perhaps via direct operator input) and predicting the duration of incident conditions, based in part on input path splits and in part on exogenous data on incident parameters and OD demands. The guidance map would have to generate messages derived from the incident variable; again, these messages could be arbitrary. However, the routing map would have to be able to predict path splits based on these messages, and these splits would then be input to the network loading map. Even with this extremely basic notion of the network condition variable and the corresponding component maps, the composite framework maps remain well-defined and the notion of guidance consistency as a composite mapping fixed point retains its meaning. Similar examples can easily be provided using alternative condition variables such as queue lengths.

3.2.4.2 Paths and subpaths

The framework supposes that all feasible paths are explicitly enumerated in a pre-processing step. Furthermore, an origin-destination path is determined via a sequence of choices made at decision nodes; each choice concerns the subpath to follow from that decision node to the destination, and may be revised at decision nodes further downstream. If en route path switches are not allowed, the definitive OD path choice is made at the origin, as in a conventional full information model.

In the past, transportation algorithms involving explicit path enumeration were generally ruled out *a priori* on the grounds that the very large number of paths in a network would lead to excessive computer storage requirements. More recently, however, increases in the amount of memory routinely available on computers, together with improvements in the data structures used to store and access path definitions, have made such approaches more attractive. Computer time/space tradeoffs that were not available earlier have now become viable software design options. Although explicit path enumeration cannot yet be considered a routine procedure for medium- to large scale networks, it is nonetheless feasible enough to be a workable part of the framework. A number of papers [Leurent, 1995; Cascetta et al., 1997] have proposed algorithms for static problems that exploit explicit path availability. As an indication of its growing acceptance, several state-of-the-art dynamic traffic network modeling software systems (DYNASMART, MITSIM, DynaMIT) include some form of explicit path enumeration.

One of the advantages offered by explicit path enumeration is that the identity and attributes of each element of the path choice set are available for use by a path choice model. This permits the application of complex choice models that would not otherwise be available. Another advantage is that the shortest path calculation, a bottleneck in many network applications, can become in some cases a trivial and low-cost operation[3].

Discussions of route guidance systems sometimes distinguish the *guidance network* from the full traffic network. For political or other reasons, it may not be acceptable to direct traffic, via route recommendations, towards certain paths in the network (for example, paths that use local neighborhood streets). The guidance network is defined by the set of paths that are considered to be acceptable for route recommendations. On the other hand, drivers are free to choose any path they like through the full network. If the guidance network is a seriously restricted subset of the network, it is unlikely that a traffic network model including only this subnetwork would be

[3]However, this may not be true in all cases. Indeed, brute force calculations of time-dependent travel times on enumerated paths may require *more* computational effort than efficient dynamic shortest path algorithms such as the DOT algorithm of Chabini [1997].

sufficiently accurate for guidance purposes. Therefore, the set of feasible paths, as defined here, will generally be a superset of the acceptable guidance paths. Any restrictions on the paths acceptable for route recommendations would be enforced in the guidance map.

In guidance applications, the path enumeration process must take account of the possible blockage of one or more links in the network by incidents. If all the enumerated paths to a destination pass through blocked links, for example, the guidance model would be useless. On this basis, there would be a tendency to include certain paths which might not be attractive under normal traffic conditions, thus expanding the size of the feasible path set. An alternative approach might be to redefine the feasible path set whenever an incident occurs, using the modified paths for traffic modeling throughout the duration of the incident, and reverting to the original path set when the incident has cleared. In yet another approach, DYNASMART [Mahmassani et al., 1994] uses a set of k-shortest paths that is periodically recomputed during a simulation.

Path generation in practical work will require careful balancing of the need to accurately represent the path choice logic with the storage and processing implications of a large path set.

3.2.4.3 Flow variables

As explained above, path splits are the time-dependent fractions of trips leaving a decision point (origin node or en route switching location) and going towards a destination via the various subpaths exiting from the decision point. Through the dynamic network loading map, path splits ultimately determine dynamic path, subpath and link flows.

In conventional full information models, with fixed OD demand trajectories and pre-trip path choice, the framework's flow variable can be defined in terms either of origin-based path flows or path splits. These variables enjoy a number of convenient properties. Subject only to simple bounds constraints, feasible values of either variable can be arbitrarily specified. The feasible sets of both variables are convex (they are simplices), so convex combinations of feasible values result in another feasible value. The two variables are related to each other by an obvious linear transformation and, in fact, are essentially equivalent.

This equivalence between path splits and path flows does not hold in dynamic models that allow path switches at en route decision points. In such networks, dynamic subpath flows at en route decision points are determined endogenously by the network loading map. They depend on flows and network dynamics at upstream locations. Time trajectories of subpath flows cannot be arbitrarily specified without risking violations of flow feasibility. Since network dynamics are nonlinear, the set of feasible subpath flows is non-convex:[4] convex combinations of feasible subpath flow trajectories do not necessarily result in feasible flows. This severely restricts the applicable fixed point solution algorithms. On the other hand, path splits at en route decision points form a unit simplex, and so are convenient to work with. Since the splits are independent of the magnitude of flows through the decision points, the splits can be arbitrarily specified, subject only to the simplex constraints. For this reason path splits were retained as the framework flow variable.

A number of researchers have found flow splitting-type variables to be an analytically tractable way to represent network flows. The METANET macroscopic traffic simulator [Messmer and

[4]Note that this conclusion differs from that reached in Carey [1992]. There, it is shown that a FIFO assumption on link flows introduces a non-linearity that leads to a non-convex set of feasible link flows. Here, the argument involves subpath flows when en route path switching is allowed, and the non-linearity of the dynamic network loading map results in non-convexity; this conclusion does not depend on FIFO assumptions.

Papageorgiou, 1990], for example, is based on a state space model that expresses flow propagation via link splitting rates [Papageorgiou, 1990] at all nodes. Powell et al. [1995] analyze general dynamic networks and argue that a splitting rate flow formulation avoids performance problems that are intrinsic to a flow based formulation of dynamic network problems. Wisten and Smith [1997] also prefer a splitting rate formulation of a dynamic traffic assignment problem. These authors do not consider situations of en route path switching or confront the non-convexities that it introduces, which add yet another reason to adopt a flow splitting approach.

Link splitting rates are not precisely the same as path splits: splitting rates are the fractions of the flow exiting node i at time t for destination d via each of i's outgoing *links* (i,j), whereas path splits are the the fractions of the flow exiting decision node i at time t for destination d via each of i's outgoing *paths*. Clearly, link splitting rates can be computed from path splits, but not conversely. Path split variables inherit the analytical and practical advantages of splitting rates. Furthermore, most models of driver path choice behavior assume that drivers assess the properties of and decide between entire paths or subpaths and, in this case, path splits are a direct statement of the path choice probabilities.

Splitting rates, on the other hand, represent a more local decision between outgoing links at a particular node, which is independent of the links chosen at prior nodes on the trip. Thus, the probability of choosing a path is determined *a posteriori* as the product of the link splitting rates at its successive nodes. This approach resembles the static assignment model of Maher [1992], which is based on a similar assumption of independent probabilistic link choices at nodes. The Markovian nature of this process can lead in some cases to unrealistic path choice probabilities. Maher and Hughes [1997, pp. 344–345] provide an example contrasting the path choice probabilities that result from path-based *vs.* link-by-link choice.

Trip movements on a network could also have been defined using link flow variables. The reason for not using these variables here is that they are arguably less fundamental to the full information and RGG problem than path splits or flows: they can be derived from path flows (this is done during evaluation of the dynamic network loading map), but no inverse relationship allows path splits or flows to be determined from link flows.

A further reason for not including link flow variables is that, in dynamic network models, the precise definition of a link flow is highly model-specific. To determine network conditions from link flows, different performance models might require information about the total number of trips on a link, or the number of trips in queuing and moving segments, or the trips' positions on the link, or something else again. The definition of a path split or flow, on the other hand, is unambiguous and applicable to all models. By not using link flows as a framework variable, problem-specific details of the link flow representation and performance model are subsumed in the network loading map.

3.2.4.4 Condition variables

Although the framework encompasses general definitions of network conditions, it was agreed above to use link traversal times as an example of network condition variables because some specific illustrative variable was needed, and the link time variable is a common choice in transportation network modeling. Furthermore, it is a variable on which many other frequently-used network condition variables are based, and is generally needed to establish traffic dynamics.

The argument for preferring link to path impedance variables in the framework specification is similar to that for preferring path splits to link flows: for condition variables that are additive from

link to path level, path conditions can be derived from link conditions but not inversely. Thus, link conditions appear as a more fundamental representation of overall network state than path conditions.

3.2.4.5 Message variables

In discussions of route guidance, it is sometimes felt useful to distinguish between prescriptive guidance (route recommendations) and descriptive information (data about traffic conditions). In practice, this may be a needlessly fine distinction. The messages sent by a system to drivers may well contain both prescriptive and descriptive elements ("30 minute delays ahead, take alternate route XYZ"). Furthermore, the descriptive and/or prescriptive content of the messages may be arbitrarily imprecise ("delays ahead, take alternate route"). It is for this reason that the words "guidance" and "information" are used interchangeably in this dissertation, and that the generic concept of "messages" having arbitrary content and format is preferred. As explained above, the logic of the framework is sufficiently flexible to handle this generality.

Similar reasoning applies to the often-made distinction between user- and system-optimal guidance as an objective pursued by the guidance provider when generating messages. Again, this distinction may be excessively fine in practice. Guidance providers will generally pursue some single objective when generating guidance. Drivers generally will consider a much wider range of link or path attributes in making their path choices. Furthermore, their opinion about the guidance system itself will influence their reaction to the guidance. Drivers' resulting path choice decisions involve the integration of messages from the guidance system with these various other factors. While the resulting decision protocol could perhaps be deemed user-optimal in a tautological sense ("drivers choose the path that seems best to them, all things considered"), such an interpretation is far removed from the conventional meaning of user optimality.

Thus, in this sense, it is somewhat misleading to speak of user- or system-optimal guidance; it would be more precise to refer to predicted time information, or minimum marginal cost path recommendations, or the like. In practice, a guidance provider may well pursue more complex objectives when providing guidance: objectives involving environmental quality standards or traffic management strategies, for example.

As long as the path split response to guidance messages can be predicted, the particular objective that was the basis for generating the messages is not relevant to the modeling framework.

Of course, there would be little benefit in developing a guidance system to generate messages that most users ignore, even if the extent and consequences of this reaction could be accurately predicted by the routing model. From this point of view, it would be best for a system to generate guidance using objectives that correspond (at least in part) to criteria important to most drivers, and to be clear and honest about the basis for the generated guidance.

3.2.4.6 Routing map

The routing maps considered in this dissertation are one-to-one relationships from the message domain to the path split range. Routing models based on underlying random utility path choice models generate one-to-one relationships but those based on conventional deterministic user optimality assumptions (single shortest path routing, for example) do not. In fact the latter assumption results in a routing correspondence (one-to-many map) rather than a proper function since, in the

event that multiple paths have minimum utility, any such path might be (but need not be) chosen with positive probability.

A number of researchers have investigated properties of deterministic user optimal (DUO) traffic patterns using fixed point approaches. These include Braess and Koch [1979], Daganzo [1983] and Cantarella [1997] for static network models, and Kaufman et al. [1998] for dynamic network models. Although, as these research efforts show, fixed point methods for correspondences can be effectively applied to analyze and solve DUO network problems, the use of these methods is considerably more cumbersome than function-based methods. Reasons for these difficulties can be variously traced to the general awkwardness of dealing with multiple-valued maps; to the non-differentiability of the satisfaction function (the expected maximum utility of a choice set as a function of its elements' attribute values) [Cantarella, 1997]; or to the need to introduce complementary slackness conditions in the routing map [Kaufman et al., 1998].

Furthermore, as driver behavior modeling becomes increasingly sophisticated, deterministic user optimality assumptions appear increasingly simplistic and inappropriate for use in traffic and guidance applications.

For these reasons, it was decided in this research to avoid the use of DUO assumptions and of routing models based on them.

3.2.4.7 Conclusions

The preceding paragraphs have argued that path split, network condition and message variables, together with the causal relationships between them, constitute a basic and logically complete set of components that define a framework for route guidance generation.

Other variables, such as path impedances or link flows, could be derived from these. Similarly, additional or modified mappings could be defined to utilize the derived variables. For example, a modified network loading map might output path impedances, and a modified guidance map might directly utilize these impedances to generate messages.

Many of the research efforts reviewed in section 3.4 below utilize variables and/or mappings that are derived from the framework variables and mappings proposed here. This illustrates some of the variety possible in guidance problem formulations and approaches.

It is possible that for specific modeling assumptions, a derived formulation may allow a simpler specification and analysis of the guidance generation problem. Similarly, it is conceivable (although by no means evident) that one of the many possible derived formulations may provide computational advantages compared to the basic formulations. In some cases, some basic properties of the derived formulations can be inferred from abstract considerations (for example, path times can be expected to exhibit less relative variability than a series of independent link times), but ultimately the practical worth of such alternative formulations will have to be explored through computational testing. These and related questions are not investigated further here, and are identified as topics for future research. The research described here focuses exclusively on formulations that use the framework variables proposed above.

3.3 Framework for the RGG problem: stochastic case

3.3.1 Nature of model stochasticity

Up to this point in the discussion, the various component maps have implicitly been considered to be deterministic. Traffic phenomena, however, seem intrinsically random because they are the emergent outcome of individual drivers' reactions to stimuli that, in general, cannot be known to others. These reactions may influence both the driver's (pre-trip and en route) choice of path, as well as the progression along the chosen path towards a destination. Random external factors (weather, extent of travel friction created by other road users, etc.) may also influence drivers' reactions.[5] In these conditions, it follows that the particular flow patterns and network conditions that prevail at a given time can be considered, not as deterministic quantities, but rather as realizations of underlying random variables.

For this reason, it may be appropriate for modeling purposes to treat both the network loading map and the routing map as stochastic. (The guidance map may conceivably be stochastic as well, although the possibility will not be considered further in this dissertation.) Stochasticity in the network loading map can result from random aspects in either the trip propagation relationships or the link performance relationships (or both). Stochasticity in the routing map typically results when this map is obtained by applying an underlying probabilistic path choice model independently to individual members of a population of trips. Stochasticity may also be introduced to facilitate the simulation processing, for example via an intentionally randomized order of processing links or vehicles.[6]

If any of the component maps is stochastic, the various composite framework maps will be stochastic as well. Different evaluations of a framework map with identical input variable trajectories may result in different numbers of trips being associated with the various available paths (from the routing stochasticity) and/or progressing from link to link at different rates (from the network loading stochasticity.) In other words, for a given framework variable input trajectory, the output of the framework map is a stochastic process—a family of random variables, defined over the framework variable's values, and indexed by time. Consequently, even for identical input trajectories, the predicted dynamic path splits and network conditions will generally vary, systematically and perhaps significantly, from one evaluation of the framework map to the next.

3.3.2 Approaches to guidance generation in the presence of stochasticity

From the point of view of guidance generation, stochasticity in the network condition predictions is the principal cause of concern (although this may result, of course, from stochasticity in the path splits). This stochasticity has two major effects on guidance generation:

- the rate of progression of traffic through the network is stochastic. Consequently, a vehicle may not arrive at the next en route decision point at the time that the model expects based

[5] However, precise data that focus on stochastic aspects of driver behavior and traffic propagation are scarce.

[6] Note that, because of the focus of this thesis on route guidance issues, variability in the magnitude and time trajectory of OD flows is not included here among the factors contributing to model stochasticity, although in practice the statistical errors in dynamic OD matrix estimates could be a major source of stochasticity. However, as will be seen in Chapter 5, the other factors mentioned here can by themselves result in highly stochastic model outputs.

on the current realization of link times. This means that the vehicle may receive different guidance messages than expected;

- conditions downstream from the vehicle may not be what is predicted by the current realization of the condition variables, so that guidance messages based on these predictions may be inappropriate for the actual conditions.

Uncertainty regarding the actual arrival time of vehicles at a decision point is not a major issue, since guidance messages will generally be computed and made available for all time steps. Thus, whether the vehicle arrives at the time predicted from the current link time realization or at a different time, a guidance message (possibly null) will nonetheless be received. The message will be appropriate for the actual time of arrival; however, the vehicle's subsequent decisions and experience may be different from what the model expected during the guidance computation, so that the quality of the model's predictions will be adversely affected.

The issue of uncertain downstream conditions is more serious, since it affects the nature of the guidance that should be provided. Dynamic programming addresses the determination of optimal actions in situations of decision making over time. When future states are stochastic, it is known from dynamic programming that optimality is not generally attained by a predetermined fixed sequence of actions, but rather by an adaptive strategy (also called a decision rule)—a complete contingency plan specifying what action to take in each decision situation, depending on the actual system state (assumed known) at the time of the decision.

In a dynamic network context, for example, Hall [1986] showed that in dynamic networks with stochastic link traversal times, the minimum expected travel time between two points is attained, not by following a fixed path, but rather by applying a set of decision rules that specify the next link to follow out of each node based on the actual time of arrival at the node. (More generally, if guidance was not available at each node, the decision rule would specify a subpath to follow from each decision point to the next, possibly many links away.) Subsequent papers by Fu and Rilett [1998], Miller-Hooks and Mahmassani [2000] and Chabini [2000] have led to deeper understanding of the problem as well as to computationally efficient algorithms for determining minimum expected time decision rules, given knowledge of the link traversal time stochastic processes (for example, the mass functions giving the probabilities of different discrete link traversal time values in different simulation time steps). This suggests that, rather than generating a fixed sequence of messages to disseminate at each decision point over the guidance computation interval, it would be advantageous to compute adaptive message generation rules that determine the particular message to disseminate at a decision point based on prevailing conditions (the time of arrival at the node as well as any other conditions that can be determined.)

To the user, time-dependent prescriptive messages containing next subpath recommendations would appear identical regardless of whether they were obtained from an adaptive decision rule calculation for a stochastic network or from a conventional shortest path calculation for a deterministic network. However, depending on the technology operating in beacons and vehicles, the internal form of the disseminated guidance messages could be different. Adaptive systems relying on substantial in-vehicle computing resources might transmit to the vehicle a complete decision table, to be accessed according to the time at which intermediate decision points are reached; non-adaptive systems might transmit a fixed sequence of subpaths. In any case, the computational details involved in generating messages would be quite different in the two situations. To calculate

adaptive decision rules, for example, the (stochastic) dynamic network loader would have to be applied repeatedly in each iteration to the iteration's input data to generate a sample distribution of output link traversal time values in each time step.

Use of local controllers is another way of confronting the variability produced by stochastic relationships. Imagine, for example, that guidance beacons, in addition to receiving information from the guidance center and disseminating it to vehicles, were also able to measure and immediately respond to local traffic conditions. The guidance center would compute message generation rules that depended only on the prevailing conditions at each decision point. A decision point's beacon would match measured conditions against the relevant decision rules to determine the particular message to disseminate. While this approach is theoretically suboptimal compared to one using decision rules based on full state information, in practice it could adapt to changing traffic situations more quickly than a method requiring complete recomputation of all guidance, and more effectively than a control law that does not incorporate global information. A more sophisticated system would allow a guidance beacon to exchange information with other nearby beacons, so that it could incorporate additional state measurements in the decision rules it applies. Further elaboration of these ideas is beyond the scope of this thesis and is a topic for future research.

This dissertation will therefore concentrate on guidance systems that produce fixed message trajectories rather than adaptive message generation rules. On what basis should these messages be produced? There are at least three possibilities:

- use the most recent realization of network conditions. Guidance messages would be derived solely from the most recent realization of the network condition stochastic process. Descriptive messages might simply report the predicted conditions without change, for example, and prescriptive messages might recommend minimum paths computed from these conditions.

- use the expected value of network conditions or, more precisely, the mean value trajectory computed from the network condition stochastic process. This would require the generation of a number of realizations of the network condition stochastic process, using fixed path split inputs, in order to estimate the corresponding mean conditions. Unfortunately, without some guarantee regarding the properties of the distribution of condition values in each time step, use of expected values could lead to meaningless results. Suppose, for example, that half the realizations predicted free-flow conditions on a particular link, while half predicted severe congestion there. The average of these two ("moderate congestion" on the link) would correspond to an event of probability 0. Should this be the basis for guidance generation?

- use the full network condition stochastic process. Knowledge of the full network conditions process would suffice to generate the full stochastic process of messages in response. However, it is difficult to see how this information could be used in a guidance system.

The above reasoning suggests that, for the generation of anticipatory guidance in a stochastic setting, guidance messages should be based on the most recent realization of network conditions, and this approach will be followed in the remainder of this research.

3.3.3 Guidance consistency in the presence of stochasticity

Recall the general definition of guidance consistency, which is that the assumptions on which the guidance was based prove to be verified, within the logic of the predictive model, after drivers receive

the guidance and react to it. In the deterministic case, consistency translated into a requirement of equality between the framework variable time trajectories that are input to and output from a framework map. As was seen above, this requirement implied that the trajectory is a fixed point of the corresponding framework map.

What might guidance consistency mean in a stochastic context? Clearly the requirement of equality of the framework map's inputs and outputs cannot be maintained in this case because of the randomness of the output trajectories. Rather, the assumptions on which guidance is based relate to the characteristics of the input and output stochastic processes themselves. The input to the map is a realization of a stochastic process. Conditional on this realization, the output is also a realization of a stochastic process. Consistency means, not that the realizations are numerically the same, but rather that they are drawn from stochastically equivalent processes (i.e., stochastic processes that differ at most on a set of measure zero).

Consider, for example, a path split stochastic process; it indicates, at each decision point and in each time step, the fraction of trips from the decision point to a destination that use each available subpath. The composite path split map $D \circ G \circ S$ takes a particular realization of an input path split process, and produces a realization of an output path split process.

Focusing on one particular decision point and one time step, the input process is simply a multinomial distribution of path splits over the k (say) available subpaths to the destination: $p^I = (p^I_1, \ldots, p^I_k)$, and the output process is similarly $p^O = (p^O_1, \ldots, p^O_k)$. The input and output processes are nothing more than the entire set of such distributions, over all decision points and time steps; these can be written P^I and P^O, respectively.

Consistency means simply that $P^I = P^O$, or equivalently that $p^I = p^O$ for each decision point and each time step. But since the output and input processes are related via the composite path split map, it follows that $P^I = P^O = D \circ G \circ S(P^I)$ or, suppressing the superscript, $P = D \circ G \circ S(P)$.

Similar reasoning could be stated for the composite network condition and composite message maps.

Thus, it is seen that the fixed point interpretation of guidance consistency applies as well to stochastic framework maps where, in this case, consistency (the coincidence of framework map inputs and outputs) means that they are both realizations of the same stochastic process.

3.4 Review of related work

This section reviews a selection of research efforts that are related in some way to the guidance generation framework proposed here. Some of these efforts investigate the guidance generation problem itself, while others adopt a fixed point formulation of conventional assignment problems. The review attempts to cover approaches to both static and dynamic network problems because of the insight that can be gained by viewing the issues from this broader perspective. The discussion concentrates primarily on problem formulation and analysis issues, although in some cases it also presents the proposed solution methods when these are inextricably related to the formulation itself or when, because the methods are not based on a fixed point analysis, they will not be separately discussed in Chapter 4.

The approaches discussed below include:

- fixed point approaches for static and dynamic network equilibrium models, including the

analyses by Kaufman, Smith and Wunderlich of full information dynamic traffic models for route guidance;

- stochastic process network models;
- Engelson's analysis of self-fulfilling and recursive forecasts;
- Bovy's and van der Zijpp's analysis of dynamic traffic management and driver information systems;
- the rational expectations model in economics;
- Van Schuppen's game theory approach to anticipatory guidance generation; and
- Kaysi's systems analysis of driver information systems.

3.4.1 Fixed point approaches for network equilibrium models

While they have never been in the mainstream of analytical or algorithmic developments for network equilibrium models, fixed point approaches have nonetheless been applied to useful effect in a number of investigations of these models over the past several decades. The fixed point formulation of user optimality conditions is very natural and is perhaps more intuitive than some alternative formulations, for example in terms of equivalent nonlinear optimization, nonlinear complementarity or variational inequality problems.

The connection between equilibrium and fixed points is part of the culture that transportation systems analysis inherited from economics. This connection was exploited for computational purposes by Kuhn [1977], who developed an algorithm for computing deterministic user-optimal equilibria in static traffic networks via fixed point approximation methods. Early analytical work along these lines was done by Braess and Koch [1979], who investigated the existence of deterministic user equilibrium in multi-class static networks with asymmetrical link cost function Jacobians. Stochastic user equilibrium (SUE) [Daganzo and Sheffi, 1977] is a fixed point concept by its very definition, although this does not appear to have been explicitly pointed out in early articles on the subject. However, Daganzo [1983] later used fixed point methods to extend the work of Braess and Koch [1979] to include static stochastic user equilibrium, principally in the case of fixed demand and multiple vehicle types, and identified an assignment algorithm suitable for handling such problems via fixed point computation. Fisk [1984] also presented a general framework for formulating certain static network equilibrium problems as fixed point problems; this framework is an extension of the analysis she carried out in Fisk and Nguyen [1981] of a static asymmetric two-mode model originally formulated by Florian [1977]. She derived existence and uniqueness conditions for this fixed point formulation, and illustrated her conclusions by reexamining Florian's two-mode model as a particular application.

By far the most general fixed point formulation to date of the static network assignment problem is due to Cantarella [1997], whose work builds on that of Daganzo [1983]. Cantarella investigated a multi-mode, multi-user class model with elastic demand, path and hyperpath routing choices, asymmetric link performance functions, and both link-additive and non-additive path impedances. He expressed and analyzed both stochastic and deterministic user equilibrium for this model in a single fixed point framework.

The variables in Cantarella's framework are (vector) link flows $\mathbf{f}$ and link costs $\mathbf{c}$. Link costs are directly dependent on flows through the link performance function $\mathbf{c} = c(\mathbf{f})$. Link flows are indirectly dependent on costs: link costs determine hyperpath costs that determine hyperpath systematic utilities. The systematic utilities determine the expected maximum utility of the hyperpath choice sets, and this fixes the level of demand for each OD pair. The utilities also determine the choice probability of each hyperpath. Combining the OD demands with the hyperpath choice probabilities and the (fixed) link-hyperpath incidence matrix finally gives the link flows. Cantarella terms this latter composition of functions the network loading map $\mathbf{f} = f(\mathbf{c})$. (Clearly, this is not at all the same thing as the network loading map defined in the framework discussed above. Terminology in this field is far from being standardized.) From these two component maps Cantarella defined and analyzed two composite map fixed point problems, the solutions of which express the notion of static user equilibrium in his framework:

- in link flow space: $\mathbf{f}^* = f(c(\mathbf{f}^*))$ and
- in link cost space: $\mathbf{c}^* = c(f(\mathbf{c}^*))$

Cantarella's particularly lucid analysis of static network user equilibrium using fixed point methods, and his demonstration of the power of these methods when applied to such problems, were an inspiration for the approach adopted in this research.

Fixed point approaches have also been proposed for the full information dynamic traffic assignment problem. Again, however, such approaches are not on the mainstream of investigations into dynamic network problems. Perhaps because the dynamic assignment problem has been less intensively investigated than the static problem, examples of such applications are rarer.

One of the few applications of these approaches to the dynamic case is the SAVaNT (Simulation of Anticipatory Network Vehicle Traffic) route guidance method developed by Kaufman et al. [1991]. SAVaNT models guided and unguided vehicle classes. Unguided vehicles are assumed to follow minimum paths based on free-flow travel times, while guided vehicles are assumed to comply fully with time-dependent minimum time path recommendations available at every node; a single path is computed for each node-destination pair and each time step. In short, SAVaNT assumes fully informed choice by guided vehicles and so can be considered a deterministic user-optimal full information model with fixed route background traffic consisting of unguided vehicles. However, the authors presented it as a route guidance generation system, and as such used it to highlight and discuss issues such as guidance consistency.

SAVaNT uses a mesoscopic traffic simulator, INTEGRATION, to move vehicles along their respective paths and to determine the resulting traffic conditions. The version of INTEGRATION used by Kaufman et al. was essentially deterministic: its only random component involved vehicle inter-departure times at the origins, and the authors reported that the stochasticity introduced in this way was very minor. To generate guidance in SAVaNT, time-dependent minimum paths are computed and guided vehicles are moved along them, resulting in a new set of conditions and new minimum paths, which in turn are used to compute minimum paths in the next iteration. Because of INTEGRATION's essentially deterministic dynamics, and SAVaNT's use of single-path guidance and assumption of full driver compliance, the guidance generation process evolves deterministically, and can assume only a finite number of states following initialization. Thus, it necessarily must either converge to a fixed point (the minimum paths are repeated in iterations k and $k+1$) or else cycle (minimum paths are repeated in iterations k and $k+n, n>1$). In fact, Kaufman et al.

found that SAVaNT invariably cycled when using the time-dependent link traversal times from the mesoscopic simulator as the basis for the minimum path calculation.

The SAVaNT method has a straightforward interpretation in terms of the guidance analysis framework. The mesoscopic traffic simulator implements the network loading map. Guidance consists of a time-dependent minimum path recommendation based on predicted time-dependent link times for the guided vehicles. Since guided drivers comply fully with the guidance, the routing map simply has all the OD flow of each class follow the recommended path.

Again, terminology in this area is not standardized. What we refer to as the network loading map is called the *assignment map* $\mathcal{A}$ in the work of Kaufman et al.; what we refer to as the guidance map (minimum path recommendations) is called the *routing map* $\mathcal{R}$; what we call guidance messages are referred to as a *routing policy* Π; and what we refer to as the routing map (resulting from the SAVaNT assumption that drivers comply fully with the guidance recommendations) is not given a specific name.

Despite formulating the fixed point problem in the space of routing policies, SAVaNT tests for convergence by comparing the time-dependent link traversal times obtained in successive iterations. This is required because in deterministic shortest path assignments the routing map is a correspondence (one-to-many map). Fixed points of correspondences are defined in the sense of set inclusion: if $T : X \mapsto 2^X$, then $x^* \in X$ is a fixed point of T iff $x^* \in T(x^*)$. (The notation 2^X denotes the *powerset*, or set of all subsets, of X.) It is meaningless to attempt to test for a fixed point of a correspondence by testing the equality of the results of two evaluations. The travel times that result from a routing policy Π do not suffer from this problem.

In many ways, Kaufman et al. [1991] was a seminal work. It was one of the earliest papers to recognize the issue of consistency in anticipatory guidance. It identified guidance generation as a fixed point problem over a composite routing-assignment map $R \circ A : \Pi \mapsto \Pi$, which is a form of the composite guidance message map $(D \circ I) \circ S : P \mapsto P$ discussed above. However, the SAVaNT approach lacks the generality of the framework proposed here:

- it does not identify driver response to information as a key problem component;
- it is based on a particularly simple assumption of driver path choice behavior (full compliance with minimum path routing);
- the guidance generation objective is assumed to match the driver path choice criterion;
- it assumes that guidance information is perfect (ubiquitous, precise, up-to-date) and does not provide for any relaxation of this assumption;
- it does not recognize alternative possible fixed point problem formulations.

Finally, the solution approach proposed by Kaufman et al. to compute fixed points of their composite routing-assignment map is not generally convergent, and the authors were obliged to resort to a heuristic to make it so. This will be discussed in greater detail in section 4.3.1 below.

3.4.2 Stochastic process and related network models

A number of researchers have investigated probabilistic aspects of static traffic network models, generally using dynamic process methods.

One line of investigation has generalized the path choice component of static stochastic user equilibrium (SUE) models. In standard SUE models, the flow on an OD path is exactly equal to the product of the OD demand and the path choice probability output by a choice model. An alternative approach is to consider the path flows to be realizations of a multinomial distribution defined by the OD demand total and the different path choice probabilities. In this approach, the expected flow on any path is equal to the value predicted by the standard SUE model, but other flow values are also possible in accordance with the multinomial distribution. The outcome of the model is a joint distribution of flows over different paths, from which distributions of link flows, link times and path times can also be derived. Hazelton et al. [1996] and Hazelton [1998] propose Gibbs sampling as a numerical method to approximate the distribution of network variables (path and link flows) resulting from this assumption. (Gibbs sampling constructs an artificial Markov process whose stationary distributions replicate the probabilities of a joint distribution that is difficult or impossible to determine directly. By simulating the Markov process transitions, a sample from the joint distribution is generated. It is discussed in Section 4.1.)

Another line of research has investigated the properties of the outputs generated by successive applications of static traffic assignment models, where the inputs (link times or flows) used in one model application result from a filtering of the model input and output values from prior applications (so-called "day-to-day" static models). The sequence of model outputs from successive applications constitutes a dynamic process, the properties of which can be analyzed using methods from nonlinear dynamical systems theory.

Cascetta [1989] was an early research effort along these lines. It introduced the idea that, due to random variations in factors affecting travel demand and conditions, the sequence of states occupied by a network over successive epochs (e.g., days) is a realization of a stochastic process. It derived sufficient conditions for stationarity of the path split process in terms of the properties of the path choice model, and showed the implications of stationary path splits in terms of the link flow and time processes. Relationships between the properties of these stochastic processes and the outputs of an SUE model were also derived.

Subsequent efforts extended these results. Cascetta and Cantarella [1991] proposed a "doubly dynamic" model that represented both within-day and day-to-day dynamics as stochastic processes; in Cascetta et al. [1991] this model was applied to the analysis of traffic control measures. Davis and Nihan [1993] considered day-to-day stochastic process models and showed that, for some particular model forms, as the number of individual travelers becomes large, model outputs can be approximated as the sum of a nonlinear deterministic process and Gaussian white noise. Cantarella and Cascetta [1995] analyzed the consequences of different assumptions regarding travelers' learning and forecasting filters, which integrate the effects of prior days' travel expectations and experiences on the current day's choice. By varying filter parameters, they were able to exhibit qualitatively different kinds of dynamic process behavior, including convergence towards different kinds of dynamical system attractors (fixed point, periodic, quasi-periodic and aperiodic). Watling [1996, 1999] considered relationships between properties of a dynamical process traffic model and the stability of the SUE problem.

It is striking that stochastic process approaches to dynamic traffic assignment models have not be pursued to a comparable extent. The research of Nagel and co-workers [see Nagel, 1997, for example] is one of the few efforts that clearly recognized the stochastic process nature of the outputs from stochastic dynamic traffic simulators; they furnished an example where use of different

random number seeds led to significantly different outputs in otherwise identical simulation runs. However, although TRANSIMS is a stochastic model, its solution algorithm (discussed in section 4.1.2) suppresses the stochasticity in the system outputs.

3.4.3 Self-fulfilling and recursive forecasts

Engelson [1997] presented a rather general analysis of predictive information provision in situations where users' responses to the predictions are likely to affect the future being predicted; he used anticipatory route guidance as an example of such a situation.

Key to Engelson's analysis are the notions of *self-fulfilling* and *recursive* forecasts.

> A self-fulfilling forecast is defined by the following property: If the message containing the result of the forecast is reported to the drivers before they make their route choices, then the expected value of travel time for each driver and each route will equal the value reported in the message. [Engelson, 1997]

In other words, a travel time forecast is said to be self-fulfilling if, for each route and driver, the mean travel time that actually results on the network after the forecast is disseminated is what was stated in the forecast.

In contrast to this, the notion of a recursive forecast is model-based: it is only defined in terms of the models that are used to perform forecasts.

> ...a model of users' behavior and a model of service response [are needed]. The first model uses values of performance indicators which the agent includes in the message as input. The result of the model is a description of users' behavior, that is, the number of users choosing each action. The second model uses the result from the first model to estimate performance indicators, or to predict the service response.... [A] recursive forecast is defined as a set of values which equates input of the first model with output of the second. [Engelson, 1997]

Engelson carries out his analysis in the context of static network equilibrium models. His performance indicators C are just static path times and his user behavior map $D : C \mapsto P$ is essentially a path choice model that predicts the static path flows P. Similarly, his service response model $S : P \mapsto C$ is a static network loading model. He states clearly that finding a recursive forecast amounts to solving either $S(D(\bar{c})) = \bar{c}$ for a set of path times $\bar{c} \in C$, or $D(S(\bar{p})) = \bar{p}$ for a set of path flows $\bar{p} \in P$, and notes that in stochastic models "only approximate equality with some precision level" can be required. In the deterministic case, however, if both D and S are continuous, then Engelson notes that applying Brouwer's theorem to the mapping $S(D(\cdot))$ (which is defined over a finite-dimensional compact set P) guarantees the existence of a fixed point.

Engelson's approach is noteworthy for its careful analysis of the notion of consistency in reality ("self-fulfilling" forecasts) and in modeling ("recursive" forecasts). He provides a general mathematical formulation of the two notions. Stochasticity is allowed, although he assumes that its effects can be adequately captured via the expected values of the involved random variables. In the context of travel time forecasts for route guidance, he recognizes the variety and importance of models of driver response to guidance information and provides an interesting discussion of the implications of various common travel behavior models on the applicability of the Brouwer theorem to prove the existence of a guidance solution. Finally, he recognizes that a fixed point definition of consistency can induce multiple fixed point problem formulations.

Engelson's analysis, however, is couched in the context of the static network equilibrium problem. (He recognizes the generalization of his problem to dynamic networks but does not undertake its analysis.) In a static problem, all decisions are intrinsically pre-trip; thus, there is no reason for him to consider dissemination systems providing en route guidance although, as has been seen, this is a central issue in the analysis of dynamic guidance. Further, the assumption that random variables can be represented by their expected value—a common approach in static network equilibrium modeling—hides many of the dynamic effects that stochasticity can create in a more general setting. Guidance messages are not represented, so providing messages that are different from network conditions is not possible in his model. Finally, he recognizes the potential computational difficulty of determining a guidance fixed point in large networks, but does not address this problem.

3.4.4 Bovy's and van der Zijpp's analysis

Bovy and van der Zijpp [1999] present an analysis of anticipatory travel guidance systems that is based on a framework (a *conceptual model*, in the terminology of the paper) that is quite similar in many respects to the one presented here.[7]

The authors consider dynamic traffic prediction models incorporating both traffic control and guidance information. They single out a number of variables and relationships between them as key to modeling the traffic network guidance and control problem. All relationships discussed in the paper are implicitly deterministic. Sources and effects of model stochasticity are not examined, although the authors recognize that the underlying uncertainties of traffic predictions pose a considerable challenge to guidance applications. Variables used in the conceptual model include:

- routing fractions (or path splits). Actual fractions can differ from pre-trip fractions due to en route path switches, although the paper does not discuss this in detail;
- network conditions. The paper focuses on link volumes and traversal times as examples, but this choice could be generalized;
- information. General information is generated based on network conditions (link volumes), then filtered using an information availability map that determines (based on the dynamic link-path incidence matrix) what specific information is available at particular times and locations on the network.

Relationships between these variables are not explicitly named in the paper, and do not necessarily correspond one-for-one with those proposed in this dissertation. For example, the authors explicitly represent a number of low-level relationships (between path flows and link volumes, between link volumes and link times, and between link times and the dynamic link-path incidence matrix) that are here considered internal details of the dynamic network loading map. The authors' routing map (which outputs routing fractions) takes as inputs both received information and also link traversal times; their message map is based on predicted link volumes. These differences

[7]Their work was originally presented in 1995 at the 7th World Conference on Transportation Research in Sidney, Australia, but was not published in the WCTR proceedings. The paper was first published in 1995 in a Dutch technical journal [Bovy and van der Zijpp, 1995], and reprinted with small changes in the 1999 collection cited above. I was unaware of this important work until Prof. Bovy drew my attention to it during his review of a near-final draft of this dissertation.

between the approach of Bovy and van der Zijpp and the approach developed in this dissertation are not fundamental.

The authors recognize that guidance consistency corresponds to a fixed point of the various model relationships, and provide a fixed point formulation expressed in terms of routing fractions. They suggest that an iterative approach might be used to find the routing fraction fixed point but do not pursue this idea.

There is a suggestion regarding the possible form of a driver response to information model taking account of prior days' experiences as well as messages received during a particular trip. Compliance and information acquisition behavior are not represented explicitly, although their effects are captured in model parameters.

The framework proposed in Bovy and van der Zijpp [1999] is remarkably similar to the one developed in this dissertation and the paper overall is an impressive contribution to the literature that deserves to be better known.

3.4.5 Rational expectations models

The rational expectations hypothesis (REH) was introduced in economic theory by Muth [1961]. It applies to situations in which the behavior of an economic actor with incomplete information is determined in part by her expectations about the future state of the economy—and this behavior, of course, influences in turn the future state of the economy. The REH postulates that, over time, an economic actor develops an internal model of the economy that allows her to make inferences about the probability distribution of economic variables from publicly-available information. These inferences are derived, explicitly or implicitly, from an individual's internal model of the relationship between the (limited) information she receives and the actual state of the economy. On the other hand, the true relationship is determined by individual actors' behavior, and hence by their expectations. Hence there are feedback routes from the true relationship to the individual expectations. A rational actor will be motivated to revise her expectations if she observes differences between expectations and experiences. An equilibrium of this system, in which the individual subjective distributions are identical to the true distributions, is called a rational expectations equilibrium (REE). The theory was extensively developed and applied in the 1970s and is now one of the standard approaches to both macro- and microeconomic analysis in the presence of uncertainty [Radner, 1982].

In a series of papers, Kobayashi [1993, 1994] and Kobayashi and Tatano [1999] considered the application of the rational expectations approach to network problems, and particularly to the analysis of route guidance systems. Kobayashi [1993] developed an analysis of general networks of interacting actors (which he termed "logistical networks"). He extends the theory of games with incomplete information to include the formation of rational expectations under a Bayesian learning protocol, and characterizes and provides conditions for the existence of a rational expectations equilibrium. This paper provides a very simple example of REE in a traffic network.

These ideas were applied more directly to the analysis of traffic network route guidance systems in Kobayashi [1994]. Here Kobayashi considers a traffic network used by drivers with imperfect information about path travel times. Drivers receive guidance messages, which condition their expectations about path times. They then choose a path, and experience its true travel time; this may lead them to update their subjective expectations. Kobayashi shows that each driver's

subjective expectations converge to rational expectations under mild conditions on the learning process.

In Kobayashi and Tatano [1999] these ideas are further pursued. The authors use rational expectations theory as a basis for determining the welfare benefits of route guidance systems.

Engelson [1997], discussed above, has questioned the validity of a rational expectations approach to route guidance, arguing that the one-shot nature of incidents precludes drivers from forming an internal model of the relationship between guidance messages and true network conditions in these situations.

3.4.6 Game theory approach to route guidance

3.4.6.1 Background

There are close connections between Nash equilibria in game theory and various definitions of equilibrium that have been proposed in traffic network analysis. These connections have been recognized at least since the treatments of the static traffic assignment problem presented in Dafermos and Sparrow [1969] and Dafermos [1971].

A static non-cooperative N-person game in normal form is defined as a set of N players $1, \ldots, N$, each of whom plays by independently selecting a strategy from her strategy space X_i. When player i deterministically selects to play a single element x_i from the strategy space, x_i is referred to as a *pure* strategy.

The outcome, or quantitative *payoff*, to player i of playing a strategy depends on the selected strategy but also on strategies played by each of the other players in the game. In the applications considered here, game outcomes are generally travel times or impedances, so the term *penalty* will be used instead of payoff, and players seek to minimize their penalty. Each player's individual outcome from a choice of pure strategies $x_1, \ldots, x_N$ by all players is thus given by a penalty function $\phi_i : X \mapsto \Re$ defined on the joint pure strategy space $X = \prod_{i=1}^{N} X_i$.

A player i may also choose to play a *mixed* strategy, which means that he or she selects randomly from among the available strategies $x_i^j \in X_i$ in accordance with a probability distribution p_i^j. The penalty to player i of playing a mixed strategy ξ_i is defined in an expected value sense, and each player's problem is to determine the probabilities p_i^j to minimize the expected penalty.

Given the pure strategies $x_{-i} \equiv \{x_j | j \in \{1, \ldots, N\},\ j \neq i\}$ selected by the other players, player i would only select a strategy from among the minimizers of $\phi(\cdot, x_{-i})$. Since in general there may be more than one such minimizer, the relation between the strategies selected by other players and player i's response is a correspondence, called i's *best response correspondence* $\psi_i(x_{-i})$. Formally, $\psi_i(x_{-i}) = \{x_i^* \in X_i | \phi_i(x_i^*, x_{-i} = \min_{x_i \in X_i} \phi_i(x_i, x_{-i})\}$. Knowing the strategies x_{-i} chosen by the other players, player i's best response correspondence ψ_i gives the set of strategies in X_i resulting in minimum penalty.

A point $x^* \in X$ is a *Nash equilibrium* of the non-cooperative game if

$$\phi_i(x_1^*, \ldots, x_{i-1}^*, x_i^*, x_{i+1}^*, \ldots, x_N^*) \leq \phi_i(x_1^*, \ldots, x_{i-1}^*, x_i, x_{i+1}^*, \ldots, x_N^*), x_i \in X_i, \forall i = 1, \ldots, N$$

or, equivalently,

$$x^* \in \prod_{i=1}^{N} \psi_i(x_{-i}^*)$$

.

In other words, in a Nash equilibrium, each individual player's strategy choice is optimal within his strategy space and with respect to his individual penalty function, given the strategies chosen by the other players. The equivalent definition states that a Nash equilibrium is a fixed point of the players' joint best response correspondence.

3.4.6.2 Static network case

A number of researchers have recognized and developed similarities between the Nash equilibrium concept and various equilibrium concepts in traffic networks.

As noted above, Dafermos [Dafermos and Sparrow, 1969; Dafermos, 1971] had already mentioned this connection in some of her earliest work. Considering OD pairs to be players, the set of feasible allocations of OD flow across available paths to be a strategy set, and the maximum cost of travel on any OD path receiving positive flow to be the penalty function, she recognized that user-optimal flow patterns are equivalent to Nash equilibria for the non-cooperative game so defined.

Rosenthal [1973] studied a version of the traffic assignment problem in which path flows are considered to be the result of minimum travel cost path choice decisions made independently by a finite number of tripmakers, and hence are restricted to be integers. He analyzed this situation as a non-cooperative game in which the players are individual tripmakers, a player's strategy space is the set of available paths, and a player's penalty function is the OD travel time. A Nash equilibrium of this game is equivalent to a user-optimized flow pattern with unit flow swaps. Under an assumption of separable link costs, Rosenthal showed that pure strategy (i.e. integer flow) Nash equilibria always exist and in fact that any solution to a discrete analogue of Beckman's user-optimal equivalent optimization objective function is such a pure strategy equilibrium (the converse is not true, however).

This result was extended to the continuous case by Devarajan [1981]. He defined the OD pairs to be players and the allocation of OD demand across available routes as a (possibly mixed) strategy. As before, separable link costs are assumed. Here the penalty function for OD pair rs is actually defined to be Beckman's equivalent optimization objective function,

$$\phi_{rs} = \sum_{a \in \mathcal{L}} \int_0^{f_a} t_a(x)dx,$$

so that the Nash equilibrium is equivalent to a Wardrop equilibrium.

Haurie and Marcotte [1985, 1986] considered relationships between game theoretic equilibria and static traffic network equilibria in a more general setting. In their analysis, a number of players may be associated with an OD pair, with strategy sets representing the allocation of the corresponding OD demand across paths, and the penalty function being the cost of the chosen routes. The game obtained in the limiting case, when the number of players in each OD pair tends to infinity, while sharing the same strategy, is shown to be equivalent to a Wardrop equilibrium.

3.4.6.3 Game theory and route guidance

It is clear that game theory approaches could also be applied to dynamic traffic network problems, although there appear to be relatively few examples of such applications described in the literature.

One exception is found in the research of van Schuppen [1997], carried out as part of the DACCORD project.[8] Van Schuppen applied game theory concepts to the design of control laws (i.e., guidance maps) for generating route directives. In his analysis, each player controls a variable direction sign (VDS) that provides route recommendations to flows for one OD pair. At any particular time, a VDS displays one from among a finite set of route directives for its OD pair. The display of a directive is maintained for a predetermined time interval, but can be changed thereafter. Over the fixed duration of the game, a sequence of directives is displayed on each VDS; clearly, there is a finite number of such sequences.

When traffic from an OD pair passes by its VDS, the directive that is being shown at that particular time affects the traffic's choice of subsequent path in some (unspecified) way. The combined path choices of traffic from all OD pairs determine the link and path travel times. These both are time-varying, reflecting the dynamics of OD demands entering and propagating through the network.

The route directive displayed for an OD pair is determined by a corresponding control law, which maps the network state (extended to include forecasts of exogenous factors such as OD demands) into a guidance directive. A player's strategy in this game consists of a choice of a control law.

A player's penalty function is the travel time needed to go from the origin (or, more precisely, from the VDS location) to the destination. It clearly depends on the player's individual control law but, since different OD flows interact on the network, it depends on the control laws adopted by other OD pairs as well. A set of individual control laws adopted by each OD pair is called a control law for the game.

More formally, let us define the following notation:

- $\mathcal{G}_{rs}$ is the class of admissible control laws for OD pair rs;
- $g_{rs} \in \mathcal{G}_{rs}$ is an individual control law for OD pair rs;
- $g = \{g_{rs} \in \mathcal{G}_{rs}, \forall rs\}$ is a control law for the game;
- $J_{rs}(g)$ is the travel time (penalty function) for OD pair rs when control law g is applied.

Van Schuppen restricted attention to control laws $g^* = \{g^*_{rs} \in \mathcal{G}_{rs}, \forall rs\}$ that are Nash equilibria:

$$J_{rs}(g^*) \leq J_{rs}(g), \forall rs$$

where $g = g^*$ for all OD pairs other than rs, and $g_{rs} \in \mathcal{G}_{rs}$ is arbitrary. This is a logical generalization of the interpretation of static Wardrop equilibria as Nash equilibria, and reflects the objective of minimizing individual experienced OD travel times while not favoring one OD pair at the expense of another.

It is virtually impossible to compute or even to approximate Nash equilibrium routing control laws because of the high dimensionality of the problem and the complexity of the class of admissible control laws. As an alternative, Van Schuppen suggested attempting to approximate the guidance directives that would be output by a such a law. For this purpose, he proposed the heuristic summarized in Figure 3.2:

[8] DACCORD is a project funded by the European Community. Its objective is the design, implementation and evaluation of advanced dynamic traffic management systems for the integrated and coordinated control of interurban motorway corridors.

```
given initial time trajectories of link conditions and route directives

for each stage t in the rolling horizon
    prioritize the set of OD pairs
        for each OD pair in priority order
            generate a set of route directive time trajectories
                for each route directive trajectory
                    compute the resulting OD time
                select the route directive trajectory with minimum time
    apply the selected route directive trajectory for the next stage
```

Figure 3.2: Approximate computation of Nash equilibrium routing directives [van Schuppen, 1997]

This can be seen to be a rolling horizon procedure in which the processing of each stage involves online input design using look-ahead over a finite horizon. The key steps of this algorithm are the computation of the OD travel time resulting from a particular set of guidance directives; the generation of a set of route directive time trajectories to consider for each OD pair; and the ordering of OD pairs for processing.

In Van Schuppen's approach, OD travel time is defined as the average travel time experienced by vehicles that depart in the middle of the initial time period. It is determined by forecasting the evolution of traffic conditions from a given initial state and under the effects of the selected guidance directives. This is the finite look-ahead step.

The set of possible route directive time trajectories for an OD pair is simply determined as the Cartesian product of possible directives in one time period across the number of rolling horizon stages. The examples discussed in van Schuppen [1997] involve two possible route recommendations ("turn left" or "turn right") that are fixed for three ten-minute intervals, giving a look-ahead horizon of thirty minutes. Thus, there are only eight (two possible paths per time step over three time steps or 2^3) directive trajectories to consider per OD pair over the entire horizon. However, as Van Schuppen pointed out, the number of trajectories that this approach generates will increase combinatorially with the size of the problem: if there are T time steps and D possible route directives per OD pair, D^T distinct trajectories will be generated per pair.

For each such trajectory, traffic predictions must be carried out to determine the corresponding OD travel time. The directive trajectory that results in minimum travel time for the OD pair is selected. The combination of generating possible trajectories, determining the cost of each and selecting the one with minimum cost is the online input design step of Van Schuppen's procedure.

The processing order of OD pairs is established by first selecting OD pairs whose route directives send flow through "congested" (in some suitably-defined sense) links, then taking pairs whose directives send flow through links on paths that are alternatives to the paths with congested links, then pairs whose traffic uses links on alternatives to the alternatives, and so on. The process is continued until no more OD pairs are identified, at which point any remaining pairs are included with lowest priority. It can thus be considered a greedy algorithm.

This procedure is an approximation because it makes a single pass through the OD pairs: once the directive for an OD pair is determined, subsequent modifications to directives for lower-priority OD pairs do not cause it to be reconsidered. By definition, however, any such modification

invalidates the Nash equilibrium verification for OD pairs considered earlier in the procedure. The changes in traffic flows and conditions resulting from the modified directive could potentially require changes to directives chosen for OD pairs considered earlier. However, since the procedure makes a single pass, already-processed OD pairs are not reconsidered. The processing order of OD pairs is chosen in a way that attempts to minimize the impacts of this approximation.

In de Waal et al. [1999], the above Nash equilibrium approach to guidance generation was tested by simulation. For this purpose the authors programmed a discrete-time deterministic traffic simulator derived from the continuous-time stochastic model of Smulders [1996]. A state space model of traffic propagation over a single link was extended to a general motorway network. The model allows destination-specific routing directives to be provided at intersections, and takes account of imperfect driver compliance with the directives. Data for the Amsterdam ring network and its traffic were obtained and used to develop program inputs. In this network, only binary routing directives were meaningful, and viable path alternatives were available to four OD pairs. Simulation of thirty minutes of traffic on the network took about seven seconds of computation time on a Sun Ultra10 workstation.

Instead of applying the approximation procedure described above, the authors decided instead to consider constant directive trajectories (i.e., the same directives are displayed over the entire thirty-minute period) and to carry out an exhaustive search for Nash equilibria over all such trajectories. Given the problem characteristics presented above, there were sixteen (two possible paths for each of four OD pairs or 2^4) distinct directive combinations to consider, requiring roughly 100 seconds of computation time. The authors investigated a number of situations including normal conditions, and single and multiple bottlenecks; in each of these situations, a Nash equilibrium was found and was unique.

Lack of data on time variations in OD demand rates led the authors to assume constant OD demand rates over the analysis horizon. Because of this, restricting consideration to constant guidance trajectories was perhaps a reasonable simplification. In a more general situation, however, with dynamic inflow rates, such a simplification could be questionable. Furthermore, since there is no general relationship between the existence and properties of dynamic Nash equilibria calculated at very different time discretizations (i.e., with directives able to change every ten *vs.* every thirty minutes), the conclusions reached about the existence and uniqueness of Nash equilibria for the constant trajectory case do not necessarily generalize.

Again, this procedure is computationally very demanding. With R OD pairs, each having D routing directives able to change over T time periods, an exhaustive search for a Nash equilibrium would entail D^{RT} separate traffic predictions. For example, if directives had been allowed to change every ten minutes over the thirty minute horizon, the authors' example would have required 4096 runs of the traffic simulator, taking for the example network approximately eight hours of computation time.

There are a number of differences between Van Schuppen's dynamic game approach and the guidance generation framework presented earlier in this chapter:

- In Van Schuppen's formulation, one VDS is associated with each OD pair; it disseminates route directives to all tripmakers for that pair. The definition of M in the guidance generation framework is intended to allow more general situations such as multiple information sources available to an OD pair, OD pairs with no information sources, different user classes with access to different types of guidance, a variety of message formats, etc.

- Strictly speaking, a control law g maps the current network state estimate (and any forecasts of exogenous inputs) into a set of current guidance directives. As was discussed above, the rigorous determination of a Nash equilibrium control law is, for all practical purposes, impossible. Consequently Van Schuppen proposed an online input design heuristic for identifying directives that approximate the outputs of a Nash equilibrium control law. The heuristic procedure plays a role similar to the guidance map $G : C \mapsto M$.

- The penalty function $J(g)$ predicts the OD travel times that result from control law g and implicit parameters such as OD demand. To do this it must incorporate a model of (i) how users choose paths based (in part) on route directives and (ii) how these reactions affect OD travel times.[9] Consequently, the penalty function is a form of composite $(S \circ D)$ map in which the distinct properties of the dynamic network loading map and the routing map are intertwined.

- In Van Schuppen's approach, penalty functions are defined in terms of travel times on the shortest path between origin and destination, corresponding to an assumption that tripmakers choose minimum time paths. A more general route choice model would require a (perhaps substantial) modification of the definition of network conditions.

Nonetheless, there are deep similarities between Van Schuppen's approach and that proposed here; these derive from the close relationships between Nash equilibria and fixed points. Some of the differences mentioned above are superficial and could probably be removed in a more general formulation of the game problem. Perhaps the most fundamental differences are that the framework treats explicitly and uniformly each component of the guidance generation problem (network, tripmakers and guidance) and, because of this, three symmetric formulations of the consistency problem emerge in a straightforward way. When a problem has an inherent symmetry it is natural, at least in initial investigations, to prefer methods that exploit this property. The game theory approach does not allow this. Consequently, the framework approach proposed here would seem to offer greater opportunities both for modeling each component of the problem, and for attacking the problem as a whole.

3.4.7 Kaysi's analysis

In his Ph.D. dissertation, Kaysi [1992] carried out a comprehensive system-level analysis of real-time travel guidance systems, focusing primarily on route guidance but also considering issues related to departure time and mode choice guidance. (Related works co-authored by Kaysi include Ben-Akiva et al. [1991], Kaysi et al. [1993] and Kaysi et al. [1995].) The analysis considered a wide range of functional components required for guidance systems including traffic surveillance, OD matrix updating, driver behavior and network performance modeling, self calibration capabilities and guidance generation. The research emphasized anticipatory guidance.

Kaysi's analysis was a pioneering effort that was among the first to identify and investigate many of the central issues of route guidance system design. Among other things, the work:

[9] "It will be assumed that there is available a prediction program that, if a specification is made of the setting of all road control measures and of the traffic measurements, will produce an estimate of the travel times for the traffic flow of all relevant OD pairs." [van Schuppen, 1997, page D.18]

- defined and analyzed the need for consistency in predictive guidance;
- identified consistency as equivalent to a fixed point of the combined congestion prediction (related to the composite map $S \circ D$ presented here) and control and routing (related to the component map G presented here) relationships;
- recognized information imperfections and driver response to guidance as key issues in the guidance generation problem;
- traced possible negative effects of guidance (overreaction) to problems in guidance quality (including inconsistency);
- discussed at length driver behavior and network performance modeling requirements, including a particularly detailed treatment of the modeling of driver perception updating and decision making;
- recognized that the network model has to take account of driver reactions to guidance, including guidance received en route;
- carried out simulation experiments investigating the impacts of many guidance system design options and parameters; and
- noted the presence of stochasticity in simulation outputs and identified some of its causes.

On the other hand, Kaysi did not fully explore some of the ideas that he identified in his research. In a real sense, the present dissertation is an attempt to deepen and extend in some areas the broad investigation begun by Kaysi. The following paragraphs indicate some of these areas, and describe differences between Kaysi's research and that presented here.

Kaysi recognized the need for a congestion prediction (COP) capability, but for him this was a monolithic enhanced traffic prediction model able to account for driver response to guidance. (In the framework of this chapter, Kaysi's COP function is the composite $S \circ D$ map, which outputs traffic conditions in response to travel guidance.) By not reducing the problem to its fundamental relationships (the three component maps S, D and G), the full implications of consistency (implying constraints on flows, conditions and guidance) remain obscured.

There is not a clear distinction in Kaysi's work between what is here called the guidance map, which generates guidance messages in response to any particular set of network conditions, and the algorithm used to compute consistent guidance. His control and routing CAR relationship included both the guidance response to forecast situations, as well as the computational algorithms that attempt to ensure mutual consistency between network conditions, guidance messages and driver response to them. Thus, Kaysi's analysis implicitly focused on the composite $G \circ S \circ D$ message map.

Kaysi's examples consider only a finite (and small) guidance message dictionary, and the guidance generation algorithm that he applies is essentially an exhaustive search over the set of possible messages: for each message it runs the enhanced traffic model to check for consistency (which he viewed as a fixed point over the guidance messages). The applicability of guidance generation method to large networks is uncertain, and its generalization to continuous messages is not obvious (in any case it was not attempted). Kaysi's formulation of consistency and the problem relationships is that it more or less imposes the one solution approach (application of an enhanced

traffic model to verify consistency) that he proposed. When the more basic maps are recognized as fundamental, a greater variety of possible solution methods becomes available, as will be seen in Chapters 4 and 5. (It must be pointed out, however, that Kaysi's research was primarily concerned with conceptual rather than computational issues.)

Finally, although Kaysi recognized the presence of stochasticity in the outputs of the simple simulation model that he used to investigate guidance system design issues, and was able to trace the causes of the stochasticity back to simulation model assumptions, he did not fully consider the implications of this stochasticity (i.e., model output variables forming a potentially arbitrary stochastic process).

These comments are not intended to be critical of Kaysi's work. His research remains remarkable for the extent to which, in the very early days of research into travel guidance, it clearly identified and explored the main issues involved in the design of anticipatory guidance systems.

3.4.8 Advantages of a fixed point approach

Although fixed point approaches have been applied in analyses of both static and dynamic network problems, they have generally been less widely used than other formulations such as equivalent optimization or variational inequality problems. Nonetheless, a fixed point formulation of consistent anticipatory guidance has a number of advantages.

First, fixed point formulations capture the concept of consistency between supply, demand and guidance in a natural and relatively simple way and, as was seen above, they lead to alternative expressions of this concept. Perhaps equally importantly, they provide a framework for determining whether a particular model system is structurally capable of generating consistent guidance. Fixed point consistency conditions are directly expressed in terms of the primary variables manipulated by a simulator, and do not require transformations (e.g., derivatives or integrals) of these; such transformations could be a source of numerical instabilities in solution methods. As simulators become more realistic, their basic maps will likely become more difficult to analyze mathematically, yet the composite map fixed point formulation of guidance consistency remains well-posed and relatively tractable despite the potential complexities of the individual maps.

Finally, a fixed point problem can be transformed into an alternative equivalent mathematical formulation (such as a game theory or variational inequality problem) if it is advantageous to do so for analytical or computational reasons. The equivalences between the different approaches are well established.

Chapter 4

Fixed Point Algorithms for Guidance Generation

> PHYSICIST TURNED SPY PRETENDING TO BE A MADMAN WHO THINKS HE IS EINSTEIN:
> *And I had to learn to play the fiddle. It was torture for someone like me with no ear for music.*
>
> Dürrenmatt [1964, Act II]

This chapter is a survey and synthesis of various mathematical methods that can be applied to the solution of fixed point problems occurring in stochastic, quasi-deterministic and deterministic guidance generation models. The chapter provides a focused review that is not intended to be a comprehensive discussion and rigorous analysis of the methods, but rather to furnish a minimum background for understanding what each method is and when it might be advantageously applied.

The intent of this chapter is to provide a unified overview of applicable fixed point computation methods, showing how a variety of solution methods that have been proposed in the literature for both static and dynamic network problems can be understood from the viewpoint of fixed point computations. At the same time, these methods are assessed in terms of their possible use for guidance generation computations.

Emphasis here is on methods that are mathematically valid and computationally feasible for route guidance generation applications. The route guidance generation fixed point problem is characterized by:

- the fact that the function of interest is a composite map;
- the general unavailability of analytical forms for the component functions;
- the possible presence of stochasticity in function outputs;
- the high computational cost of a composite function evaluation (involving, as it does, a dynamic network loading); and
- its dynamics;
- its very large dimensionality;
- (in an operational setting) the need to compute solutions in better than real time.

These characteristics impose considerable constraints on the set of feasible fixed point solution methods. For example, many "classical" methods, based on gradient or Hessian evaluations, are unlikely to be attractive for guidance problems because the derivatives have to be numerically evaluated at considerable computational cost and possibly with a high degree of unreliability (due to the stochasticity). Methods involving line searches may also be uncompetitive for the same reasons.

However, one of the conclusions of this chapter's review is that entire families of provably convergent solution methods with theoretically attractive properties remain untried in transportation applications, and indeed seem largely unknown to the transportation community. The most promising of these are implemented and tested in Chapter 5.

This chapter is organized as follows. Stochastic methods are discussed in Section 4.1, quasi-deterministic methods in Section 4.2 and deterministic methods in Section 4.3. In each case, presentation of the mathematical background to each method or class of methods is followed by a survey of transportation-related applications of the method that have been reported in the literature. In some cases, problems in the applications can be traced to a failure to recognize the underlying mathematical principles. The applications surveys also cover some applications that resemble yet differ in some key way from the solution methods presented here; these were included both to provide perspective on the principal methods and because of their intrinsic interest.

4.1 Gibbs sampling and related methods

This section focuses on methods that apply to dynamic stochastic network models, the outputs of which are realizations of a general stochastic process. Primary among these methods is Gibbs sampling, a procedure for generating random samples drawn from the full output stochastic process. Any process statistic of interest can be determined to a desired confidence level from the sample statistics of the random draws. Also discussed are the TRANSIMS route replanning algorithm, which deals with process stochasticity using a different approach; and a fictitious play method for finding system optimal routing patterns in dynamic networks, which resembles but actually is quite different from Gibbs sampling.

4.1.1 Gibbs sampling

4.1.1.1 Mathematical background

Consider a system made up of a finite number of components $i = 1, \ldots, N$, each of which assumes a state x_i from among a finite set of possibilities. The components interact, in the sense that the state of one component depends at least partially on the states of other components. If the interactions between the components are probabilistic in nature, then the resulting system state will also be probabilistic. Indeed, a complete description of the outcome of the mutual interactions is provided by the joint probability distribution of the combined states of the components. Less exhaustive descriptions may also be of interest, including notably the marginal distributions of individual components' states. However, the analytical determination of these distributions for a system of probabilistically interacting components is generally difficult to impossible.

One way of characterizing the probabilistic interactions is in terms of the conditional probability distribution of each component's state, given the states of all of the other components; these

are called the system's *full* conditional distributions. Provided that it is feasible, such a specification can capture both direct and indirect as well as deterministic and stochastic mechanisms of interaction between components.

Is specification of the full conditional distributions sufficient, in principle, to determine the joint distribution of the outcome of the component interactions? Besag [1974, pp. 195–196] answers this question in the affirmative, provided that a certain *positivity* condition holds. This condition requires that, if components $1, \ldots, N$ can individually be in states $x_1, \ldots, x_N$, then it is possible for the system as a whole to be in state $(x_1, \ldots, x_n)$. Formally, positivity requires that if, for particular state values $x_1, \ldots, x_N$, the values of the individual conditional distributions $\Pr(x_i) > 0 \; \forall i$, then the values of the joint distribution $\Pr(x_1, \ldots, x_N) > 0$ for the system as a whole, and is frequently verified in practice. Note that this very general result is the counterpart of equilibrium existence results for deterministic problems.

Even under the positivity condition, however, direct determination of the joint distribution from the full conditionals is not straightforward. Consider first the bivariate case (two components, say X and Y, which we now allow to take on values from a continuous domain). The full conditional distributions are $f_{X|Y}$ and $f_{Y|X}$. We begin by finding f_X, the marginal distribution of X:

$$\begin{aligned}
f_X(x) &= \int f_{X|Y}(x|y) f_Y(y) dy \\
f_Y(y) &= \int f_{Y|X}(y|x) f_X(x) dx \\
f_X(x) &= \int f_{X|Y}(x|y) \int f_{Y|X}(y|t) f_X(t) dt \, dy \\
&= \int \left[\int f_{X|Y}(x|y) f_{Y|X}(y|t) \, dy \right] f_X(t) \, dt \\
&= \int h(x,t) f_X(t) dt
\end{aligned}$$

The above defines an integral transformation for which the required marginal distribution, $f_X(x)$, is a fixed point. Denote the integral transformation by $T(g(x)) = \int h(x,t) g(t) dt$, where $h(x,t) = \int f_{X|Y}(x|y) f_{Y|X}(y|t) dy$; T simply transforms one integrable function, $g(x)$, into another, $T(g(x))$. Tanner and Wong [1987, Theorems 1 and 2] showed that, under mild conditions on the conditional probability densities (more specifically, on the transition function $h(x,t)$ that they induce), T is pseudocontractive in the L_1 norm ($\|f\|_{L_1} \equiv \int |f|$), and that functional iteration[1] converges from an arbitrary initial distribution (more accurately, from almost any initial distribution) to the unique distribution function that is a fixed point of the integral transformation. Thus, solving the integral fixed point equation allows the determination of the marginal distribution from the full conditionals. In the bivariate case the joint distribution can then be computed from (for example) $f_{XY} = f_{X|Y} f_Y$.

In solving such problems, it may be difficult to manipulate the functions analytically; indeed, the analytical forms may not be available at all. In these cases, it is necessary to resort to numerical methods in which the integrals are approximated through Monte Carlo sampling from the various involved distributions. With a large enough number of drawings, the distributions or any of their properties can be determined to arbitrary accuracy from the sample properties.

These ideas can be applied to find a fixed point distribution using functional iteration in the following general way. Generate a drawing x^0 from an initial (assumed) distribution of X, f_{X^0}.

[1] Called successive substitution or the substitution algorithm in the integral equations literature.

Next, using x^0, generate a drawing y^0 from the conditional distribution $f_{Y|X^0}(y|x^0)$. Finally, with y^0 and the conditional distribution $f_{X|Y^0}(x|y^0)$, generate a new drawing x^1 from distribution X^1, which is a better approximation to the true distribution of X. (This can be seen as an evaluation of the transformation T for one point.) Continuing in this way, the underlying distribution from which the drawings x^i are being generated will converge to the distribution f_X, and at convergence the drawings themselves will be coming from the desired marginal distribution. (Analogous statements obviously hold for the drawings y^i as well.) This procedure is known as *substitution sampling.*

If analytical forms are available for the conditional distributions, then the most efficient estimate of the marginal distribution $f_X(x)$ is obtained by generating a sample of values $y^j, j = 1, \ldots, m$ after the substitution sampling process has converged, and computing

$$\hat{f}(x) = \frac{1}{m} \sum_{j=1}^{m} f_{X|Y}(x|y^j),$$

a Monte Carlo integration that corresponds to the relationship $E_Y\left[f_{X|Y}\right] = \int f_{X|Y} f_Y(y)dy = f_X$ [Gelfand and Smith, 1990]. (The generated drawings of x^i are not involved.)

When more than two variables are involved, the above reasoning can be extended in various ways, depending on the availability of the different possible conditionals (e.g., $f_{X|(Y,Z)}$ or $f_{(X,Y)|Z}$); collectively these different extensions are all referred to as substitution sampling. If the full conditionals are used for each variable, the resulting method is known as *Gibbs sampling.* It was originally proposed by Geman and Geman [1984] in the context of image restoration, and was justified using reasoning based on Markov processes rather than the integral operator approach of Tanner and Wong (who recognized the Markovian property of successive iterates but did not further pursue the idea.)

It can be seen that the generated sequence of drawings x^i (and y^i) is a realization of a Markov chain (i.e., a discrete time Markov process, where "time" is in fact the iteration counter) with transition probabilities dictated by the conditional distributions. It is known from the theory of Markov processes that, if the one-step transition probabilities are all strictly positive, then the k-step transition probabilities converge to stationary (invariant) values as $k \to \infty$. The marginal probabilities determined above are then fixed points of the stationary transition function, and these fixed points can be obtained by iteratively applying the one-step probabilities to an arbitrary set of initial probabilities. These conclusions of Geman and Geman [1984] parallel those reached by Tanner and Wong regarding the solution of the fixed point equation from an operator theory approach. Gelfand and Smith [1990] highlighted the essential equivalence of the approaches, for example using results from Geman and Geman [1984] to strengthen results derived by Tanner and Wong [1987]. Because of this deep connection, substitution sampling, Gibbs sampling and a variety of related procedures are sometimes called Markov chain Monte Carlo methods [Neal, 1993].

Let the set of variables be $X = \{X_i, i = 1, \ldots, N\}$. The joint distribution of X is f_X and the marginal distribution of component X_i is f_{X_i}. A drawing from the distribution of X_i made in iteration k will be indicated X_i^k. The Gibbs sampling algorithm is then

1. start with arbitrary values $X_i^0, i = 1, \ldots, N$; set $k = 1$
2. generate X_1^k by sampling from $f(X_1|X_2^{k-1}, \ldots, X_N^{k-1})$;
   generate X_2^k by sampling from $f(X_2|X_1^k, X_3^{k-1}, \ldots, X_N^{k-1})$;

generate X_3^k by sampling from $f(X_3|X_1^k, X_2^k, X_4^{k-1}, \ldots, X_N^{k-1})$;

and so on up to the generation of X_N^k by sampling from $f(X_N|X_2^k, \ldots, X_{N-1}^k)$.

This constitutes one *cycle* over the variables. Set $k = k + 1$.

3. carry out m such cycles, generating samples $(X_1^m, \ldots, X_N^m)$.

Geman and Geman [1984] showed that, under the conditions stated above, as $m \to \infty$, the sample joint frequency distribution of the sample $(X_1^m, \ldots, X_N^m)$ converges in distribution to the true joint distribution of the variables. From this it follows that portions of the sample can be extracted to represent any subset of the full set of variables; in particular, the sample distribution of X_i converges in distribution to the marginal distribution of X_i. They also proved the ergodicity of the sample generation process, establishing in particular that, for any measurable function $T(X_1, \ldots, X_N)$ whose expectation exists,

$$\lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} T(X_1^k, \ldots, X_N^k) \xrightarrow{a.s.} E(T(X_1, \ldots, X_N))$$

In practice, values generated during an initial "warm up" phase are discarded since they are drawn from distributions that have not yet converged [Geweke, 1996]. When convergence has been achieved, the generated values will be drawn with the correct probabilities from the true distribution; these values can be used to estimate the joint or marginal distributions, or any properties of these distributions. However, successive drawings may exhibit high correlations, which can be a shortcoming in some applications. A number of schemes have been proposed to circumvent this problem, including generating multiple independent Markov chains from which successive samplings are taken, and generating a single very long chain but only using every kth sample to reduce the correlation. For the latter approach, an analysis of the generated sequence's autocorrelation function (coefficient of correlation between elements as a function of their separation in the sequence) can suggest a suitable value for k.

When other sets of conditionals (for example, some full and some partial) are available, alternative sampling schemes are possible. These can be shown [Gelfand and Smith, 1990] to be essentially equivalent to the Gibbs sampler, in the sense that the alternative procedures ultimately generate a sample from the full conditional distributions. These procedures do not necessarily draw from the full conditionals in the sequential order used by the Gibbs sampler; however, if a sampling scheme draws from each conditional distribution infinitely often in the course of an infinite sequence of draws, the scheme can be shown to be convergent. In some cases modest improvements in convergence properties have been reported by exploiting the ready availability of particular partial conditionals.

4.1.1.2 Applications

Although Gibbs sampling is used in a wide variety of fields, the only published examples of its application in transportation network modeling appear to be the analyses by Hazelton et al. [1996] and Hazelton [1998] of the static stochastic user equilibrium (SUE) problem.

The static SUE problem is characterized by:

1. a conventional representation of a transportation network and its supply characteristics (link performance functions that depend on link flows);

2. a static network loading process that instantaneously propagates a path flow over all the links in the path, such that a link's flow is the accumulation of the flows on paths that go through it. Link costs are a function of the link flows.

3. a multipath routing model that predicts the split of flow on each available alternative path as a function of the costs of those paths. These path costs are the sum of the costs of the links that make up the path; and

4. the assumption that OD flow divides among the available paths exactly in accordance with the path splits.

Stochastic user equilibrium is obtained when the path costs that formed the basis for the path flow calculation (by 3 and 4) are the same as those that actually result when the path flows are loaded onto the network (by 1 and 2); or alternatively when the loading of a set of path flows on the network (1 and 2) results in path costs that lead (via 3 and 4) to the prediction of an identical set of flows. Either of these is obviously a fixed point condition on a composite supply-demand or demand-supply map [Cantarella, 1997]. It can be seen that, despite its name, SUE is actually a *deterministic* equilibrium concept.

Hazelton analyzed the consequences of modifying assumption (4) above. Specifically, he considered the case where total OD flow is disaggregated into integer flow units, each of which independently evaluates the path splits in (3), and selects from among the paths in accordance with the resulting probabilities. The total flow on paths is then the aggregate of the individual path choices. Under this approach, path flows are multinomially distributed random variables and, consequently, link flows and costs and path costs are random as well.

Hazelton's analysis considered a single period within which all travelers interact. In this context, the path costs taken into account by a traveler in evaluating the choice probabilities logically depend on the contemporaneous path choices made by the other travelers in the network. Through the loading process, these choices determine link volumes and costs, and so lead to the path costs and path choice probabilities which the traveler himself considers. This sequence of steps expresses, in effect, the conditional distribution of each traveler's path choice probabilities, given the path choices made by all other travelers; Hazelton calls this conditional stochastic user behavior. As long as no route is assigned zero probability by any traveler, the positivity requirement mentioned above is verified, so the individual conditional probabilities combine to determine the unique joint probability distribution of all travelers' path choice decisions; this Hazelton calls a conditional stochastic user equilibrium.

As indicated above, Hazelton proposed Gibbs sampling to determine the characteristics of the resulting distribution. Suppose the travelers are numbered $1, \ldots, N$. The method consists of assuming initial path choices by travelers $2, \ldots, N$; loading these travelers on their chosen paths and determining the resulting link and path costs; evaluating the path choice probabilities of traveler 1 based on those path costs; and randomly choosing a path for traveler 1 in accordance with the path choice probabilities. The method then moves on to traveler 2: traveler 1 is loaded on the just-chosen path, along with travelers $3, \ldots, N$ on their prior choices. The path costs resulting from this loading are determined, then traveler 2 evaluates the path probabilities based on their costs and chooses one in accordance with the probabilities. Traveler 3 is next, using the recent path choices of travelers 1 and 2, and the prior choices of travelers $4, \ldots, N$; and so on until all travelers have been processed. This pass over all travelers constitutes one iteration of the method. Iterations

are continued until the process is judged to have converged, at which point the generated choices can be used to determine the distributions of flows, costs, or other quantities of interest. Although in some ways the method resembles a day-to-day assignment model, the differences between the two are clear:

- in day-to-day models the iterations represent successive days (or other time period). In Hazelton's approach, the iterations are simply algorithmic steps towards the determination of the required probability distribution, with no particular temporal interpretation; the method is fundamentally atemporal.

- in day-to-day models, travelers all choose routes based on costs and/or flows from previous iterations (days); the travelers are then loaded and the entire set of costs is updated. In Hazelton's Gibbs sampling approach, costs are updated after each individual traveler is loaded, and each traveler chooses a route based on the latest set of costs.

The requirement that costs be updated after every individual traveler has been loaded might seem excessive; after all, the effect of one trip on link and path costs may not be significant, particularly in a heavy traffic situation. However, the consequences of approximating the process by updating the loadings and costs in blocks, where a block is some number of trips between 1 and the total number in the network, does not appear to have been systematically investigated. Hazelton presents numerical examples that show significant differences between the flow distributions that result in a simple network from updating after each trip has been loaded (Gibbs sampling) and updating after all trips have been loaded (a day-to-day model). He also provides a heavy traffic result, showing that if flow units are subdivided into m independently assignable flow fractions, each using $1/m$ unit of capacity, then as $m \to \infty$ the CSUE link flow pattern distribution converges in probability to the standard (deterministic) SUE link flow pattern.

4.1.2 The TRANSIMS rerouting algorithm

TRANSIMS is a micro-simulation system for transportation planning that has been designed and implemented by the Los Alamos National Laboratory. It represents at the level of individual tripmakers a very broad range of factors involved in determining traffic levels and conditions, including daily activity pattern planning, trip-specific decisions such as those concerning departure time and path, and vehicle propagation through the network. Because of the complexity and experimental nature of a full model system of this type, TRANSIMS also provides a simplified operational mode in which time-dependent OD demands are directly provided by the users (rather than derived from higher-level models), and the system dynamically assigns these to the network.

Assignment is based on an iterative *route replanning* and loading procedure. Each traveler is allocated to an OD path at all times. In a given iteration, TRANSIMS selects some portion of the travelers and re-assesses their routing options, assuming that travelers choose dynamic minimum time paths. The dynamic network loader then loads all vehicles (those who were selected to reconsider their path choice as well as those assumed to remain on their prior path), using a cellular automata approach to move vehicles along their paths. Statistics on link entry and exit times are collected as the vehicles are moved; the statistics are used to compute average time-dependent traversal times by link, and these are used for route replanning in the next iteration.

Although route choice is assumed to be based on a deterministic user optimal minimum time criterion, the actual determination of link times in done in a way that introduces a stochastic element into the choice. Before computing the minimum time path of each traveler selected for route replanning, TRANSIMS multiplies the prior iteration's average time-dependent link times by a random number drawn from a uniform $\mathcal{U}[0.7, 1.3]$ variate, and the traveler's dynamic minimum path is computed with respect to these perturbed time values. According to Nagel and Barrett [1997], this approach was adopted in order to damp strong oscillations between routes that were observed when unmodified link times were used for the minimum path calculation. (This method bears a very strong resemblance to one proposed by Burrell [1968] to effect multipath routing in static traffic networks with deterministic user optimal path choice.) In any case, the end result is to introduce a stochastic element into traveler route choice decisions. The cellular automata network loader also has stochastic elements.

It follows that TRANSIMS is based on a stochastic routing-loading map so, as discussed in Chapter 3, its outputs in a particular run will in general be random trajectories sampled from a stochastic process. Some evidence for this, in the form of non-Gaussian noise affecting the output trajectories, is presented in Nagel [1997]; Nagel [personal communication, 1998] has in fact observed runs with identical input data but different random number realizations leading in some cases to gridlock and in others to acceptable traffic conditions.

This model could be solved using Gibbs sampling to determine the stochastic process outputs resulting from the path choice and cellular automata models. However, rather than attempting to determine the full stochastic process, the TRANSIMS route replanning method is designed to compute a single trajectory. TRANSIMS first determines the fraction of trips that will replan routes. This fraction is generally a decreasing step function: typical numbers are three iterations with 20% of the trips replanning, followed by three iterations with 10%, three more with 5% and the remainder with 2% and 1%. The specific trips to be replanned might be chosen randomly, or with a probability that increases with the number of iterations since the last replanning [Rickert and Nagel, 1998].

TRANSIMS documentation presents this procedure as a reasonable-seeming heuristic that can be made to give realistic results when used with the other components of the TRANSIMS system. It strongly resembles methods, such as simulated annealing, that are used in statistical physics to determine the most probable states of large systems having stochastic interactions between individual components. Rather than speaking of "convergence", the documentation refers to the objective of the iterative process as "relaxation": a state in which, according to the solution logic of the route replanning algorithm, the system has no tendency to change state in subsequent iterations. Kelly and Nagel [1997] discuss a variety of criteria by which the relaxation of the iterative process can be judged.

4.1.3 Fictitious play methods

Fictitious play is an iterative method for solving zero-sum games, in which each player keeps track of the history of moves by all the other players, and in each iteration chooses the pure strategy that is optimal against the mixed strategies defined by the sample frequency distributions of the opponents' moves up to that time. After a sufficient number of iterations, the sample frequency of a player's choice of the different possible pure strategies is taken as an approximation to the

player's optimal mixed strategy.

The fictitious play method was originally proposed by Brown [1951] as a solution method for solving two-person zero-sum games. The validity of this application was shown by Robinson [1951] shortly thereafter. However, it is not a generally valid game solution procedure. Various particular types of game have been identified for which the method is valid; such games are said to possess the *fictitious play property.*

Monderer and Shapley [1996] showed that, in games where all players have identical payoff functions,[2] the frequencies with which players choose each strategy eventually stabilize, and that this limiting distribution is a mixed strategy Nash equilibrium of the game. (Note, however, that a Nash equilibrium is a local although not necessarily a global minimizer of the game's payoff function.)

Despite a certain similarity of appearance, fictitious play and Gibbs sampling methods are fundamentally different. In fictitious play each player generates a pure strategy (i.e., makes a deterministic choice) based on the mixed strategies (sample probability distribution of choices) effectively pursued by other players up to that point, whereas in Gibbs sampling each player generates a mixed strategy (i.e., determines the probability distribution of choices) based on the pure strategies (deterministic choices) currently followed by other players.

Garcia et al. [2000] have applied fictitious play methods to the problem of computing system optimal routings in dynamic networks. In this problem the objective is to route traffic in such a way that the total trip time of vehicles in the network is minimized. The problem can be expressed as a game in which each player is an individual vehicle whose possible strategies are the available route choices. The payoff is calculated using a network loading map that moves the vehicles along their chosen routes and determines the resulting trip times. Each vehicle chooses a route so as to minimize the total trip time of all vehicles in the network. This clearly defines a game with identical interests, and so the above convergence result applies.

The authors took a number of shortcuts in implementing the fictitious play method for this problem, however. Principal among these is that they do not update the sample frequencies and times after each individual vehicle's route choice, but rather after all vehicles have made their choices. The loading that is then done includes all vehicles, and the resulting times are used by all vehicles when making their routing decisions in the next iteration.

With these shortcuts the processing steps in the authors' implementation strongly resemble those that are applied in day-to-day assignment models (except that day-to-day models usually assume some form of user optimal routing). As was seen in Section 4.1.1, when similar steps are carried out in the context of user optimum assignment, the resulting distribution can differ significantly from the correct distribution as obtained by Gibbs sampling. Here the authors did not examine the effect of the shortcuts on the solutions computed by their method.

[2]Or, more generally, in games that are equivalent in terms of mixed strategy best responses to a game with identical payoff functions. These are called *games with identical interests.*

4.2 Stochastic approximation methods

4.2.1 Introduction

Stochastic approximation is a family of methods for addressing numerical problems such as root finding, unconstrained or constrained optimization, and system identification and control, when evaluation of the function of interest returns a result that is affected by noise. The methods are non-parametric: they do not require knowledge of the analytical forms of the function or the noise distribution. However, they assume that one can select the specific points at which the function is to be evaluated during the solution procedure (these are sometimes called *design points*).

Equation-solving algorithms are the only stochastic approximation methods of interest in this thesis. For these problems, it is generally assumed that noise values are independent realizations from a zero-mean distribution. Results generated by the procedure in prior iterations are used to compute the design point for the current iteration, and to estimate the desired equation root.

The stochastic approximation procedures considered here are recursive averaging, of which the method of successive averages (MSA) [Sheffi and Powell, 1982] is a typical example; and a two-pass method called iterate averaging. Both the DYNASMART [Mahmassani et al., 1994] and DynaMIT [Ben-Akiva et al., 1997] mesoscopic simulation models use solution techniques similar to the MSA. It appears, however, that iterate averaging methods have not yet been applied to transportation network problems.

Although there exist stochastic approximation methods which can handle constraints on the feasible solution set, all problems considered here are effectively unconstrained: the feasible set is convex, the procedures used to generate trial ("auxiliary") solutions guarantee their feasibility, and new solution estimates are generated as convex combinations of prior (feasible) estimates and the auxiliary solutions, and so fall within the feasible set.

These methods only apply to problems defined on continuous value domains. Consequently, they are only applicable to guidance generation based on the composite link condition and path flow maps, or to the composite message map when the guidance messages can be considered as continuous. (The discreteness of the time domain is not relevant in this discussion.)

References on stochastic approximation include Wasan [1969], for good coverage of developments through the 1960s; Ruppert [1991], which reviews work through the late 1980s; and Kushner and Yin [1997], which provides a fairly comprehensive treatment of the subject including many developments, such as iterate averaging and distributed methods, through the mid-1990s. Stochastic approximation is an important family of numerical methods that continues to be the subject of considerable research activity fifty years after it was originally proposed.

4.2.2 Recursive averaging

4.2.2.1 Mathematical background

With $\Re$ the field of real numbers and $\Re^n$ designating n-dimensional Euclidean space, suppose that we are interested in finding the root x^* of a function $T(\cdot) : \Re^n \to \Re$, so that $T(x^*) = 0$, but that evaluations of T return a value that is affected by noise: the result is $\tilde{T}(x) = T(x) + \varepsilon$ rather than $T(x)$, where ε is a random zero-mean noise term.

Recursive averaging involves the basic step

$$x^{k+1} = x^k + \alpha^k \tilde{T}(x^k);\ x^0 \in \Re^n,\ k = 1,2\ldots$$

(where the sequence α^k is chosen so that $\sum \alpha^k$ diverges but $\sum (\alpha^k)^2$ converges) to solve for a root of T, despite the noisy evaluations of $\tilde{T}$.[3] It is easily seen that applying recursive averaging to the function $(T(x) - x)$ results in a method for finding fixed points of $T(\cdot)$ despite the noise. In this case the basic step is

$$x^{k+1} = x^k + \alpha^k(\tilde{T}(x^k) - x^k);\ x^0 \in \Re^n,\ k = 1,2\ldots,$$

which is a small step size method of the type discussed in Section 4.3.1.

Robbins and Monro [1951] considered functions T defined on $\Re$ and proved that, under mild conditions, recursive averaging would converge in probability to a root of T. Blum [1954] extended this result, under somewhat more stringent conditions, to multi-dimensional functions and almost sure (a.s.) convergence. The methods of proof used in the original papers have since been generalized [Robbins and Siegmund, 1971] and the convergence conditions have been considerably relaxed. Conditions for a.s. convergence include a requirement for the existence of a stochastic potential function that is reduced "on average" as the iterations proceed, and restrictions on the step sizes α^k. Typically, the step sizes are required to be chosen as $\alpha^k = ak^{-\beta}$, with $\frac{1}{2} < \beta \leq 1$; these are sufficient to allow the algorithm to reach a solution from an arbitrary starting point, while ensuring that the variance of the successive iterates decreases to zero so that the sequence converges to a single value. Conditions for weak convergence are even less stringent [Kushner, 1977].

The initial work by Robbins and Monro [1951] on root finding and by Kiefer and Wolfowitz [1952] on function maximization led to extensions of the theory in a number of directions. Chung [1954], Sacks [1958] and Fabian [1968] showed, under increasingly more general conditions, that in the one-dimensional problem the normalized distribution of $(x^k - x^*)$ converges asymptotically to a normal distribution with zero mean and a variance matrix depending, in part, on the gradient of T evaluated at x^*. For example, with Gaussian disturbances, the sequence $k^{-\beta/2}(x^k - x^*)$ is asymptotically normal if $\beta < 1$ or alternatively if $\beta = 1$ and $aT'(x^*) > \frac{1}{2}$. A noteworthy consequence of these results is that minimal asymptotic variance is attained for the step size choice $\alpha^k = (kT'(x^*))^{-1}$. Similar results are available for the multidimensional case (see Nevelson and Hasminskii [1973], for example).

Of course, immediate application of this result to determine optimal step sizes is usually not possible since the quantity $T'(x^*)$ or $\nabla T(x^*)$ will not be known in general. However, Venter [1967] proposed an adaptive procedure that uses information generated by the function evaluations in different iterations to estimate the gradient information and to compute improved step sizes accordingly; under some restrictions, this procedure can be shown to attain the minimal asymptotic variance. Along similar lines, Lai and Robbins [1979] proposed a procedure to estimate the gradient using the result of the most recent function evaluation, while Frees and Ruppert [1990] examine a more general procedure that performs a least squares fit of all prior data to estimate the root of $T(\cdot)$ and its gradient there.

In further developments, Abdelhamid [1973] and Anbar [1973] showed that in situations of non-Gaussian noise, transformations of the observations could, in effect, modify the problem so that

[3]The term *recursive averaging* presumably derives from the fact that the iterate in step $k+1$ is expressed as a combination ("average") of data generated in step k and the iterate in step k, and so is recursively defined.

the conclusions on asymptotic normality and optimal step sizes continue to hold for the modified problem.

Chung [1954] and Fabian [1973] also showed that Robbins-Monro type recursive averaging methods are asymptotically locally minimax procedures for determining equation roots in the presence of noise.

Subsequent investigations into the stochastic approximation problem produced more general analyses and considerably extended the applicability of the method. Ljung [1977, 1978] showed that the convergence properties of many recursive algorithms such as stochastic approximation could be determined through study of an associated deterministic ordinary differential equation; this approach has had wide application in a number of fields. At roughly the same time, Kushner [1977] introduced weak convergence methods that generalized and extended many prior results and proof methods in the area; these methods are one of the centerpieces of his books [Kushner and Clark, 1978; Kushner and Yin, 1997].

In a different line of analysis, Kersting [1977] and Ruppert [1982] showed that, under broad conditions, analysis of stochastic approximation methods could be reduced to study of the properties of weighted averages of the disturbances, and this too led to a simplification and generalization of many prior analyses and results.

4.2.2.2 Applications

The classical application of recursive averaging to a transportation problem is by Sheffi and Powell [1982], who used it to minimize a twice continuously differentiable functional (the objective function of the unconstrained convex optimization formulation of the SUE problem). In their application, the function $\Upsilon(\cdot)$ provided a noisy descent direction while the stochastic potential function was provided by the objective function itself. They used step sizes $\alpha^k = 1/k$, which lead to a design point estimate for iteration k that is the numerical average of the function evaluation results in the preceding iterations; for this reason, their algorithm is called the method of successive averages (MSA). Cantarella [1997] also proved the applicability of the MSA (under certain conditions) to solve his two general fixed point formulations of the static stochastic user equilibrium problem.

As noted above, the MSA has also become a popular heuristic for solving dynamic traffic network models, despite the lack of rigorous proof of its applicability to these problems. In its favor, it seems to give reasonable results, it avoids (potentially expensive) step size calculations, and it works directly with model outputs without requiring derivative calculations or other transformations. Against this must be set the lack of rigorous convergence proof for dynamic networks, and its empirically observed convergence properties: generally effective performance in the initial iterations followed by a pronounced "tail" effect, resulting in overall slow convergence.

The transportation community does not seem to be generally familiar with the research on stochastic approximation that followed the work of Robbins and Monro [1951] and of Blum [1954]. While the asymptotically optimal methods mentioned above have remained largely untried in transportation applications, considerable effort has been spent trying to find heuristic procedures that improve the convergence rate of recursive averaging without requiring overly expensive calculations. Some researchers have achieved noteworthy improvements in the MSA's convergence speed with such heuristics but, because the conditions of applicability and convergence properties of these methods have not been rigorously proved, questions remain about the validity and reliability of their use.

As an example, Cascetta and Postorino [1998] observed that in the MSA an iteration's estimate is affected by the results from all prior iterations, including those from early iterations that are presumably far from the solution. Furthermore, later iterations, which are presumably closer to the solution, receive smaller weights when computing a new estimate. Accordingly, Cascetta and Postorino propose a heuristic method that from time to time restarts the MSA (i.e., resets the iteration counter k back to 1) using the last computed value as the new initial point. By restarting, the direct influence of earlier iterations is eliminated, and larger step sizes are applied to subsequent iterations. The frequency with which such restarts are carried out decreases as the number of iterations increases, via a user-specified "refreshing modulus" n. The first restart is done after n iterations, the second after $2n$ iterations following the first restart, the third after $3n$ iterations following the second restart, and so on. The authors report that, compared to the standard MSA, this method significantly reduces the number of iterations needed to achieve a given convergence level. Moreover, each increase in n from 2 to around 7 generally improves the convergence speed, while increases in the modulus beyond 7 generally result in slower convergence (but that remains nonetheless superior to the MSA).

4.2.3 Iterate averaging

4.2.3.1 Mathematical background

It is characteristic of recursive averaging that the design points generated by successive iterations of the algorithm are also taken to be the successive estimates of the equation solution. This was noted by Frees and Ruppert [1990], who pointed out the advantages of using one method to select the design points, and a different method to estimate the solution. Use of methods adapted to each purpose could allow, for example, a more aggressive exploration of the feasible space and a more effective exploitation of the results generated during that exploration to estimate a solution. One family of methods that exploits this idea is called *iterate averaging*.

Let $x^{k+1} = x^k + \alpha^k \tilde{T}(x^k)$ be a recursive averaging process involving a noisy function $\tilde{T}(\cdot)$. Suppose that the process converges to a limit x^*. In iterate averaging one also computes a running unweighted average of the iterates x^k, say $\xi^k = \sum_{i=1}^{k} x^i/k$. The sequence ξ^k also converges to the limit x^*. It is known that if $\alpha^k = O(1/k)$ then the convergence properties of ξ^k are no better than those of x^k; in other words, applying iterate averaging to an MSA-type process is of no benefit. Polyak [1990] and Polyak and Juditsky [1992] showed, however, that if $\alpha^k \to 0$ "slower than" $O(1/k)$, then the sequence ξ^k converges to x^* at an optimum rate, in a suitably-defined sense. Specifically, suppose that $\alpha^k/\alpha^{k+1} = 1 + o(k)$. Then $k^{1/2}(\xi^k - x^*)$ converges in distribution to a Gaussian random variable with zero mean and minimum asymptotic variance over a large class of averaging methods; and the convergence does not depend on the particular choice of step sizes α^k, provided that the step size condition mentioned above is met.

Remarkably, therefore, this simple procedure theoretically equals or surpasses the asymptotic performance of any other averaging method, including methods that derive and exploit gradient information to determine optimal step sizes. As remarked by Kushner and Yang [1993, p. 1046], "In the past, a great deal of attention was given to the problem of choosing optimal sequences [of step sizes] via 'adaptive' and generally unreliable means. The importance of this problem is now much reduced." The procedure is sometimes called Polyak averaging. (Ruppert [1988] proposed a one-dimensional version of the method for linear models; see also Kushner and Yang

[1993, Chapter 11].) This result holds provided that recursive averaging (for example, the MSA) is itself applicable to the problem being solved. Thus, in the context of stochastic dynamic user equilibrium or guidance generation problems, it is no less valid than applying the MSA or other recursive averaging method to solve for a fixed point of the problem map.[4]

Note that the calculation of ξ^k is done "off-line", in the sense that iteration $k+1$ of the recursive averaging process makes use of x^k and not ξ^k. However, after performing n steps of the recursive averaging process, the value of ξ^n will provide an optimal estimate of x^*.

A somewhat different averaging approach was proposed by Bather [1989]. Here, the design point in iteration $k+1$ is derived from a combination of the average of previous design points with the average of the previous observations:

$$x^{k+1} = \bar{x}^k - k\alpha^k \bar{T}^k$$

where $\bar{x}^k$ is the average of the design points selected in previous iterations, and $\bar{T}^k = \frac{1}{k}\sum_{i=1}^{k} \tilde{T}(x^i)$ is the average of the corresponding function evaluation results. As in the Polyak method, the observations are taken at the design points x^k while the root is estimated by $\bar{x}^{k+1}$. Since $(k+1)\bar{x}^{k+1} = k\bar{x}^k + x^{k+1}$, Bather's recursion can also be expressed as $\bar{x}^{k+1} = \bar{x}^k + \frac{k}{k+1}\alpha^k \bar{T}^k$, so it resembles the basic Robbins-Monro procedure with the design points replaced by their averages.

Schwabe and Walk [1996] have shown that Bather's algorithm has the same asymptotically optimal properties as does Polyak's method (i.e., it converges to a solution with minimal asymptotic variance). However, they also show that the method is theoretically less sensitive to poor starting points. Specifically, in the Polyak procedure (and in the Robbins-Monro method with optimal step lengths), the influence of starting values decays like $\frac{1}{n}$ (with n the number of iterates) whereas in the Bather procedure it decays at a rate substantially faster than this. They assert that, for small to moderate sample sizes, this property should lead to better performance of the algorithm although no computational results have been adduced to support this claim.

Kushner and Yang [1995] considered "on-line" iterate averaging methods, in which the results of the iterative averaging procedure are suitably fed back to and incorporated in the primary iteration. They focus on linear models with constant step sizes and show that, in many cases, these methods too offer advantages over recursive averaging. However, they have not extended these results to the non-linear case.

4.2.3.2 Applications of iterate averaging

Although iterate averaging methods have been applied, generally with good success, in areas such as signal processing, the author is not aware of any transportation applications of these methods.

4.3 Deterministic methods

This section surveys methods for solving deterministic fixed point problems that arise in guidance generation.

[4] Iterative averaging could also be applied to the static SUE problem, and holds promise for accelerating the well-known slow convergence of the MSA in this application.

Nonlinear equation systems for which fixed points must be found in guidance applications are typically very large.[5] In some cases, analytical versions of the involved maps may be available, from which derivatives can be computed as required. In many cases, however, evaluation of the maps and their derivatives may only be possible using simulation. The amount of computation required in these cases to compute numerical derivatives is so great as to preclude the use of classical derivative-based methods; consequently, these methods were not examined here.

Two families of methods are considered: one based on functional iteration, and the other on simplicial decomposition. It will be seen that neither is generally applicable to route guidance problems, although each has interesting properties that justify an examination in this section.

4.3.1 Functional iteration and related methods

4.3.1.1 Mathematical background

Functional iteration, also known as the method of successive approximations, is a well-known method for computing the fixed point of a contractive mapping.

A mapping $T : X \mapsto X \subseteq \Re^n$ is said to be *Lipschitz continuous* on a set $X_0 \subseteq X$ if there is an $\alpha > 0$ such that $\|T(x^1) - T(x^2)\| \leq \alpha\|x^1 - x^2\|, \forall x^1, x^2 \in X_0$, where $\|\cdot\|$ is a norm. T is *contractive* if $\alpha < 1$. (The contractive property is norm-dependent, so that a mapping may be contractive with respect to one norm but not another.) T is *pseudocontractive* if $\|T(x) - x^*\| \leq \alpha\|x - x^*\|$ where $x^* \in X$ is a fixed point of T and again $\alpha < 1$. T is *nonexpansive* on X_0 if $\|T(x^1) - T(x^2)\| \leq \|T(x^1) - T(x^2)\|, \forall x^1, x^2 \in X_0$, and *strictly nonexpansive* on X_0 if strict inequality holds whenever $x^1 \neq x^2$.

Functional iteration begins at an arbitrary point $x^0 \in X_0$ and repeatedly applies the iterative step $x^{k+1} = T(x^k)$.

The *contraction mapping theorem* provides the basic result relating contractive mappings and fixed points: if T is contractive on a closed set $X_0 \subseteq \Re^n$, then T has a unique fixed point in X_0. The proof proceeds by establishing that the iterates $x^{k+1} = T(x^k), k = 0, 1, \ldots$ form a Cauchy sequence in X_0 and invoking the completeness of $\Re^n$. (In fact the theorem and method of proof carry over without change to an arbitrary Banach space.) Functional iteration is relatively efficient in this case since, when the map is contractive, the iterate sequence converges geometrically to the fixed point. Similar results hold for pseudocontractive maps.

If a map is not contractive, there is no guarantee that functional iteration will find a fixed point (if, indeed, one exists). However, consideration of the method suggests several lines of development towards potentially more generally applicable methods.

The first approach is simply to take smaller steps. Rather than jumping from x^k to $T(x^K)$ in a single step

$$x^{k+1} = T(x^k) = x^k + (T(x^k) - x^k),$$

a small step size approach only moves part of the distance from x^k to $T(x^k)$:

$$x^{k+1} = x^k + \alpha^k(T(x^k) - x^k),$$

where $0 < \alpha^k < 1$ is predetermined. Properties of this method in a quasi-deterministic problem setting were discussed in Section 4.2.2 above.

[5] Roughly speaking, the number of variables equals the product of the number of time steps times the cardinality of the network components (such as links or destination-specific subpaths involved in the formulation.

A second approach uses the outputs of the functional iteration method but averages them. Baillon [1975] showed that averaging functional iterates results in a method that can find fixed points of non-expansive maps. Specifically, if $T(\cdot) : C \mapsto C$ is defined on a convex, closed and bounded subset C of a real Hilbert space and is non-expansive, and if, for $x \in C$

$$S_n(x) = \frac{x + T(x) + \cdots + T^{n-1}(x)}{n}$$

then $S_n(x)$ converges weakly to a fixed point of T.

A third approach also takes smaller steps, but explicitly computes the step size; this is known as adaptive step size determination. If the scheme for determining the step sizes is effective, then the total computational effort to convergence (the average effort per iteration to determine the step sizes times the required number of iterations) will be less than the effort to convergence taken by a method with predetermined step sizes.

Magnanti and Perakis [1997a,b] suggest a scheme to determine the step sizes α^k by iteratively minimizing a potential function along the line defined by the current iterate x^k and its image under the map T. Specifically, if $P(x)$ is such a potential function, they propose to compute α^{k+1} via the following one-dimensional auxiliary minimization problem:

$$\alpha^{k+1} = \arg\min_{a \in \mathcal{S}} P(x^k(a))$$

where the function $x^k(a) = x^k + a(T(x^k) - x^k)$ and $\mathcal{S} \subseteq \Re$ is some step size search set. The auxiliary minimization may be carried out using either exact or inexact (e.g., Armijo-type) line search methods.

Magnanti and Perakis identify and investigate a variety of potential functions for the fixed point problem. They prove that the use of specific potential functions with particular classes of mappings leads to a convergent algorithm for finding the fixed point of those mappings, and show that the theoretical rate of convergence can in some cases be faster than averaging with predetermined step sizes. (Note that the theoretical rate of convergence does not take account of the effort required to determine the step size, but only of the progress towards the fixed point made in each iteration, and so many not be an accurate indication of the total computational effort required by an algorithm to approximate a fixed point to a given degree of accuracy.)

Possible potential functions are often suggested by intuitive considerations. One obvious approach, for example, is to define the potential in terms of the distance between an iterate and its image under the map T:

$$P1(x^k(a)) = \|x^k(a) - T(x^k(a))\|^2$$

Clearly if x^* is a fixed point of T then $P1(x^*) = 0$ and conversely. Magnanti and Perakis show that use of this potential with contractive mappings leads to a convergent algorithm with a better theoretical rate of convergence than functional iteration.

Another approach is based on the general idea that, to better explore the solution space, iterate x^{k+1} should be forced away from iterate x^k unless the iterative process is near a fixed point. This leads naturally to the following potential function:

$$P2(x^k(a)) = \|x^k(a) - T(x^k(a))\|^2 - \beta\|x^k(a) - x^k(0))\|^2$$

where $\beta > 0$ is a "repelling" factor that tends to keep x^{k+1} away from $x^k(0) \equiv x^k$.

A natural extension of the preceding idea is to force iterate x^{k+1} away from both x^k and $T(x^k)$ unless the iterative process is near a fixed point. The resulting potential function is then

$$P3(x^k(a)) = \|x^k(a) - T(x^k(a))\|^2 - \beta\|x^k(a) - x^k(0))\|\|x^k(a) - x^k(1))\|$$

where $x^k(1) \equiv T(x^k)$.

Both P2 and P3 have attractive convergence properties when applied to nonexpansive maps and, in general, these methods can apply to maps that satisfy even weaker conditions than non-expansiveness. Magnanti and Perakis derive sufficient conditions, depending on both the mapping and the weights, for the convergence of general processes of this type.

As mentioned above, the auxiliary minimization problem that determines the step size can be solved using either exact or inexact line search methods. The justification for use of inexact methods in this case is very similar to the justification when such methods are applied in mathematical programming problems: since at each iteration we are dealing with a subproblem that only is a local approximation to the actual problem, there is little point in devoting significant effort to compute a precise solution to the subproblem.

Magnanti and Perakis propose an Armijo-type inexact line search rule for the auxiliary minimization problems that, for the combinations of problem and potential functions considered above, results in generally convergent algorithms.

4.3.1.2 Applications

Functional iteration One of the earliest methods applied to the static traffic assignment problem, the *iterated all-or-nothing assignment heuristic* [Patriksson, 1994, p. 21], repeated the basic steps of (i) finding minimum time OD paths, (ii) loading all traffic from each origin to each destination on the minimum time path and (iii) updating link times based on the loading. This procedure can be viewed as functional iteration involving a composite routing (selecting the minimum time path) and network loading (putting traffic on paths and updating link costs) map. It was quickly observed, however, that the method did not generally converge, and more satisfactory assignment methods were eventually developed. This behavior is not surprising, considering that there is no guarantee that the composite map is a contraction. In fact there has been little work reported in the literature investigating the Lipschitz properties of composite supply-demand maps. The question is still of some interest, since it is possible that some generally non-contractive maps may be contractive (or pseudocontractive) on subsets of their domain near a fixed point and thus allow iterated assignment to be applied there. This approach would be attractive because it would require no extraneous calculations and would converge geometrically.

In the area of dynamic network models, Kaufman et al. [1991], discussed in Section 3.4.1, applied functional iteration as the method to compute a fixed point of a composite routing-assignment map for a user optimal (based on minimizing actual travel times) dynamic traffic assignment problem (SAVaNT). The method iterates between determining time-dependent minimum time paths and dynamically loading traffic on these paths to obtain new link times; the process stops when either cycling over paths is detected or link times have approximately converged. As indicated, the authors found that functional iteration consistently cycled. Since SAVaNT essentially utilizes a dynamic generalization of the iterated all-or-nothing assignment heuristic, the non-convergence is again not surprising.

Pursuing this work, Wunderlich [1994] developed a heuristic called SAVaNT-CNV, which avoids cycling by applying a time bias to the dynamic link traversal times computed from simulator outputs: a simulated vehicle link traversal time corresponding to link entry in simulation time period t, for example, is intentionally imputed to time period $t + n, n \neq 0$. Time-dependent minimum paths are then computed from these biased times and traffic is loaded on the computed paths as before. In this method, convergence is assured for sufficiently large values of n.[6] Wunderlich varied the magnitude of the bias so as to reduce guidance inaccuracies while still ensuring the convergence of the method. This can be seen as a way of modifying the guidance mapping to make it contractive. Note that, in general, the method does not compute consistent guidance since the conditions it predicts, and from which its guidance is generated, are likely to be systematically different from those which would actually be experienced by drivers. In other words, although SAVaNT-CNV converges, the values to which it converges are not the solution to the original problem.

Baillon's method To the best of the author's knowledge, Baillon's method has not been used in transportation applications. There is in fact no proof that the composite maps typical in network analysis are non-expansive over all or part of their domains, so Baillon's method cannot be expected to give correct results in such applications.

Application of the method to a composite $S \circ D$ network equilibrium problem would begin with an arbitrary set of (feasible) link times C^0. The composite map would be applied by computing the path splits resulting from the link times and then loading the OD demands in accordance with the splits to obtain a new set of link times C^1, and so on. The calculation in step k is simply $C^k = S \circ D(C^{k-1})$, just as in functional iteration. However, unlike functional iteration, the estimate of the equilibrium times after n steps is $C^* = \frac{C^0 + C^1 + \cdots + C^n}{n+1}$.

This is somewhat similar to some early heuristic capacity restraint procedures that performed a series of loading and time updating steps, then averaged the results. It seems likely, however, that these early procedures, in fairly wide use[7] before the development of convergent equilibration algorithms, were developed without reference to the Lipschitz properties of the composite supply-demand maps used in the models.

Adaptive step size determination methods At the time of this writing, no applications of the Magnanti and Perakis adaptive step size determination methods have been published. Small-scale computational experiments carried out by the author on the static stochastic user equilibrium problem indicated some speedup compared to the MSA, but this must be considered extremely preliminary. In a dynamic network problem, each evaluation involved in the step size determination involves the full composite function including guidance generation, routing and network loading (in some order); even with inexact line search methods, this implies a considerable computational effort just to determine the step size to use in one iteration. The challenge is to develop potential functions and line search methods that, when applied to transportation-related fixed point problems, result in less effort to convergence than do the predetermined step size methods that are currently used in most solution approaches.

[6] For n sufficiently large negative or positive, the congestion effects produced by assigned traffic are pushed to times either before or after the analysis period, leaving an uncongested network loading problem that can be solved in one iteration.

[7] For example, the CAPRES module included in the FHWA's PLANPAC/BACKPAC suite of mainframe computer programs.

4.3.2 Triangulation methods

4.3.2.1 Mathematical background

The groundbreaking work of Scarf [1967] initiated a period of intense research activity in the computation of fixed points using so-called triangulation methods. These methods can be most easily understood by considering fixed point problems defined on simplices, but can readily be extended to more general domains.

A (closed) *simplex* S of dimension n (n-*simplex*) is the convex hull of $n+1$ affinely independent vectors in $\Re^{n+1}$, $v^1, \ldots, v^{n+1}$, called its *vertices*: $S = \{x \in \Re^{n+1} | x = \sum \alpha_i v^i, \alpha_i \geq 0, \sum \alpha_i = 1\}$. The representation of an $x \in S$ in terms of the α_is is unique; they are called x's *barycentric coordinates*. The *diameter* of a simplex is the maximum distance between any two of its vertices.

The *unit* n-*simplex* U^n is the simplex whose vertices are the unit vectors in $\Re^{n+1}$: $U^n = \{x \in \Re^{n+1} | x_i \geq 0, \sum x_i = 1\}$. On a unit simplex the barycentric and Cartesian coordinate systems coincide.

An n-simplex S has $n+1$ *facets* which are themselves simplices of dimension $n-1$. Each facet is said to be *opposite* a particular vertex v^j, and consists of the vectors $x \in S$ whose jth barycentric coordinate $\alpha_j = 0$.

A *simplicial subdivision* or *triangulation* of an n-simplex S is a set of simplices $S^1, \ldots, S^K$, also of dimension n (its *subsimplices*), whose union $S^1 \cup \cdots \cup S^K$ exactly covers S, and which has the property that the intersection $S^i \cap S^j$ of any two subsimplices is either empty or else is a facet common to both of them. The vertices of a triangulation consist of the combined vertices of all of its subsimplices (a vertex common to two or more subsimplices is included only once.) The *mesh* of a triangulation is the maximum diameter of any of its subsimplices.

An *integer labeling* of a triangulation of an n-simplex is a mapping that assigns to each vertex of the triangulation an integer (the vertex's *label*) drawn from the set $\{1, \ldots, n+1\}$. (A vertex shared by two or more subsimplices receives one and only one label.) A subsimplex in the triangulation is said to be *completely labeled* if each of its vertices is assigned a distinct element from the set $\{1, \ldots, n+1\}$.

Suppose that a triangulation of S is given an integer labeling that can be arbitrary except for the requirement that vertices located on a facet of S must not be labeled with the number of the vertex of S opposite that facet. (This is called a *proper labeling.*) Then *Sperner's lemma* states that the number of completely labeled subsimplices in the triangulation must be odd; in particular there must be at least one completely labeled subsimplex.

To see why this is of interest, consider for concreteness the unit simplex U^n and let $f(\cdot) : U^n \mapsto U^n$ be continuous. It follows that $f_i(x) \geq 0$ and $\sum f_i(x) = 1, \forall x \in U^n$. Construct a triangulation of U^n and label each vertex v of the triangulation with integer j if $v_j > 0$ and $v_j \geq f_j(v)$ (ties can be broken arbitrarily). (To verify that this labeling scheme is well-defined, suppose that for some vertex there were no j such that $v_j > 0$ and $v_j \geq f_j(v)$. Then for each $i \in \{1, \ldots, n+1\}$, either $v_i = 0$ so $v_i \leq f_i(v)$ or else $v_i > 0$ and $v_i < f_i(v)$. The latter alternative necessarily holds for at least one coordinate. Then $1 = \sum v_i < \sum f_i(v) = 1$, a contradiction.) Note that if v lies on the facet of U^n opposite its jth vertex, then $v_j = 0$ and the labeling scheme ensures that such a v will not be assigned the label j. The scheme therefore establishes a proper labeling of the triangulation, so it satisfies Sperner's lemma and a completely labeled subsimplex exists. If the mesh of the triangulation is small, any point in the completely labeled subsimplex will be "close" to its image

under $f(\cdot)$ and so can be considered an approximate fixed point of $f(\cdot)$.

Consider specifically a sequence of triangulations of U^n, each one labeled in this way, whose mesh tends to 0 in the limit. Each triangulation will include at least one completely labeled subsimplex. Since U^n is compact, the resulting sequence of completely labeled subsimplices contains a convergent subsequence that, because of the decreasing mesh, will tend in the limit to a single point in U^n, say x^*. As before, $x_i^* \geq 0, \sum x_i^* = 1, f_i(x^*) \geq 0$ and $\sum f_i(x^*) = 1$. Furthermore, since $f(\cdot)$ is continuous, $x_i^* \geq f_i(x^*), \forall i$. This can only hold if $x_i^* = f_i(x^*) \forall i$, so that $x^* = f(x^*)$; in other words, x^* is a fixed point of $f(\cdot)$ on U^n.[8]

It can be further shown [Border, 1985, Theorem 10.5] that, for any $\epsilon > 0$, there exists a $\delta > 0$ such that, if the mesh of a triangulation is less than δ, a completely labeled subsimplex of the triangulation lies in an ϵ-neighborhood of some fixed point of f on U^n. The value of δ can in fact be analytically determined. (For continuously differentiable maps, for example, this depends on the map's Lipschitz constant.) Thus, triangulation methods are capable not only of approximating the fixed point of continuous mappings, but also of providing an indication of the accuracy of a computed approximation.

A generalization of the method involves labelling vertices with vector rather than scalar labels. (The vectors defining the simplex receive labels equal to themselves.) Instead of looking for a completely labeled subsimplex, the search attempts to find a subsimplex with vector vertex labels $\ell_1, \ldots, \ell_n$ for which the linear equation system

$$y_1\ell_1 + \cdots + y_n\ell_n = b$$

has a non-negative solution $y_1, \ldots, y_n$ for a positive vector b. When seeking the fixed point of vector-valued functions, this method can be more effective than scalar labeling, which in effect throws away information when it reduces a vector function value to a scalar label.

Simplicial triangulation is the basis for a large family of algorithms that compute approximate fixed points of continuous functions on U^n by locating completely labeled subsimplices of a suitably-defined triangulation.[9] In general, starting from some initial subsimplex in the triangulation, these algorithms execute a systematic (and potentially exhaustive) search procedure that moves from one subsimplex to an adjacent one (i.e., to one that shares n vertices with it) by dropping one vertex and adding another in a procedure somewhat similar to LP pivoting. Specific triangulations have been proposed that facilitate the computations needed to identify a new vertex and that apply different approaches to cover the unit simplex. Early methods suffered from difficulties such as constraints on allowable initial points and limited restart capabilities; however, subsequent work has led to the development of particular triangulations and application of homotopy methods that eliminate most of these inconveniences [see Todd, 1976; Yang, 1999].

4.3.2.2 Applications

During the 1970s, as experience accumulated with the application of triangulation-based methods, the weaknesses of the approach began to become apparent. While theoretically very robust and

[8] This is in fact a proof of Brouwer's fixed point theorem on the unit simplex. The proof of the theorem for an arbitrary simplex is a straightforward modification of the argument given here. More general versions of the theorem (on an arbitrary convex compact domain D) can be established by embedding the domain in an enclosing simplex S and extending the function f to the composite function $f \circ P_D$, where P_D projects elements of S onto D.

[9] The methods used to prove Brouwer's theorem on more general domains can also be used to adapt these algorithms to domains other than U^n.

powerful, these methods had difficulties in solving medium- to large-scale problems in reasonable amounts of time. The basic issue is that no technique except direct search has been developed to locate completely labeled subsimplices. For general maps, this leads to a worst-case complexity that is exponential in the dimension and the number of digits of accuracy in the computed solution. While in practice triangulation methods usually only visit a very small fraction of the total number of vertices in a triangulation before stopping near a fixed point [Kuhn, 1976], the combination of the number of vertices visited and the need to evaluate the function at each visited vertex result in a procedure that is generally too computationally burdensome to apply to any except small-scale problems. Harker and Pang [1990] mention the disappointing experience in applying triangulation methods to the PIES (Project Independence Evaluation System) energy market equilibrium model during the late 1970s, and specifically cite this as one of the factors that drove the development of finite-dimensional variational inequality methods for (among other things) equilibrium analysis.

Kuhn [1977] developed one of the few published applications of triangulation methods to a transportation-related problem. He considered the static traffic assignment problem with separable link cost functions and fixed demand, formulated in the space of path splits. (Note that the set of feasible path splits for an OD pair constitutes a unit simplex, and the set of feasible path splits for all OD pairs is a Cartesian product of unit simplices, called a *simplotope*.) He derived a labeling rule such that completely labeled subsimplices identify user equilibrium flow patterns, then proposed a method for generating and searching subsimplices. The procedure was systematized in an algorithm called Pathfix.

In his Ph.D. thesis, Aashtiani [1979] commented on experiences up to the late 1970s in applying triangulation methods, including Kuhn's, to traffic network problems. He remarked favorably on the accuracy of the computed solutions, but observed that the computation time even for small networks was so high that application of the method to larger problems appeared to be impractical.

Triangulation methods remain of interest today for solving certain fixed point problems in economics and game theory, and they may be effective in solving specific network problems as well, but the paucity of applications to general traffic network problems since the work cited above would seem to bear out Aashtiani's observation.

4.4 Conclusions

This chapter has reviewed fixed point solution methods for possible use in route guidance generation, including both a discussion of the relevant mathematics as well as a summary of published applications of the methods. The discussion has focused exclusively on rigorously justifiable and computationally feasible methods. A wide range of published research efforts was surveyed under the unifying viewpoint of fixed point solution methods.

The Gibbs sampler is a method for generating samples from a joint distribution whose full conditional distributions are readily available. It was seen that this method could also be applied to generate sample trajectories from a general discrete-time stochastic process. The preceding chapter argued that if the network loading or routing maps are stochastic then the various composite maps that define the guidance generation problem will also be stochastic and that, without further information, their outputs must be considered to be sample trajectories from a general stochastic process. Thus, Gibbs sampling can be applied to generate sample trajectories from the stochastic process outputs of general guidance generation problems. It then becomes possible to investigate

the empirical properties of the solution processes to stochastic dynamic traffic models. Although Gibbs sampling has been used to investigate extensions of the static SUE problem, the identification here of its potential applicability to dynamic models with more general sources of stochasticity appears to be new.

In cases where model outputs can be viewed as noise-affected deterministic values, then stochastic approximation methods are applicable. To the transportation community, the method of successive averages (MSA) [Robbins and Monro, 1951; Blum, 1954] is without doubt the best-known stochastic approximation procedure. However, numerous other stochastic approximation methods have been developed since the MSA. The review identified iterate averaging [Polyak, 1990; Polyak and Juditsky, 1992], which is applicable whenever the MSA is but has optimal asymptotic properties and can be easily and efficiently implemented, as a particularly promising method for route guidance generation. Iterate averaging does not appear to have been used to date in transportation applications.

Deterministic methods reviewed included functional iteration and extensions as well as simplicial decomposition. Simplicial decomposition appears to be impractical for the size of problem typical in route guidance applications. Functional iteration as such is only applicable to contractive or similar maps. An averaging method due to Baillon applies only to non-expansive maps. Extensions developed by Magnanti and Perakis extend the domain of applicability to slightly more general maps but require line searches that are likely to be computationally burdensome. It is not known whether the maps commonly used in transportation applications satisfy the properties necessary for the rigorous application of these methods; this verification is likely to be difficult to carry out for simulation-based models. A final point is that simulation models are frequently stochastic or noisy (this point is discussed at length in Chapter 5), and the robustness of these methods when function evaluations return random values is not known. Consequently, their applicability to general guidance generation problems remains to be determined.

The next chapter investigates the application of Gibbs sampling, the MSA and iterate averaging to solve route guidance generation problems.

Chapter 5

Computational Tests

The moment you sit down at a harpsichord it says, "I will present your digital qualities as clearly as any instrument can do for you."

Gould [1963]

5.1 Introduction

A software system was developed as part of this research to operationalize and explore properties of the route guidance fixed point framework presented in Chapter 3, and to investigate the performance of the various algorithmic approaches identified in Chapter 4. The system is coded in C++ and provides capabilities in a number of areas:

- a class for each framework variable (link conditions, path splits, messages);
- a distinct function for each component map (DNL, routing, guidance);
- a version of each of the composite framework maps;
- a variety of composite map fixed point solution algorithms.

Together, the first three areas implement the capabilities of an enhanced traffic simulator (in the sense defined in Chapter 2), which will be referred to here as the *simple simulator*. The fourth category provides solution logic that can be wrapped around the simple simulator to compute consistent guidance. The system allows solution algorithms to be easily coded and tested. It is intended as a test bed for investigating alternative problem formulations and solution methods.

The organization of this chapter is as follows. Section 5.2 describes the functional design of the software system. It also identifies and discusses some of the major implementation issues.

The software system was applied to a number of problems in a series of test runs. These runs were intended to provide an empirical understanding of the route guidance analysis framework and to screen the different approaches (combinations of problem formulations and solution algorithms). Section 5.3 describes the test problem. Section 5.4 presents results of initial investigations into simulator properties, focusing particularly on the sources and amounts of stochasticity in simulator outputs. These tests lead up to the computation of a stochastic process solution for a full information dynamic equilibrium problem on the test network. Section 5.6 then presents results

from an extensive set of tests that examined the convergence properties of each of the three composite formulations, and of the MSA and Polyak averaging solution methods, when applied to three different problems. Conclusions of the tests are presented in Section 5.7. Graphic outputs from the tests are collected together for ease of reference in Section 5.8 at the end of the chapter.

5.2 Description of the simple simulator

The software implements intentionally simple versions of the network loading, routing and guidance component maps as separate C++ functions. The functions represent the component maps' basic properties without providing an elaborate set of modeling options. The design makes it relatively straightforward to combine the component maps in the various orders corresponding to the different composite framework maps identified above.

As will be discussed in the sections below, a number of design issues had to be addressed while developing the software. Many of these issues are inherent to simulation-based route guidance generation and, indeed, it is likely that any traffic simulator used for guidance generation would have to confront a very similar set of issues. While it is probably impossible to avoid design idiosyncrasies, a conscious effort was made to create a system that carries out the required tasks as generically as possible. Thus, there is reason to think that many of the conclusions obtained from the development, testing and use of this software system will be applicable to other simulation-based guidance generation systems as well.

5.2.1 Network loading map

The software system's dynamic network loading map is a discrete-time vehicle-based traffic simulator that implements a store-and-forward protocol [Gazis, 1974] with blocking (spillback). The loader's inputs are a network description in terms of nodes, links and paths[1]; a schedule of time-dependent trip (departure) rates by OD pair; and a table of time-dependent path splits at all network decision points (origins or en route points). Trip rates and path splits can be specified by user and/or guidance class. The loader's output is a table of average link traversal times by link and by time of link entry. Link volume profiles (i.e., the number of vehicles that entered, exited and remained on each link in each time period) can also be output on request. To compute these outputs, the loader generates vehicles in accordance with the input OD trip rates, allocates them to paths in accordance with the input path splits, and simulates their movement through the network, tracking their time of entry and exit from each link.

The distinguishing feature of this traffic simulator is its representation of links as deterministic FIFO (first-in-first-out) single-server queues with given exit (service) and storage capacities. Link attributes include connectivity (end node numbers), length, fixed speed, number of lanes and per

[1]Network paths are stored in a recursive data structure that, for each link and each destination, represents each path joining them by identifying the next link on the path, and the subsequent path to follow out of that next link. This recursive data structure is stored as a set of tables, one per link, and the individual paths are identified by their position in the table. This data structure is created prior to the start of the simulation from a file of explicitly enumerated OD paths that is prepared by the user. Multiple paths from an origin to a destination are allowed. Preprocessing converts the OD paths into tables of (sub)paths from each link to each destination, then eliminates duplicate entries and compresses the tables into the recursive [next link, next path] format. This data structure is accessed whenever a vehicle is ready to leave the link it is on and to continue towards its destination.

lane exit capacity. When a vehicle enters a link, its earliest possible exit time is calculated from the link's length and fixed speed; no account is taken in this calculation of other vehicles on the link. The vehicle is then placed at the tail of the link's queue and the link's available storage capacity (calculated from its length and number of lanes) is reduced accordingly. As each successive vehicle at the head of the queue is processed and moves on to a downstream link, each following vehicle advances in position until it too arrives at the queue head.

For a vehicle at the head of a queue to advance to the next link, it must (i) be able to leave its current link and (ii) be accepted on the next link. A vehicle is only able to leave the current link if its earliest possible exit time is less than or equal to the current simulation time, and if the link that it is on has unused exit capacity remaining in the current time step. If these conditions are met, and if the link is an en route decision point, the vehicle individually reselects a path in accordance with the input path splits corresponding to that location and the current time step; otherwise it retains its current path. In either case, the next link on the vehicle's path is determined.

An exiting vehicle advances if the next link on its path has storage capacity available, or if it has arrived at its destination. In advancing, it is removed from its current link and inserted into the next one, or removed from the network if it has arrived at its destination. The exit capacity remaining in the time step on the departing link is reduced, while the available storage capacities of the departing and receiving links are incremented and decremented, respectively.

A next link with no remaining storage capacity causes spillback: a vehicle wishing to enter that link remains blocked on its current link. Once the vehicle at the head of a link's queue is blocked, no further exits from that link are allowed until the vehicle becomes unblocked in a later time step; in effect, links are treated as if they have a single exit lane. Blocked vehicles do not attempt to bypass the capacitated link by choosing a new path.

As a special case, links with length 0 are considered to have infinite storage capacity (i.e., they do not spill back). Such links can be useful for modeling centroid connectors or other purely topological connections. The time vehicles spend in queues on such links is not included in the calculation of total path times. Note that any accumulation of vehicles on centroid connectors (e.g., because of spillback from downstream network links, or because the total trip production rate exceeds the centroid connector's exit capacity) results in a discrepancy between the input schedule of OD trip departures and the time trajectory of vehicles actually appearing on the physical network. This may not be desirable but, under these circumstances, it cannot be avoided.

In each simulation time step, the loader makes multiple passes over the network's links. Each pass advances one vehicle from the head of each link that has a vehicle ready to exit. Links are selected in random order so as to avoid systematic biases that could occur with a fixed link processing sequence: for example, always allowing vehicles from one link to traverse an intersection ahead of those from another link. The head vehicle on a selected link is examined to determine if it is able to advance. If it is not (for any of the reasons explained above), the link is removed from consideration in subsequent passes during the time step. When no more links are available for consideration, processing for the time step is terminated, and the simulation clock advances to the next time step, when all links are again considered.[2]

[2] An early version of the loader chose the links in random order, but processed all the exiting vehicles on a chosen link before proceeding to the next link. However, when problem data were such that more than a few vehicles could exit a link in a time step, this procedure was found to be unsatisfactory: it led to the unrealistic formation of platoons of vehicles transferring from link to link, biased the loader outputs and limited how closely a fixed point could be approached.

The time step duration determines the time resolution of traffic dynamics and is decided by the user. A short time step limits the number of link entries and exits per step, and so increases the variance of the traversal time estimates output by the loader (see below). On the other hand, the time step must not be so long that vehicles can completely traverse a link in less than one step, because this will make the loader unstable. In all work described here, a time step of one second was used.

As was mentioned above, the loader outputs a table of average link traversal times by link and by time of link entry. To compute this table, the simulator records each vehicle's time of entry and exit on each link that it traverses. From these it computes the vehicle's link traversal time, which is associated with its time of link entry. These traversal time values are separately accumulated for each link and entry time and, at the end of the simulation, the arithmetic mean of the values accumulated for each link and entry time is computed and used as the corresponding average traversal time. This calculation method results in traversal time values that correspond to the space mean speed of the vehicles that entered the link in each time step.

Instead of computing traversal times at the end of the simulation, a running average could be updated at each vehicle exit from a link, or the average corresponding to a particular entry time could be computed immediately after the exit of the last vehicle that entered the link at that time. As an implementation matter, it was simplest to compute the averages for all links and entry time steps at one time (i.e., at the end of the simulation). These averages are not needed during the simulation run that generates them, but only later: for example, in the next iteration of a guidance generation calculation.[3]

Similarly, other techniques for computing link traversal times from the individual vehicle times are possible. Wunderlich [1994], for example, proposes for this purpose an exponential smoothing filter $c_\ell^n(t) = (1-\alpha)c_\ell^{n-1}(t) + \alpha\tau_\ell^n$, where $c_\ell^n(t)$ is the traversal time for link ℓ computed after the exit of vehicle n in time step t, τ_ℓ^n is the traversal time experienced by vehicle n on link ℓ, and $0 < \alpha < 1$ is a constant. However, there seems to be no compelling reason to prefer such methods to simple averaging (for which $\alpha^n = 1/n$).

If no vehicles enter a link in a particular time step, the averaging method clearly cannot be used to compute the traversal time. In this case various rules and heuristics are applied. If the link is empty in the time step, it is given the traversal time that corresponds to its free flow speed. For a time step when a link is not empty but receives no arrivals, the traversal time is estimated by linear interpolation between the two closest bracketing values that are known either from averaging simulated times or because the link is empty. If a second bracketing value cannot be determined because there are vehicles still on a link when the simulation ends, a free flow value is arbitrarily assumed.

The accurate estimation of missing link time values may seem at first to be an unimportant nuisance, but in fact the quality of the treatment of this problem can strongly influence a simulator's accuracy and practical utility. The issue arises for example when links are blocked due to incidents: determination of an effective incident response strategy requires good estimates of traversal times after traffic begins to flow again, but simulation data for these estimates will not be available. Another example arises when the simulation time step duration is reduced in order to track traffic

[3]However, in a simulator that also represents real-time traffic control measures it might be preferable to update traversal time estimates at each vehicle exit from a link. This would better replicate the time estimates made by a real-time system.

dynamics at a finer time resolution: this reduction will increase the number of steps for which traversal times will have to be estimated, but poor estimates will negate the desired increase in simulation precision. This is a topic requiring further research.

As a final comment on the link traversal time calculation, note that it is possible to base traversal times on either the time of link entry or of link exit. However, when one is interested in computing path times or in determining dynamic minimum paths for a given departure time towards a given destination, the calculations are much more straightforward if entry-based traversal times are used—and these are precisely the types of problem of interest in route guidance applications. On the other hand, all data required to calculate the exit-based traversal time for time k is available at the end of simulation time step k whereas, as was seen above, the data needed to compute the entry-based traversal time for time k only becomes available some time later; consequently, the former value is slightly easier to calculate. Some traffic simulators calculate exit-based link traversal times yet compute path times as if they were using entry-based link times; they have generally had to resort to heuristics in an attempt to patch up the errors induced by this incorrect approach.

A number of researchers have developed traffic simulators based on a store-and-forward protocol, although incorporation of link blocking due to spillbacks is perhaps less common (despite the fact that Yagar [1971] presented a number of ideas along these lines thirty years ago). The general idea for this simulator is based on the dynamic network loading component of the QM model described in Simon and Nagel [1998] and Gawron [1998]. The most obvious difference between the simple simulator and the QM model is that the simple simulator simulator uses path splits and accounts for en route path switches whereas the QM model uses path flows and allows only pre-trip path choice. Furthermore, there are very many other differences in the designs of the two systems, and their implementations are completely distinct.

Despite the model's evident simplicity, it captures many of the basic characteristics of dynamic traffic flow in a network. Its link performance model has a reasonable-looking fundamental diagram (relationships between traffic flow, speed and density on a link) [Simon and Nagel, 1998]. It clearly represents the forward propagation and backward spillback of traffic from link to link. Perhaps the main feature of traffic flow that is not represented is the finite backward propagation speed of kinematic shock waves in queues. The model's blocking logic may also be unrealistic for multilane approaches to intersections with multiple possible turning movements. Nonetheless, it has been shown to produce results relatively close to those of much more complex traffic simulation models such as TRANSIMS and others [Nagel et al., 1998]. Moreover, its execution time is very modest compared to these larger models. These features make it well suited to exploratory investigations of guidance algorithms.

5.2.2 Guidance map

The simulator can represent ubiquitous (i.e., long-range) descriptive, short-range descriptive and short-range prescriptive guidance. Descriptive guidance consists of information on path times, while prescriptive guidance is a destination-specific path recommendation. Short-range guidance is only available on specific links but can be accessed by any vehicle on the link; it is intended to represent variable message signs or similar short-range collective guidance dissemination technologies. Ubiquitous guidance, on the other hand, is available anywhere on the network, but only to a subset of vehicles; it is intended to represent individualized guidance disseminated by technologies

requiring special equipment to access it. All forms of guidance are time-dependent, and relate to the paths that are available from the guidance source (origin or en route beacon) to accessible destinations. While other types of guidance are obviously possible, these options constitute a reasonable range of capabilities that illustrate typical guidance system features.

Descriptive guidance is implemented as tables of time-dependent path times from decision points to destinations, with separate tables maintained for ubiquitous and for short-range guidance. These tables are computed by the guidance map from the link time tables output by the dynamic network loading, by simply adding up the traversal times of links along a path, and taking account of the simulation time step during which each link will be reached (i.e., these are *experienced* path times). Link traversal time values are available from the loading map for each simulation time step, but the time step used in the guidance tables can be different; for example, link traversal times determined by the simulator might correspond to link entry times measured to the nearest second, while the guidance table might provide path times averaged over five minute entry intervals, for example to avoid confusing drivers with too-frequent changes in guidance messages. In the computational experiments carried out here, the ubiquitous guidance table contained estimates of time-dependent path times computed by the most recent iteration of the guidance generation algorithm being tested. In many experiments, the short-range guidance table also contained these estimates, but in some cases alternative values were used, as explained below.

For prescriptive guidance, the guidance map simply designates a path having minimum travel time from the guidance link to each possible destination, using the latest estimates of time-dependent link traversal times for the path time comparisons. A different recommendation may be given in each simulation time step but, again, need not be.

5.2.3 Routing map

The simulator's routing map is based on an underlying model of driver route choice behavior that predicts path choice probabilities. These become path splits in one of two ways, at the user's option. In an aggregate application of the probabilities, they are assumed to apply to a homogeneous group of vehicles as a whole. In this case, the path splits are identical to the choice probabilities. In a disaggregate application, each individual vehicle is allocated a path based on a independent draw from the choice probabilities. The path splits then emerge as the overall result of the separate draws by each vehicle.

As the discussion in Section 3.2.1.3 indicates, a behaviorally realistic driver behavior model for route guidance applications can be quite complex, and development of such a model represents on its own a considerable challenge. Since the focus here is on the guidance generation problem itself, it was preferable to base the software system's routing map on a simple approach that highlights the main issues of driver behavior modeling in route guidance applications, without attempting sophisticated modeling of drivers' information acquisition and decision processes. In this context, the main issues include:

- distinguishing between the behaviors of guided and unguided drivers;
- distinguishing between driver response to pre-trip and en route guidance; and
- distinguishing between driver response to descriptive and prescriptive guidance.

The simple simulator recognizes *equipped* and *unequipped* vehicles. (Although the software is also capable of distinguishing vehicles with different behavioral classes such as trip purposes or values of time, in the computational experiments described in this chapter all vehicles were assumed to be in one class.) Equipped vehicles are those that can receive ubiquitous guidance, which they use to make their path choice. Unequipped vehicles base their path choice decisions on background travel times (free-flow times in the experiments performed here), unless they arrive on a link where short-range guidance is available. In this case, they use the available guidance to reconsider en route their current path, then proceed on a possibly different path.

Drivers' pre-trip path choice decisions are represented by a logit-form path choice model in which a path's disutility is proportional to the ratio of its travel time to the travel time of the fastest path currently available from the origin to the destination. The coefficient of this ratio was set to a high number to approximate an all-fastest-path route choice model; however, paths whose times are nearly minimum also receive some traffic. A coefficient value of -10.0 was deemed to give reasonable results: in a two route situation, for example, a path whose time is is 25% more than the minimum has a choice probability of 8%.

The pre-trip path choice model is applied both to equipped vehicles, using link traversal times from the ubiquitous guidance source, and to unequipped vehicles, using background link traversal time values. In each case, the appropriate link traversal time values are used to compute the experienced path travel times corresponding to the vehicle's time of departure from the origin.

While this model obviously lacks the sophistication of state-of-the-art path choice models, it does possess a number of advantages for the applications intended here. First, since the routing approximates a fastest path logic, rough verification of the simulation system outputs is easily done by checking that, in each time step, the fastest paths receive most of the path flows. On the other hand, the routing map is continuous and one-to-one; compared to all-or-nothing (single fastest path) path choice, this has the effect of reducing the abrupt variations in path flows and avoiding the theoretical complications that would accompany the use of a path choice correspondence (one-to-many map). Finally, the behavioral assumption that drivers evaluate a path's time relative to the time of the fastest path is not totally unreasonable, although obviously such an assumption would have to be validated by empirical work before being applied in practice.

The model is applied somewhat differently in en route guidance situations. If short-range descriptive en route guidance is received, the currently followed path is reconsidered using a logit model that incorporates a switching penalty to represent the hysteresis resulting from drivers' reluctance to abandon an already selected path. The penalty is implemented as a multiplier that increases the disutility of all paths other than the current one. Note that, after considering the alternatives, the driver may nonetheless decide to continue using the original path.

A penalty multiplier value of 1.1 was used. With this value, an alternative route (different from the current one) whose travel time is 89% of the current route's time would have a 50% probability of being switched to. This produces interesting results in the computational experiments described later in this chapter, and is felt to be in the range of reasonable values. In practical applications, of course, this parameter would have to be determined through empirical work.

Absence of switching hysteresis in pre-trip path choice is simply handled by setting the switching penalty multiplier to unity for all paths. Thus, the same basic logit model form applies to the pre-trip path choice as well as to the en route path reconsideration decisions:

$$p_i = \frac{e^{\beta \gamma_i c_i / c^*}}{\sum_{j \in \mathcal{F}} e^{\beta \gamma_j c_j / c^*}}$$

where

p_i is the probability of choosing path i from among the set $\mathcal{F}$ of F available paths;

β is the utility function scale coefficient;

γ_j is the switching penalty multiplier; if i designates the currently followed path, $\gamma_i = 1$ and $\gamma_j > 1$ for other paths $j \neq i$;

c_j is the cost (traversal time) of path j, as obtained from background information or from the route guidance system;

$c^* = \min_{j \in \mathcal{F}} c_j$.

As is well known, the logit model assumption of independent error terms is likely to be inappropriate when the choice set includes overlapping paths, leading to unrealistic path choice probabilities in this situation. For this reason, probit models, which allow a more general error covariance structure, are sometimes preferred for path choice modeling. However, estimating probit models requires significantly more computational resources than estimating simple logit models. Researchers have recently proposed alternative approaches that retain the logit form (with its computational advantages) but, by adding a correction term to the systematic utilities, produce more realistic choice probabilities in overlapping path situations. Such models include C-Logit [Cascetta et al., 1996] and path size logit [Ben-Akiva and Ramming, working paper, 2000].

Because the intent here is to approximate fastest-path routing, path overlap corrections are less important than they are for general logit-form path choice models. For this reason, they were not implemented in the software. Adoption of a more general path choice model would require the software to be changed to accommodate the calculation of the overlap corrections and their inclusion in the logit model systematic utility; this is a relatively trivial software modification. However, development of a realistic routing model for practical applications would clearly require considerable empirical work to guide both the estimation of a path choice model and the implementation of path overlap corrections.

Driver response to short-range prescriptive en route guidance is represented by a simple compliance model that applies a fixed probability of accepting the recommended path to the destination. If the guidance recommendation is rejected, the vehicle continues on its current path.

Equipped vehicles ignore short-range guidance if they receive it, on the assumption that the ubiquitous guidance available exclusively to them is of higher quality than short-range collective guidance.

As mentioned above, the software system allows as a user option the path choice model described above to be applied either to individual vehicles in a disaggregate way, or to homogeneous groups of vehicles in an aggregate way.

In the disaggregate application, each vehicle departing from its origin, and each unequipped vehicle about to exit from a link containing a short-range guidance source, samples independently from the available paths in accordance with the path choice probabilities in order to determine its subsequent path. The combination of these individual decisions determines the total path flows

continuing on from the decision point. The path flows $f_1 \ldots f_F$ generated by N vehicles sampling independently from path choice probabilities $p_1 \ldots p_F$ will be a realization of a multinomial variate $\mathfrak{F}$, and in general will differ from one simulation run to the next:

$$\Pr(\mathfrak{F} = (f_1 \ldots f_F) | \sum_{i \in \mathcal{F}} f_i = N) = \frac{N!}{f_1! \cdots f_F!} p_1^{f_1} \cdots p_F^{f_F}$$

Consequently, the flow on any particular path i will be a realization of a binomial variate $\mathfrak{F}_i$:

$$\Pr(\mathfrak{F}_i = f_i) = \frac{N!}{f_i!(N - f_i)!} p_i^{f_i} (1 - p_i)^{N - f_i}$$

In the aggregate application of the behavior model to a homogeneous group of N vehicles, output path flows will always be $f_i = Np_i$, $i = 1 \ldots F$. (However, these products must also be rounded to integer values as described below.) This is, in effect, a large sample approximation, the accuracy of which increases with the size of the homogeneous group.

In pre-trip route choice modeling, a homogeneous group consists of a set of vehicles departing from the same origin in the same time step, going to the same destination and having identical user class and other parameters that condition the outputs of the route choice model. For en route choice models with switching hysteresis, the set of conditioning parameters includes the same factors as well as the vehicle's current path. Because of this additional stratification for en route decisions, the size of en route homogeneous groups will generally be less than that of pre-trip homogeneous groups. Particularly with simulation time steps of short duration, the size of en route homogeneous groups becomes so small that the "aggregate" route choice model application would in effect be equivalent to a disaggregate application. In fact the software enforces this conclusion: en route path switching probabilities are always applied independently, one vehicle at a time.

5.2.4 Sources of stochasticity in the simulator

A number of features of the software system make its outputs stochastic rather than deterministic. These features are common to many simulation models used for dynamic traffic assignment and route guidance generation, and for the most part do not represent gratuitous quirks of the system's design or implementation. Consequently, conclusions reached here regarding model stochasticity and its effects are likely to hold true even for more complex and elaborate simulation-based software systems.

In general terms, stochasticity in the model outputs results from (i) randomizing the order in which trip departures and links are processed, so as to avoid systematic biases in model outputs, and (ii) rounding to integers any non-integer numbers of vehicles, link traversal times, flow rates or link capacities per time step.

The individual sources of stochasticity are discussed below. However, because the issues derived from rounding are common to a number of the sources, it is appropriate to discuss these first.

5.2.4.1 Rounding issues

The simple simulator rounds scalars, vectors and matrices in the course of its processing. Although the general idea is always to convert non-integer values to integers, different requirements and procedures apply to each case:

scalar rounding considers each non-integer value to be rounded independently and without constraint;

vector rounding begins with a (one-dimensional) list of non-integer values that sum to an integer total, and attempts to round the values to integers in such a way that the rounded values sum to the same total;

matrix rounding is similar to vector rounding, but begins with a (two-dimensional) table of non-integer values having integer row and column sums. It attempts to ensure that the rounded values preserve the original row and column totals (as well as the grand total, of course).

Because of the constraints that must be respected, vector and matrix rounding are more complex than scalar rounding. Rounding vectors and matrices subject to additivity constraints is generally known as *controlled* rounding. This topic has been extensively investigated by statisticians, who sometimes need to present tabulations of rounded results without invalidating the inherent additivity relationships. Many of the methods developed by statisticians for this purpose involve multiple passes through the data. However, multi-pass methods may not be appropriate for some of the rounding applications needed in the simulator, when data must be rounded and used immediately after it is processed. For such applications, one-pass methods would be required.

The following paragraphs discuss in greater detail the various rounding requirements and applications in the simulator.

Scalar rounding The simple simulator uses scalar rounding to convert computed average link traversal times to the nearest time unit. For this it applies the usual method of rounding a non-integer value to an integer (i.e., x is rounded to $\lfloor x + 0.5 \rfloor$, where $\lfloor \cdot \rfloor$ denotes the integer part of a value).

Users also have the option of applying scalar rounding in situations where vector rounding should actually be used, in order to avoid the (slight) computational overhead of vector rounding. An example of this is the rounding of link exit capacities per time step. Such capacities are typically non-integer and may be fractional (less than 1), yet their sum over an extended period of time should be an integer approximately proportional to the link's exit capacity. Note that in this case the usual method of rounding can easily lead to unsatisfactory results because of its determinism. Converting an exit capacity of 1799 vehicles/hour into a per second integer equivalent in this way, for example, would block the link.

To avoid this problem, the simulator implements *random rounding* [Nargundkar and Saveland, 1972] of link exit capacities when scalar rounding is chosen. Random rounding generates an independent realization of $\mathfrak{U}[0,1)$ (a uniform variate on [0,1)) for each value to be rounded, rounding up to the next higher integer if the value's fractional part exceeds the realization, rounding down otherwise. Of course, this method does not guarantee that the sum of the rounded values in the sequence will equal the correct total. However, its properties in this regard can be seen from two representative situations.

Consider a finite sequence of J non-integers $n_1 \ldots n_J$ with integer sum N. Suppose first that the J values in the sequence to be rounded are all the same. For ease of exposition, suppose further that the non-integer values N/J are fractional and can each be written $0.p$. Then random rounding of each value produces a Bernoulli variate taking value 1 with probability p, and the result of J realizations is a binomial $\mathfrak{B}(J,p)$ variate. For sufficiently large J, the binomial variate will be well-approximated by a normal variate $\mathfrak{N}(N, Jp(1-p))$. Returning to the link exit capacity example

used above, suppose that the exit capacity is 1800 vehicles/hour, that the simulation time step is one second, and that the simulation is run for one hour. The "effective" link capacity resulting from random rounding would be approximately distributed as $\mathfrak{N}(1800, 30^2)$. This is probably a tighter range than is obtained for most estimates of roadway capacity from traffic measurements, so might be considered acceptable in many applications. Note that when $p = 0.5$, as it is in this case, the variance of the sum is maximized. For $p = 0.1$, for example, the sum is approximately distributed as $\mathfrak{N}(1800, 18^2)$.

To consider a second representative situation, suppose that the fractional parts of all the values to be rounded are independently distributed as $\mathfrak{U}[0, 1)$. Then the probability of rounding up any particular value is the same as the probability that the $\mathfrak{U}[0, 1)$ fractional part exceeds the $\mathfrak{U}[0, 1)$ realization generated by the random rounding, which is Bernoulli distributed with $p = 0.5$. The situation thus reduces to the first example considered in the preceding paragraph.

In conclusion, random rounding is easy to apply and, although it is a scalar rounding procedure and so cannot guarantee that the sum of the rounded values will be correct, over a large number of applications the sum will tend to converge to the correct value.

Vector rounding The simulator implements the *bucket rounding* method of vector rounding. This is always used to determine the number of vehicles departing from an origin in each time step. At the user's option, it may also be used to determine link exit capacities per time step.

Starting at some random initial location i in the sequence $n_1 \ldots n_J$ that sums to N, bucket rounding rounds the value n_i up or down to $\tilde{n}_i$ in the usual way. This creates a discrepancy $\delta_{i+1} = n_i - \tilde{n}_i$ which is added to the next value n_{i+1} in the sequence. The sum $n_{i+1} + \delta_{i+1}$ is rounded in the usual way to obtain $\tilde{n}_{i+1}$, and a new discrepancy $\delta_{i+2} = n_{i+1} + \delta_{i+1} - \tilde{n}_{i+1}$ results. The procedure continues in this way, wrapping around at the end of the sequence from n_J to n_1 until all J values have been rounded. (The starting location i is chosen randomly to avoid a systematic bias resulting from the fact that the initial discrepancy $\delta_i \equiv 0$.) Because of the vagaries of rounding the final value, the sum of the rounded values may not exactly equal N; however it will not differ from N by more than ± 1. Bucket rounding can be considered a stochastic procedure because it begins at a random location in the value sequence; however, once started it is deterministic. This is most clearly seen when the procedure is applied to a set of equal values: its outputs then cycle repeatedly through a fixed sequence of rounded values. For example, if all values to be rounded are 0.25, the outputs consist of repetitions of some fixed permutation of the sequence 0, 0, 0, 1.

Note that for any j, the discrepancy δ_j contains all the information needed to apply the bucket rounding procedure to the entire sequence. In particular, it is possible to make a dummy pass through the sequence, calculating the δ_is but not storing the rounded data values, until the beginning of the sequence is arrived at and δ_1 is obtained. A normal pass can then begin with $i = 1$ and the corresponding δ_1; this saves on storage in applications where the rounded values are used in sequential order and do not otherwise need to be stored (e.g., when any value in the sequence can be easily computed as needed.)

Matrix rounding The need for matrix rounding arises in the simple simulator when a table of non-integer path flows by time step must be converted to integer numbers of vehicles while preserving both the total trip productions in each time step and the total number of vehicle departures on

each path over the considered time period.

Bacharach [1966] examined the problem of rounding a matrix in a way that preserves consistent row and column sums, and showed that the problem is equivalent to a maximum flow problem in a network with lower and upper bounds on link capacities. Such problems might have a multiplicity of solutions. Subsequently, Cox and Ernst [1982] imposed additional constraints on the problem, notably that the l_p norm of the difference between original and rounded values be minimum, and that any integer elements of the matrix round to themselves. They showed that this problem could be converted to a standard capacitated transportation problem for which a solution always exists.

It was not felt justified to implement max-flow or transportation problem solvers as components of the simple simulator solely for the purpose of matrix rounding: the complexity of the required processing was out of proportion to the utility of the results that it would produce.[4] Consequently, a heuristic approach was adopted. This heuristic ignores the constraint on trip production in a time step and applies a simplified form of bucket rounding to each path flow across time steps. The simplification consists of beginning the bucket rounding in the initial time period rather than selecting a random starting point (which would have required the availability of future path flows). Checks on intermediate outputs of the heuristic show that both trip production and path departure totals are generally preserved. This procedure also regularizes the number of vehicle departures on an individual path from time step to time step, as was noted in the discussion of bucket rounding above.

5.2.4.2 Individual sources of stochasticity

The following paragraphs discuss the various operations that produce stochasticity in the simple simulator. To the extent that most vehicle-based traffic simulators need to carry out similar operations, they too are likely to experience stochasticity in their outputs.

Integer rounding of the number of departing vehicles per time step. OD trip rates are input in units of vehicles/hour and typically apply over some shorter period of simulated time, say 10–15 minutes. The first operation in each processing step is to convert the applicable OD trip rates into an integer number of departing vehicles in that time step. If the integer rounding were done independently for each time step, there would be no guarantee that the total number of generated vehicles would equal the number expected from the trip rate and its period of applicability. This may or may not be considered realistic, but it would certainly complicate the interpretation of simulation results: different simulation runs with identical input data could result in significantly different numbers of vehicles being loaded on the network. To prevent this, bucket rounding is applied to the set of vehicle departures in each time step to ensure the correct total over the trip rate's period of applicability. However, since bucket rounding itself is a probabilistic procedure, the number of vehicles generated in each time step is stochastic.

Integer allocation of vehicles to paths—aggregate case. As explained above, aggregate application of path splits involves multiplying the number of vehicles in a homogeneous group departing

[4] After most of the runs described in this chapter had been completed, it was learned that Cox [1987] had proposed a much simpler procedure for controlled matrix rounding. Because of time constraints, Cox' procedure could not be incorporated in the software, but would be worth implementing for future work.

from a decision point for a destination by the appropriate path splits. This will generally result in a non-integer number of vehicles on each available path. As was noted above, it is desirable to respect trip production and path departure totals, but rigorous methods for doing this appear to be excessively complex. Consequently, a heuristic bucket rounding procedure was used, which introduces stochasticity both in productions per time step and cumulative path flows.

Integer allocation of vehicles to paths—disaggregate case. Disaggregate application of path splits ensures that the correct integer number of vehicles will always be allocated across the various available paths. However, as explained above, in this case the number of vehicles choosing the different paths is a realization of a multinomial distribution, and would generally vary in subsequent samplings. If the simulation time step is relatively brief, then only a small sample will be drawn from the distribution in a given time step, and the sample properties will have relatively high variance.

Randomizing the loading order of departing vehicles. For each time step and each origin and destination, the preceding operations determine the integer number of vehicles departing on a particular path between them. This number of vehicles is then created by the simulator and inserted on a list of vehicles waiting to enter the network at the origin. If this list were created by processing each path in turn (e.g., in a loop), and if the individual vehicles on the list were loaded in sequential order, the result would be platoons of vehicles with identical destinations and other characteristics entering and traveling together through the network. To avoid this unrealistic situation, the vehicle list is shuffled prior to loading to randomize the order of departing vehicles.

Randomizing the order of link processing. Links are processed (i.e., vehicles on them are moved if able to do so, capacities are adjusted and traversal time statistics are recorded) in random order so as to avoid biases that a fixed processing order would produce (e.g., vehicles on one link always moving before those on another).

Integer rounding of link exit capacities. Exit capacities in vehicles/hour must be converted to an equivalent capacity per simulation time step. If the conversion results in a non-integer value, it is converted to an integer using either random or bucket rounding, at the user's option.

Calculation of link traversal times by averaging. Link traversal times by link and by time of link entry are calculated as the average of corresponding individual vehicle traversal times observed during the simulation. Considering the stochasticity of the individual vehicle times, the average itself will be stochastic, with a variance that decreases with the number of vehicles used for the calculation.

Integer rounding of link traversal times. The averages computed as described above are then rounded to the nearest integer in the usual deterministic way. This can be thought of as introducing noise into network loading map outputs since the true values are modified by the addition of a (signed) random fraction.

The rounding of the number of departing vehicles per time step, randomized order of link processing and rounding of link exit capacities tend to introduce stochasticity into the rate of

propagation of vehicles along their paths. Conversely, the randomized order of vehicle departure processing, the averaging of individual times and integer rounding of the averages, and the integer allocation of vehicles to paths tend to introduce stochasticity into the distribution of vehicles across paths.

As indicated above, "amount" of stochasticity introduced by the various sources can generally be expected to depend on the number of vehicles involved in the processing: as this number increases, the variance of the different random effects will generally be reduced. Accordingly, the simple simulator provides the option to *scale* trips: a trip by one vehicle can be converted into N trips of 1/N "vehicle fractions", each fraction having guidance and user class characteristics identical to that of the original single vehicle.[5] The effect of N trips of 1/N vehicle fractions is identical, in terms of link exit and storage capacities, to the effect of one trip by one vehicle. However, each vehicle fraction is processed as an independent flow element by the loader, and so is independently allocated to a path and moved along it. By using a large trip scale factor, the variance-reducing effects of high flow rates can be obtained without changing the number of vehicles loaded on the network. Put differently, higher trip scale factors cause the simulator outputs to approximate more closely those of a continuous flow model.

Similarly, the simulator provides the option to scale link traversal times. As noted above, the average traversal time computed from simulation outputs for a link and entry time step is rounded to the nearest integer number of time steps (equal to one second in this work). A time scale of 10 causes the simulator to round the computed average to the nearest tenth of a second, and similarly for other scale factors. This allows some control over the contribution of travel time rounding to simulator output stochasticity.

5.2.5 Implementation issues

There are various ways in which the framework developed in Chapter 3 could be implemented using the software components described above. A key implementation decision concerns the manner in which each of the component maps is evaluated.

The dynamic network loading map has unique properties in this regard because its outputs for time step t (link traversal times for link entry in time t) can only be determined some time later, when the vehicles that entered links in time t have exited from the links. Thus, while it is meaningful to talk of moving vehicles in one time step t according to the prevailing path splits $P(t)$ as part of the network loading process, it does not make sense to speak of evaluating the network loading map at a particular time step. The loading map must be evaluated for all time steps, and its outputs only become finalized when the simulation has finished.[6] Thus, evaluation of the dynamic network loading map S necessarily incorporates a loop over all simulation time steps and produces the full $C(\cdot)$.

[5]This operation is the inverse of grouping a number of vehicles with homogeneous characteristics into a single "packet" that is moved through the network as a unit. Use of packets is sometimes proposed as a means of reducing the amount of required simulation processing by decreasing the number of individual vehicles that have to be moved and tracked (although dynamically grouping vehicles into packets also requires computer processing, of course). It can be seen here that use of packets may also have the effect of increasing the simulator stochasticity, which may be an unwanted side-effect.

[6]More precisely, determination of link conditions corresponding to entry in a particular time step can be completed in a "window" the duration of which depends on the time needed by all the vehicles that entered links in that step to exit them. The duration of this window cannot be known *a priori* since it depends on dynamic network conditions.

The guidance and routing maps, on the other hand, can be evaluated for a particular time step and network feature. An evaluation of the guidance map, although it requires the full $C(\cdot)$ as input (to compute anticipatory time-dependent path conditions), need only produce messages for one particular time step and decision point n: $M_n(t)$. Similarly, an evaluation of the routing map need only consider $M_n(t)$ to produce the path splits for the time step and subpaths r exiting node n: $P_r(t)$. Either of these two maps can thus be embedded within a loop over the simulation time steps and evaluated for specific arguments as needed; this will be called *on-the-fly* evaluation of the component maps. On-the-fly evaluation restricts any required auxiliary processing (such as vector or matrix rounding) to one-pass methods.

On the other hand, if the required inputs are available, the guidance or routing maps could also be pre-computed for all time steps and network elements, with the complete set of results $M(\cdot)$ or $P(\cdot)$ stored for later use. This will be called *all-at-once* evaluation of the component maps. In the case of the guidance map, all-at-once evaluation would entail a loop that computes the messages to disseminate in each time step at each decision point; given a complete set of input link condition data $C(\cdot)$, the procedure is straightforward. The routing map, on the other hand, might depend not just on input guidance messages but also on conditioning parameters such as user class or prior path. If these conditioning parameters take on a finite set of values, all-at-once evaluation of the routing map would then require a loop over time steps and over all possible values of the conditioning parameters, with storage of the complete set of results. If the conditioning parameters are continuous and cannot be satisfactorily discretized, then all-at-once evaluation is not possible.

These contrasting approaches for component map evaluation lead to similarly contrasting implementations of each of the composite framework maps. These will again be called all-at-once evaluation (i.e., the component maps are completely evaluated for all input values prior to their use) and on-the-fly evaluation (i.e., the component maps are evaluated only when required and only for needed input values.) All-at-once evaluation corresponds to a literal-minded implementation of the composite function framework, while in some cases the on-the-fly evaluation fits more naturally in the logic of a simulation. The following paragraphs amplify these comments in the context of each of the framework maps.

5.2.5.1 The composite link condition map

This composite map, $S \circ D \circ G : C \mapsto C$, applies the guidance map ($G : C \mapsto M$) to an input set of time-varying link condition forecasts in order to determine appropriate guidance messages. When drivers receive the messages and react to them, the path splits are affected ($D : M \mapsto P$). The routing of OD demand according to these path splits results in a new set of link condition forecasts ($S : P \mapsto C$).

As can be seen in Figure 5.1, the all-at-once implementation is a straightforward translation of this logic. (In the following figures presenting pseudo-code for the implementation options, indented lines are within a loop, italicized lines are comments, and expressions within curly brackets indicate the function evaluation being carried out by a section of pseudo-code.)

Figure 5.2 shows that the on-the-fly implementation of the $S \circ D \circ G$ map is much more condensed.

In the all-at-once implementation, the evaluation of one component map is fully completed before that of the next (in the order defined by the composite map) is begun. This requires storage for the total number of framework variable values: condition forecasts for all links and time

given a complete set of link condition forecasts $C(\cdot)$

fully evaluate the guidance map $\{M(\cdot) = G(C(\cdot))\}$:
for each time step t and decision point n
    generate guidance messages based on link condition forecasts $\{M_n(t) = G(C(\cdot))\}$

fully evaluate the routing map $\{P(\cdot) = D(M(\cdot))\}$:
for each time step t, decision point n and conditioning parameter value
    compute path splits based on guidance messages $\{P_r(t) = D(M_n(t))$

fully evaluate the dynamic network loading map $\{C(\cdot) = S(P(\cdot))\}$:
for each time step t
    move vehicles according to path splits $P(t)$
compute a complete set of output link condition forecasts

Figure 5.1: All-at-once implementation of the $S \circ D \circ G$ map

given a complete set of input link condition forecasts $C(\cdot)$

for each time step t
    generate guidance messages based on link conditions $\{M_n(t) = G(C(\cdot))\}$
    compute path splits based on guidance messages $\{P_r(t) = D(M_n(t))\}$
    move vehicles according to path splits $P(t)$
compute a complete set of output link condition forecasts $\{C(\cdot) = S(P(\cdot))\}$

Figure 5.2: On-the-fly implementation of the $S \circ D \circ G$ map

steps; guidance messages for all decision points and time steps; and path splits for all conditioning parameters (current path and user class), decision points and time steps.

Furthermore, in the all-at-once implementation the routing map may be evaluated for values of conditioning parameters that do not actually occur during the simulation. For example, at a particular en route decision point and a particular time, path splits would be computed for each possible current path that a vehicle could be following. Later, when vehicles are actually being moved by the simulator, it might happen that the vehicles at that decision point at that time were only following one of the possible paths; computation of splits conditioned on the other possible paths would then turn out to have been unnecessary.

In contrast to the above, the on-the-fly implementation of the $S \circ D \circ G$ map only evaluates the guidance and routing maps at the times and for the values that are needed to move vehicles, and it only maintains message and path split values for the current simulation time step. In this sense, it is a more efficient procedure than the all-at-once implementation.

5.2.5.2 The composite path split map

This composite map, $D \circ G \circ S : P \mapsto P$, starts with a complete set of path splits, forecasts the corresponding link conditions using the dynamic network loading map ($S : P \mapsto C$) and generates guidance messages accordingly ($G : C \mapsto M$); these are disseminated to drivers and cause them to react in some way, leading to a new set of path splits ($D : M \mapsto P$).

Again, the all-at-once implementation is a straightforward translation of the sequence of map evaluations (Figure 5.3).

This is clearly a trivial permutation of the algorithm steps used for the $S \circ D \circ G$ composite map. The required changes to the software are nearly as trivial: this is an advantage of the all-at-once implementations for research purposes. On the other hand, the storage requirements are again significant, and there is again the possibility that some evaluations of the routing map may prove to be unnecessary, in the sense that the map is evaluated for situations that do not arise during the simulation.

The on-the fly implementation of the $D \circ G \circ S$ map turns out to be very similar to its all-at-once implementation (Figure 5.4). The only difference is that the on-the-fly implementation includes both the guidance and routing map evaluations in the same time step loop, whereas these evaluations are performed in separate time step loops in the all-at-once implementation.

Consequently, the on-the-fly implementation of this map does not offer as many advantages as that of the $S \circ D \circ G$ map.

5.2.5.3 The composite message map

This map, $G \circ S \circ D : M \mapsto M$, begins with a set of time-dependent messages, predicts the resulting path splits ($D : M \mapsto P$), forecasts the ensuing link conditions ($S : P \mapsto C$), then determines the appropriate new set of guidance messages ($G : C \mapsto M$).

Again, the all-at-once implementation is straightforward (Figure 5.5).

The on-the-fly implementation is more advantageous than that of the $D \circ S \circ G$ map but less so than that of the $S \circ D \circ G$ map (Figure 5.6). This organization avoids the need to pre-compute and store path splits for all situations, but still requires a full evaluation of the guidance map.

given a complete set of path splits $P(\cdot)$

fully evaluate the dynamic network loading map $\{C(\cdot) = S(P(\cdot))\}$:
for each time step t
    move vehicles according to path splits $P(t)$
compute a complete set of link condition forecasts

fully evaluate the guidance map $\{M(\cdot) = G(C(\cdot))\}$:
for each time step t and decision point n
    generate guidance message based on link condition forecasts $\{M_n(t) = G(C(\cdot))\}$

fully evaluate the routing map $\{P(\cdot) = D(M(\cdot))\}$:
for each time step t, conditioning parameter value and subpath r
    compute path splits based on guidance messages $\{P_r(t) = D(M_n(t))\}$

Figure 5.3: All-at-once implementation of the $D \circ G \circ S$ map

given a complete set of path splits $P(\cdot)$

evaluate the dynamic network loading map
for each time step t
    move vehicles according to path splits $P(t)$
compute a complete set of link condition forecasts $\{C(\cdot) = S(P(\cdot))\}$

for each time step t
    for each decision point n
        generate guidance messages based on link condition forecasts $\{M_n(t) = G(C(\cdot))\}$
    for each conditioning parameter value and subpath r
        compute path splits based on guidance messages $\{P_r(t) = D(M_n(t))\}$

Figure 5.4: On-the-fly implementation of the $D \circ G \circ S$ map

given a complete set of messages $M(\cdot)$

evaluate the routing map $\{P(t) = D(M(t))\}$:
for each time step t, decision point n and conditioning parameter value
    compute path splits based on guidance messages $\{P_r(t) = D(M_n(t))\}$

evaluate the dynamic network loading map $\{C(\cdot) = S(P(\cdot))\}$:
for each time step t
    move vehicles according to path splits $P(t)$
compute a complete set of link condition forecasts

evaluate the guidance map $\{M(t) = G(C(\cdot))\}$:
for each time step t and decision point n
    generate guidance messages based on link condition forecasts $\{M_n(t) = G(C(\cdot))\}$

Figure 5.5: All-at-once implementation of the $G \circ S \circ D$ map

given a complete set of messages $M(\cdot)$

for each time step t
    compute path splits based on guidance messages $\{P(t) = D(M(t))\}$
    move vehicles according to path splits $P(t)$
compute a complete set of link condition forecasts $\{C(\cdot) = S(P(\cdot))\}$

for each time step t and decision point n
    generate guidance messages based on link condition forecasts $\{M(t) = G(C(\cdot))\}$

Figure 5.6: On-the-fly implementation of the $G \circ S \circ D$ map

5.2.5.4 Simulator implementation decisions

Based on the considerations discussed in the preceding paragraphs, it was decided to prepare computer codes for the on-the-fly implementation of the composite link condition map and the all-at-once implementations of the composite path split and message maps. After one of the all-at-once implementations was programmed, preparation of the other was essentially a trivial matter of rearranging the order of function calls and ensuring proper initialization of the data structures. Code for the on-the-fly implementation of the composite condition map required substantially different coding. The effort was felt to be worthwhile because this combination appears to provide the most storage-efficient and in some ways the more straightforward implementation of any of the composite maps—although of course its execution efficiency remains to be determined. In view of the reduced advantages of the on-the-fly versions of the composite path split and message maps compared to their all-at-once versions, and of the all-at-once version of the composite link condition map compared to its on-the-fly versions, it was not felt worthwhile to program these.

5.3 Description of the test network

Runs were made using a simple 14-link network with a single OD pair and eleven OD paths. Figure 5.7 shows this network. All links are single lane and 1 km long except for links 2 (100-201) and 9 (201-200), which are 1.5 km long. Vehicles are assumed to be 7.5 m long, so the link storage capacities are about 133 and 200 vehicles, respectively. Free speed on all links is 100 km/hr. Except for the centroid connectors, all links normally have an exit capacity of 3,600 vehicles/hour.[7] In some runs, as noted below, link 6 (102–200) has a reduced capacity of 900 vehicles/hour throughout the entire simulation period. This might be thought of as an unplanned road blockage that occurs during the night (when the network is empty), and about which tripmakers only become aware through ubiquitous guidance or the VMS.

The OD demand rate, duration of inflows and percentage of equipped vehicles depend on the experiment. Both the origin and the destination centroid connectors (links 1–100 and 200–2, respectively) have infinite storage capacity (i.e., they do not spill back). The origin centroid connector meters vehicles onto the network at the rate of 3 vehicles/second unless one of its downstream links spills back. Equipped vehicles receive the estimates of time-dependent link traversal times from the most recent iteration of the fixed point algorithm. There is also a source of short-range link condition (descriptive) guidance on link 1 (100–101); in the interest of brevity, this will be called a VMS. Any unequipped vehicles on this link can access the VMS information to reconsider their path choice at node 101. Different types of information were disseminated there in different runs.

The simulation time step is 1 second in all runs. All simulations start with an empty network and continue until the last vehicle has cleared the network; this generally takes 30 minutes or less of simulated time.

5.4 Investigation of simulator properties

For ease of reference, all graphs mentioned in this discussion have been collected in Section 5.8 at the end of the chapter.

[7] The intent here was to have a convenient per second capacity value, not to be realistic.

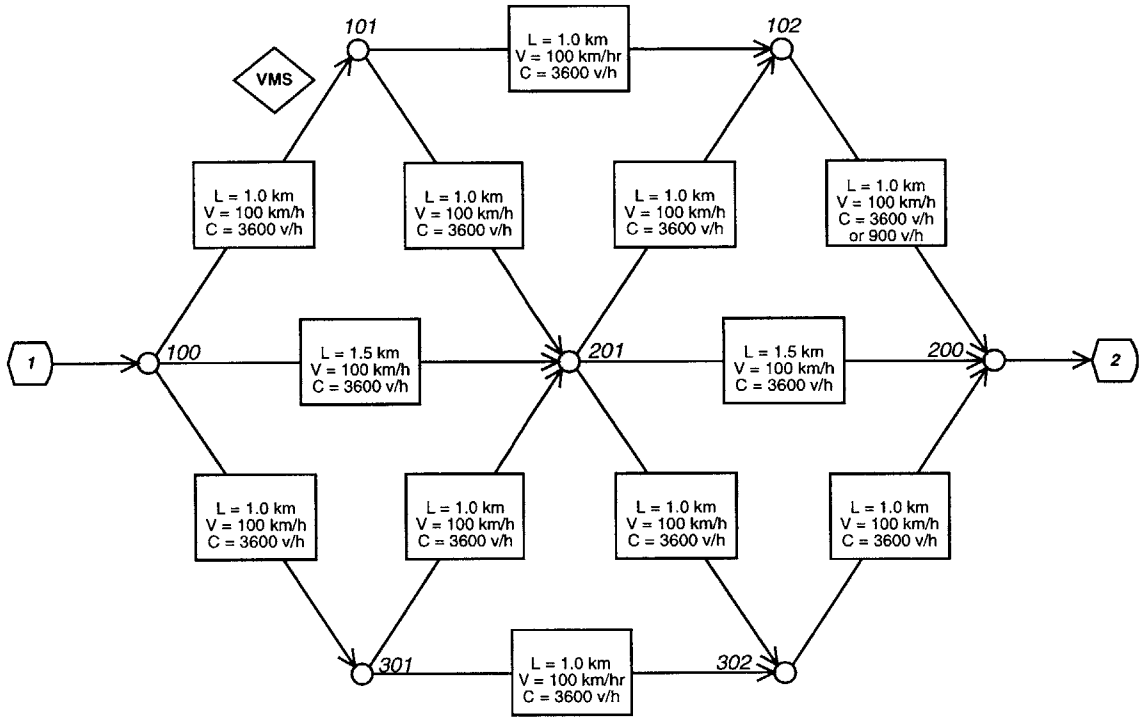


Figure 5.7: Test network

5.4.1 Stochasticity in the network loading map

Stochasticity in the network loading map affects the rate of propagation of vehicles proceeding along already-chosen paths. As was seen in the discussion of stochasticity sources earlier in this chapter, this stochasticity derives from a number of factors including randomizing the order of vehicle departures and link processing, and rounding link capacities and traversal times.

This section presents computational experiments intended to assess, in a qualitative way, the "amount" of stochasticity introduced into simulation outputs by these various factors acting individually and in combination. To this end, the simulator was presented with fixed schedules of vehicle departures *by path*, so that no path choice processing needed to be carried out. The program simply moved the vehicles along their predetermined paths in accordance with the logic of the network loading map. Either random or bucket rounding was used to determine link exit capacities per time step, as noted in the discussion of individual experiments. In each experiment, five separate replications of the network loading application were carried out, with identical input data but different random numbers affecting the processing order and rounding. The resulting link volume trajectories were then plotted. (The link volume is the number of vehicles remaining on a link at the end of a time step.) By visually examining the differences in plotted link volume trajectories from each replication, an impression of the magnitude of the stochastic effects can be formed.

In discussing these stochastic effects, the term *variability* will be used to refer to stochasticity occurring between replications, and the term *noise* to refer to such effects within a single replication.

Table 5.1 summarizes the various runs that were carried out to investigate network loading

model properties. For ease of reference, graphics presenting the results of these runs are collected in Section 5.8 at the end of the chapter.

Run NLM-1 The first experiment introduced flow rates of 3,600 vehicles/hour on each of the 11 OD paths of the test network over a period of 240 seconds; thus, exactly 240 vehicles were loaded on each path. All network links were assumed to be operating at full capacity (3,600 vehicles/hour); consequently, although random rounding was selected for this run, it did not have any effect because of the 1 second time step.

The centroid connector had an exit capacity of 10,800 vehicles/hour. Because of the disparity between the centroid connector inflow and outflow rates, a considerable queue builds up on the centroid connector (this is not shown on the graphs). Inflows into the actual network are the vehicles leaving this queue.

The network was empty at the beginning of the run, and by 1,200 seconds all vehicles had exited the network. Once a vehicle was assigned to a path at the origin, no further en route path choice was allowed; therefore, there was no need to distinguish guided from unguided vehicles. Note that, by the choice of path flow rates and link capacities, little or no stochasticity was introduced as a result of the rounding of these two variables.

Figure 5.8(a) plots the link volume trajectories obtained in this way. An explanation of the main features of the traffic pattern presented in the figure may be helpful before discussing its stochastic aspects.

Of the eleven paths out of node 100, four traverse link 1, four traverse link 3, and three traverse link 2. The origin centroid connector upstream of node 100 allows 3 vehicles/second to exit. Thus, an average of 3 * 4/11 = 1.09 vehicles/second will go to link 1, the same average number will go to link 3, and an average of 3 * 3/11 = 0.82 vehicles/second will go to link 2.

In the absence of congestion (queueing), vehicles will take 36 seconds (1 km / 100 km/hour) to traverse links 1 and 3, and 54 seconds (1.5 km / 100 km/hr) to traverse link 2. For this amount of time following the start of the simulation, links 1, 2 and 3 accumulate but do not discharge any vehicles. This can be seen in the initial steep portion of the link volume plots. Then begins a phase where the links are both receiving and discharging vehicles. Since the discharge rate is 1 vehicle/second, links 1 and 3 continue to accumulate vehicles at an average rate of 1.09 - 1 = 0.09 vehicles/second, starting with the approximately 39 vehicles (1.09 vehicle/second * 36 seconds) already present at the end of the initial phase. Since the rate of vehicle arrivals at link 2 (0.82 vehicles/second) is less than the discharge rate (1 vehicle/second), the link 2 volume remains at the level it achieved at the end of the first phase, i.e., approximately 44 vehicles (0.82 vehicles/second * 54 seconds).

As long as there is traffic queued on the centroid connector, inflow to links 1, 2 and 3 will continue. The centroid connector queue empties after approximately 880 seconds (net accumulation

Run	*Path flow*	*Link exit capacity*	*Capacity rounding*	*Figure*
NLM-1	3600 v/h	3600 v/h	random	5.8
NLM-2	1800 v/h	1800 v/h	random	5.9
NLM-3	1800 v/h	1800 v/h	bucket	5.10

Table 5.1: Computational tests of the network loading map

rate of 11 vehicles/second inflow - 3 vehicles/second outflow = 8 vehicles/second * 240 seconds = 1920 vehicles; 1920 vehicles / 3 vehicles/second outflow rate = 640 seconds + 240 seconds = 880 seconds). The maximum volume accumulated on links 1 and 3 is around 115 vehicles (39 vehicles + 0.09 vehicles/second * (880 seconds - 36 seconds); this is less than the 133 vehicles (1000 meters / 7.5 meters/vehicle) that would cause spillback. Link 2, of course, maintained an approximately constant volume of 44 vehicles throughout this phase.

Once the inflows from the centroid connector have ceased, links 1, 2 and 3 discharge their accumulated traffic at the average rate of 1 vehicle/second until they are empty. This is the third phase of the volume diagram, where the volumes decline steeply to 0. Links 1 and 3 empty at around 995 seconds (880 seconds + 115 vehicles / 1 vehicle/second), while link 2 empties at around 924 seconds (880 second + 44 vehicles / 1 vehicle/second).

Traffic exits links 1, 2 and 3 at the average rate of 1 vehicle/second. Traffic from link 1 is divided among links 4 and 5; that from link 2 is divided among links 8, 9 and 10; and that from link 3 is divided among links 11 and 12. Thus, none of the links immediately downstream from link 1, 2 or 3 receives traffic at a rate that exceeds its exit capacity. The only accumulation of traffic on these downstream links is that due to the initial free flow traversal times; once vehicles begin exiting from the link, a steady state volume is maintained until the cessation of the inflows arriving from links 1, 2 and 3.

Links 8, 9 and 10 all receive direct inflows from links 2, 5 and 11. Again, the combined inflow rates do not exceed the exit capacities, so a steady state volume is maintained following an initial build up. Link 9 attains a higher steady state volume than links 8 and 10 because it is longer: it takes more time to traverse, so more vehicles accumulate during the initial build up. Traffic from link 2 stops before that from links 5 and 11 (because the latter two receive traffic via links 1 and 3, respectively), so volumes on links 8, 9 and 10 decline to 0 in two steps, corresponding to the cut-off of traffic from link 2 and later simultaneously from links 5 and 11.

Link 6 receives traffic from links 4 and 8, and link 13 receives traffic from links 10 and 12. Each link's combined inflow rate from its two upstream links is 1.02 vehicles/second, slightly exceeding its exit capacity, so there is a gradual buildup of traffic following the initial accumulation. The four paths leading to each of these links involve different numbers and traversal times of links; thus, in principle there are a number of steps in the initial accumulation, the buildup, and the discharge phases. However, the superposition of the inflow trajectories tends to mix these effects so much that they are barely discernible in the graph.

Turning now to the question of model stochasticity, the close correspondence of the individual graphs with the above calculations indicates the low degree of stochasticity present in the model outputs. The plotted results look almost as if they were obtained by analytical methods: there is no visible variability between replications, and very little noise within replications.

A slight amount of noise is evident in the pure inflow and outflow phases at the beginning and end of each plot. The noise in the middle phase (of combined inflow and outflow) is due to the randomized order of trip departures from the origin. Specifically, in each time step one trip per path (for a total of 11 trips) is generated. The generated trips are loaded on the origin centroid connector in randomized order. The centroid connector releases three trips per time period to its downstream links (unless these have spilled back). The number of vehicles that enter each downstream link depends on the individual paths of the released trips, and in particular on the first link of these paths. Thus, a given link may receive between zero and all three released trips.

On the other hand, the outflows from these links are essentially constant due to the choice of exit capacities. The result is that the accumulated link volumes fluctuate up and down within small limits, and vary slightly from one replication to another. As traffic propagates downstream from the first three links, the same phenomenon affects the inflows and accumulated volumes of subsequent links. These fluctuations produce the thick lines seen in the middle phase. (The same noise is actually present in the inflow phase, but is less visible because it only affects the upward slope of the lines.)

Link traversal time trajectories that are output from this dynamic network loading are shown in Figure 5.8(b). As was seen from the link volume plots, the only significant queueing takes place on links 1 and 3, and to a lesser degree on links 6 and 13. This is reflected in the link time plots. The graphs for links 1 and 3 show a linearly increasing traversal time corresponding to the steady net accumulation of vehicles. At around 880 seconds, the origin centroid connector queue empties and no more vehicles arrive at links 1, 2 and 3. The simulator must estimate the traversal times for later arrival times; it does this by interpolating between its last computed time and the free flow time of 36 seconds when the link empties. Changes in the traversal time curves for links 6 and 13 are smoother because of the superposition of multiple vehicle arrival trajectories. Again, the variability of results between replications is very minor.

Similar results with respect to path times are seen in Figure 5.8(c). (These path times are computed in a straightforward fashion from the link traversal times output from the loading map.) Changes in the path time trajectories are due to corresponding features of the traversal time trajectories of links 1, 3, 6 and 13. There is so little variability in the outputs that it is impossible to tell from examination of the graphs that results for five replications are being superposed.

The clear conclusion that emerges from examination of these plots is that the stochasticity introduced by the network loading map is in this case very minor. No variability in output trajectories is evident; and the stochasticity can essentially be regarded as negligible noise.

Run NLM-2 The next series of figures (5.9(a) through 5.9(c)) present results from a run similar to the preceeding one, except that the (predetermined) path flow rates and link exit capacities are 1,800 vehicles/hour, or half of their values in the preceding experiment; random rounding is used to determine the exit capacity per time step. It might be expected that the results of this experiment would be an exactly scaled version of the earlier results. (Strictly speaking, proper scaling of the problem data would also require that link storage capacities be halved. It was seen, however, that no spillback occured in run NLM-1, so this was not an issue here.) In the preceding experiment, however, the basic link exit capacity was 1 vehicle/time step and so did not need to be rounded (or, more precisely, the applied rounding had no effect); in this experiment, on the other hand, the exit capacity is 0.5 vehicle/time step and is randomly rounded. Any significant difference in the nature of the outputs of the two runs can be attributed to the random rounding of exit capacities.

Although the shapes of the link volume trajectories are broadly similar to those seen in run NLM-1, a higher level of stochasticity in link volume trajectories is immediately apparent from examination of Figure 5.9(a). Not only do individual links have generally greater volume variability between replications, but there is also some evidence of individual random effects accumulating to produce qualitatively significant differences between some replication results. For example, this can be seen on link 3 where, in one of the replications, more favorable random rounding permitted earlier clearance of the accumulated link volumes, and avoided a large queue buildup. These results

are reflected in the link and path time outputs shown in Figures 5.9(b) and 5.9(c). It is seen that both the variability and the level of noise are much greater than in run NLM-1.

Run NLM-3 To verify that random rounding is indeed the cause of the stochasticity here, a run with the scaled data of run NLM-2 was made using bucket rather than random rounding of link exit capacities. The output link volume trajectories are shown in Figure 5.10. There is very little variability in the volumes, and the trajectories do indeed appear to be a scaled version of those output in run NLM-1 (cf. Figure 5.8(a)).

By bucket rounding link exit capacities, the stochasticity of dynamic network loading map outputs can be significantly suppressed. Bucket rounding does two things: it ensures that the exit capacity over any time period is proportional to the hourly capacity, and it produces a very regular (possibily unrealistic) cyclical pattern of vehicle outflows from a link. The tradeoff between the difficulties introduced by stochasticity and the artificiality that results from bucket rounding is one that individual studies will have to evaluate based on available empirical data.

In conclusion, this section has shown that apparently reasonable and seemingly innocuous design and implementation decisions in a simulation-based dynamic network loader can lead to significant amounts of stochasticity in the loader outputs. The principal cause of stochasticity is the rounding of non-integer link exit capacities per time step; however, if bucket rounding is used for this purpose, the stochasticity can be reduced to low levels, albeit at the cost of reduced realism in representing trip propagation. The other potential sources of stochasticity (randomizing the order of link processing and of vehicle departures from the origin) do not appear to contribute in an important way to network loader output stochasticity.

5.4.2 Stochasticity in the composite routing/loading map

In the next set of experiments, OD flow was split by the routing map among the 11 OD paths in the test network and moved by the network loading map over the network to the destination. The splitting rate at the origin was the same for each path (1/11, or approximately 0.091); this was achieved by setting the travel time scale coefficient to 0 in the underlying path choice logit model, so that splits were determined solely by the number of competing paths. No further en route path switches were allowed.

The purpose of the experiments was to determine the amount of simulator output stochasticity that results from the independent application of splitting rates to individual vehicles. Although the distribution of trips among paths is multinomial in this situation, the draws in a given time period generally constitute a small sample with high variability, and nonlinear network dynamics might amplify the effects of differences in trip departures by path, or of transient surges or lulls in the numbers of vehicle departures on individual paths.

It can be seen that these experiments resemble those described in the preceding section. In the present experiments, however, the splitting rates are applied independently to each vehicle departing from the origin to determine its path, whereas in the preceding experiments the vehicles' paths were predetermined. In the present experiments, the expected fraction of trips on each path is 1/11 but the exact amount may vary based on the specific draws from the multinomial

Run	*Mean path flow*	*Path splits*	*Link capacities*	*Capacity rounding*	*Figure*
SD-1	3600 v/h	disaggregate	3600	bucket	5.11
SD-2	1800 v/h	disaggregate	1800	bucket	5.12
SD-3	1800 v/h	disaggregate	1800	random	5.13
SD-4	3600 v/h	aggregate	3600	bucket	no figure
SD-5	1800 v/h	aggregate	1800	bucket	5.14
SD-6	x 13	aggregate	x 13	bucket	no figure

Table 5.2: Computational tests of the combined routing/loading map

distribution; in the preceding experiments these path fractions were exactly 1/11 by construction. The preceding path choice process was deterministic, whereas here it is stochastic.

Vehicles allocated to a path by the stochastic routing map are moved through the network by the network loading map. Model outputs reflect the combined stochasticity introduced by both the routing and the network loading maps. As was seen in the preceding section, the amount of stochasticity that the loading map contributes to model outputs depends on the link capacity rounding procedure. Both bucket and random rounding were investigated here, as noted in the discussion of each particular run.

Table 5.2 summarizes the runs that were made to investigate the composite routing/loading map. Again, graphics are grouped at the end of this chapter.

Run SD-1 In the first experiment, an OD flow of 39,600 vehicles/hour over 240 seconds was split by a disaggregate routing map application among the 11 OD paths in the test network (for a mean path flow rate of 3,600 vehicles/hour), and moved over the network to the destination. All links had full (i.e., 3,600 vehicles/hour) capacity. Bucket rounding was used for exit capacity determination. Five replications were performed. Figure 5.11 presents the results of this run.

Figure 5.11(a) shows the cumulative path departures on each of the eleven paths for the different replications. (Because of the problem data, the actual number of vehicle departures per path is either 0 or 1 per time step; a plot of this quantity does not give visually intelligible results.) Variability in the path flows is clearly evident, as is noise on the individual path flow trajectories themselves.

These path flows translate into the link flows shown in Figure 5.11(b). As expected, the additive effects of the path flow variations produce relatively greater variations in link flows than were seen in the preceding experiments. This is particularly so on links (those such as 1 and 3 near the origin and 6 and 13 near the destination) traversed by many paths. Links traversed by a lesser number of paths tend to exhibit somewhat less stochasticity in their volumes, although it is noticeable that the level is still considerably higher than that due to the network loading process alone.

The path flows produced in this experiment build up on links 1 and 3, which meter them to the rest of the network. As a result, these two links experience significant amounts of queueing and congestion delay. Convergence of traffic near the destination on links 6 and 13 also produces some congestion delays, but the rest of the network is relatively free of congestion. The amount of congestion delay on links 1, 3, 6 and 13 changes considerably between replications. Figure 5.11(c) illustrates these results.

Finally, the cumulative effects of link traversal time stochasticity can be seen in the trajectories

of dynamic path times shown in Figure 5.11(d). Although some path time trajectories appear almost deterministic, for others different replications give times that differ by nearly 100%.

Run SD-2 As in the test of the network loading map, a second run was carried out here with a total OD flow rate of 19,800 vehicles/hour (mean path flow rate of 1,800 vehicles/hour) and link capacities of 1,800 vehicles/hour. Again, bucket rounding was used to determine link exit capcities, and five replications were performed. Results are shown in Figures 5.12(a) through 5.12(d).

The graph of cumulative departures by path in Figure 5.12(a) generally resembles that in Figure 5.11(a). *A priori*, one might expect slightly more relative variability in the path flows obtained here because the coefficient of variation of the binomial distribution is 8.7% in this run *vs.* 6.2% in the case of higher OD flows; but this difference is not apparent in the samples drawn here. Similarly, link volumes (Figure 5.12(b)) do not exhibit noticeably more variability than they did in Figure 5.11(b).

Link and path times, however, are considerably more random than they were in the higher volume case (Figures 5.12(c) and 5.12(d) *vs.* Figures 5.11(c) and 5.11(d)). This is probably due to the fact that a smaller number of individual vehicle time measurements are involved in the averaging process.

Run SD-3 A run was made to investigate the combined effects of the independent splitting rate application and random rounding. Figures 5.13(a) and 5.13(b) show the main results from this run, which used the 19,800 vehicles/hour OD flow rate and 1,800 vehicles/hour link exit capacity. It can be seen that, compared to the preceding run, there is generally more noise in individual link volume and traversal time trajectories, although the amount of variability between replications is not noticeably different. The greater high frequency variability is probably a result of greater fluctuations in link inflows and outflows from one time step to the next when random rounding is used.

Run SD-4 A run was also made to investigate the effects of aggregate rather than individual application of path splits to departing trips. Bucket rounding was used to determine link exit capacities. As expected, when the total OD flow rate was 39,600 vehicles/hour (1 vehicle/second on each path), the model outputs were identical to those presented in Figures 5.8(a) through 5.8(c) in the discussion of run NLM-1. (The plots for this run are not shown in order to save space.) In the former run the path flows were input whereas in this run they were computed, but the end result was the same. In this case, then, stochasticity in the routing map can be completely suppressed by aggregate application of path splits.

Run SD-5 However, with the same problem parameters as in run SD-4 but an OD flow rate of 19,800 vehicles/hour (an average of 0.5 vehicles/second departing on each path), the outputs were again noticeably stochastic. Figure 5.14 shows the variability in the path flows, and these produced correspondingly stochastic link flows, and link and path times. It might at first seem surprising that aggregate application of the path splits should lead to stochastic path flows. The problem is that the fractional numbers of departures per path that result from application of the splits to the OD flow rate are rounded using bucket rounding, but this is done independently in each time step. Bucket rounding ensures that, in each time step, the sum of all path flows equals the time step's

OD flow; but it does not constrain individual path flow sums across time steps. More complex matrix rounding procedures would probably reduce or eliminate this effect.

Run SD-6 When the individual path flows to be rounded are large in magnitude, the relative adjustment due to rounding is small, and both the individual and cumulative flows are likely to be close to their correct values. This was verified by re-running the model after scaling the problem data by a factor of 13 (so that the individual path flow rate was 6.5 vehicles/sec and the storage and exit capacities were factored accordingly); apart from the magnitude difference, the outputs were indistinguishable from those shown in Figures 5.8(a) through 5.8(c). (Again, the results are not shown for reasons of space.) However, with small flows, as in the present case (0.5 vehicle/second per path), the adjustments are relatively much more important, and produce significant fluctuations in the path flows and resulting conditions.

The principal conclusion of this section is that independent path selection by individual drivers introduces considerable stochasticity into model outputs via the multinomial variations in path flows per time period and the follow-on effects of these flows propagating through the network. This conclusion holds regardless of the assumptions made concerning the rounding of link exit capacities, although random rounding of capacities appears to superpose additional noise on the variability that results from the path choice process. Aggregate application of path splits at high flow values can virtually eliminate the stochasticity in the composite routing/loading map, but at low flows the stochasticity generally remains even when this procedure is used.

5.5 Gibbs sampling solution of a stochastic $D \circ G \circ S$ model

It was argued in Chapter 3 that, in the presence of stochasticity in the framework component maps, the outputs of the composite maps must be considered stochastic processes. To investigate this claim and develop an understanding of the nature of the stochastic process outputs, Gibbs sampling (discussed in Section 4.1.1) was used to compute the stochastic process outputs for the fully stochastic $D \circ G \circ S$ map.

Implementation and run details Implementation of the Gibbs sampler was relatively straightforward. Input data consist of a network description, and a schedule of OD vehicle departures by time step; in the run carried out here, there were 3 departures per time step (one second) over an inflow period of 1200 steps. All vehicles were guided. The process is initialized by randomly allocating a path to each vehicle departing in each time step. Then the first vehicle departing in the first time step is considered.

In the basic step, a single departure (one vehicle in a given time step) is considered. All other vehicles are dynamically loaded on the network, taking account of their departure time and allocated path. The loading results in trajectories of dynamic link volumes and traversal times. The link time (network condition) data is used by the guidance map to generate guidance messages, and the routing map converts the messages into path splits. The considered vehicle then selects a path in accordance with the splits. Processing then proceeds to consider the next vehicle: after all vehicles departing in a time step have been considered, the subsequent time step is then selected. One cycle (iteration) of the Gibbs sampling process consists of a pass through all the vehicle departures in this

way, and Gibbs sampling requires many iterations to determine the stochastic process distribution fixed point.

It can be seen that the procedure is based on evaluations of the D ∘ G ∘ S composite map. This formulation seemed most amenable to Gibbs sampling methods because it is straightforward to compute an individual vehicle's path choice probabilities, given the path choices made by other vehicles (and the path times that result), by simply applying the routing map described in Section 5.2.3. Approaches based on the other two composite map formulations do not enjoy this property.

The process was allowed to "burn in" for 25 iterations, after which it was run for an additional 100 iterations. Statistics on path choices and link volumes by time step were collected during the 100 iterations and then plotted. Sample outputs are presented in Figure 5.15, at the end of this chapter. It is of particular interest, in examining these results, to look for evidence that relates to the validity of a mean trajectory approximation. An example of such evidence might be a framework variable trajectory with, in each time step, bimodal distributions having little probability density around the mean.

Results and conclusions Figure 5.15(a) shows mean path split trajectories computed from the 100 realizations. The dispersion pattern is indicative of the variability of the stochastic process results from one time step to the next.

Figure 5.15(b) shows the sample frequency distribution of volumes by time step for link 2. The axis with values from 0 to 1800 corresponds to the simulation time steps. The axis with value from 0 to 200 corresponds to the link volume in a particular time step. The third (vertical) axis shows the frequency (number of times out of 100 replications) with which the particular volume was encountered in a particular time step. Note that a slice through the graph at a given time step provides the sample frequency distribution for different link volume values.

Although the statistics were collected for every simulation time step, Figure 5.15(b) presents results for every 50 time steps and interpolates between them. Unfortunately, the graphs that were obtained when presenting the results at closer intervals were too densely drawn to be intelligible. On the other hand, some of the peaks that appear on the graph are artifacts of the interpolation process.

It can be seen that for the first 700 time steps or so, most of the link volume trajectories are concentrated in a narrow band of values centered around a volume of about 60 vehicles. This situation changes dramatically between steps 800 and 1300, where the trajectories uniformly cover a range of volumes between roughly 60 and 120 vehicles. After time step 1200, the volumes generally decrease to 0 in trajectories that tend to decrease in spread as the mean value itself decreases.

In the first and last portions of the plot, the trajectories are fairly concentrated while in the middle portion they are much more spread out. What seems clear is that the trajectories are not organized in a bimodal or multimodal pattern. This conclusion is reinforced by examination of plots for the other links (not shown here) and by generation of histograms of volume values from slices of the trajectory plots at particular time steps (also not shown).

Reoptimization of dynamic network loading Execution of Gibbs sampling for this problem requires literally tens of thousands of D ∘ G ∘ S composite map evaluations. In the full information model used here, each such evaluation consists of a dynamic network loading followed by a recalculation of the path splits; dynamic network loading is by far the more computationally burdensome step.

Successive evaluations differ only in the path choice decision of a single vehicle yet, by the logic of the simulator, require the loading and path split calculation of the entire set of vehicles.

A rather similar situation arises in some iterative static network algorithms: each iteration requires a minimum path calculation with problem data that differ only slightly from those of the previous iteration. In these cases, it has been found to be much more efficient to "reoptimize" the prior problem's minimum path solution rather than computing the minimum paths *ab initio*. Reoptimization generally involves initializing the minimum path algorithm's data structures (for example, node labels) with values retained from the prior solution, and then adjusting these values until a correct solution for the current problem data is obtained.

In dynamic network models, a reoptimization procedure for the dynamic network loading step would be equally advantageous. The potential benefits of reoptimization are perhaps easiest to see in the case of Gibbs sampling, where the problem data commonality between successive iterations is maximum (since all path choices but one remain unchanged between iterations). However, there are likely to be computational gains in many iterative algorithms having DNL as a subproblem, including in particular those for analytical or simulation-based dynamic traffic assignment and guidance generation problems. The DNL reoptimization problem does not appear to have been recognized or reported on in the literature, and is a subject for future research.

5.6 Computational tests of guidance generation algorithms

The absence of features such as multimodal probability distributions of link flows suggests that the stochastic guidance model with its general stochastic process stationary solution might be approximated by a deterministic process whose outputs are affected by zero-mean unimodal noise (a *quasi-deterministic* model) having a deterministic fixed point trajectory. Stochastic approximation methods can then be applied to solve the quasi-deterministic model for a fixed point. It must be emphasized, however, that this approach is an engineering approximation. As will be seen, the approach gives reasonable results in all situations examined, but its validity and conditions of applicability need to be subjected to rigorous mathematical analysis. This is a subject for future research.

In this section, two stochastic approximation fixed point algorithms are applied to find fixed points of the three different framework formulations developed in Chapter 3. A large number of runs was made in the course of testing the various formulations and solution algorithms considered below. Many of these runs examined the effects of varying different model and problem parameters such as time and flow scaling, presence or absence of incidents, etc.; they served to validate *a priori* qualitative expectations about the effects of changes in parameter values and to verify the overall reasonableness of the model's behavior. It would be mind-numbing to present here the detailed results of all of these runs.

For this reason, it was decided to select and present runs that highlighted the differences between two contrasting modeling approaches: one involving low stochasticity model relationships and the other involving high stochasticity relationships. The intent underlying this decision was to exhibit the range of output behaviors that a quasi-deterministic model (a model with deterministic but noise corrupted outputs) can produce as the degree of stochasticity in the model relationships is changed. Other aspects of the model may also tend to exaggerate the effects of stochasticity: the (all) shortest path routing assumption and 10% switching penalty, the 1 second interval between

message updates, the use of a relatively small network with a single OD pair, etc.

The simulator parameters used in these runs were not chosen in an attempt to produce the most accurate possible representation of reality. Indeed, since good quality empirical data on many of the questions important in route guidance modeling—issues such as driver response to guidance messages, or the actual nature and extent of stochasticity in route choice and vehicle propagation—are currently scarce, it is not even possible to say which set of model parameters leads to the most realistic results.

However, since the simulator is capable of producing a wide range of degrees of stochasticity, it could potentially be "calibrated" to better reflect reality when such data became available. Furthermore, the techniques it uses to influence the degree of stochasticity in its outputs could be applied to the same purpose in other, similar traffic simulation models.

Runs investigating the the low stochasticity model applied path splits at an aggregate level and used flow and time scaling factors of 10. In contrast to this, runs with the high stochasticity approach applied disaggregate path splits and used a flow scale factor of 1 (although the time scale of 10 was maintained for comparability.) While the low stochasticity approach does not always succeed in completely eliminating noise and variability from model outputs, it often is reasonably effective at doing so, and it certainly serves to illustrate the significant differences in the outputs produced by the two different degrees of stochasticity in model relationships.

5.6.1 Composite network condition formulation

This section describes guidance generation computational tests made using the on-the-fly implementation of the $S \circ D \circ G$ composite network condition map. Guidance was generated by approximating a fixed point of the composite map. Progress of the fixed point algorithm towards convergence was measured using a link time inconsistency norm, defined as

$$IC_C = \left[\sum_{\ell} \sum_{t} \{C_\ell(t) - [S \circ D \circ G(C(\cdot))]_\ell(t)\}^2 \right]^{\frac{1}{2}}$$

where, clearly, the inconsistency norm attains the value 0 at a deterministic fixed point of the $S \circ D \circ G : C \mapsto C$ map. Convergence to a constant but non-zero value could be (but is not necessarily) an indication that the stochastic process of link condition trajectories generated by successive iterations of the solution algorithm has become stationary. Other possible behaviors of the IC_C norm and explanations for them are discussed below.

If IC_C were divided by the number of time periods and the number of links, the result could be considered a root mean square (RMS) discrepancy between the "input" and "output" link conditions, which are measured in units of seconds, for each link and time period. This quantity cannot be rigorously interpreted, however. Conditions on an empty network are by definition free-flow conditions, and comparisons of such conditions result in structural zeros. These zeros do not contribute to the total norm value, of course, but dividing the total norm by a count that includes the zeros would produce misleading results. Nonetheless, dividing the norm by the product of the number of links and the (approximate) number of time steps needed to clear the network can give a very rough indication of the RMS discrepancy between simulator inputs and outputs at the level of an individual link and time step.

As an alternative consistency measure, a weighted norm in which each link and time period's discrepancy is multiplied by the corresponding traffic volume was also considered but rejected. The problem with this measure is that link volumes may change between replications, making impossible a meaningful comparison of norm values.

For these reasons the IC_C norm, which is simply a vector 2-norm $\|C - S \circ D \circ G(C)\|$ of the difference between "input" and "output" conditions, was used in all tests of fixed point solution methods for the composite link condition map. Its value is reported in units of seconds of time.

Before concluding this discussion of the IC_C norm, it is worthwhile to examine the effect of model stochasticity on the results of the norm computation itself. One way of examining this effect is by applying the composite link condition map repeatedly to a fixed input C^0 to generate a sequence $C^i = S \circ D \circ G(C^0)$, $i = 1 \ldots$ of link condition trajectory realizations, and then examining the variability of the resulting norm values $IC_C^i = \|C^0 - C^i\|$. Figure 5.16(a) does this using as C^0 the link time trajectories computed in the 100th iteration of the solution by the MSA of one of the problems discussed below (run SDG-2); the problem involved disaggregate path split application and fractional link exit capacities per time step that were bucket rounded. The plot shows the individual norm values computed over 100 replications, as well as the mean and sample cumulative distribution of the norm's value. The full range of values is roughly 33% of the mean, with most values are clustered in a range approximately 18% of the mean. Similar results are shown in Figure 5.16(b), where the time trajectories from iteration 1 in the solution of this problem were used as C^0. Here the full range is approximately 21% of the mean, with most values clustered in a range that is roughly 14% of the mean. If this amount of variability is present in evaluations of $\|C^0 - S \circ D \circ G(C^0)\|$ for a single value C^0, it is clear that interpreting the convergence properties of a fixed point algorithm by examination of $\|C - S \circ D \circ G(C)\|$ must take into account the inherent stochasticity of the problem.

5.6.1.1 MSA algorithm

Recall from the discussion in Section 4.2.2 that the MSA fixed point solution algorithm computes the conditions C^{i+1} in iteration $i+1$ in terms of those in iteration i as

$$C^{i+1} = C^i + \frac{1}{i+1}(S \circ D \circ G(C^i) - C^i);\ i = 0 \ldots$$

The correction $S \circ D \circ G(C^i) - C^i$ that the MSA applies to iteration i's estimate C^i of the fixed point solution is weighted by the factor $1/(i+1)$, with the overall correction becoming negligible at some point.

In all runs discussed below, the initial link time trajectory estimate, C^0, was taken to be free-flow conditions on all links in all time periods. This may not be a valid cost vector, in the sense that there may be no C^* such that $S \circ D \circ G(C^*) = C^0$; note, however, that the first iteration of the MSA simply uses C^0 as a means to generate a feasible C^1 (i.e., the link condition trajectories corresponding to a routing of OD demand using guidance derived from free-flow conditions), then discards it. Thus, the possible infeasibility of C^0 is not an issue.

Table 5.3 lists the runs that were made to investigate the performance of the MSA algorithm when applied to generate guidance by finding fixed points of the $S \circ D \circ G$ composite link condition map. All of these runs used a 1 second simulation time step, employed bucket rounding to determine link exit capacities, and consisted of three replications of 300 iterations each. In all runs, the time

Run	*Network*	*Guidance*	*Path Splits*	*Flows*	*Times*	*Figure*
SDG-1	full capacity	100%	aggregate	x10	x10	5.17
SDG-2	full capacity	100%	disaggregate	x1	x10	5.18
SDG-3	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.19
SDG-4	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.20
SDG-5	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.21
SDG-6	link 201-200 at 900 v/h	25% + VMS	disaggregate	x1	x10	5.22

Table 5.3: Computational tests of the MSA applied to $S \circ D \circ G : C \mapsto C$

scale factor was set to 10 (i.e., computed link traversal times were stored and reported to the nearest 0.1 second), and the guidance messages were updated every second.

Results of each run are presented in a sequence of figures that reflects the steps of the processing. Results of the three replications appear on each plot. The sequence of figures is:

(a) a plot of the IC_C consistency norm for each iteration;

(b) the link traversal time trajectories, as computed after 300 iterations of the MSA. These trajectories are *input* to the final evaluation of the $S \circ D \circ G$ map, the results of which are presented in the remaining figures (c) through (f) of the sequence;

(c) the path time trajectories computed from (b). Although these path times are not, strictly speaking, the guidance provided to drivers, they are derived from that guidance and used in the routing calculation;

(d) the cumulative path flow trajectories that result after path splits are obtained from the path times in (c);

(e) the link volume trajectories that result from loading the path flows in (d); and

(f) the corresponding link time trajectories, determined simultaneously with link volumes by the dynamic network loading map.

It can thus be seen from the sequence of figures how the $S \circ D \circ G$ map transforms an input link time table into an output table. Comparison of the input and output tables in figures (b) and (f) provides a visual indication of how closely the process has found a fixed point of the map.

Results of the individual runs are discussed below and the output graphs are collected in Section 5.8 at the end of this chapter.

Run SDG-1 An initial application of this solution method was made to an "easy" problem: the test network had all links at full capacity; link capacities were bucket rounded; the OD flow rate was 10,800 vehicles/hour over 1,200 seconds; and all trips received pre-trip guidance. Note that, by this last assumption, the problem is very similar to a full-information dynamic equilibrium assignment problem. Simulation parameters were set in this problem to provide low stochasticity: the flow scale was 10 (vehicles were divided into 10 independent fractions) and path splits were applied to aggregate flows.

The OD flow rate amounts to 3 vehicles/second entering the network. With deterministic path choice and flow propagation relationships, this traffic should divide equally between the upper,

middle and lower paths (paths 0, 4 and 7, respectively) and be able to traverse the network without incurring congestion. Figures 5.17(a) through 5.17(f) show the results that were obtained.

Figure 5.17(a) shows the IC_C norm of the difference between $S \circ D \circ G$ map inputs and outputs in each iteration and for each of the three replications. It can be seen that, apart from some fluctuations in the first few iterations, the norm decreases monotonically, attaining a final value of around 300 seconds by the 150th iteration. The final norm is 300 seconds rather than 0 seconds in part because link times are maintained to the nearest tenth of a second; greater precision would reduce the norm value. Another reason for not attaining a norm value of 0 seconds is the small amount of stochasticity that remains even in the quasi-deterministic model. Nonetheless, considering that this is a root sum squared discrepancy over 15 links and roughly 1200 time periods, the value is completely negligible.

Figures 5.17(b) and 5.17(c) show the computed link traversal time trajectories and the corresponding path time trajectories at the beginning of the 100th iteration. The link traversal times shown in 5.17(b) are the averaged values obtained by repeated application of the MSA iterative step; they are *input* to the $S \circ D \circ G$ map in the final (300th) iteration. The input link traversal time trajectories should be distinguished from the link time trajectories computed by the $S \circ D \circ G$ map from these inputs, as shown in 5.17(f). The latter are the "raw" outputs of the composite map evaluation, and are not averaged or filtered in any way; consequently, they generally exhibit a greater amount of noise. The path times are derived from the input link times in a straightforward manner by the guidance map. The routing map uses these to determine path splits. It can be seen that the trajectories are essentially constant over the entire simulation period, and that they present no noise within or variability between replications.

Figure 5.17(d) presents the cumulative path flow trajectories that result when path splits are computed from the path times, and are applied to departing vehicles in an aggregate manner. The plot shows the cumulative departures on each path and up to each time step. It can be seen that the flow is concentrated on paths 0, 4 and 7, with lesser amounts on paths 2, 5, 6 and 9 and negligible amounts on the rest. In this problem, all links operate in free flow conditions and have the same free flow speed, so path splits are determined enitrely by path lengths. Paths 0, 4 and 7 each have a length of 3.0 km; the next shortest paths (paths 2, 5, 6 and 9) have a length of 3.5 km. The path flows result from the properties of the routing map discussed in Section 5.2.3.

Figure 5.17(e) shows the link volume trajectories that result from the propagation of the path flows through the network. The trajectories are quite regular; they exhibit no visible variability over time or across replications. The *output* link traversal time trajectories, shown in Figure 5.17(f), are similarly free of noise. Since these coincide very closely with the input link time trajectories, it is clear that a deterministic fixed point has been found.

In conclusion, this problem was accurately solved by the simple simulator applying the MSA algorithm to find a fixed point of the composite link condition map.

Run SDG-2 Figures 5.18(a) through 5.18(f) present the results for the same "easy" problem data that were used in run SDG-1. In this case, however, simulation parameters were chosen to provide high stochasticity: disaggregate path splits were applied (i.e., the results of the path choice model are sampled independently by each vehicle choosing a path from the origin) and the flow scale was 1 (vehicles were not subdivided). It can be seen that the solution is both qualitatively and quantitatively different from the deterministic path choice case just considered.

Figure 5.18(a) shows that, after sharp jumps during the first 15 or so iterations, the IC_C norm enters a phase of general decrease from iteration to iteration, with fluctuations that do not reverse the overall downward trend. After around 150 iterations the norm appears to reach a range of final values between 750 and 1250 seconds, or roughly 0.1 second RMS per link and time step. The fluctuations in the final values indicate that the MSA has converged to a stationary process: in successive iterations, process values such as link times and volumes are being drawn from the same distributions, even though the values themselves are random and not generally the same.

Link traversal time trajectories computed for input to the final (300th) iteration are shown in Figure 5.18(b). They are no longer all flat, as they were in run SDG-1, because surges in path and link inflows produce small amounts of queuing and congestion delays. Minor noise is evident although there do not appear to be systematic differences across replications. The path times (Figure 5.18(c)) also exhibit minor noise. It might be expected that path times would exhibit less relative variability than link times, but this is not clearly visible in the plots.

OD demand is now split across all available paths (Figure 5.18(d)) in varying amounts, although most of the demand is concentrated on paths 0, 4 and 7 as before. Note that some variability between cumulative path flow trajectories in different replications is apparent.

The variability is much more visible in the plot of link volume trajectories (Figure 5.18(e)), where three distinct trajectories (corresponding to the different replications of the solution process) can be seen on some links. Compared to run SDG-1, the much greater variability and level of noise in link volume trajectories is striking.

Figure 5.18(f) shows the link traversal time trajectories output by the composite link condition map; these are, of course, determined simultaneously with the link volume trajectories just discussed. Again, the greater variability and noise level of this output is striking, and there is evidence of three distinct trajectories. In the MSA algorithm, this output trajectory is combined in a weighted average with the current solution estimate to produce the next iteration's estimate, which is less noisy because of the effects of averaging. The comparison of the output link traversal time trajectory with the corresponding input trajectory produces the norm that is reported in the convergence plot.

Run SDG-3 To test the software on a somewhat more challenging problem, the network used in the preceding runs was modified by reducing the capacity of link 6 (102–200) from 3,600 vehicles/hour to 900 vehicles/hour for the duration of the run. The routing of traffic away from link 6 is sufficient to cause spillbacks on links 9 (201–200) and 13 (302–200). Apart from this change, run SDG-3 replicates run SDG-1 in that it applies low stochasticity relationships, uses bucket rounding of link exit capacities and scales times by a factor of 10. Figure 5.19 shows some of the results of 300 iterations of the MSA applied to this problem.

It can be seen from Figure 5.19(a) that the value of the IC_C norm generally decreases with increasing iterations, although because of small fluctuations the convergence is not strictly monotonic. At the 300th iteration, the norm value is around 1000 seconds. The norm appears to be approaching an asymptote, although additional iterations might possibly allow a further decrease.

Averaged link traversal time trajectories (Figure 5.19(b)), the inputs to the MSA algorithm in the 300th iteration, exhibit very limited noise. There is no evidence of multiple trajectories. The path times (Figure 5.19(c)) derived from these appear similarly deterministic.

Cumulative path flow trajectories (Figure 5.19(d)) are also deterministic, as would be expected

from the run parameters. However, they lead to link volume trajectories (Figure 5.19(e)) that on some links exhibit both slightly varying trajectories (a result of path choice stochasticity) and high frequency noise (a result of propagation stochasticity). The link traversal time trajectories output by the $S \circ D \circ G$ map in this iteration (Figure 5.19(f)) exhibit comparable variability, and are characterized on some links both by different trajectories in different replications, as well as by considerable noise on most links within individual replications.

In conclusion, although the MSA appears by 300 iterations to have converged to a stable link traversal time trajectory, the presence of the bottleneck and the associated spillbacks introduce some stochasticity into the problem, as evidenced by the variability in the output link flow and time trajectories, and the fluctuations in the computed IC_C norm values even after convergence.

Run SDG-4 Run SDG-4 resembled run SDG-3 except that simulation parameters were set to provide high stochasticity. Figure 5.20 presents the results of 300 iterations of the MSA for this problem.

Compared to run SDG-3, there is much greater variability in the IC_C norm values (Figure 5.20(a)). After around 100 iterations, there appears to be an inflection in the initial generally downward trend of norm values and fluctuations of the norm are mostly limited to the range of roughly 2500–5500 seconds. In the presence of such noise, it is difficult to see whether the trend continues with a slight downward slope, or has become horizontal. Again, the indication is that the MSA has (almost) converged to a stationary stochastic process from which the trajectories of link times and other framework variables are drawn.

The computed link traversal time trajectories in this run (Figure 5.20(b)) resemble in a general way those determined in run SDG-3. However, the trajectory features in this run have a more "rounded" aspect, compared to the sometimes abrupt changes of slope that characterized the trajectories in run SDG-3. The same comment holds for the path time trajectories (Figure 5.20(c)).

Differences can be seen, however, in the cumulative path volume trajectories (Figure 5.20(d)) where, in this run, traffic is more dispersed among paths and presents more variability between replications. Link volume trajectories (Figure 5.20(e)) are highly stochastic; most links exhibit multiple volume trajectories and considerable high frequency noise. This is also true of the output link traversal time trajectories (Figure 5.20(f)). The computed input time trajectories shown in Figure 5.20(b) appear to approximate a "mean" trajectory of the output times shown here. However, the various output trajectories exhibit systematic deviations away from the mean and it is not clear that the mean is a particularly good representation of the output trajectories with high variability.

The amount of stochasticity in the computed solution is considerable: the solution process has converged to a fixed point but, because of the noise in the composite map evaluations, the output trajectory does not appear to be particularly close to the input trajectory, and successive evaluations of the composite map, all using the same solution as input, produce very different output realizations. Similar results will be seen below in a number of applications of high stochasticity maps. As was discussed above, such results may well exaggerate the degree of stochasticity actually present in road traffic networks. A few suggestions are made in Chapter 6 about how guidance systems might address situations with such high levels of stochasticity. However, this thesis is concerned with anticipatory guidance modeling and computation, not with guidance system design; detailed investigation of appropriate system designs in such situations is a subject for future

research.

Run SDG-5 The final two runs addressed a true route guidance generation problem. In these runs, the capacity on link 6 (102-200) was again restricted to 900 vehicles/hour. Only 25% of the OD trips received pre-trip guidance in the form of anticipatory dynamic link traversal times. The remainder were unguided and based their trip decisions on free-flow travel times (note that the restricted capacity of link 6 does not affect its free-flow time). There was also a short-range guidance beacon (e.g., a VMS) on link 1 (100–101). Unguided trips that passed this beacon received dynamic link time information identical to that received by guided trips at the origin; each such trip individually reconsidered and possibly reselected its path to follow on to the destination, as described in Section 5.2.2.

Note that unguided trips can be considered as background traffic, since their path choice is not affected by network traffic conditions. However, their propagation through the network is governed by the same logic that applies to guided vehicles, and so may be more or less stochastic depending on problem parameters. Of course, this stochastic flow interacts with the propagation of guided vehicles as well.

In run SDG-5, simulation parameters were set to provide low stochasticity. Figure 5.21 presents the results.

Figure 5.21(a) shows that the MSA has reasonable convergence properties for this problem. It displays a generally downward trend. After around 200 iterations it attains its final range of values, 750–1750 seconds. The norm values exhibit more fluctuation than they did in, for example, run SDG-3; this is no doubt a consequence of the independent path switching decisions made by unguided drivers that pass the VMS on link 1.

The computed link traversal and path times (Figures 5.21(b) and 5.21(c)) are somewhat different from those determined in run SDG-3, as is to be expected. Nonetheless, the time trajectories appear logical (for example, congestion occurs on links where unguided trips make suboptimal choices) and are essentially deterministic.

The path volume trajectories reported in Figure 5.21(d) combine both the guided and unguided trip departures at the origin, but do not take account of the path switches on link 1.

Link volume trajectories (Figure 5.21(e)) are different from those obtained in run SDG-3, as is again to be expected. However, the trajectories in this problem do not present noticeably more variability than did those in the earlier run. Similarly, the output link time trajectories (Figure 5.21(f)), while different from those in run SDG-3, can be understood in terms of the suboptimal choices of unguided trips, and the path switching that occurs on link 1. The amount of variability in model outputs is again relatively limited.

In conclusion, the MSA appears able to solve an aggregate $S \circ D \circ G$ formulation of a true route guidance problem, involving short-range information sources and guided and unguided vehicles.

Run SDG-6 Run SDG-6 was similar to run SDG-5 except that the simulator was set to provide high stochasticity. Figure 5.22 presents the results.

Results of the IC_C norm computation (Figure 5.22(a)) show immediately the differences between this run and run SDG-5. The norm appears to settle after about 100 iterations into a situation of relatively large fluctuations, in the range of 2000–6000 seconds). Interestingly, this range is not much greater than than obtained in the solution of run SDG-4, and the convergence to the final

range of values appears to be slightly quicker, perhaps because of the smaller number of vehicles with an effective path choice.

The computed link traversal time trajectories (Figure 5.22(b)) resemble in a general way those obtained in run SDG-5; as noted before, the results from the stochastic run tend to be more "rounded" (i.e., with less abrupt slope changes) than those computed for the deterministic run. Path time trajectories (Figure 5.22(c)) are what would be expected from the computed link times; in particular, they do not exhibit any noticeable degree of stochasticity.

Path flow trajectories (Figure 5.22(d)), on the other hand, do show variability between replications, with the greatest variability appearing to occur on paths 0, 2 7 and 9.

Link volume trajectories (Figure 5.22(e)) are highly variable, with most links exhibiting multiple trajectories and high levels of noise. This results from a combination of the path flow variability, the effects of path switching on link 1, and the greater degree of randomness in the vehicle propagation relationships due to the flow scale factor of 1.

Similarly, the output link traversal time trajectories (Figure 5.22(f)) are highly variable. As noted before, it is not clear than a "mean" trajectory is a useful concept in this situation.

5.6.1.2 Polyak iterate averaging algorithm

As discussed in Section 4.2.3, Polyak's iterate averaging algorithm is a two-pass method. The first pass resembles the MSA except that step sizes are larger; this allows the algorithm to explore the solution space more aggressively but leads to greater variability in the outputs. The second pass is carried out offline (i.e., without influencing the first pass); it calculates a simple average of the iterates that are generated by the first pass. The average calculated by the second pass at termination is the fixed point solution estimate.

When applied to find a fixed point of the $S \circ D \circ G$ map, the Polyak algorithm can be written

$$\begin{aligned} C^{i+1} &= C^i + \alpha^i (S \circ D \circ G(C^i) - C^i) \\ \tilde{C}^{i+1} &= \tilde{C}^i + \frac{1}{i+1} C^{i+1} \end{aligned}$$

where $\tilde{C}^n$ in the final iteration n is the fixed point estimate.

In practice, iterate averaging (computation of the $\tilde{C}^i$s) is only started after the MSA-like step shows signs of stabilizing. (This is called the *window of averaging*.)

Step sizes α^i are frequently generated by a formula such as $\beta i^{-\gamma}$, with $\gamma \in (1/2, 1)$. A common choice is $\gamma = 2/3$.[8] In all runs presented here, $\beta = 1$ and $\gamma = 2/3$.

The software system required very few changes to implement Polyak averaging. When the method is selected, the software computes step sizes as described above, and writes to a file the link time tables computed in each iteration by the MSA-like step. Apart from the step size difference, processing is in all respects identical to that carried out for the MSA algorithm. After a specified number of iterations of the MSA-like step have been performed, the software begins to re-read the link time table file. A user input specifies how many of the initial iterations are to be skipped over

[8] Step sizes are sometimes computed using $\beta \ln i/i$. However, the step sizes generated in the first few iterations by this formula are inappropriate for the fixed point calculation required here. For example, the first iteration step size is 0, so it does not eliminate the initial infeasible solution. Of course, this could be corrected on an *ad hoc* basis if required, but it is more straightforward to use $i^{-2/3}$ without correction. After five iterations, the step sizes generated by $\ln i/i$ and $i^{-2/3}$ are nearly identical.

Run	*Network*	*Guidance*	*Path Splits*	*Flows*	*Times*	*Figure*
SDG-3-pol	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.23
SDG-4-pol	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.24
SDG-5-pol	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.25

Table 5.4: Computational tests of Polyak averaging applied to $S \circ D \circ G : C \mapsto C$

to reach the averaging window. The software then computes a simple running average $\tilde{C}^i$ of the successive link time tables C^i. After each successive link time table is incorporated in the average $\tilde{C}^i$, the software evaluates $S \circ D \circ G(\tilde{C}^i)$ and the IC_C inconsistency norm $\|S \circ D \circ G(\tilde{C}^i) - \tilde{C}^i\|$ of the Polyak method estimates. These evaluations are done solely to track the progress towards convergence of the Polyak method, but are not otherwise needed; in particular, they would not be carried out in a production system.

The method was applied to a number of the same problems to which the MSA was applied, as described in the preceding section. With one small exception, all of the problem data were identical in corresponding MSA and Polyak averaging runs. The difference is that the simulation was allowed to run for a longer number of time steps when Polyak averaging was used. This was required because the larger step sizes sometimes resulted in more "extreme" link time tables; when guidance was generated and vehicles routed according to these tables, longer queues would sometimes build up and the network would take longer to clear than was the case with the MSA. In all cases, however, the simulation began with an empty network and continued until all vehicles had exited; results were thus strictly comparable. As with the MSA runs, the method was run for three replications of 300 iterations each; the second pass began averaging results from an averaging window starting in iteration 50.

Runs made to test the application of Polyak averaging to finding fixed points of the $S \circ D \circ G$ map are listed in Table 5.4.

The following paragraphs discuss salient aspects of the results obtained in each of these runs. Discussion is limited to consideration of the convergence behavior of the IC_C norm; graphs of the value of this norm by iteration in the first and second passes are included at the end of the chapter. It should not be forgotten that the actual outputs of the solution process are link time tables. Consistency norms are a highly aggregate way of measuring the differences between any two such tables. Furthermore, in the second pass the calculation of this norm requires an $S \circ D \circ G$ evaluation that would not otherwise be performed and, as was discussed in Section 5.6.1, this evaluation is itself tainted by noise.

For these runs, however, there is no benefit in presenting or discussing other outputs of the procedure because the graphs of the link traversal time trajectories computed after 300 iterations by the MSA and by Polyak averaging were in all cases indistinguishable. Detailed outputs of the two procedures at intermediate iterations were not compared; however, some indication of their properties in this regard can be inferred from the IC_C convergence plots.

Run SDG-3-pol Problem data for this run were identical to those for run SDG-3 discussed above. It is a low stochasticity full information equilibrium analysis of a network on which the reduced capacity of link 6 causes spillbacks and generally complicates the flow pattern.

Figures 5.23(a) and 5.23(b) present the IC_C convergence results for this problem when Polyak

averaging is applied; this can be compared to Figure 5.19(a) for application of the MSA to the same problem. It is immediately seen (Figure 5.23(a)) that by roughly 75 iterations the first pass of Polyak averaging drives the IC_C norm to a mean value of less than 500 seconds, or less than half the norm value attained by the MSA after 300 iterations. The amount of fluctuation in the norm values is greater than was the case with the MSA; this is a result of the more "aggressive" step sizes. The second pass of Polyak averaging (Figure 5.23(b)), which begins averaging the first pass' results at iteration 50, appears to reduce this variability somewhat, although the noise level still is somewhat higher than than obtained in the MSA application. No overall trend in norm values is evident between iterations 75 and 300: the process appears to have converged.

In conclusion, the performance of Polyak averaging for this problem is remarkable: compared to the MSA, better convergence is achieved in less than one-fourth of the number of iterations, at the additional cost of a trivial averaging operation.

Run SDG-4-pol Problem data for this run were identical to those for run SDG-4: a high stochasticity full information problem in which reduced capacity on link 6 complicates the flow pattern.

IC_C norm values for Polyak averaging are shown in Figures 5.24(a) and 5.24(b), and can be compared to Figure 5.20(a) for which the MSA was applied to the same problem.

The first pass results (Figure 5.24(a)) show an inflection point in the initial downward trend at around 50 iterations, after which the trend becomes horizontal or nearly so. The norm fluctuates wildly, with most values in the range 3000–6000 seconds. Norm values obtained in the second pass (Figure 5.24(b)) fluctuate somewhat less wildly, but appear to exhibit a slightly U-shaped convex trend, with a minimum both in the trend and in the apparent intensity of the fluctuations at around 150 iterations.

It is difficult to interpret these results. Clearly, the impressive performance achieved for the quasi-deterministic case is absent here, and it does not seem possible to tell whether results of the MSA or Polyak averaging are to be preferred.

Run SDG-5-pol This run is a true route guidance problem using low stochasticity relationships. Only 25% of vehicles obtain guidance at the origin; the others base their path choice on background (free flow) times. Of these, those who use link 1 (100–101) pass a VMS that also provides guidance. Link 6 again has reduced capacity. This problem is identical to that treated in run SDG-5 above.

Figures 5.25(a) and 5.25(b) present the IC_C norms obtained from Polyak averaging, and can be compared with Figure 5.21(a) for run SDG-5 with the MSA. In run SDG-5, it was seen that the norm settled down in a range of around 750–1750 seconds after 200 iterations. Here, the first Polyak averaging pass succeeds after 50 iterations in driving the norm to a flat range of 250–1750 seconds, and the second pass reduces the variability to a range of 500–1000 seconds, with no overall trend apparent after iteration 50.

Again it is seen that, to achieve a given level of the convergence norm, Polyak averaging requires roughly one-fourth the number of iterations compared to the MSA, with the only additional cost being a trivial averaging of the link time tables generated by the first pass.

In view of the inconclusive results obtained in run SDG-4-pol, it was decided not to carry out a run of a true guidance problem with high stochasticity relationships (corresponding to run SDG-6).

In conclusion, Polyak averaging appears to be considerably more effective than the MSA in finding fixed points of quasi-deterministic $S \circ D \circ G$ maps. It was seen in the problems considered here that, compared to the MSA, Polyak averaging needs one-fourth or less the number of iterations to converge to a comparable IC_C norm value. However, the performance of Polyak averaging when applied to stochastic maps is much less clear; no obvious conclusion regarding the comparative merits of the two methods emerged from the runs carried out here.

5.6.2 Composite path split formulation

A set of runs was made to investigate the properties of the composite path split formulation and solution algorithms for it. These runs parallel in most respects the runs discussed in Section 5.6.1 that investigated the composite network condition formulation.

5.6.2.1 MSA algorithm

When the MSA algorithm is applied to the composite path split formulation, both pre-trip and en route path split tables must be averaged. To start on a roughly comparable basis with the composite network condition formulation, the path splits used in the initial iteration of the algorithm were obtained by assuming free-flow traffic conditions over the network and deriving messages from these.

Progress of the algorithm was measured in terms of the IC_P norm, an analog in the space of path splits of the IC_C norm discussed above. The norm computes the root sum squared difference between input and output path split tables; since table entries are fractions, the norm is dimensionless. The result includes both the pre-trip path split tables and the en route path split tables. In all cases, the IC_P norm computation includes each meaningful entry in the corresponding table; however, because of intrinsic differences between pre-trip and en route tables, details of the calculation are somewhat different in the two cases. For the pre-trip table, the summation is over origins, information classes, destinations, paths and time steps; only time steps during which trips enter the network need be considered. For the en route table, the summation is over decision points, destinations, from-paths, to-paths and time steps; in this case all time steps need to be considered. (Recall that en route guidance affects a decision to switch *from* one path *to* another.) As with the IC_C norm, no volume or other weighting was applied to the individual path splits, and the total norm value is reported. The measure is in effect a simple 2-norm of the difference between the two entire path split tables, considered as vectors.

It would be interesting to compare, for a given problem, the progress towards convergence of both the composite link condition and composite path split formulations. Such a comparison would require that the same measure of progress be applied in both situations. Accordingly, an attempt was made to generate IC_C norms from the composite path split formulation by saving the intermediate link time trajectories from the $D \circ G \circ S$ evaluation, and by generating an auxiliary link time trajectory from the computed path split. Put differently, instead of computing directly $P^* = D \circ G \circ S(P^i)$, the algorithm computes (in effect)

$$\begin{aligned} C^i &= S(P^i) \\ P^* &= D \circ G(C^i) \\ C^* &= S(P^*) \end{aligned}$$

where the final dynamic network loading (evaluation of the S map) is only needed in order to generate the auxiliary link time table. The link condition norm $\|C^i - C^*\|$ is then computed along with the path split norm $\|P^i - P^*\| = \|P^i - D \circ G \circ S(P^i)\|$.

This intention was thwarted by the inherent stochasticity in the problem: the link condition norm computation compares an unfiltered link time table (the output of the stochastic map S applied to the filtered P^i) with its unfiltered image through $S \circ D \circ G$ and gives, even in situations of only moderate stochasticity, highly stochastic outputs. (In contrast to this, the condition norm computation for the $S \circ D \circ G$ map compares a filtered link time table (the result of the MSA averaging) with the unfiltered image through $S \circ D \circ G$ of the filtered table, and so entails much less stochasticity.) Consequently, the comparison of the convergence rates of the two different formulations using the IC_C norm did not give meaningful results. Nonetheless, a very rough comparison of convergence rates of different formulations is possible in some cases by examining the number of iterations required for the respective norms to approach their asymptotic values when applied to the same problem data (assuming that the quality of the computed solution trajectories is comparable).

Table 5.5 lists the runs that were performed and are discussed below. It can be seen that, for a given run number, the problem parameters are identical to those in the $S \circ D \circ G$ series of runs listed in Table 5.3; thus, corresponding outputs of the two runs can be validly compared.

Run results are presented in a graphical format similar to that used for the $S \circ D \circ G$ series; however, the logical sequence of the graphs is different due to the different evaluation order of the maps within the composite. The sequence here is as follows:

(a) the IC_P and IC_C norms by iteration. As explained above, the IC_C norm is too corrupted by noise to be useful;

(b) pre-trip path split trajectories as input to the final (300th) iteration. This is the result of applying the MSA to prior iterations' results. Note that, in runs DGS-5 and DGS-6, the path split table also includes splits at node 101, where unequipped vehicles receive guidance from a VMS. These splits are not displayed in the graphics;

(c) cumulative trip departure trajectories by path, which are the result of applying the path splits (in an aggregate or disaggregate manner, according to the run) to the trip productions at the origin;

(d) link volume trajectories, which result from loading the OD flows on the network in accordance with the path splits;

(e) link time trajectories, determined simultaneously with link volumes during the dynamic network loading;

(f) path time trajectories, derived from the link time trajectories; and

(g) pre-trip path split trajectories that result from applying the routing model to the guidance messages generated from the link times (these messages are in fact the path time trajectories). Again, the path split table output by the composite path split map includes en route splits at the VMS node (node 101) for runs DGS-5 and DGS-6, but these splits are not shown here.

Run	*Network*	*Guidance*	*Path Splits*	*Flows*	*Times*	*Figure*
DGS-1	full capacity	100%	aggregate	x10	x10	5.26
DGS-2	full capacity	100%	disaggregate	x1	x10	5.27
DGS-3	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.28
DGS-4	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.29
DGS-5	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.30
DGS-6	link 201-200 at 900 v/h	25% + VMS	disaggregate	x1	x10	5.31

Table 5.5: Computational tests of the MSA applied to $D \circ G \circ S : P \mapsto P$

In short, these figures show the sequence of intermediate steps by which the composite path split map $D \circ G \circ S$ transforms an input path split table into an output path split table. Comparing figures (b) and (g) gives an indication of how close the (pre-trip) path splits calculated by the solution algorithm are to being a fixed point.

The following paragraphs discuss the individual runs. As before, all graphics are collected at the end of the chapter for easier reference.

Run DGS-1 For this "easy" problem with low stochasticity, the performance of the MSA applied to the $D \circ G \circ S$ formulation is excellent: the IC_P norm is driven to nearly 0 within a few dozen iterations (Figure 5.26(a)). (By way of comparison, Figure 5.17(a) shows that the $S \circ D \circ G$ map required roughly 150 iterations to reach its asymptote with this problem data set.) Otherwise, the detailed solution results (Figures 5.26(b) through 5.26(g)) appear identical to those obtained in run SDG-1, when they can be compared. Note that because all trips receive pre-trip guidance, the path volumes are an accurate reflection of the path splits that underlie this formulation.

Run DGS-2 This is the "easy" problem with high stochasticity relationships. The IC_P norm quickly reaches a range of values that it generally remains within (roughly 15–40 units) but exhibits considerable fluctuation from iteration to iteration within this range (Figure 5.27(a)).

The input path split and cumulative path flow trajectories (Figures 5.27(b) and 5.27(c)) show that most of the departures use paths 0, 4 and 7, and that there is some variability in the trajectories between replications. This translates into comparable variability in the link volumes (Figure 5.27(d)), link times (Figure 5.27(e)) and path times (Figure 5.27(f)). The output path splits (Figure 5.27(g) amplify the path time differences and appear much noisier than the other variables. Overall, however, the link volumes appear generally similar to those obtained in run SDG-2, and the link times appear similar to the output times obtained in that run.

Run DGS-3 DGS-3 uses the MSA to solve a low stochasticity path split formulation, applying this to a more challenging full information problem in which restricted capacity (25% of normal) on link 6 (102–200) causes spillback on a number of other links.

The graph of the IC_P norm (Figure 5.28(a)) is interesting because the initial steep norm decrease is followed by a phase between iterations 30 and 40 (roughly) in which the norm generally increases. It then enters a more gentle downward-tending phase to about iteration 100. The norm continues to decrease very gradually after iteration 100, and appears still to have a slight downward trend

when the algorithm terminates at iteration 300. Note that this gradually decreasing tail was also observed in the IC_C norm in run SDG-3.

No variability is evident in the computed path splits or corresponding cumulative path volumes in Figures 5.28(b) and 5.28(c). The volumes appear identical to those computed in run SDG-3. Link volume trajectories (Figure 5.28(d)) exhibit minor variability in a few cases; their overall shape and distinguishing features again appear very similar to those obtained in run SDG-3. The same comments apply to the corresponding link traversal times (Figure 5.28(e)), although in some cases minor differences in the trajectory's features are apparent. The link time variability carries over to the path time trajectories (Figure 5.28(f)). There is some variability in the output path split trajectories (Figure 5.28(g)) on a few paths.

Run DGS-4 This run is identical to run DGS-3 except that high stochasticity relationships are applied.

Figure 5.29(a) shows that after 50 iterations or so the IC_P norm reaches a range of values between roughly 25 and 65 units. It stays within this range for the remaining iterations, with the norm values fluctuating considerably from iteration to iteration.

The input path splits and resulting cumulative path flow trajectories show some variability between replications (Figures 5.29(b) and 5.29(c)). These, together with the intrinsic stochasticity of the model, translate into highly variable link volume (Figure 5.29(d)), link time (Figure 5.29(e)), path time (Figure 5.29(f)) and output path split (Figure 5.29(g)) trajectories. The link volume trajectories are roughly similar to those observed in run SDG-4.

Run DGS-5 This run involves both guided and unguided vehicles, with unguided vehicles able to access a short range information source on link 1 (100–101) for en route path reconsideration. Link 6 (102–200) has restricted capacity. Low stochasticity relationships are applied.

The IC_P norm reaches the range of 5–20 units after around 75 iterations (Figure 5.30(a)). By contrast, applying the MSA to the $S \circ D \circ G$ formulation required roughly twice as many iterations to reach the range of final values. The norm graph presents more variability than did the corresponding graph for run DGS-3, but much less than did that for run DGS-4.

In this case, path volumes (Figure 5.30(c)) do not completely reflect the pre-trip path splits (Figure 5.30(b)) because the volumes are an aggregate of both guided and unguided trips, but only the splits of equipped vehicles are shown. The volumes resulting in this way from the splits for the two guidance classes show no discernible variability (Figure 5.30(c)) and appear identical to those computed in run SDG-5.

The link volume and traversal time trajectories (Figures 5.30(d) and 5.30(e)) show limited variability; moderately distinct trajectories are visible on a few links. These trajectories are fairly similar to those of run DGS-5 in their overall shape and distinguishing features, although some differences of detail can be discerned; these are no doubt due to the stochasticity of the outputs. The variability in the path time trajectories (Figure 5.30(f)) reflects the underlying variability of the link times. These differences are again amplified in the output path split trajectories (Figure 5.30(g)).

DGS-6 Run DGS-6 is identical to run DGS-5 except that in this case high stochasticity relationships are applied.

Run	*Network*	*Guidance*	*Path Splits*	*Flows*	*Times*	*Figure*
DGS-3-pol	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.32
DGS-4-pol	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.33
DGS-5-pol	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.34

Table 5.6: Computational tests of Polyak averaging applied to $D \circ G \circ S : P \mapsto P$

Figure 5.31(a) again shows a fairly rapid convergence (within roughly 50 iterations) to a range of values broadly between 25 and 60 units, and exhibits strong variability within this range. The IC_C norm in problem SDG-6 also showed strong variability, but took roughly 100 iterations to attain its final range of values.

Again, the cumulative path volumes in Figure 5.31(c) do not directly reflect the pre-trip splits of equipped vehicles in Figure 5.31(b) because of the presence of unequipped vehicles as well. However, the aggregate path flows (Figure 5.31(c)) are similar to those obtained in run SDG-6, although some minor differences are apparent. The variability in trajectories between replications is clearly noticeable and is comparable between the two runs.

The resulting link volume (Figure 5.31(d)) and traversal time (Figure 5.31(e)) trajectories are highly variable; three distinct trajectories are apparent on many links. These are roughly comparable the trajectories that were obtained in run SDG-6, although the variability makes detailed comparison of the trajectories impossible. The variability in path times (Figure 5.31(f)) reflects the stochasticity of the link times, and leads to wildly variable output path splits (Figure 5.31(g)).

5.6.2.2 Polyak algorithm

The Polyak iterate averaging method was also applied to a number of the problems examined in the preceding section. The general approach is the same as that described in Section 5.6.1.2 above, with the obvious difference that path split tables rather than link time tables were saved in the first pass and averaged in the second. Again, the second pass included an evaluation of $D \circ G \circ S(\tilde{P}^i)$ for each average $\tilde{P}^i$ determined after incorporation of a new path split table P^i; the only reason for this evaluation was to compute the IC_P inconsistency norm corresponding to the averaged table, so that the convergence properties of the Polyak algorithm could be tracked. As before, discussion focuses on comparison of convergence properties of the IC_P norm, rather than on the specific path split or other trajectories that were obtained at different iterations.

Table 5.6 identifies the runs that were carried out for these computational tests.

Run DGS-3-pol As has been noted before, this is a low stochasticity full information equilibrium run for a network with restricted capacity on link 6. Figures 5.32(a) and 5.32(b) show the results of the application of Polyak averaging to this problem; these can be compared with Figure 5.28(a) for application of the MSA.

It was seen in run DGS-3 that, with the MSA, the IC_P norm approached its final range of values (around 5 units) after roughly 150 iterations, but appeared to still be gradually decreasing even after 300 iterations. In contrast to this, in the first pass of Polyak averaging, the IC_P norm reaches a range of 2.5–7.5 units after roughly 75 iterations, and does not exhibit any noticeable downward trend after that point. The second pass reduces the variability of the norm values to a

range of 2.5–5 units by 75 iterations, and the norm values remain mostly within this range, with no particular trend, through iteration 300.

This is consistent with the results obtained from the tests of Polyak averaging applied to the $S \circ D \circ G$ map, namely, that for this problem Polyak averaging seems to converge after roughly 75 iterations while the MSA has still not quite converged after 300 iterations.

Run DGS-4-pol Again, this is a high stochasticity full information equilibrium run for a network with restricted capacity on link 6. Figures 5.33(a) and 5.33(b) show the results of the application of Polyak averaging to this problem, and can be compared with Figure 5.29(a) for application of the MSA in run DGS-4.

As before, the amount of noise present in the IC_P norm evaluations makes detection of trends or convergence a difficult task. It seems valid to conclude that differences between the results of the MSA and Polyak averaging are not particularly striking. Both methods appear to reach a generally horizontal trend after roughly the same number of iterations and at this point the mean norm values are fairly similar (about 40 units). Contrary to what was seen in the analysis of the $S \circ D \circ G$ map, fluctuations in the Polyak averaging results may be less extreme than in the MSA results, but it is difficult to be sure about this.

Run DGS-5-pol This is the true route guidance problem with high stochasticity relationships; it was also treated in run DGS-5. Figures 5.34(a) and 5.34(b) present the IC_P convergence results for the Polyak algorithm, and can be compared with Figure 5.30(a) for the MSA application to this problem.

In the MSA application, the final range of IC_P norm values (roughly 7.5–12.5 units) was attained after approximately 100 iterations. The second pass Polyak averaging results show that this range is attained after roughly 75 iterations. Polyak averaging is again more effective for this problem, although its advantage is less than was found in prior applications.

The overall conclusions reached from testing the Polyak averaging method on the $S \circ D \circ G$ fixed point problem are broadly confirmed in its applications here to the $D \circ G \circ S$ problem. Measured in terms of number of iterations to convergence, the Polyak averaging method outperforms the MSA by factors of from over one to more than four when applied to noisy maps with low stochasticity. On the other hand, no clear conclusions regarding the relative effectiveness of the two methods emerge when they are applied to more highly stochastic problems.

5.6.3 Composite message formulation - continuous case

A set of runs was also made to investigate the properties of the composite message formulation $G \circ S \circ D : M \mapsto M$ and solution algorithms for it. In general, messages summarize the information contained in network condition predictions. With link traversal times used to exemplify the network condition variable in the simple simulator, a large number of possible guidance message types could be generated. It is useful to distinguish here between *continuous* and *discrete* messages, according to their variation with changing link traversal times. Algorithms for solving fixed point problems involving the two types of message are quite different: discrete variable problems are essentially

combinatorial in nature,[9] whereas continuous variable problems can in principle be attacked using the methods discussed in Chapter 4. For this reason, only continuous variable problems will be considered here. The path time variable discussed in Section 5.2.2 above was used as an example message variable for these tests.

In full information runs, only pre-trip messages (path times from an origin to a destination) needed to be considered, whereas in true route guidance generation runs both pre-trip and en route (path times from a beacon to a destination) messages were taken into account.

5.6.3.1 MSA algorithm

To initialize the MSA, free-flow traffic conditions were assumed. Messages were generated from these link times, then the full $G \circ S \circ D$ map was applied to the free-flow messages to obtain the messages used in the initial MSA iteration. It would have been possible to simply use the messages generated from the free-flow link times; however, these messages would not generally have represented feasible path times (i.e., times that correspond to possible flow conditions), and so would not have been comparable to the path times computed as intermediate variables in the other formulations. With more general continuous message variables, it might not have been necessary to undertake as elaborate an initialization. All MSA runs carried out 300 iterations.

A message inconsistency norm, IC_M, was used to measure the progress of the solution algorithms. The norm computes the root sum squared difference between input and output message tables, including both pre-trip and en route messages. Note that this entails some double counting, since some of the paths from the origin also pass through the beacon that is used for en route guidance: the overlapping portions of these paths are counted twice. The contribution of pre-trip messages to the total norm is only computed over the interval of demand inflows to the network, whereas the contribution of en route messages is determined over the entire analysis horizon. As before, no volume or other weighting is applied to the individual path times.

The comments made in Section 5.6.2 regarding attempts to compare the convergence properties of the different formulations with respect to a common norm apply here as well. Here also, link condition IC_C norms were generated during the composite message map evaluation process by saving intermediate results and carrying out some additional processing steps that would not otherwise have been necessary. The sequence of calculations to evaluate $M^* = G \circ D \circ S(M^i)$ is in effect as follows:

$$\begin{aligned} C^i &= S \circ D(M^i) \\ M^* &= G(C^i) \\ C^* &= S \circ D(M^*) \end{aligned}$$

where the final path split and dynamic network loading (evaluation of the $G \circ S$ map) is only needed to generate the auxiliary link time table. The link condition norm $\|C^i - C^*\|$ is then computed along with the message norm $\|M^i - M^*\| = \|M^i - G \circ S \circ D(M^i)\|$. Again, problem stochasticity usually rendered difficult or meaningless any comparison of the IC_C norm computed in this way

[9] Hoogendoorn and Bovy [1998], for example, describe an application of genetic optimization techniques to search through the (very large) space of route recommendation time trajectories for the network around the city of Arnhem in the Netherlands.

with the corresponding norms obtained directly from solutions of the composite network condition map or indirectly from solutions of the composite path split map.

Table 5.7 lists the runs that were carried out for the computational tests of the MSA applied to the composite message formulation. For a given run number, the problem parameters are identical to those in the $S \circ D \circ G$ series of runs listed in Tables 5.3 and 5.5; thus, outputs of comparable runs in the different series can be compared.

Run results are presented in a graphical format similar to that used for the series discussed earlier; however, the logical sequence of the graphs is different due to the different order of the component maps within the composite map. The sequence here is as follows:

(a) the IC_M and IC_C norms by iteration. As explained above, the IC_C norm is generally too corrupted by noise to be usable for comparisons with convergence properties of the other formulations;

(b) pre-trip message (path time) trajectories as input to the final (300th) iteration of the solution process. These trajectories are the result of applying the MSA to prior iterations' results. Note that, in runs GSD-5 and GSD-6, the message table also includes messages from the beacon at node 101, where unequipped vehicles receive guidance from a VMS. These messages are not shown here because of graphical presentation difficulties. Note, however, that the four paths from the beacon to the destination are obtained by removing the first link (100-101) from paths 0 through 3 shown on this graphic;

(c) cumulative path flows (trip departure trajectories) by path, which are the result of applying to trip productions at the origin the path splits (in an aggregate or disaggregate manner, according to the run) resulting from the messages;

(d) link volume trajectories, which result from loading the OD flows on the network in accordance with the path splits;

(e) link time trajectories, determined simultaneously with link volumes during the dynamic network loading;

(f) output pre-trip message (path time) trajectories, derived from the link time trajectories.

These figures show relevant intermediate steps in the sequence by which the composite message map $G \circ S \circ D$ transforms an input message table into an output message table. Comparing figures (b) and (f) gives an indication of how close the (pre-trip) path splits calculated by the solution algorithm are to being a fixed point, and also how much noise is generated by the composite map evaluation.

The following paragraphs discuss the individual runs. Run output graphics are collected at the end of the chapter.

Run GSD-1 This is the "easy" problem involving full information, no disruption of traffic conditions and low stochasticity component maps (aggregate application of the routing map and a flow scale factor of 10). The IC_M is driven to an asymptote very close to 0 seconds after roughly 150 iterations (Figure 5.35(a)); after a small amount of noise in the first two dozen or so iterations, the IC_M trajectories of the different replications are indistinguishable. This almost total absence of

Run	*Network*	*Guidance*	*Path Splits*	*Flows*	*Times*	*Figure*
GSD-1	full capacity	100%	aggregate	x10	x10	5.35
GSD-2	full capacity	100%	disaggregate	x1	x10	5.36
GSD-3	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.37
GSD-4	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.38
GSD-5	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.39
GSD-6	link 201-200 at 900 v/h	25% + VMS	disaggregate	x1	x10	5.40

Table 5.7: Computational tests of the MSA applied to $G \circ S \circ D : M \mapsto M$

noise within a replication and variability between replications characterizes all the problem variable trajectories produced by this run. The input and output message trajectories (Figures 5.35(b) and 5.35(f), respectively) can be compared to the path time trajectories reported for runs SDG-1 and DGS-1: they appear to be identical, as do the cumulative path flows (Figure 5.35(c)) and link volumes and times (Figures 5.35(d) and 5.35(e), respectively.) The three different formulations are all computing the same solution.

Run GSD-2 This run uses problem data identical to that in run GSD-1, but applies high stochasticity versions of the component maps. As seen in Figure 5.36(a), stationarity appears to be achieved after around 75 iterations, but considerable noise in the IC_M norm evaluations is evident. The stationary range of IC_C values obtained in run SDG-2 was on the order of 500–750 seconds, whereas here it is perhaps twice that; this is not surprising, considering that the IC_M norm measures path rather than link times. The message trajectories after 300 iterations are smooth, as expected (Figure 5.36(b)). However, applying the $G \circ S \circ D$ map to these values results in output trajectories that are rather variable over their entire range (Figure 5.36(f)), with path time differences between replications of 25% or more in some cases. This can be traced to the variability in the path flows computed by the disaggregate routing map from the input messages (Figure 5.36(c)). These translate into link volumes (Figure 5.36(d)) and times (Figure 5.36(e)) that exhibit variability between replications as well as the high frequency noise that has been seen to result from the unscaled network loader.

Run GSD-3 This run applies low stochasticity maps to a full information problem with reduced capacity (900 vehicles/hour *vs.* 3600 vehicles/hour normally) on link 6 (102-200). The IC_M norm descends gradually (Figure 5.37(a)) and it is not clear after 300 iterations if an asymptote has been reached. The norm exhibits noise having a range of around 500 seconds, which is considerably more than that of the IC_C norm seen in Figure 5.19(a). Nonetheless, the problem variable trajectories are fairly smooth and do not show evidence of significant variability between replications. The computed messages (Figure 5.37(b)) appear identical to the path times computed in run SDG-3. Path flows (Figure 5.37(c)) are very smooth, and again seem identical to the path flows computed in run SDG-3. There is a slight amount of noise and variability in the link volume (Figure 5.37(d)) and time (Figure 5.37(e)) trajectories on some links located on paths that avoid the capacity restriction. Accordingly, the output message trajectories show some variability, particularly towards the end of the analysis period when spillback effects are important.

Run GSD-4 This run uses the GSD-3 problem data but applies a high stochasticity composite map. It is difficult to discern the algorithm's convergence behavior from the IC_M map (Figure 5.38(a)). However, smooth message trajectories are computed by the 300th iteration (Figure 5.38(b)), and these appear identical to the trajectories determined in applications of the other composite maps to this problem data (runs SDG-4 and DGS-4). Path flows show variability on the paths that can be taken to avoid the capacity restriction (Figure 5.38(c)), whereas flows on paths through the restriction are smoother (and generally much lower). There is considerable noise and variability in the link volume (Figure 5.38(d)) and time (Figure 5.38(e)) trajectories, leading to highly variable output message trajectories (Figure 5.38(f)).

Run GSD-5 This run is a true guidance generation run: 75% of vehicles are unguided, but are able to access a short range information source on link 1 (100–101) for en route path reconsideration. Link 6 (102–200) has restricted capacity. A low stochasticity composite map is applied. As can be seen from Figure 5.39(a), the IC_M norm reaches a stationary range before 150 iterations; however, considerable noise is evident. The computed path time messages (Figure 5.39(b)) are identical to the path times determined in run SDG-5, as are the origin-based path flows (Figure 5.39(c)). Link volume (Figure 5.39(d)) and time (Figure 5.39(e)) trajectories show some variability, but the resulting variability in the output messages (Figure 5.39(f)) is limited to a fairly narrow range of path times.

Run GSD-6 Here the problem data is the same as that used in run GSD-5, but the composite map has high stochasticity, with disaggregate routing map evaluation and a flow scale of 1. No useful information can be obtained from the IC_M plot (Figure 5.40(a)) because of the noise. Nonetheless, the computed pre-trip messages are smooth and appear identical to the path times found in run SDG-6. Routing map stochasticity is evident in most of the cumulative path flows (Figure 5.40(c)). Link volume (Figure 5.40(d)) and time (Figure 5.40(e)) trajectories show considerable noise and variability, as do the resulting output message trajectories (Figure 5.40(f)): path time differences between replications exceed 50% in some cases.

5.6.3.2 Polyak algorithm

The Polyak iterate averaging method was used to solve the composite $G \circ S \circ D : M \mapsto M$ formulation of a number of the test problems investigated in the preceding section. Naturally, both the pre-trip and the en route message (path time) tables were involved in the averaging process. As was the case in applications of Polyak averaging discussed in Sections 5.6.1.2 and 5.6.2.2, the second pass evaluated $G \circ S \circ D(\tilde{M}^i)$ for each average $\tilde{M}^i$ computed after incorporation into the running averaging of a message table M^i from the first pass. This norm evaluation was done solely to compute the IC_M inconsistency norm for the averaged tables, but it must be remembered that the evaluation itself returns noisy results.

Table 5.8 identifies the runs that were carried out for these computational tests.

Run GSD-3-pol Again, this is a low stochasticity full information dynamic traffic equilibrium problem for a network with restricted capacity on link 6. Figures 5.41(a) and 5.41(b) show the results of the Polyak algorithm for this problem; these can be compared with Figure 5.37(a) for the MSA.

Run	*Network*	*Guidance*	*Path Splits*	*Flows*	*Times*	*Figure*
GSD-3-pol	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.41
GSD-4-pol	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.42
GSD-5-pol	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.43

Table 5.8: Computational tests of Polyak averaging applied to $G \circ S \circ D : M \mapsto M$

First pass IC_M norm values appear to reach stationarity in a range between roughly 500–1000 seconds by around 75 iterations; this can be contrasted with the MSA's results, in which the norm was around 2000 seconds after 300 iterations.

The second pass averaging window began at 50 iterations; this start might be slightly early, considering the first pass results. Nonetheless, second pass averaging results are clearly less noisy than the first pass results, with more norm values clustered in the range 500–1000. Stationarity again appears to be reached after around 75 iterations. Again, the Polyak method clearly outperforms the MSA: by 75 iterations it reaches a norm value that is a third of the value reached by the MSA after 300 iterations.

Run GSD-4-pol Figures 5.42(a) and 5.42(a) present the results of Polyak averaging applied to a stochastic version of the full information DTA problem discussed in the preceding paragraph. The results confirm what was found in comparable applications using the other formulations: Polyak averaging does not offer any advantages in this type of situation.

Run GSD-5-pol Again, this is a true route guidance problem involving 75% unequipped vehicles, some of which receive guidance via a VMS on link 1. It is modeled using a low stochasticity composite map. Figures 5.43(a) and 5.43(a) present the IC_M convergence norm results.

Polyak averaging compares very favorably to the MSA. The first pass norm appears to have reached stationarity by around 50 iterations, after which it fluctuates in a range of about 750–2250 seconds, with a mean of around 1250 seconds. In contrast, the MSA reached stationarity after around 150 iterations, with a mean of around 1750 seconds.

The second pass averaging window was started at 50 iterations. The second pass' range is similar to the first pass' and is comparable to the MSA's range but is reached after around 50 iterations *vs.* 150 iterations for the MSA.

5.7 Conclusions

This chapter described the design and implementation of a traffic simulation software system that operationalizes the framework presented in Chapter 3 and the fixed point guidance generation approaches discussed in Chapter 4. It also describes a number of computational experiments carried out with the simulator to investigate the performance of different problem formulations and of different fixed point computation approaches in route guidance applications.

Tests of the dynamic network loading map and the combined routing and loading maps showed that the software behaved reasonably. Furthermore, they showed that stochasticity in map evaluation outputs could be reduced to low levels, if desired, by scaling flows (dividing one vehicle into a number of vehicle "fractions") and by aggregate application of path splits to a homogeneous

population of vehicles. Conversely, moving and applying splits independently to individual vehicles was seen to result in outputs with a high degree of stochasticity in the test runs carried out.

Gibbs sampling was applied to a full information model in order to generate a sample from the equilibrium stochastic process solution. Examination of sample plots from the outputs did not reveal any evidence for problematic features such as multi-modal link time or volume distributions in a particular time step. Gibbs sampling of network stochastic processes is a computationally intensive procedure, and would benefit from availability of procedures for reoptimizing the dynamic network loading map.

Based on the Gibbs sampling outputs, it seemed reasonable to approximate the stochastic process solution to guidance models as an essentially deterministic process affected by noise, thus allowing application of stochastic approximation procedures.

The standard MSA algorithm was applied to a number of problems using the composite link condition, composite path split and composite message mappings, and using both low stochasticity and high stochasticity map evaluations. The MSA appeared to successfully handle all problems that it was presented. In the case of low stochasticity maps, it was able to drive the inconsistency norm (a measure of the distance of a trial solution from its image under the composite map) to low levels despite some noise in the composite map evaluations. In some cases, however, the MSA had not yet reached a norm asymptote after 300 iterations. In the case of high stochasticity maps, it was able to drive the inconsistency norm to a stationary range of values, although there was considerable variability in norm values within this range. For these problems, the MSA was computing a mean link time or path split trajectory, but evaluation of the composite maps would produce highly variable outputs from these mean values.

The Polyak iterate averaging method was applied to most of the problems that were solved using the MSA. It was found that, for low stochasticity maps, Polyak averaging could outperform the MSA by factors ranging from two to over four, as measured in terms of the number of iterations to reach stationarity of the convergence measure. (The additional computational cost of Polyak averaging, compared to the MSA, is very minor.) However, for highly stochastic maps the performance of Polyak averaging was not noticeably different from that of the MSA.

It was not possible to directly compare the effectiveness of the different composite link maps using a common norm because of the noise inherent in the auxiliary norm evaluations, even in low stochasticity models. However, it appeared that path-based formulations (the path split formulation or the path time message formulation) converged to their minimum norm range more quickly than the link-based formulation in most cases. The advantages appeared to be less in situations of highly perturbed traffic flow.

5.8 Graphic outputs

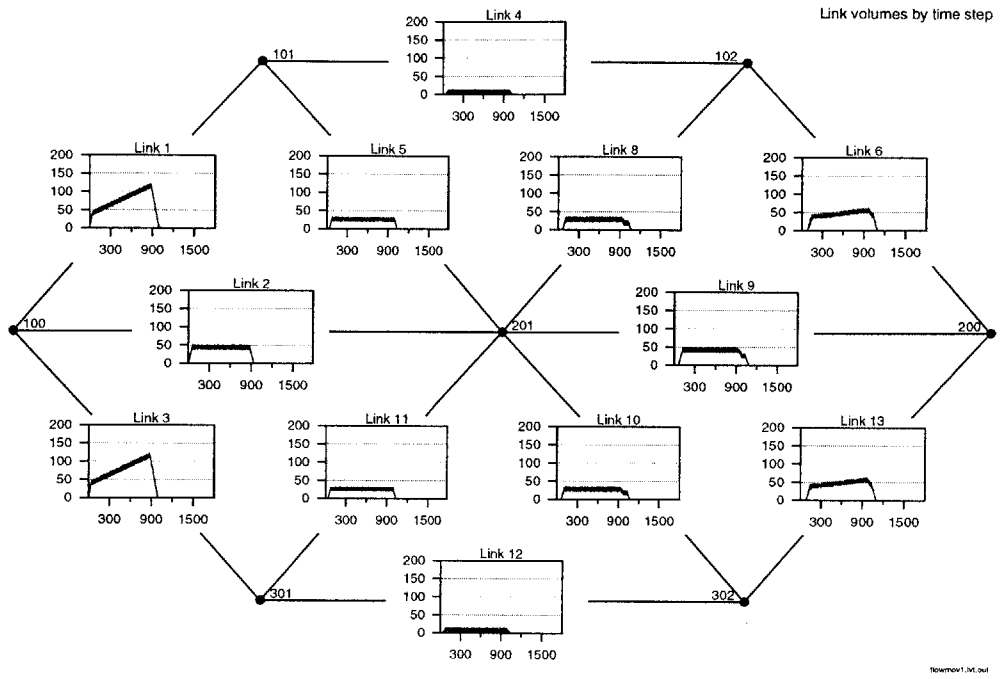
The remainder of this chapter consists of graphic outputs from the various test runs that were presented and discussed above. As an aid in identifying or locating particular outputs, the following tables indicate the main parameters of each run.

Run	*Path flow*	*Path splits*	*Link capacities*	*Capacity rounding*	*Figure*
NLM-1	3600 v/h	fixed	3600 v/h	random	5.8
NLM-2	1800 v/h	fixed	1800 v/h	random	5.9
NLM-3	1800 v/h	fixed	1800 v/h	bucket	5.10
SD-1	3600 v/h avg.	disaggregate	3600 v/h	bucket	5.11
SD-2	1800 v/h avg.	disaggregate	1800 v/h	bucket	5.12
SD-3	1800 v/h avg.	disaggregate	1800 v/h	random	5.13
SD-4	3600 v/h avg.	aggregate	3600 v/h	bucket	no figure
SD-5	1800 v/h avg.	aggregate	1800 v/h	bucket	5.14
SD-6	x 13 avg.	aggregate	x 13	bucket	no figure

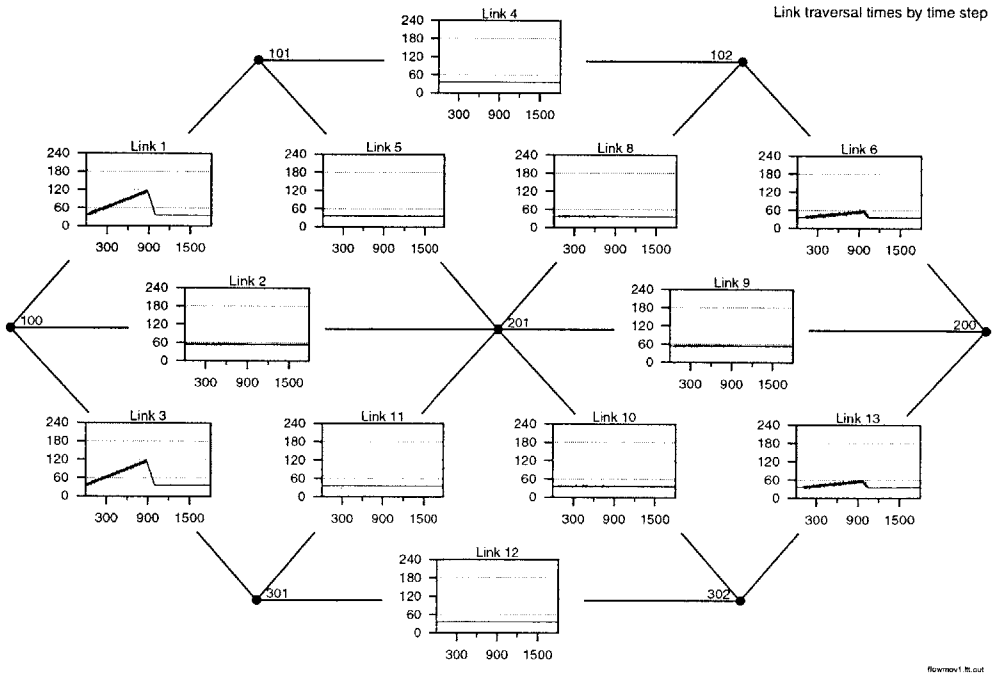
Table 5.9: Computational tests of individual and combined maps

Run	*Network*	*Guidance*	*Path Splits*	*Flows*	*Times*	*Figure*
SDG-1	full capacity	100%	aggregate	x10	x10	5.17
SDG-2	full capacity	100%	disaggregate	x1	x10	5.18
SDG-3	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.19
SDG-4	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.20
SDG-5	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.21
SDG-6	link 201-200 at 900 v/h	25% + VMS	disaggregate	x1	x10	5.22
SDG-3-pol	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.23
SDG-4-pol	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.24
SDG-5-pol	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.25
DGS-1	full capacity	100%	aggregate	x10	x10	5.26
DGS-2	full capacity	100%	disaggregate	x1	x10	5.27
DGS-3	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.28
DGS-4	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.29
DGS-5	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.30
DGS-6	link 201-200 at 900 v/h	25% + VMS	disaggregate	x1	x10	5.31
DGS-3-pol	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.32
DGS-4-pol	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.33
DGS-5-pol	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.34
GSD-1	full capacity	100%	aggregate	x10	x10	5.35
GSD-2	full capacity	100%	disaggregate	x1	x10	5.36
GSD-3	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.37
GSD-4	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.38
GSD-5	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.39
GSD-6	link 201-200 at 900 v/h	25% + VMS	disaggregate	x1	x10	5.40
GSD-3-pol	link 201-200 at 900 v/h	100%	aggregate	x10	x10	5.41
GSD-4-pol	link 201-200 at 900 v/h	100%	disaggregate	x1	x10	5.42
GSD-5-pol	link 201-200 at 900 v/h	25% + VMS	aggregate	x10	x10	5.43

Table 5.10: Computational tests of formulations and solution methods

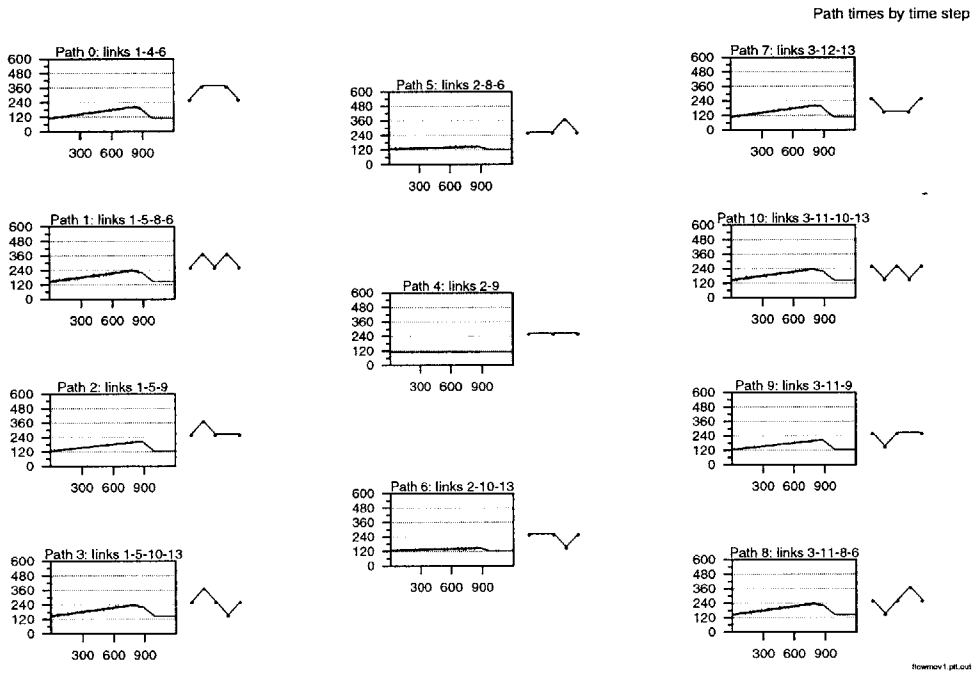


(a) Link volume trajectories (vehicles) by time step (seconds)



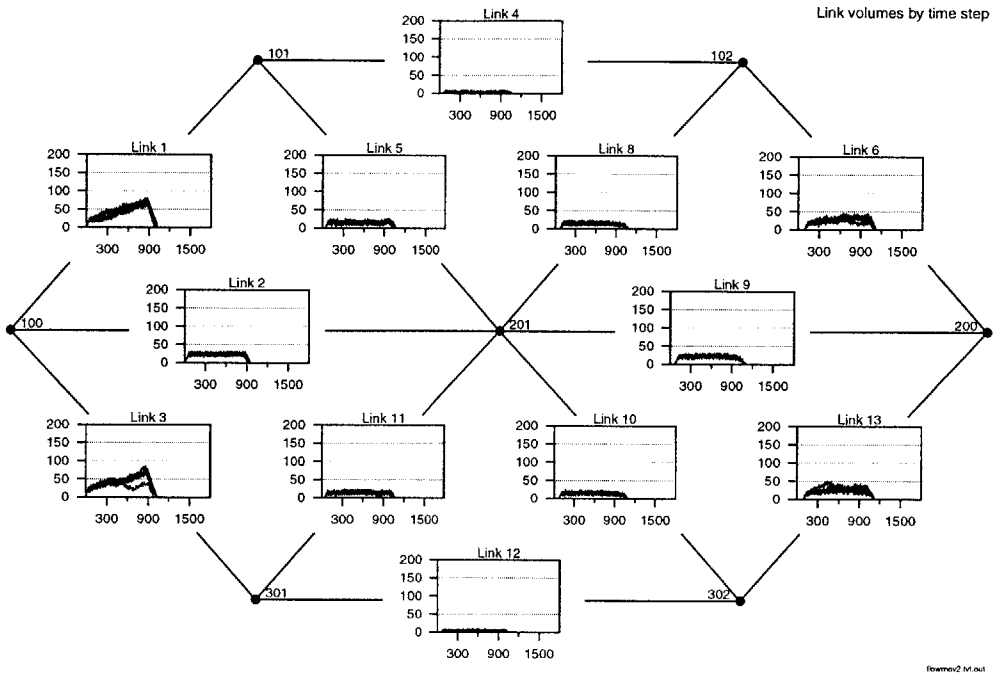
(b) Link traversal time trajectories (seconds) by time step (seconds)

Figure 5.8: Run NLM-1 (part 1/2)

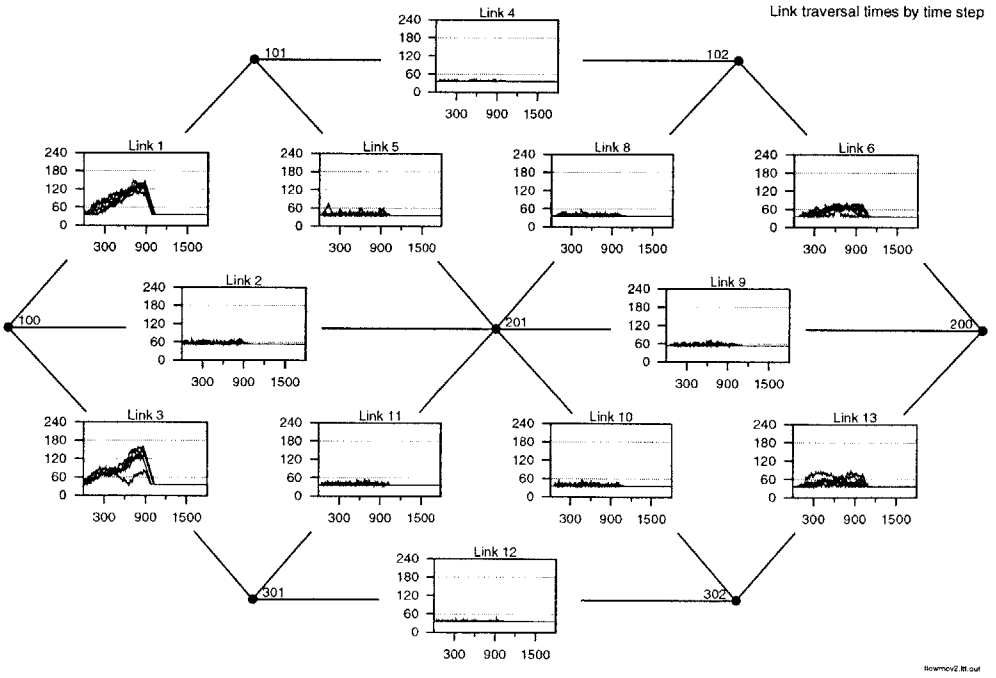


(c) Path time trajectories (seconds) by time step (seconds)

Figure 5.8: Run NLM-1 (part 2/2)

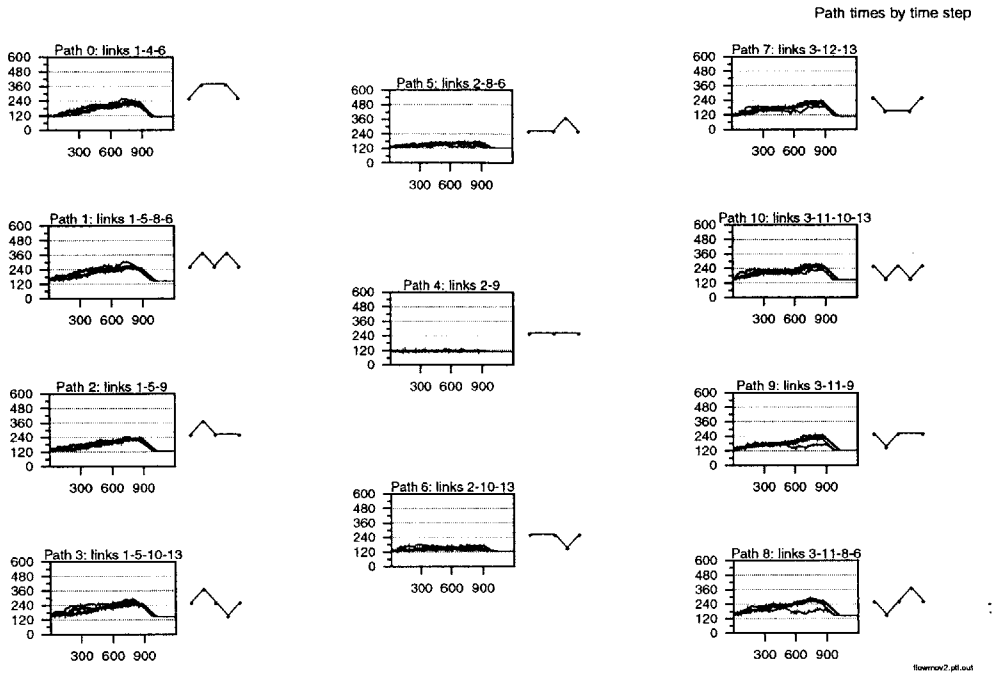


(a) Link volume trajectories (vehicles) by time step (seconds)



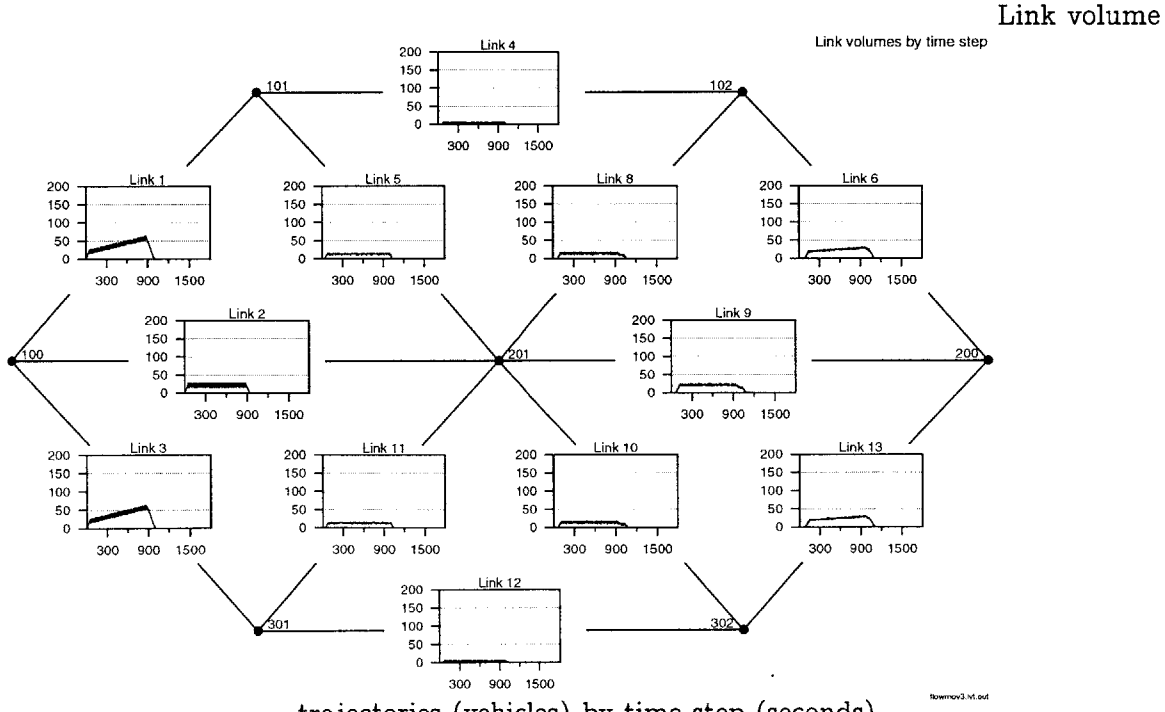
(b) Link traversal time trajectories (seconds) by time step (seconds)

Figure 5.9: Run NLM-2 (part 1/2)



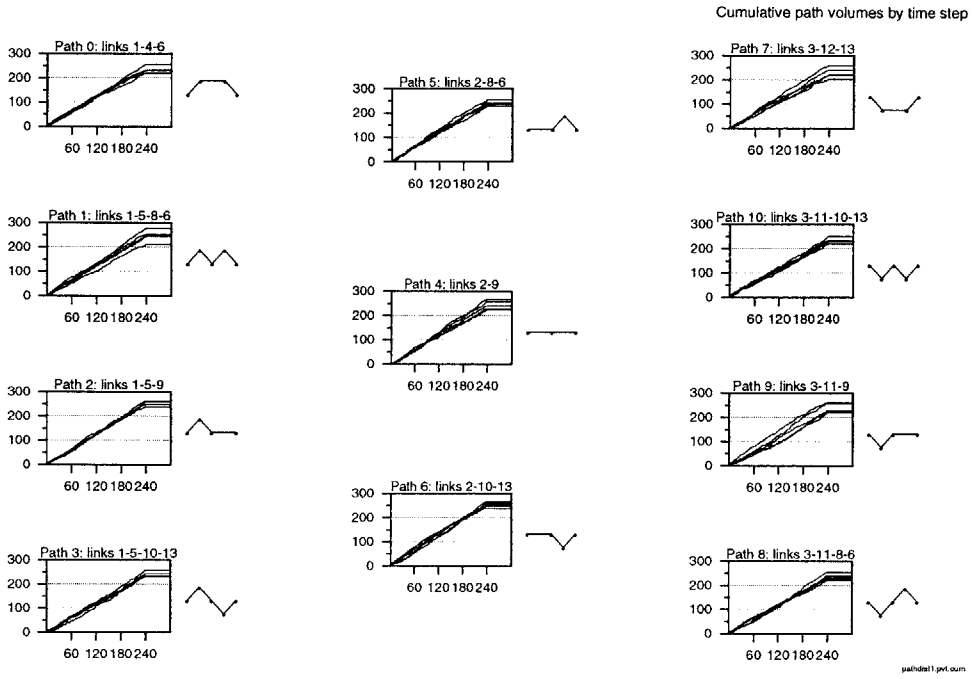
(c) Path time trajectories (seconds) by time step (seconds)

Figure 5.9: Run NLM-2 (part 2/2)

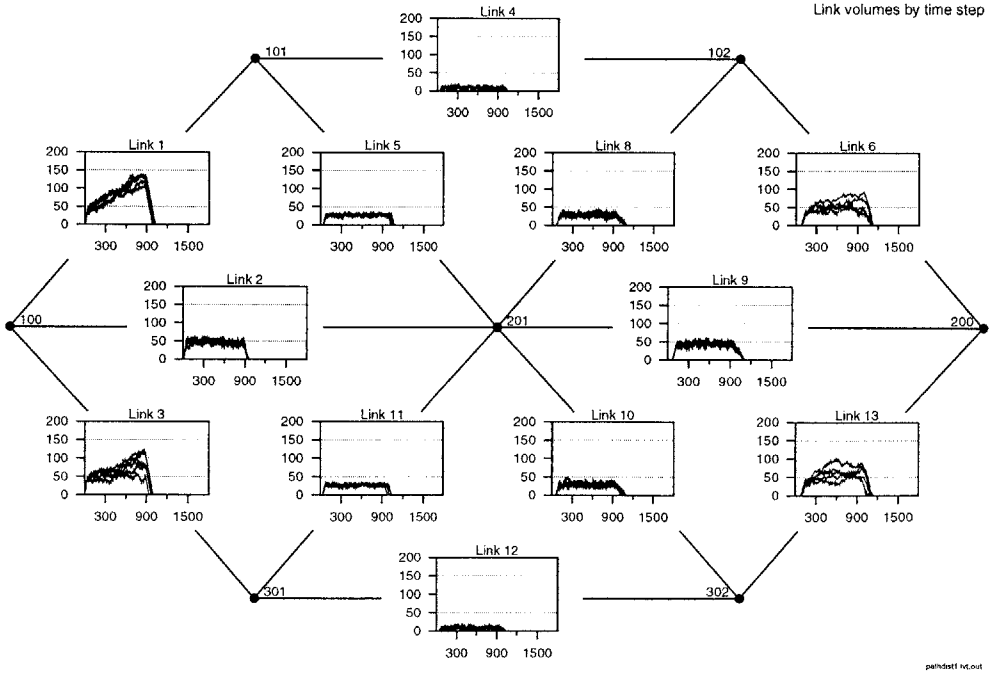


trajectories (vehicles) by time step (seconds)

Figure 5.10: Run NLM-3

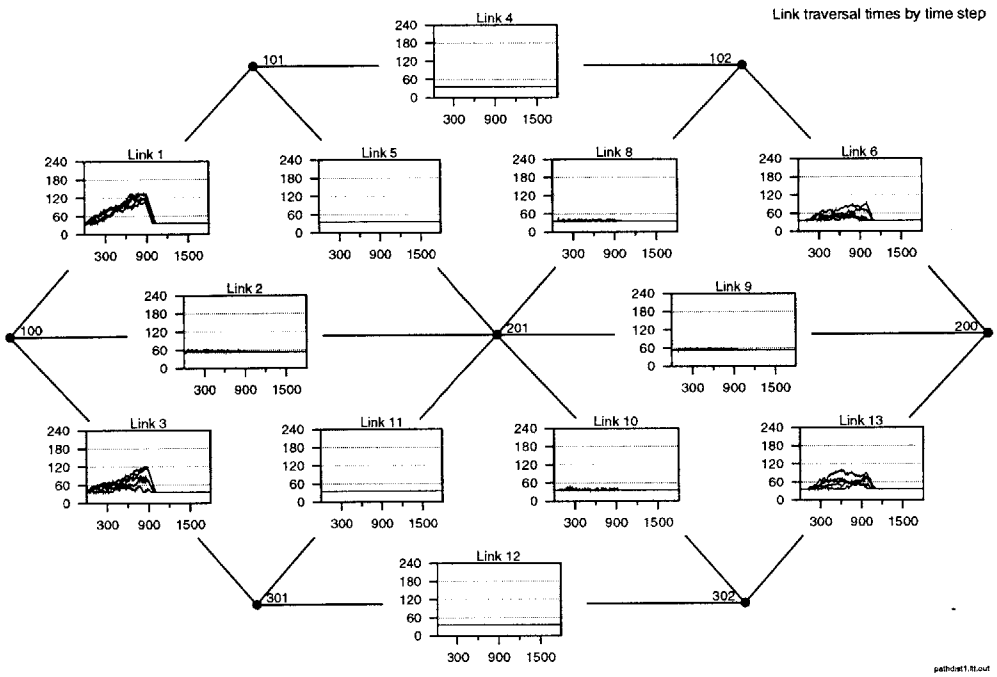


(a) Cumulative path flow trajectories (vehicles) by time step (seconds)

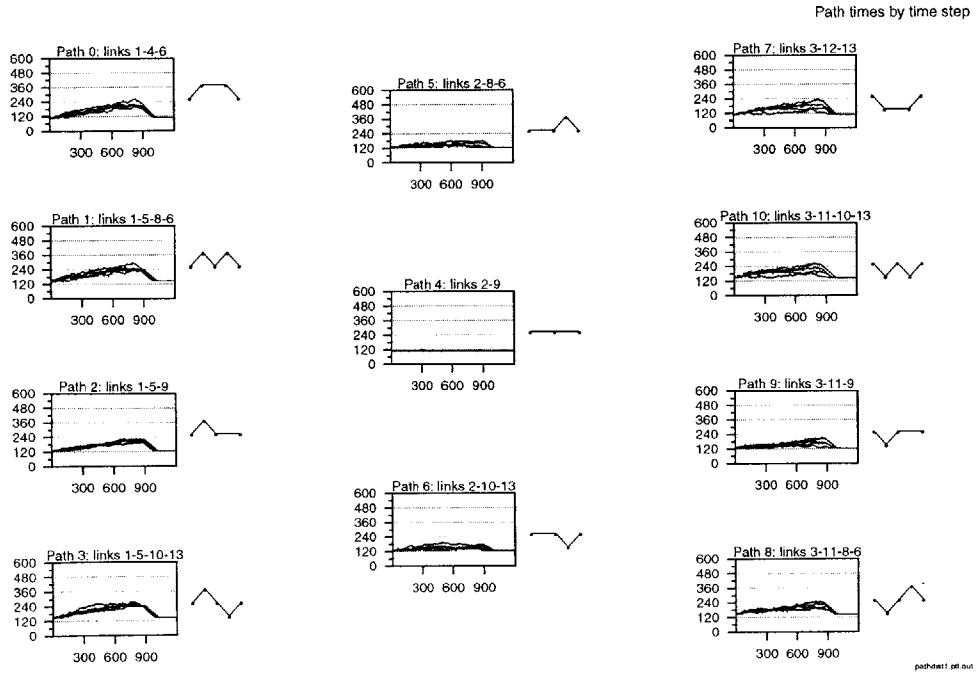


(b) Link volume trajectories (vehicles) by time step (seconds)

Figure 5.11: Run SD-1 (part 1/2)

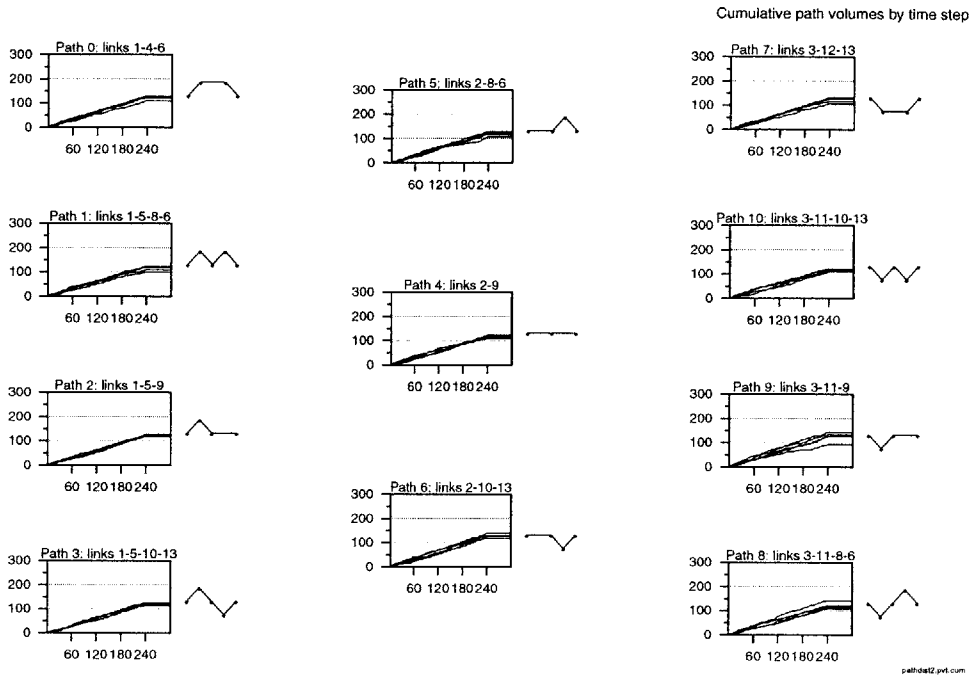


(c) Link traversal time trajectories (seconds) by time step (seconds)

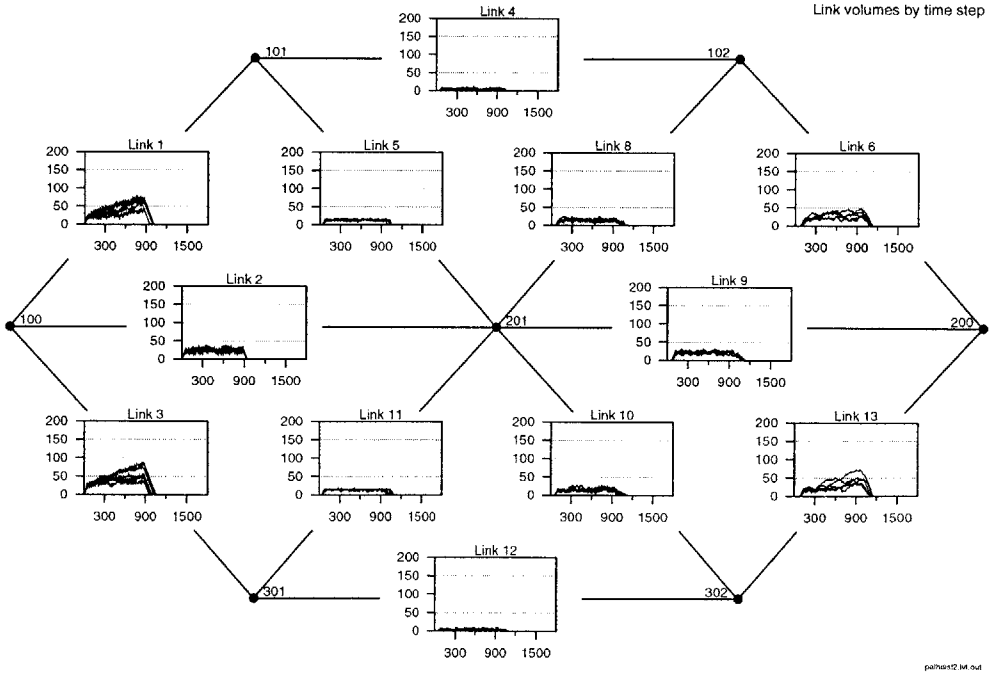


(d) Path time trajectories (seconds) by time step (seconds)

Figure 5.11: Run SD-1 (part 2/2)

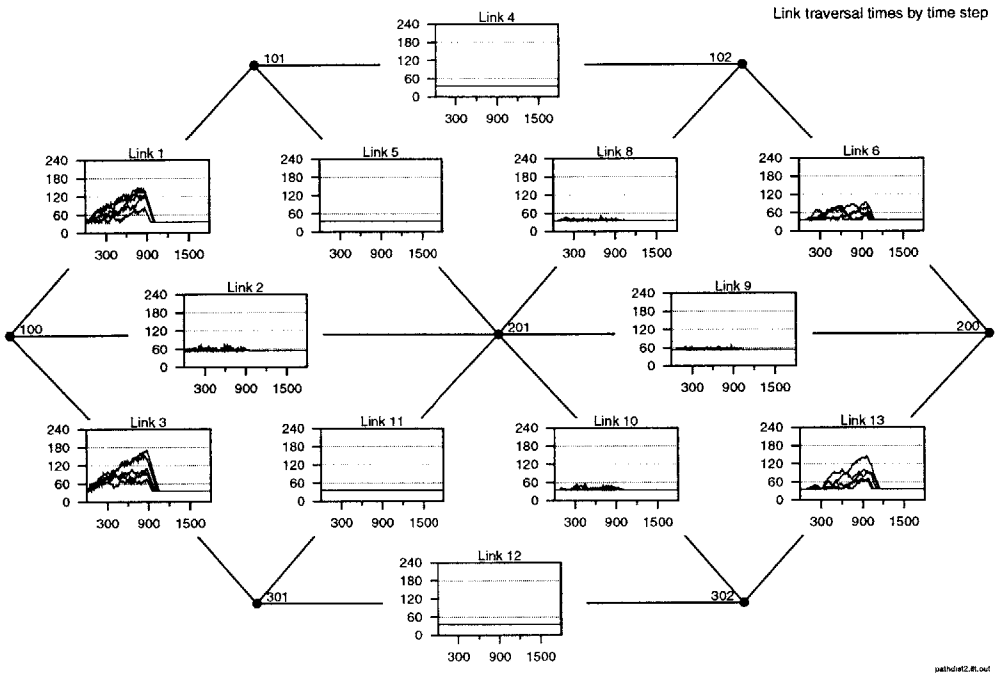


(a) Cumulative path flow trajectories (vehicles) by time step (seconds)

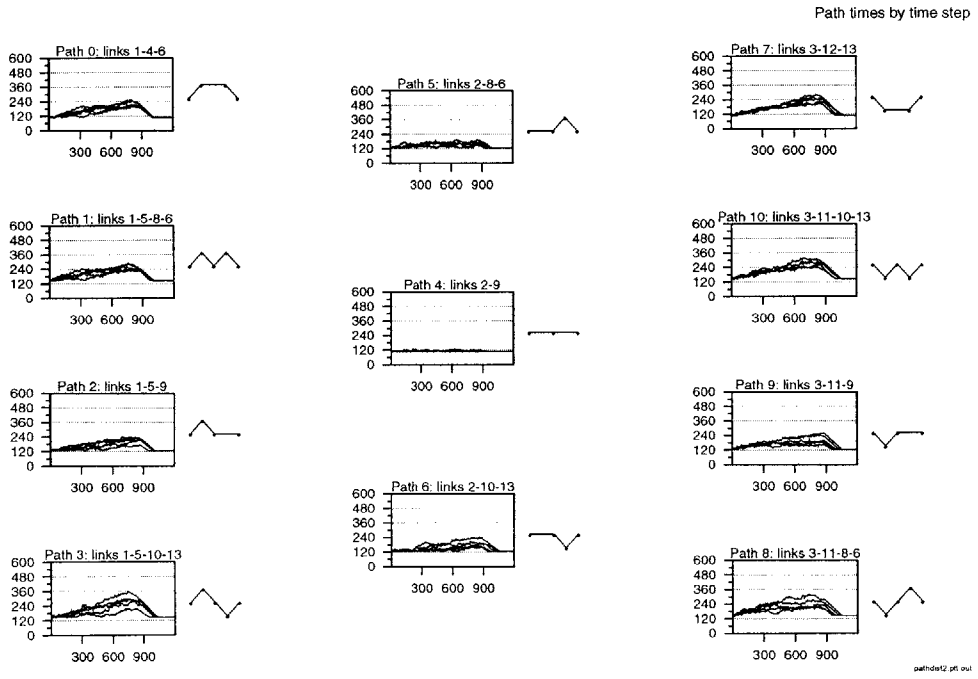


(b) Link volume trajectories (vehicles) by time step (seconds)

Figure 5.12: Run SD-2 (part 1/2)

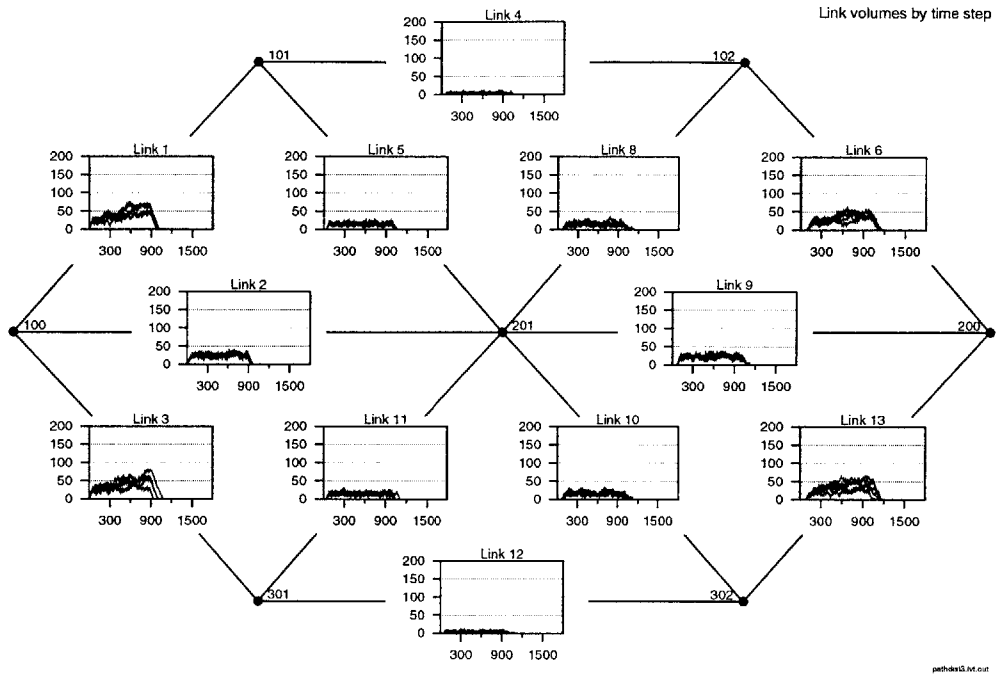


(c) Link traversal time trajectories (seconds) by time step (seconds)

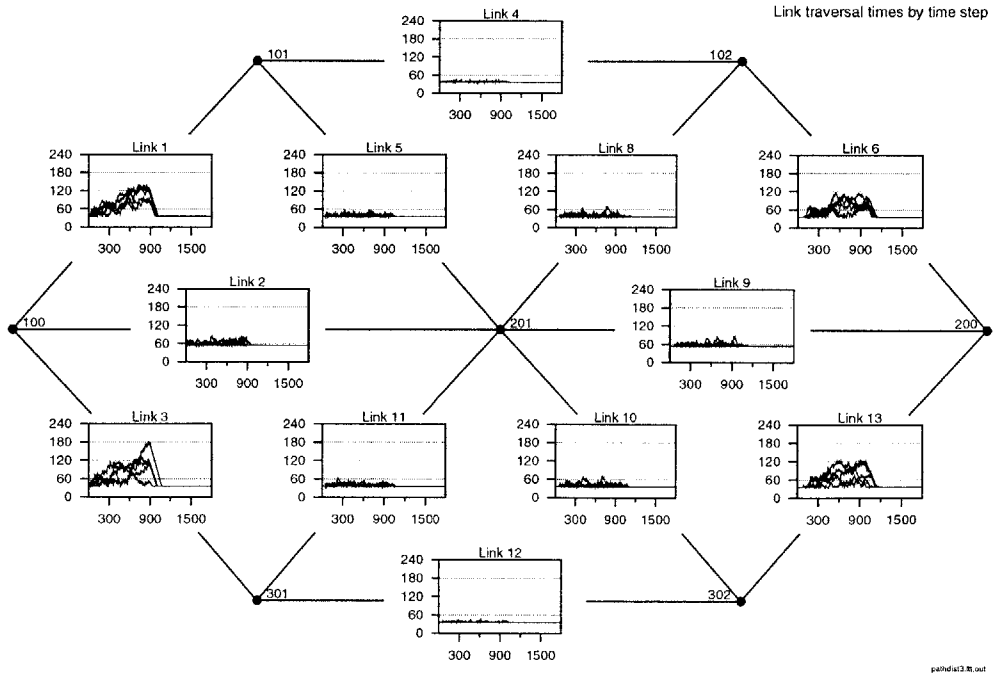


(d) Path time trajectories (seconds) by time step (seconds)

Figure 5.12: Run SD-2 (part 2/2)



(a) Link traversal time trajectories (seconds) by time step (seconds)



(b) Path time trajectories (seconds) by time step (seconds)

Figure 5.13: Run SD-3

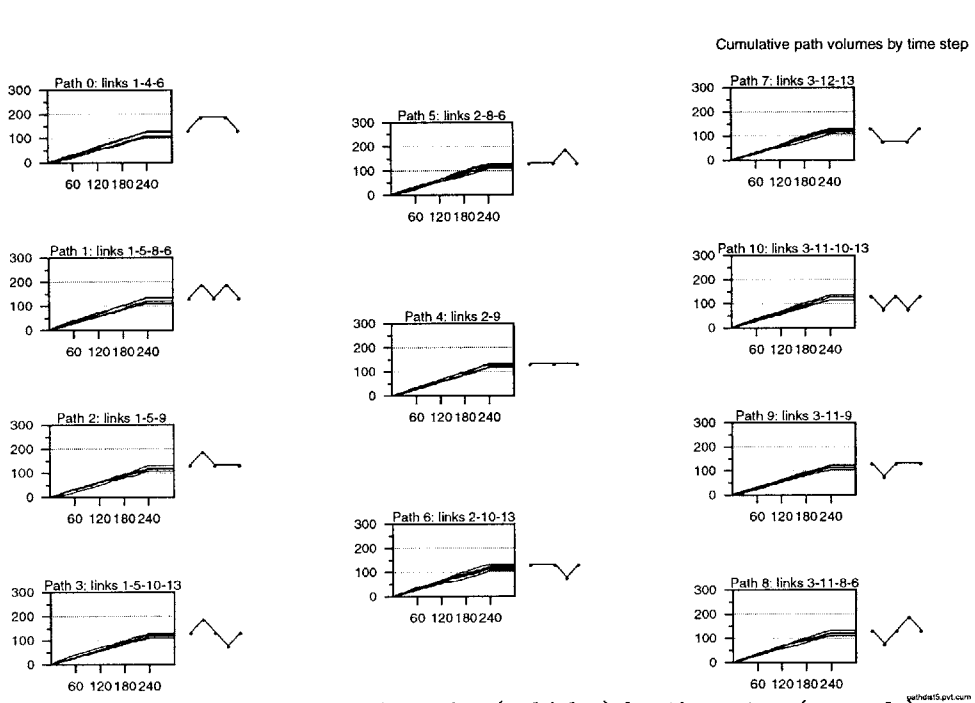
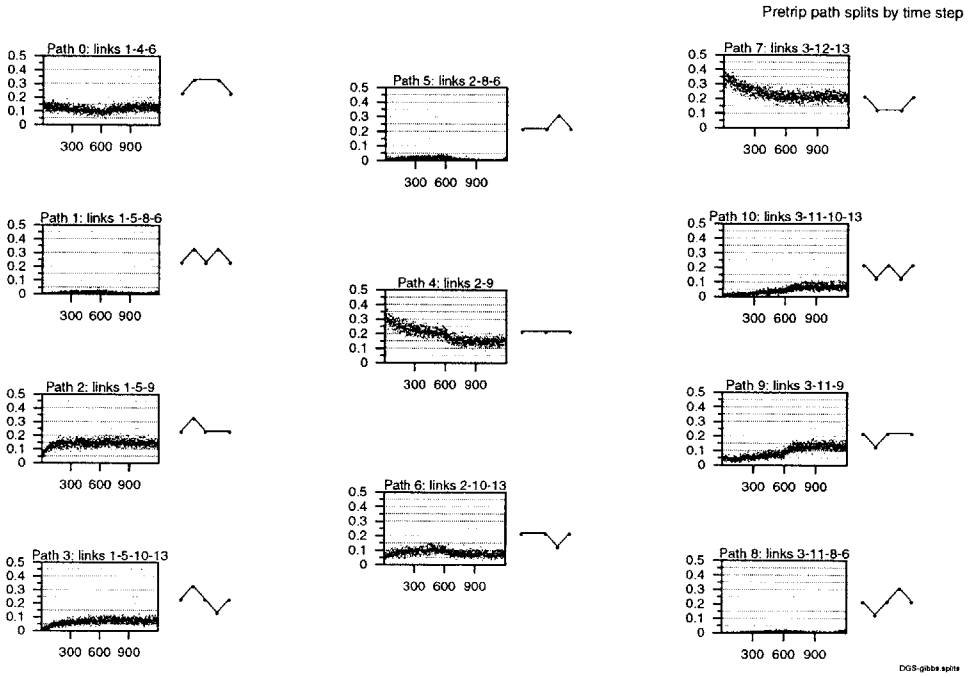
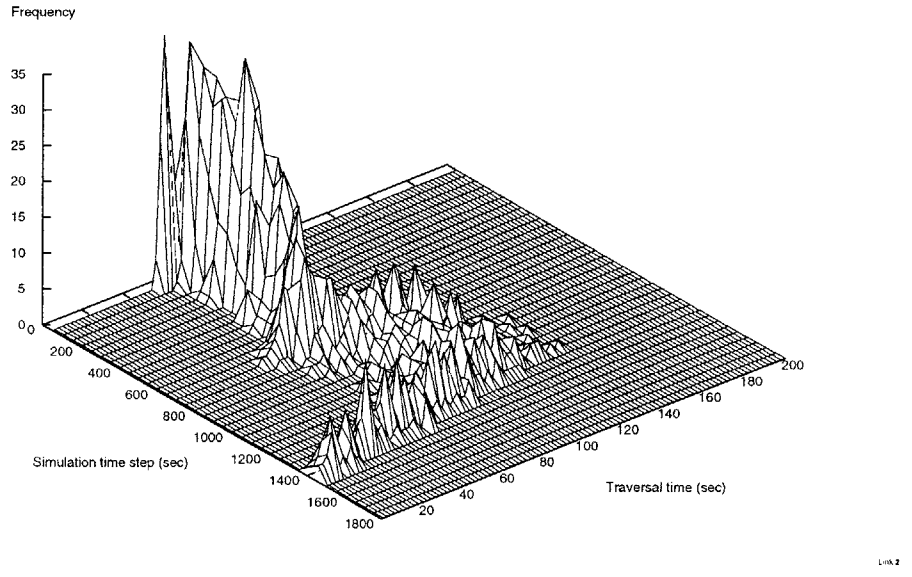


Figure 5.14: Run SD-5

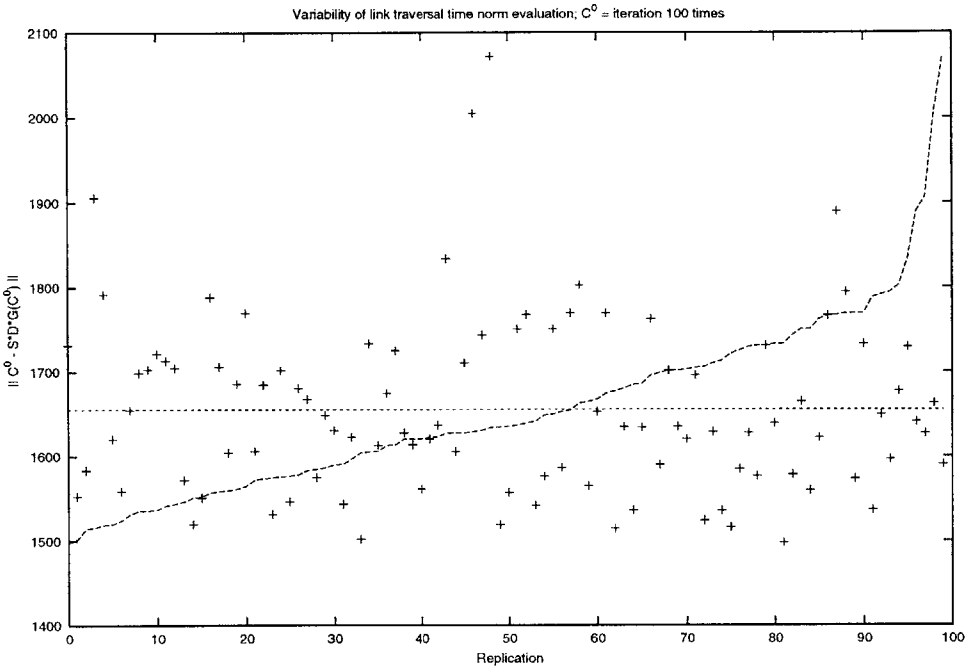


(a) Pre-trip path splits from Gibbs sampling, 100 replications

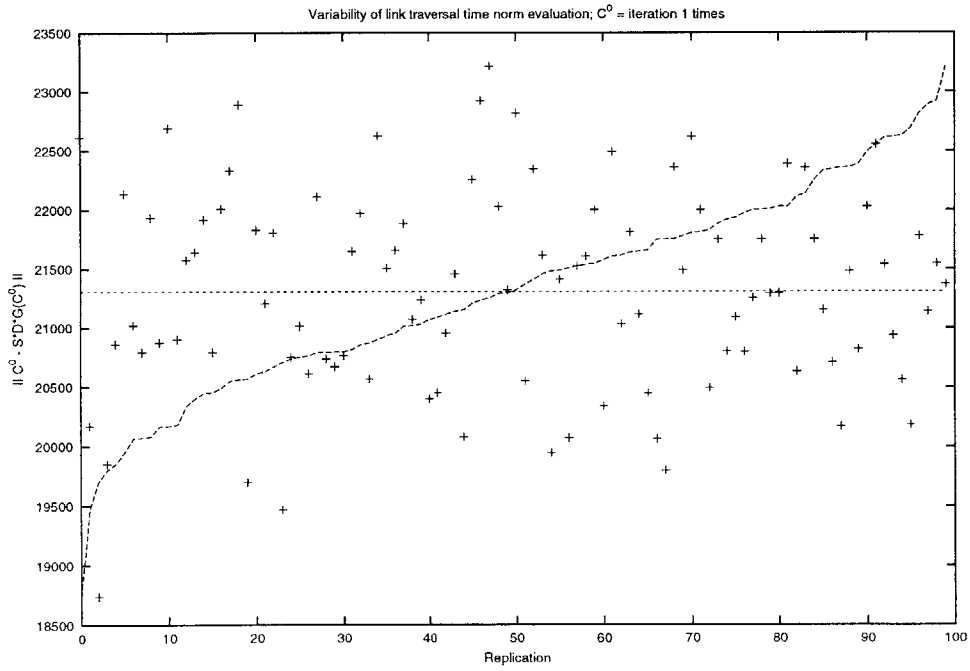


(b) Link 2 volume trajectory distribution from Gibbs sampling, 100 replications

Figure 5.15: Selected results from Gibbs sampling of D ∘ G ∘ S map

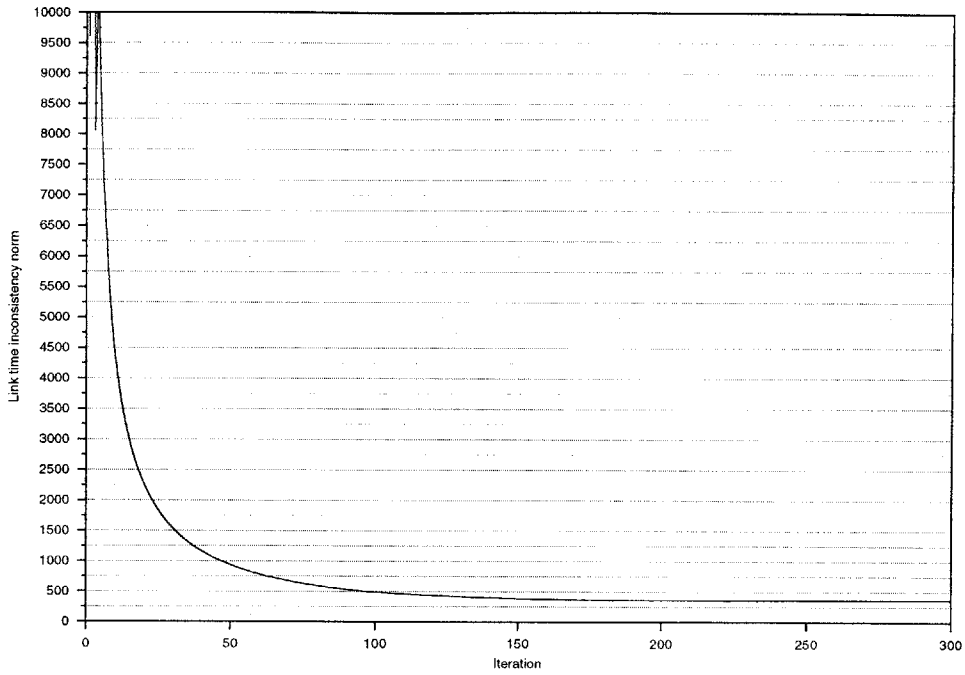


(a) Replications based on C^0 = 100th iteration link traversal times

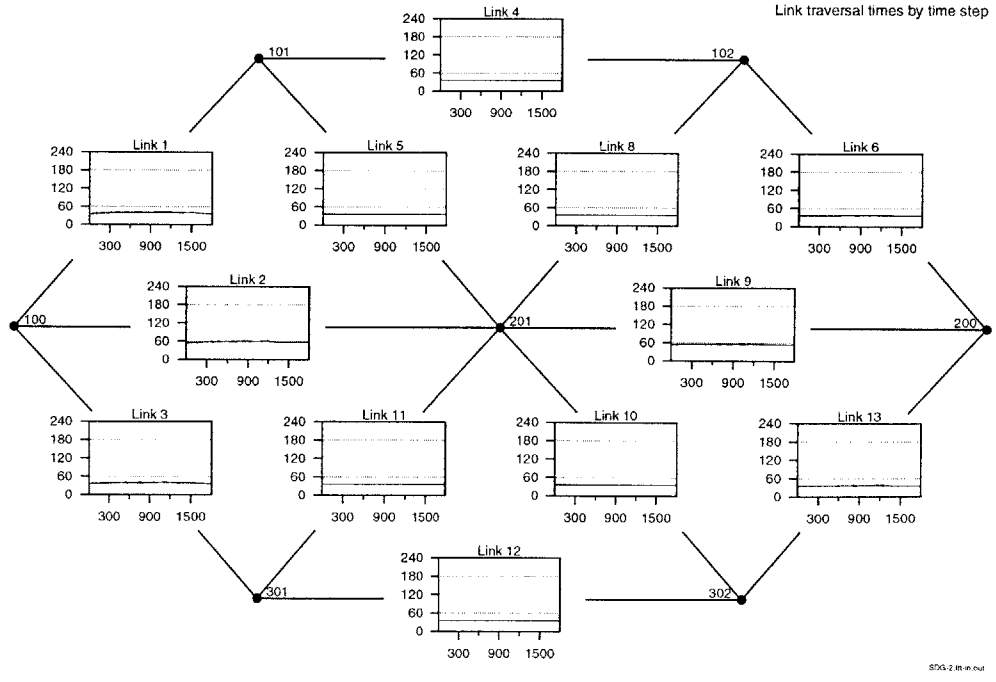


(b) Replications based on C^0 = 1st iteration link traversal times

Figure 5.16: IC_C norms evaluated from 100 replications of the $S \circ D \circ G$ map

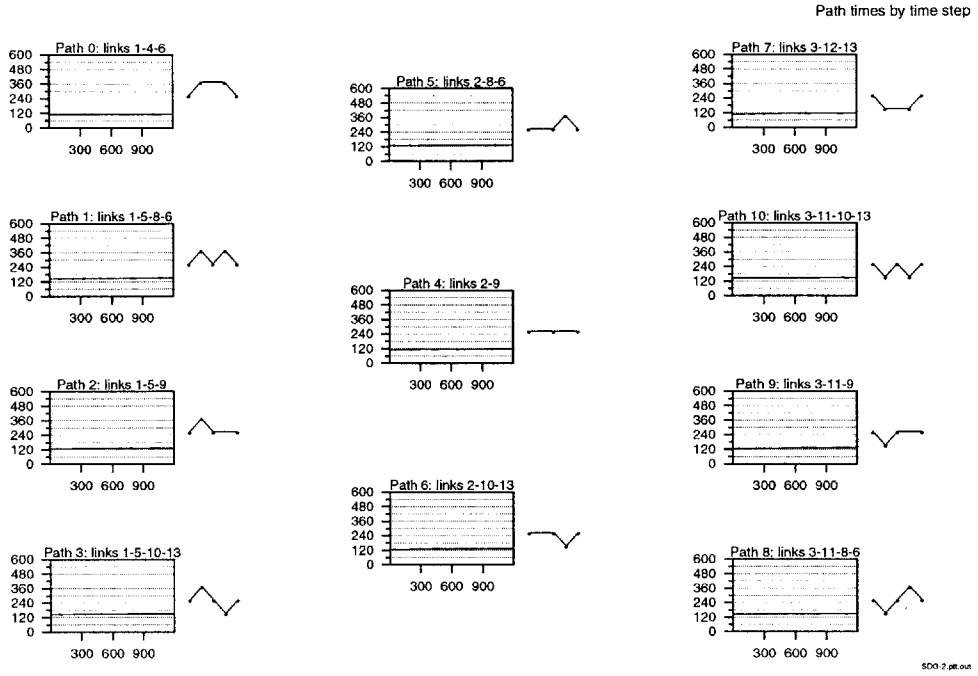


(a) IC_C convergence norm (seconds)

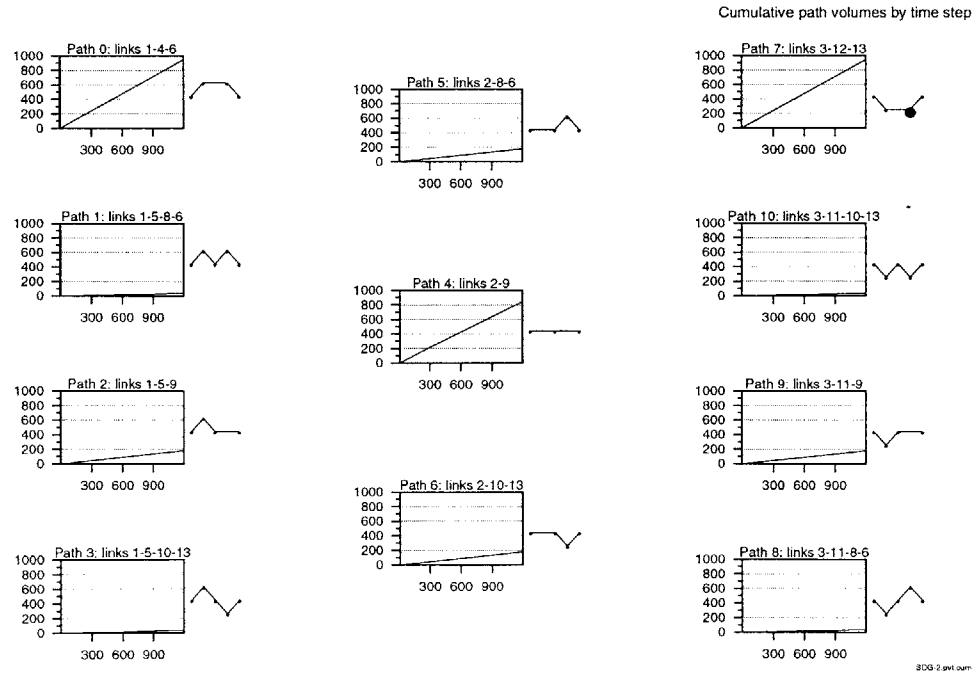


(b) Input link traversal time trajectories (seconds)

Figure 5.17: Run SDG-1 (part 1/3)

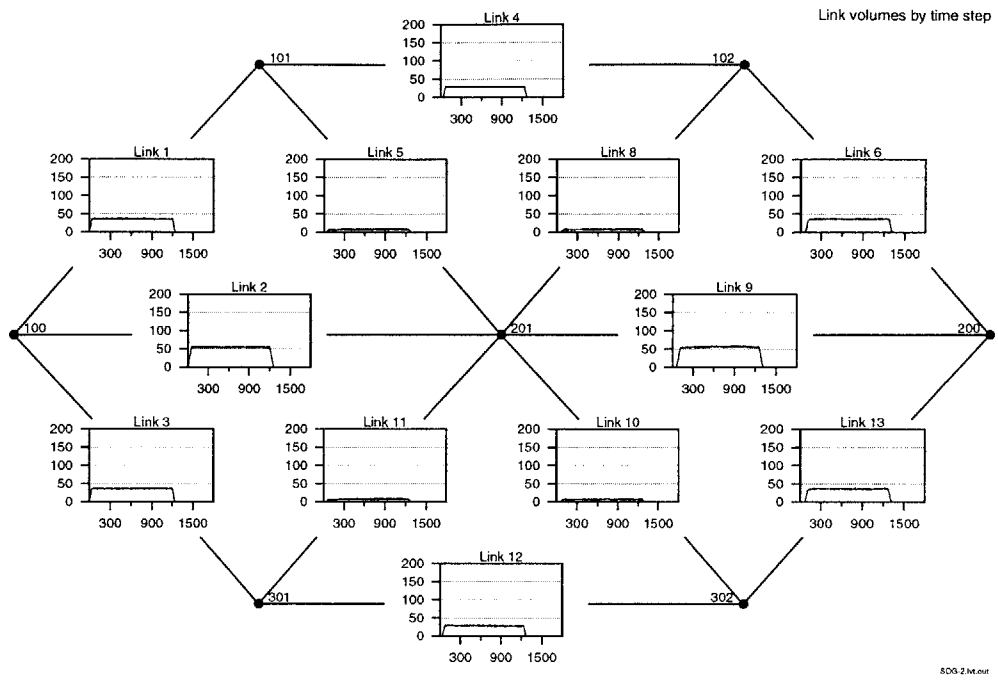


(c) Path time trajectories (seconds)

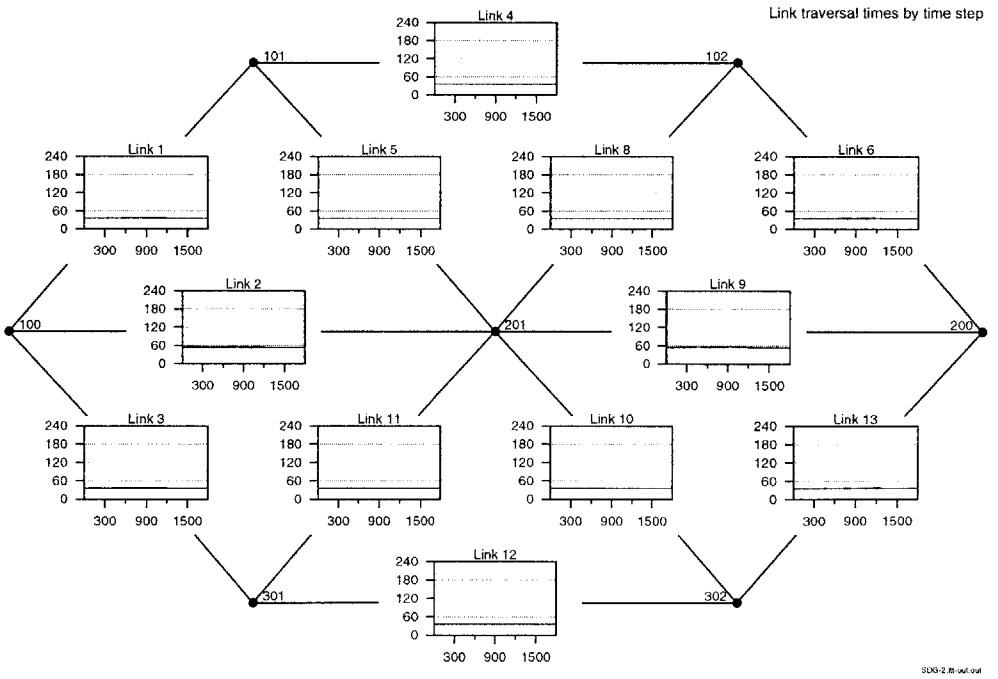


(d) Cumulative path flow trajectories (vehicles)

Figure 5.17: Run SDG-1 (part 2/3)

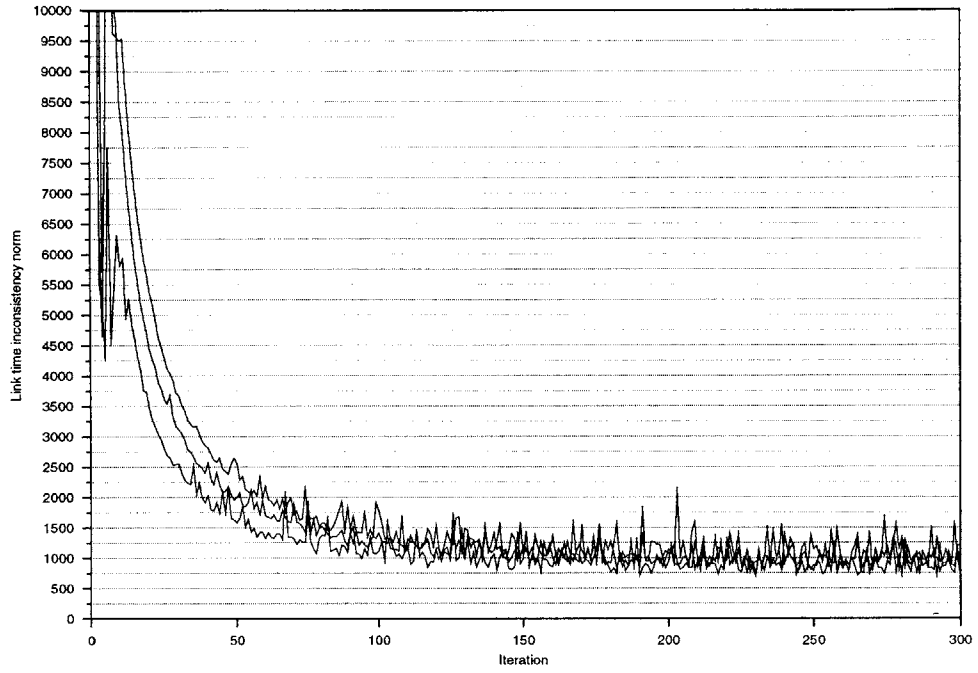


(e) Link volume trajectories (vehicles)

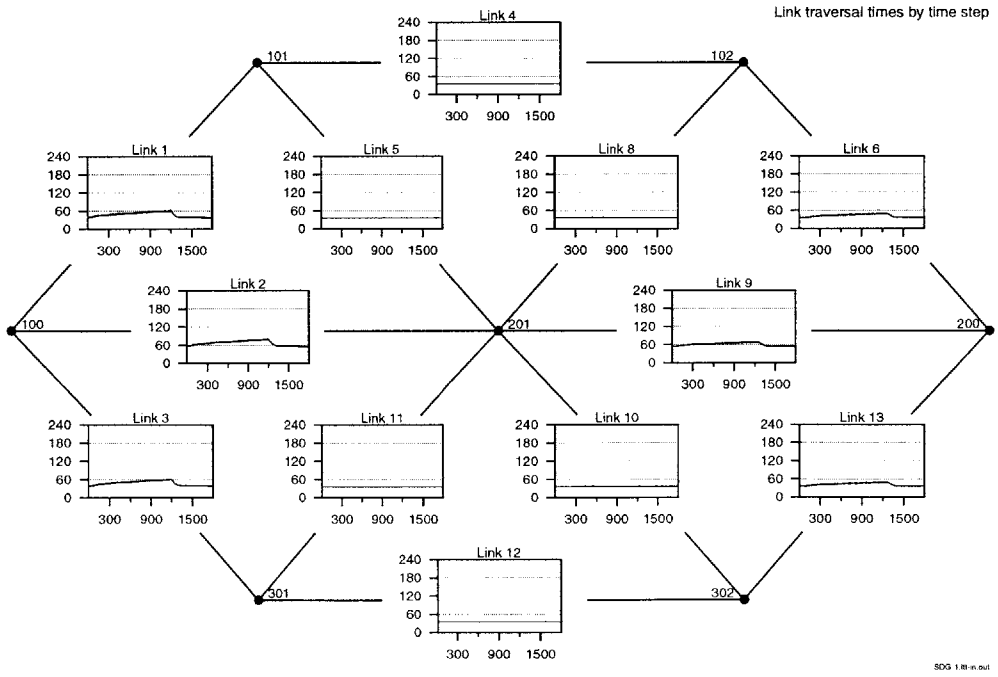


(f) Output link traversal time trajectories (seconds)

Figure 5.17: Run SDG-1 (part 3/3)

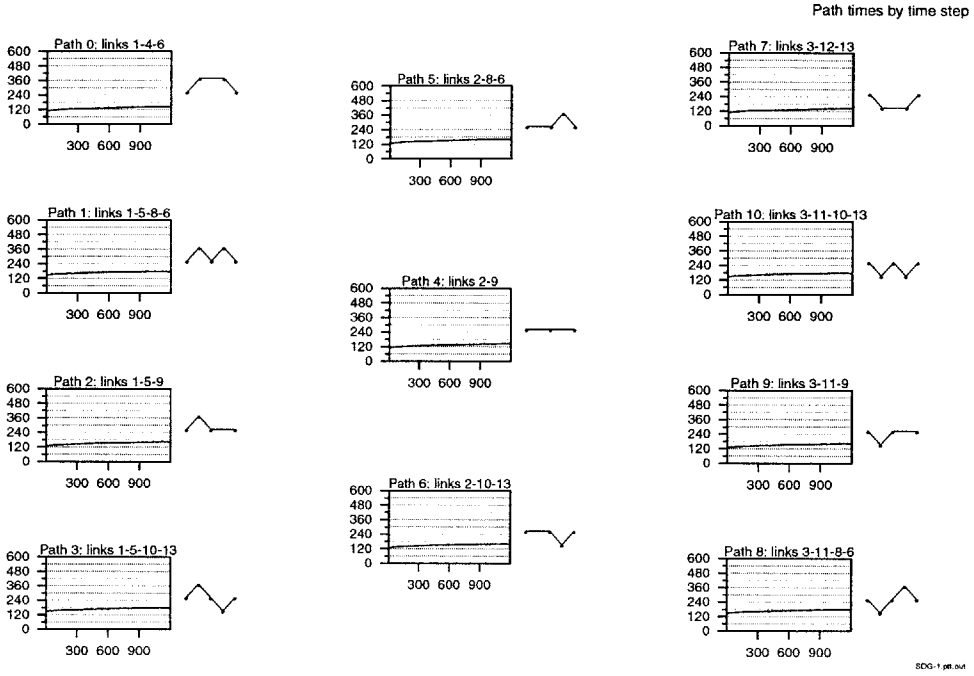


(a) IC_C convergence norm (seconds)

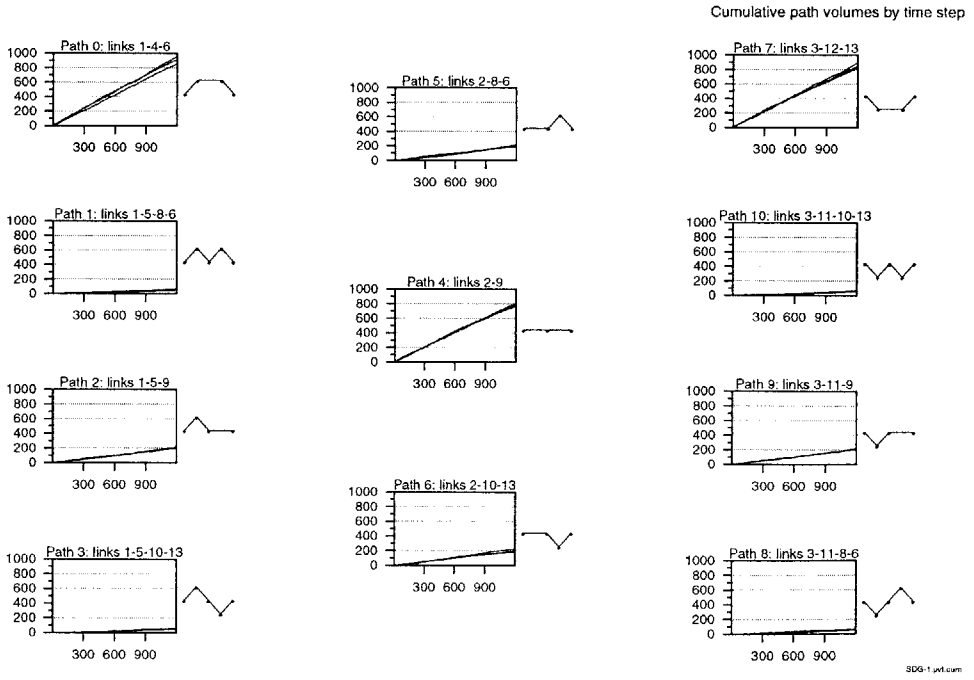


(b) Input link traversal time trajectories (seconds)

Figure 5.18: Run SDG-2 (part 1/3)

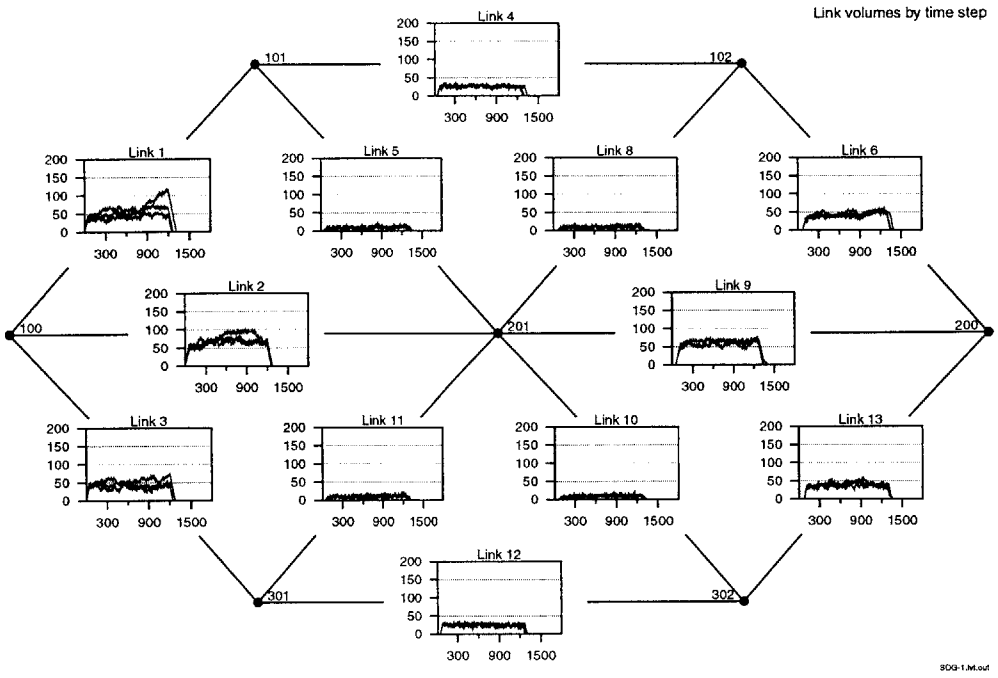


(c) Path time trajectories (seconds)

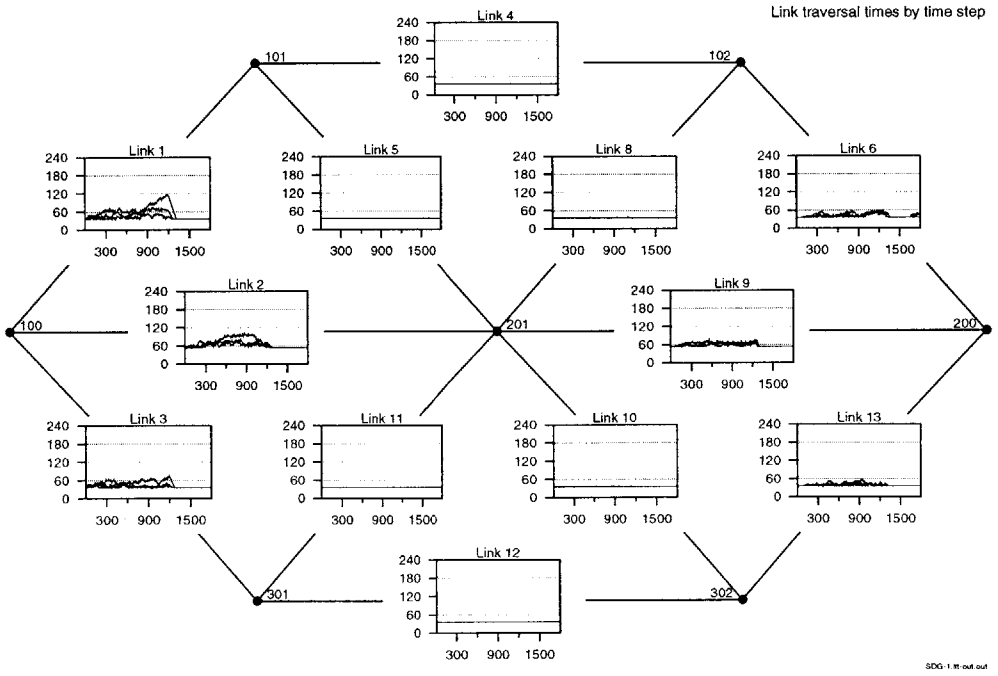


(d) Cumulative path flow trajectories (vehicles)

Figure 5.18: Run SDG-2 (part 2/3)

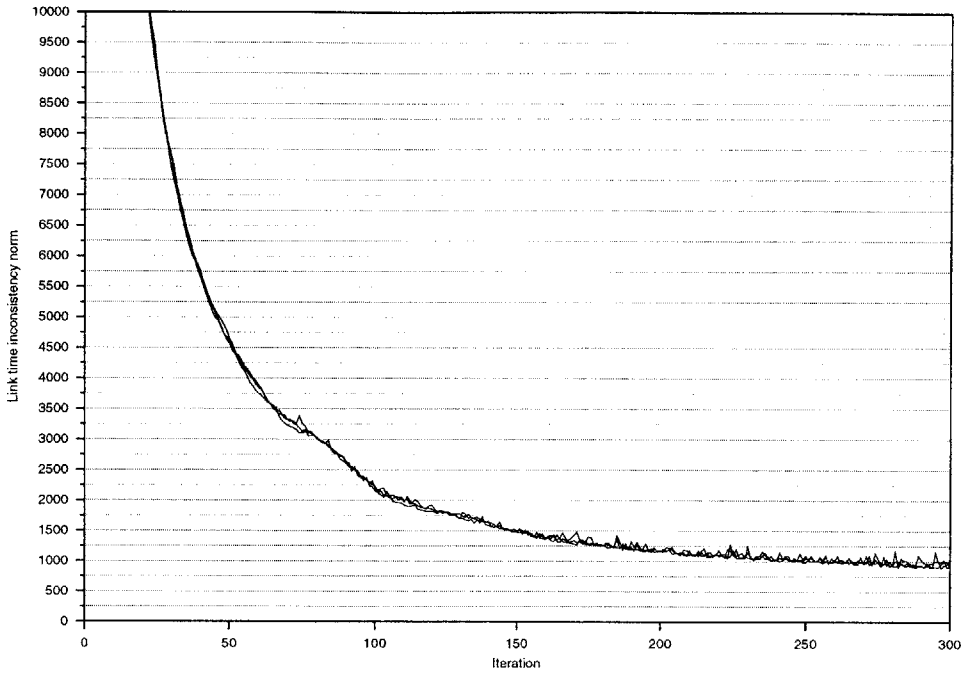


(e) Link volume trajectories (vehicles)

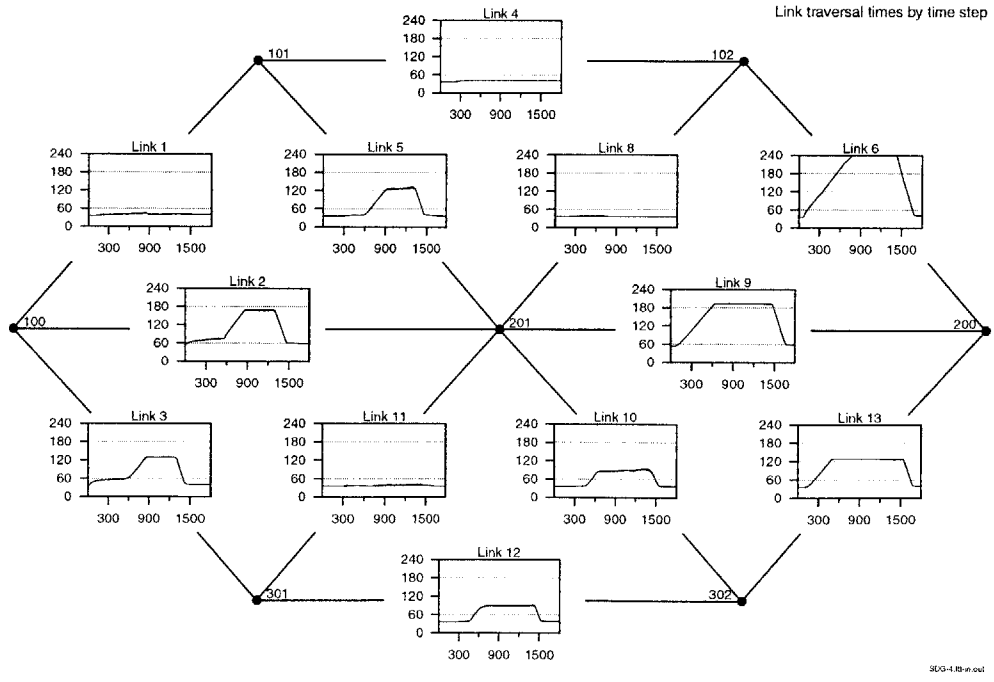


(f) Output link traversal time trajectories (seconds)

Figure 5.18: Run SDG-2 (part 3/3)

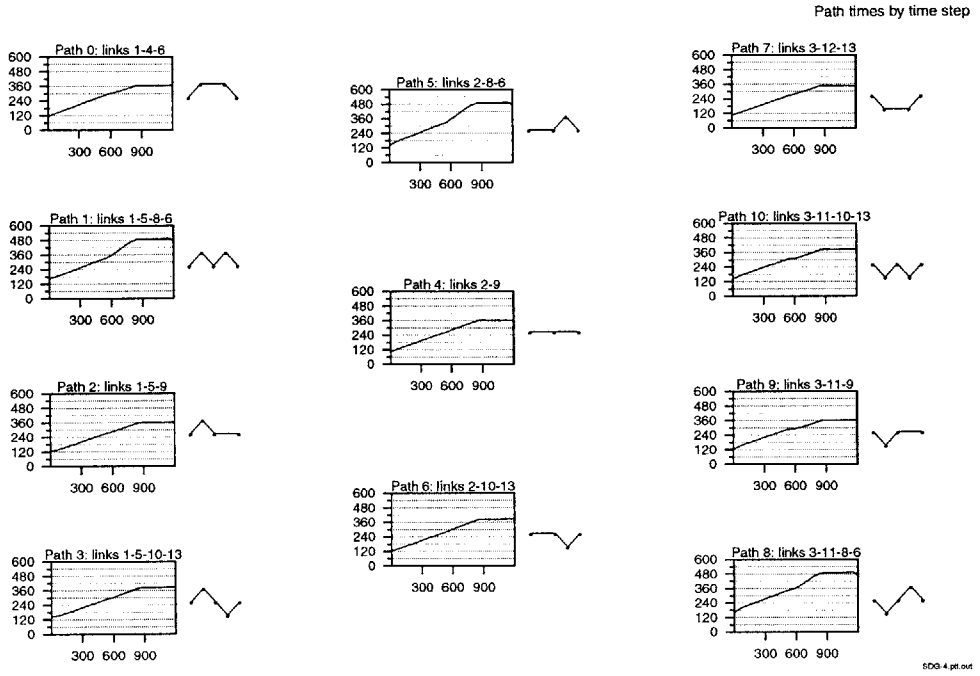


(a) IC_C convergence norm (seconds)

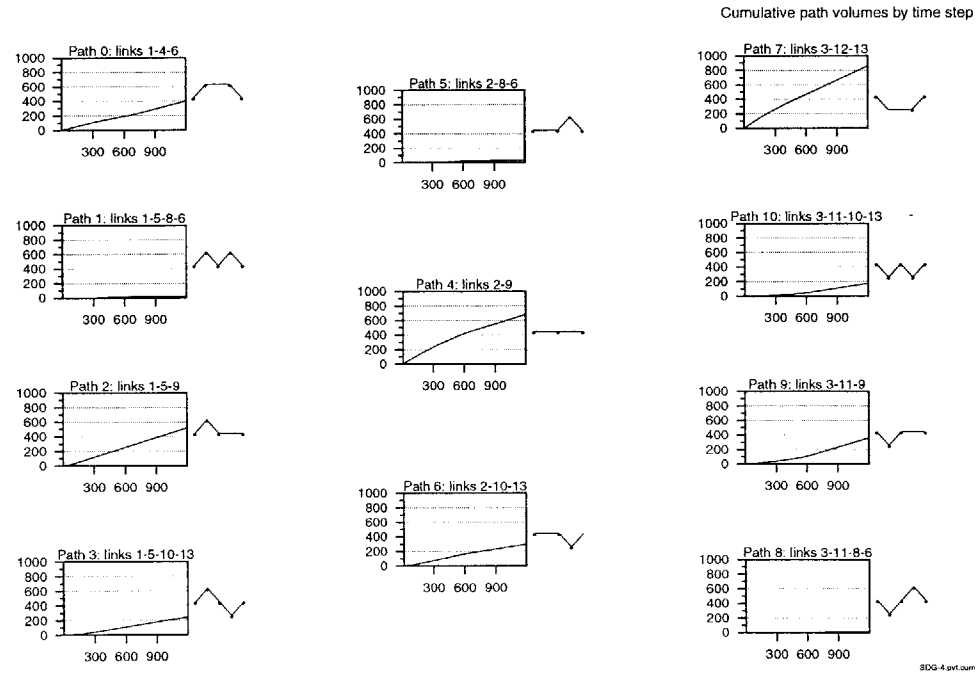


(b) Input link traversal time trajectories (seconds)

Figure 5.19: Run SDG-3 (part 1/3)

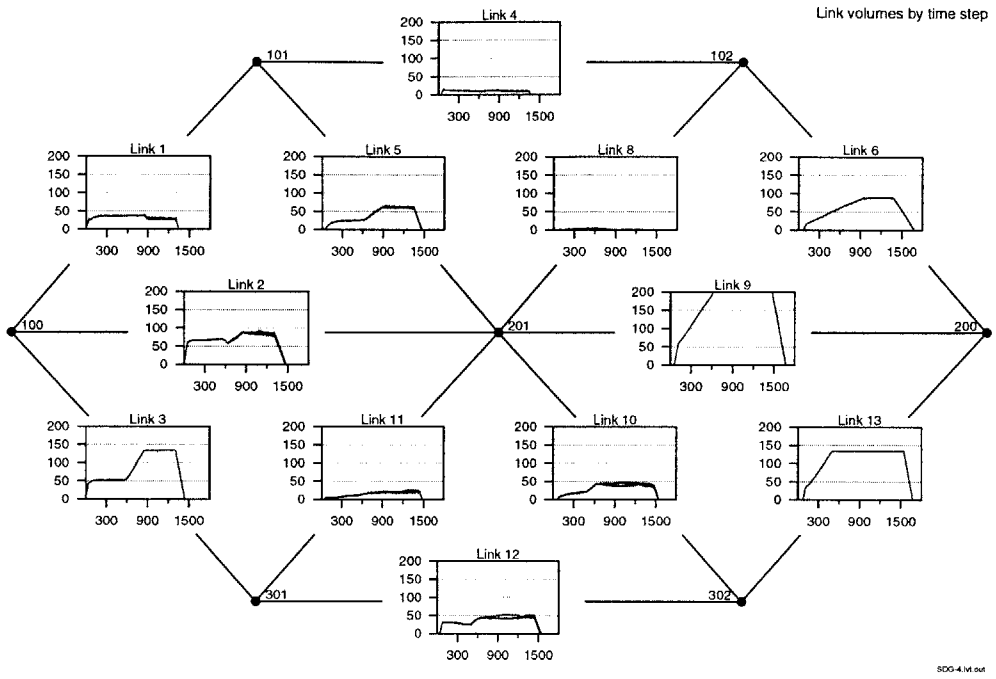


(c) Path time trajectories (seconds)

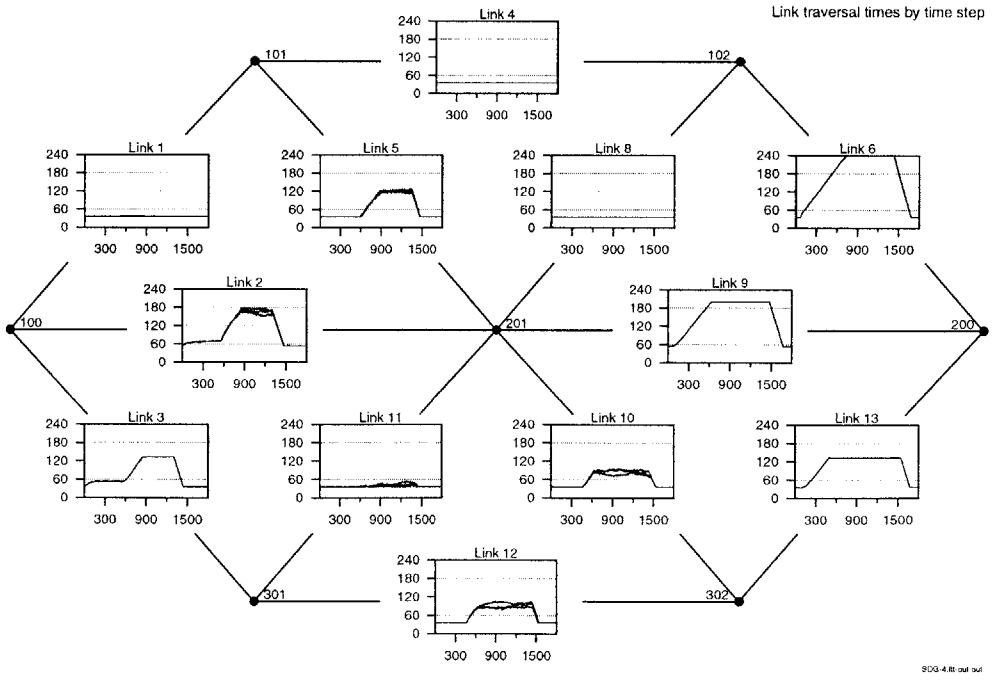


(d) Cumulative path flow trajectories (vehicles)

Figure 5.19: Run SDG-3 (part 2/3)

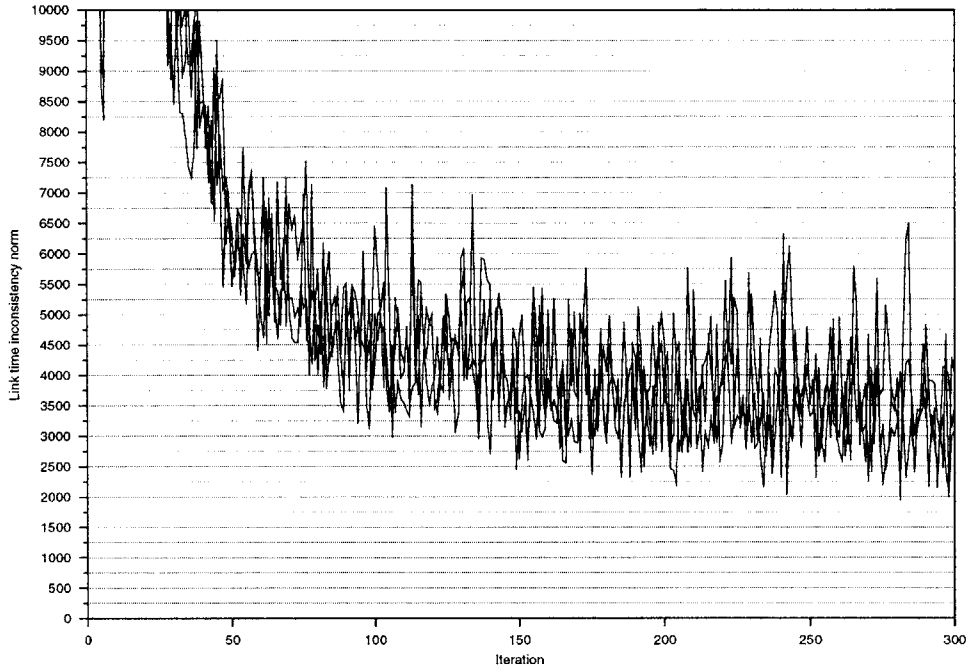


(e) Link volume trajectories (vehicles)

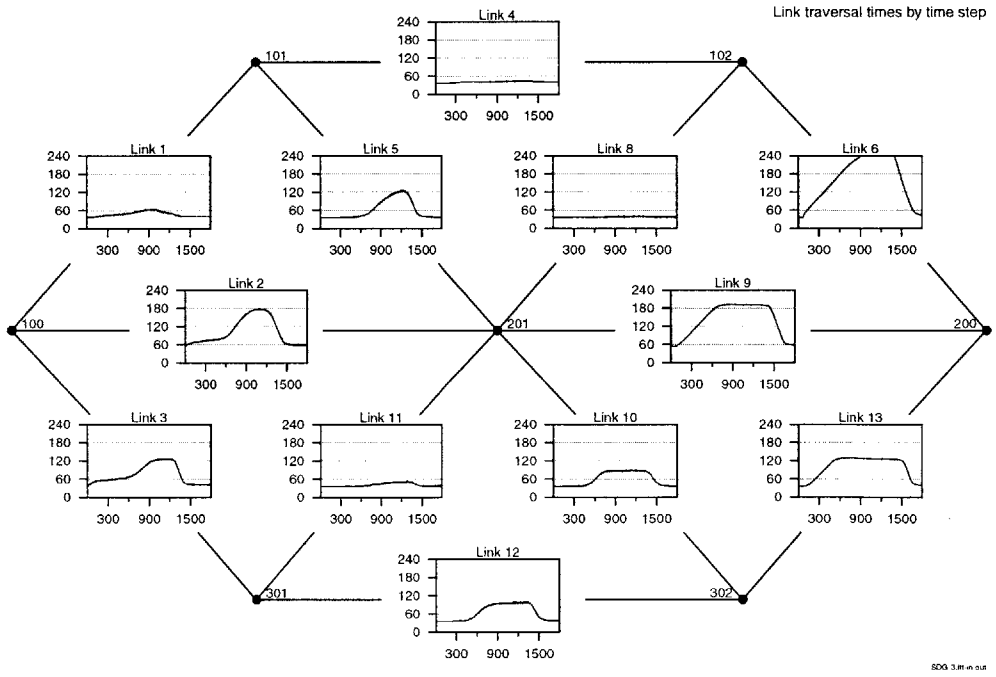


(f) Output link traversal time trajectories (seconds)

Figure 5.19: Run SDG-3 (part 3/3)

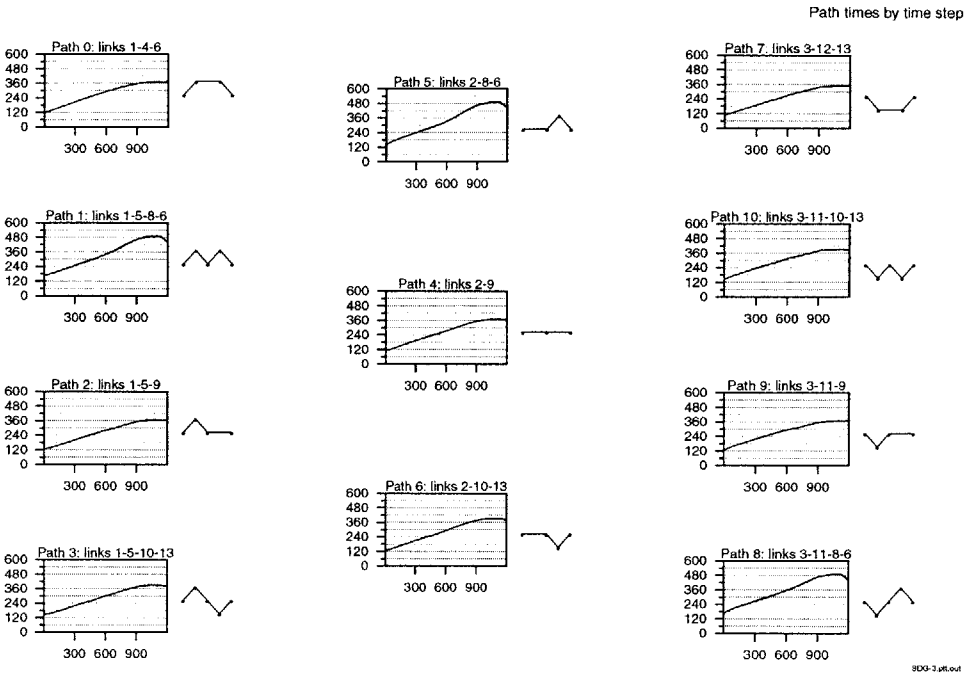


(a) IC_C convergence norm (seconds)

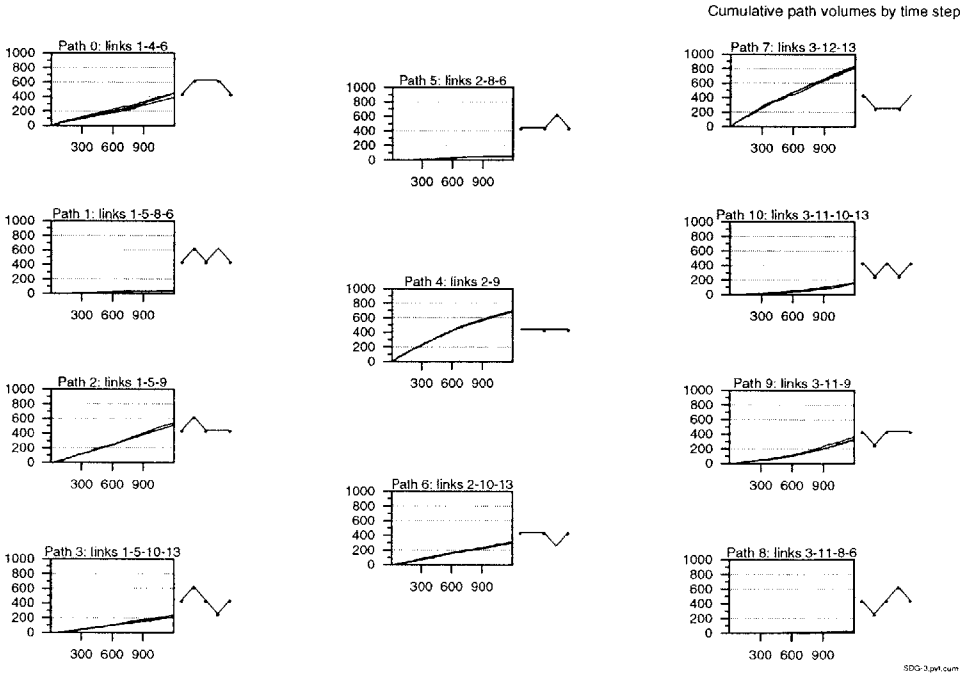


(b) Input link traversal time trajectories (seconds)

Figure 5.20: Run SDG-4 (part 1/3)

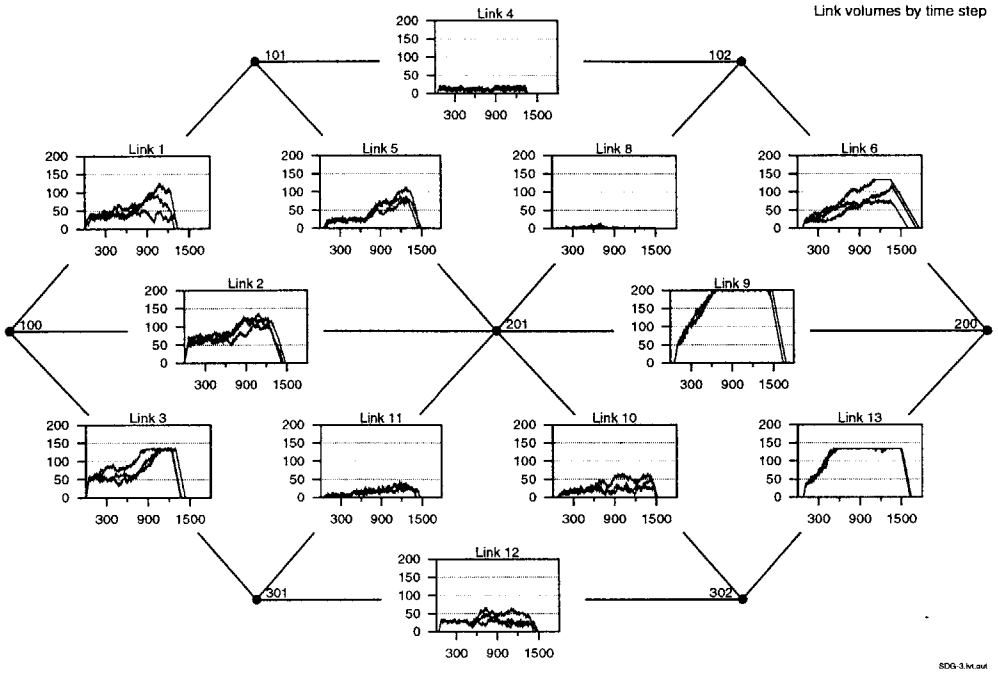


(c) Path time trajectories (seconds)

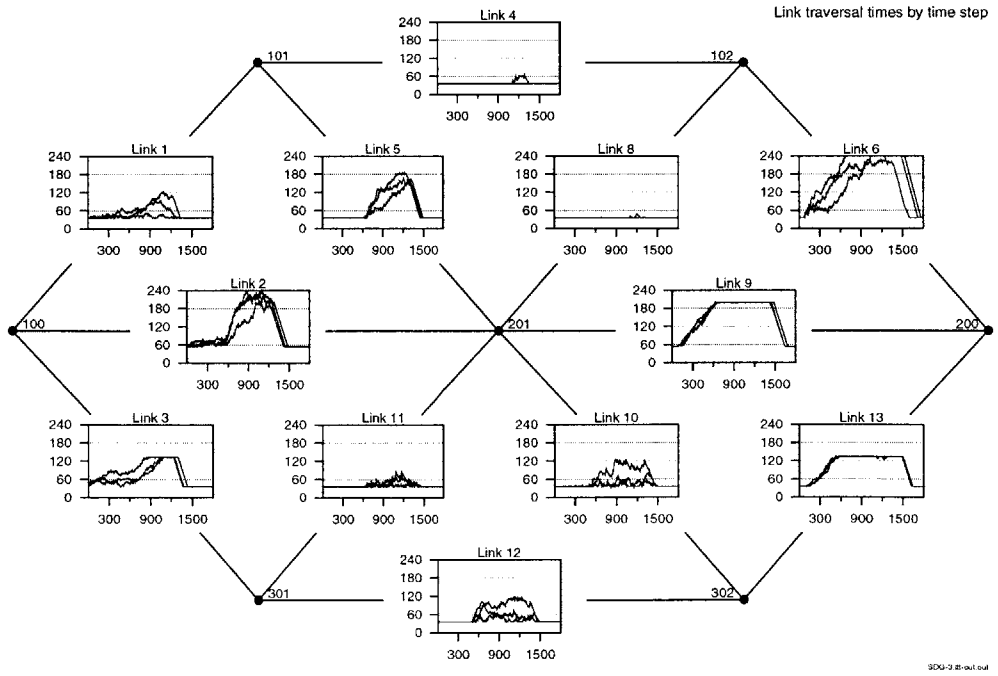


(d) Cumulative path flow trajectories (vehicles)

Figure 5.20: Run SDG-4 (part 2/3)

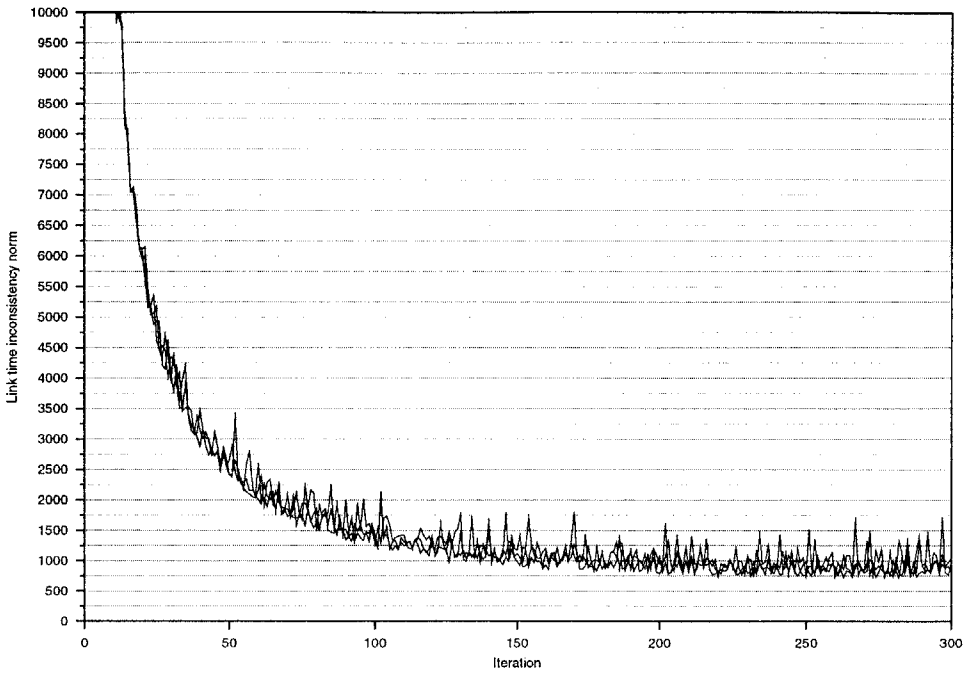


(e) Link volume trajectories (vehicles)

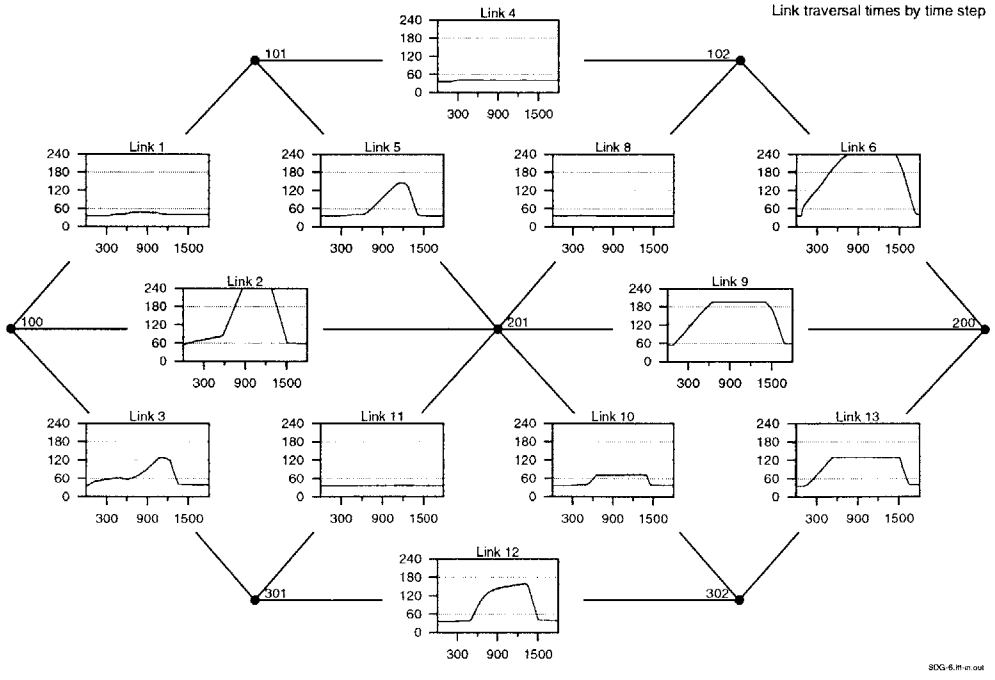


(f) Output link traversal time trajectories (seconds)

Figure 5.20: Run SDG-4 (part 3/3)

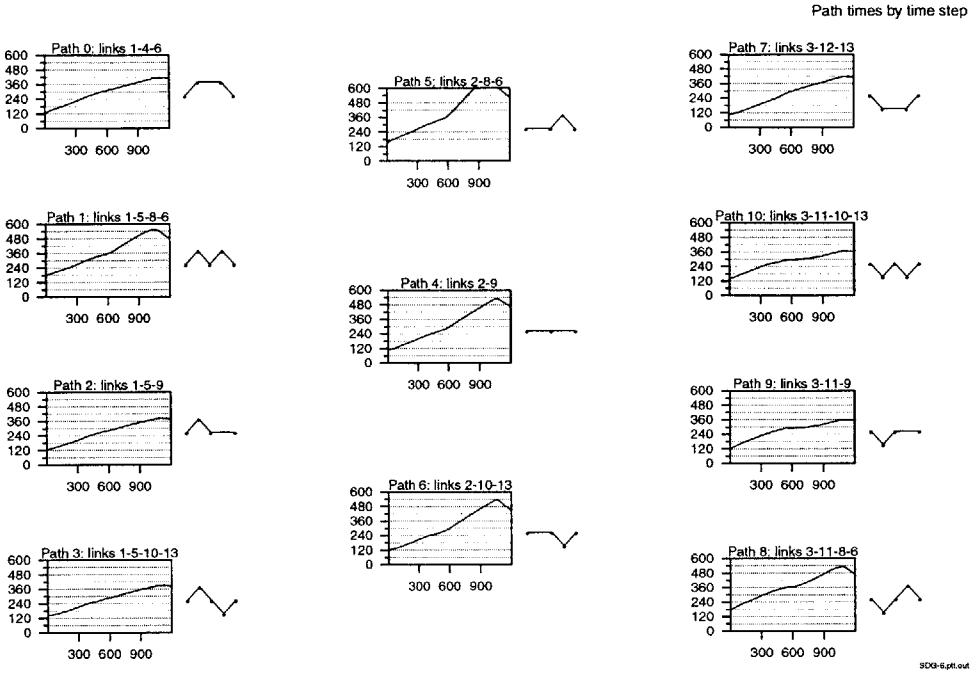


(a) IC_C convergence norm (seconds)

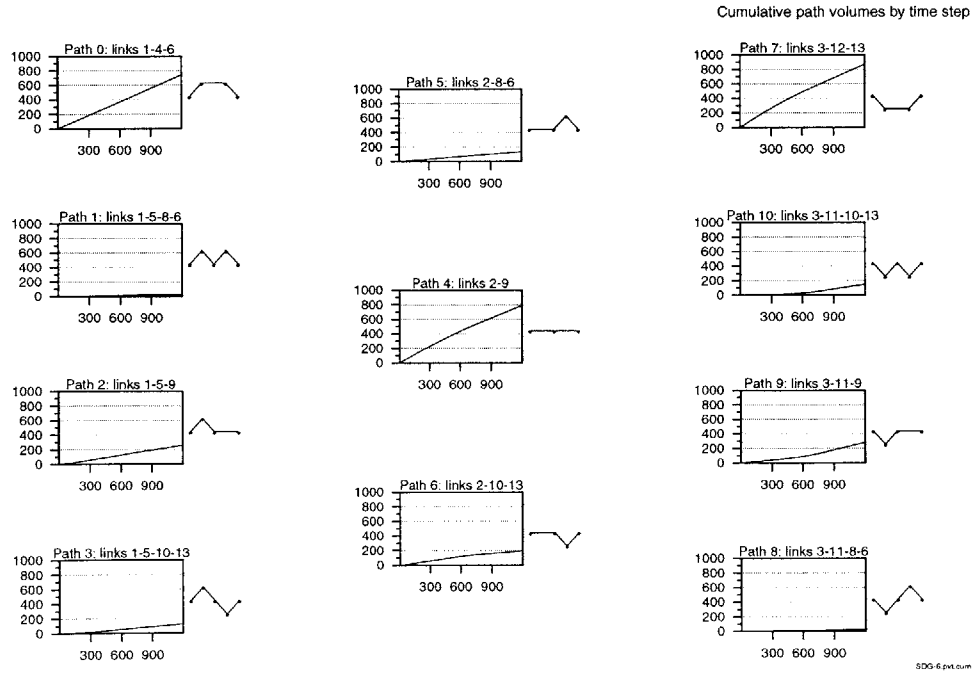


(b) Input link traversal time trajectories (seconds)

Figure 5.21: Run SDG-5 (part 1/3)

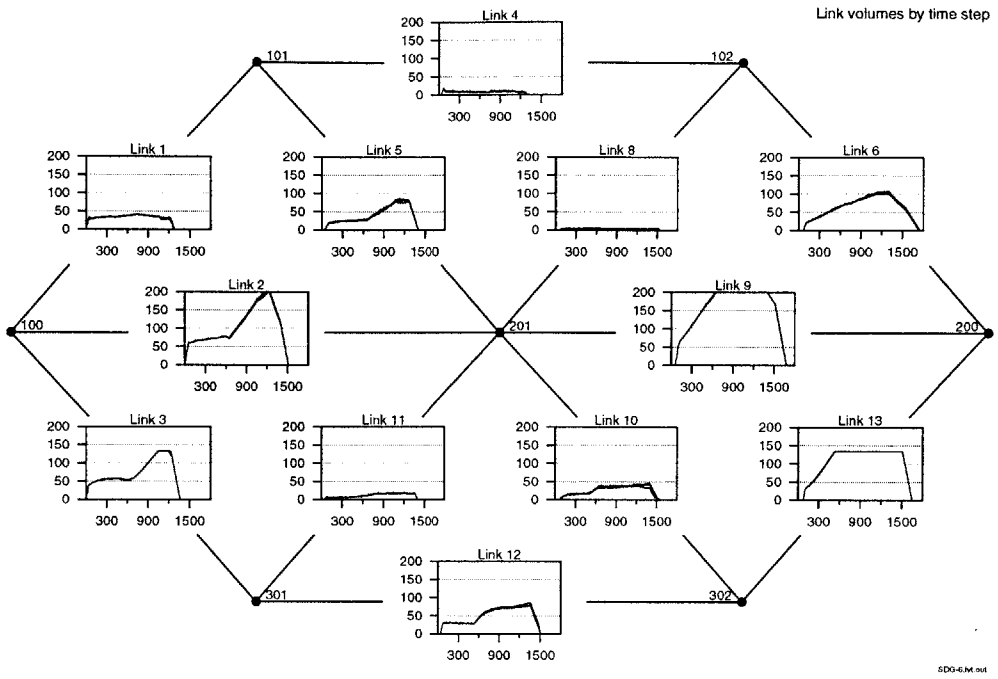


(c) Path time trajectories (seconds)

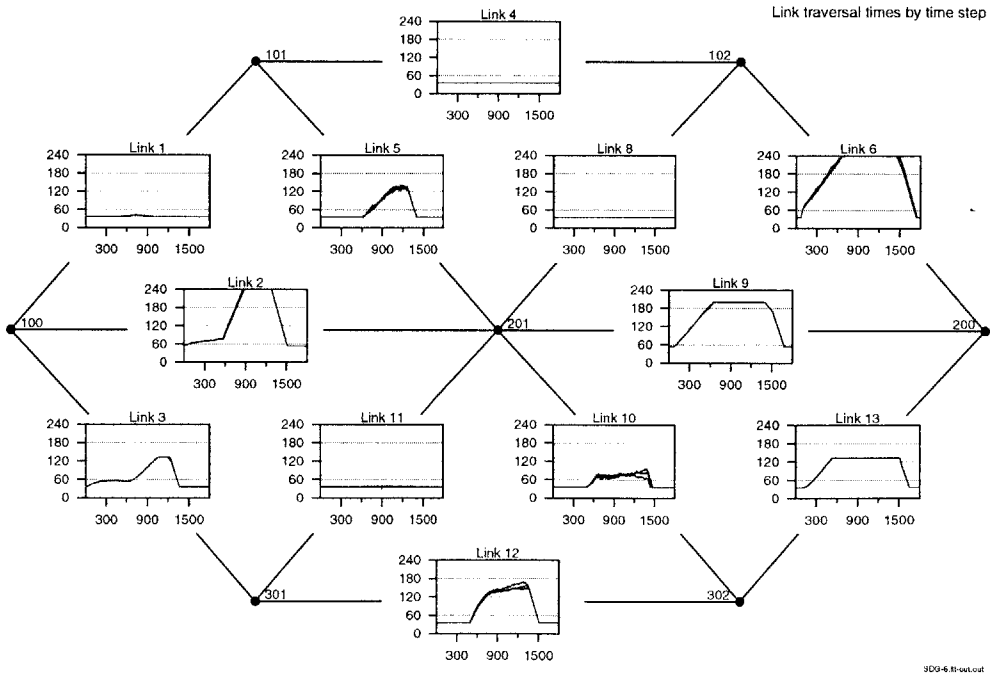


(d) Cumulative path flow trajectories (vehicles)

Figure 5.21: Run SDG-5 (part 2/3)

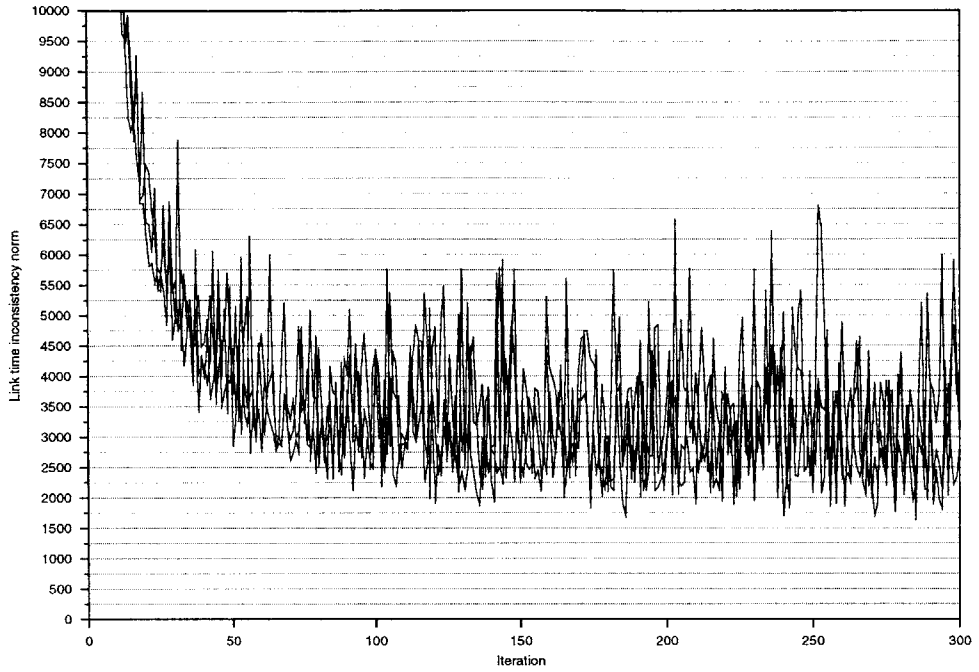


(e) Link volume trajectories (vehicles)

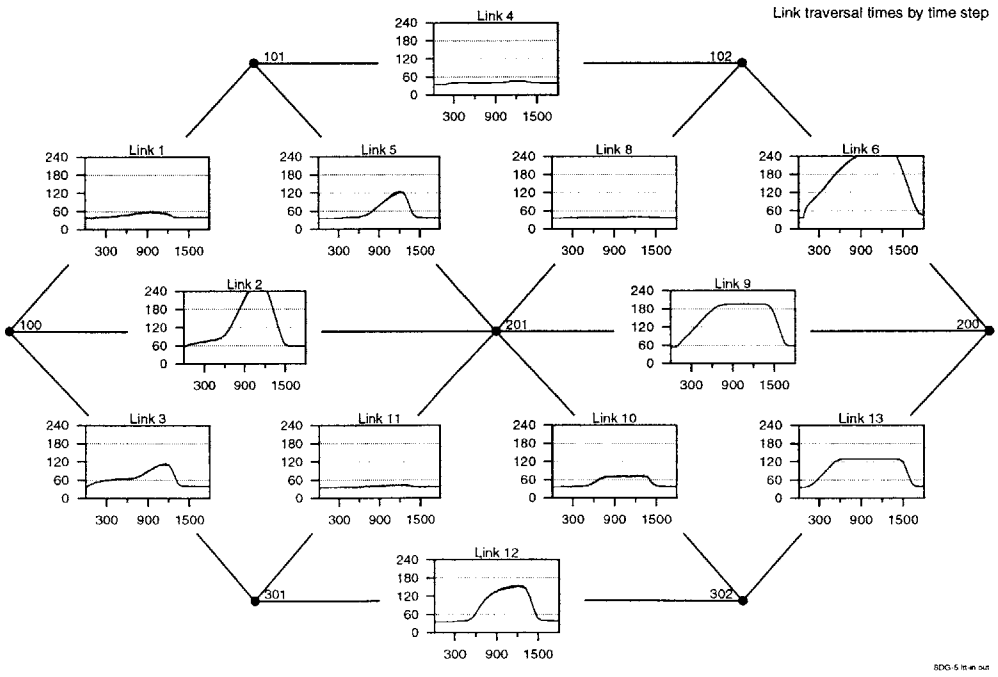


(f) Output link traversal time trajectories (seconds)

Figure 5.21: Run SDG-5 (part 3/3)

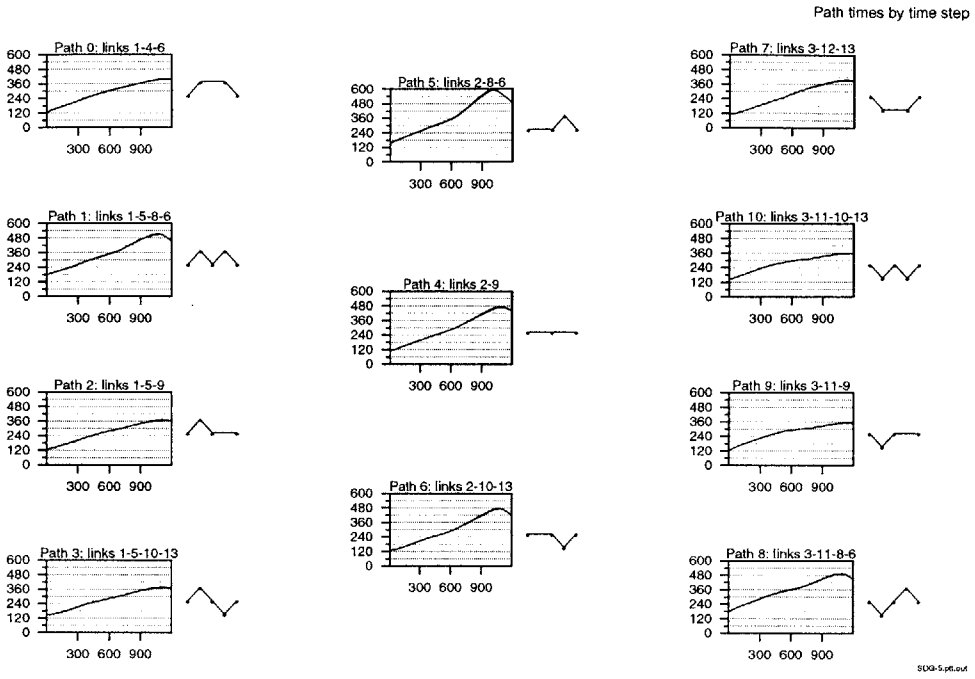


(a) IC_C convergence norm (seconds)

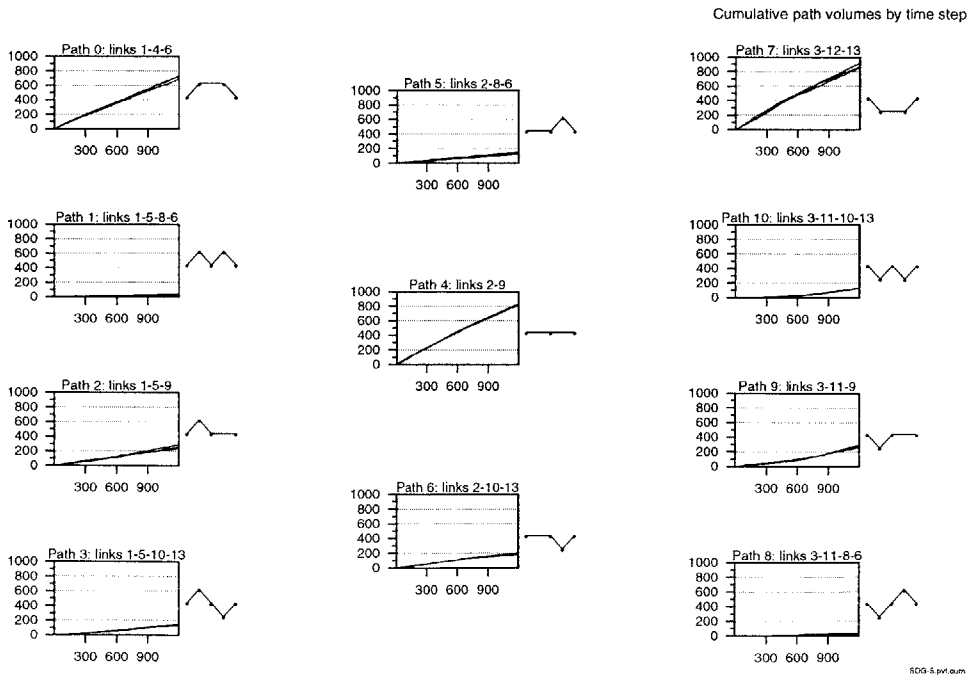


(b) Input link traversal time trajectories (seconds)

Figure 5.22: Run SDG-6 (part 1/3)

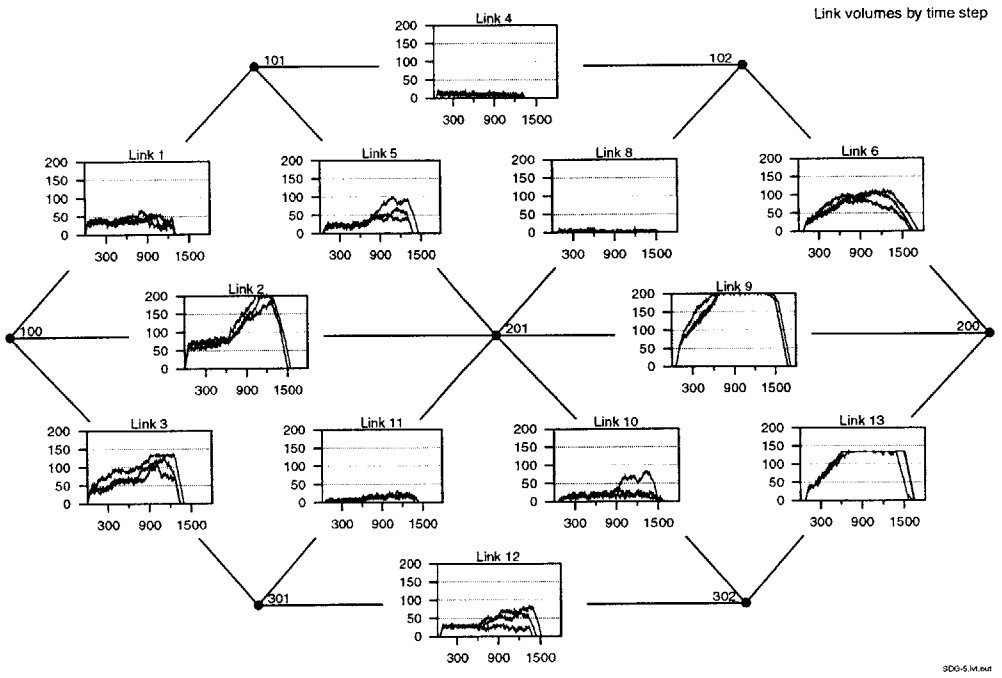


(c) Path time trajectories (seconds)

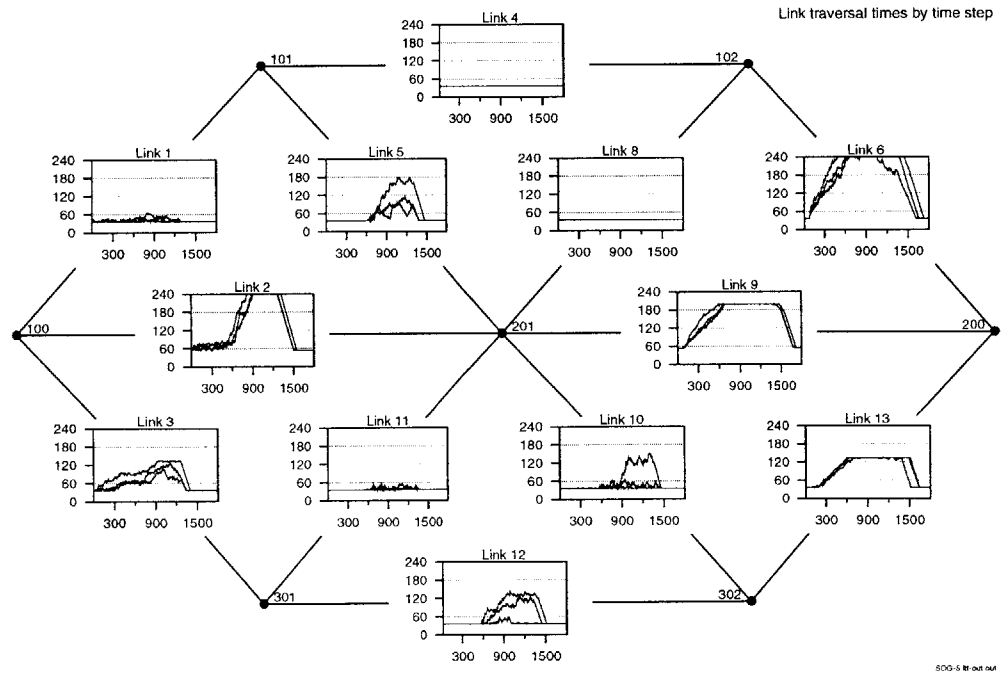


(d) Cumulative path flow trajectories (vehicles)

Figure 5.22: Run SDG-6 (part 2/3)

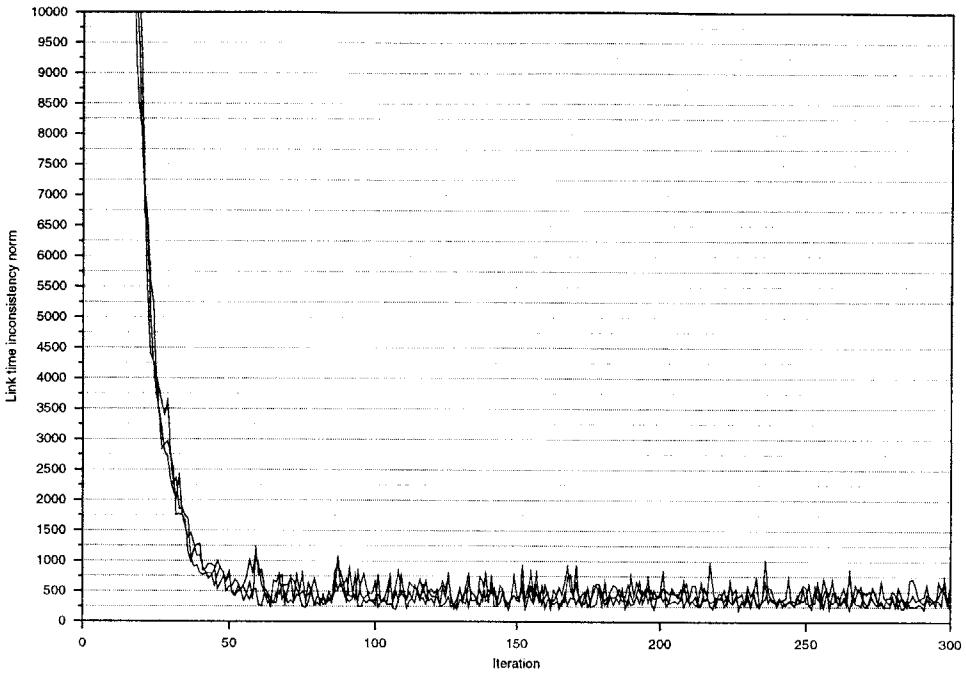


(e) Link volume trajectories (vehicles)

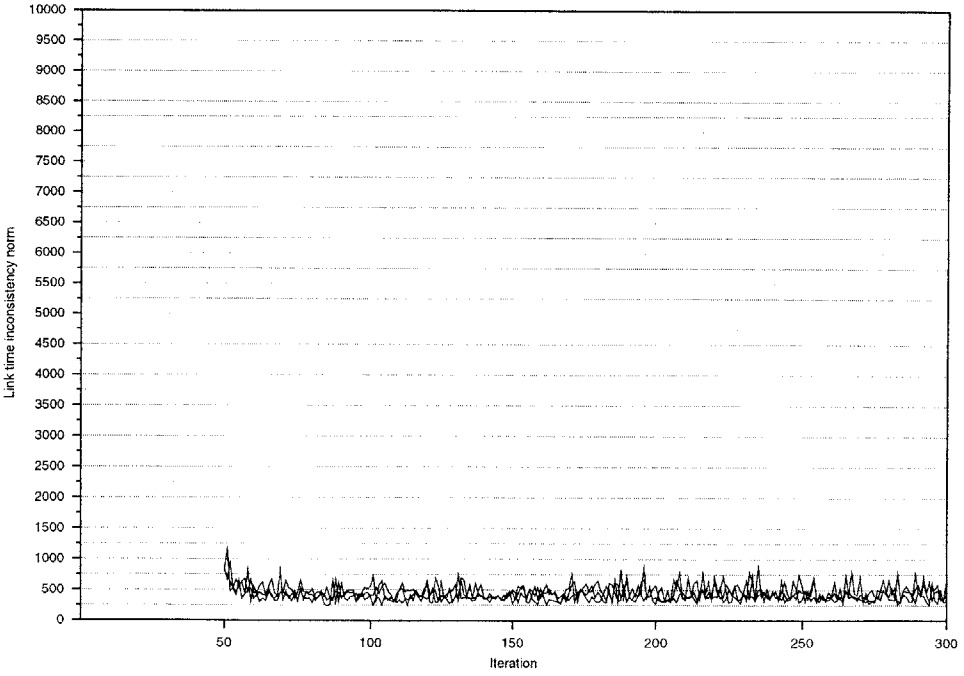


(f) Output link traversal time trajectories (seconds)

Figure 5.22: Run SDG-6 (part 3/3)

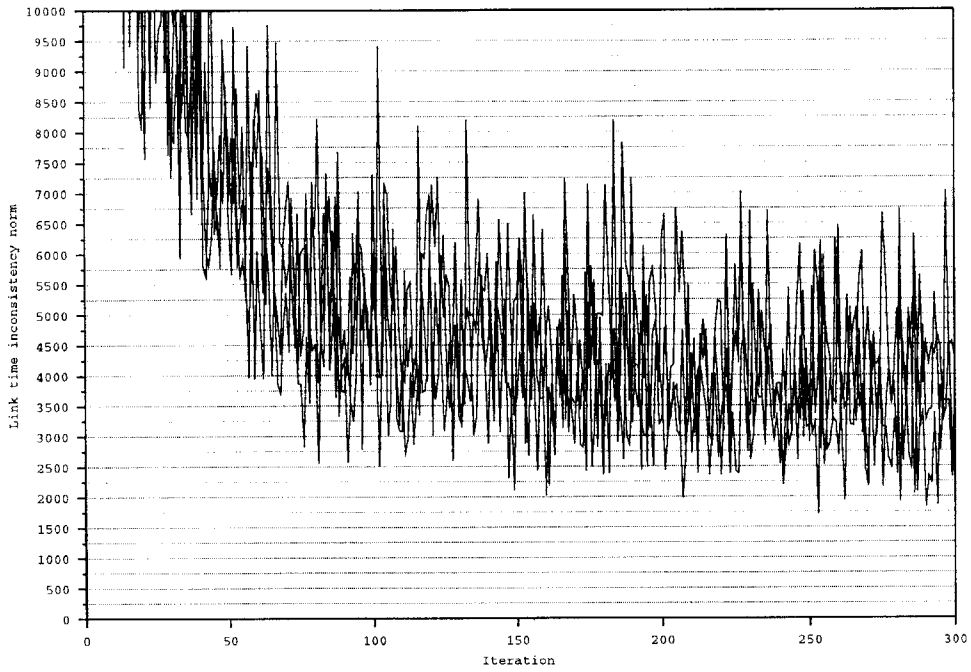


(a) IC_C convergence norm (seconds) in first pass

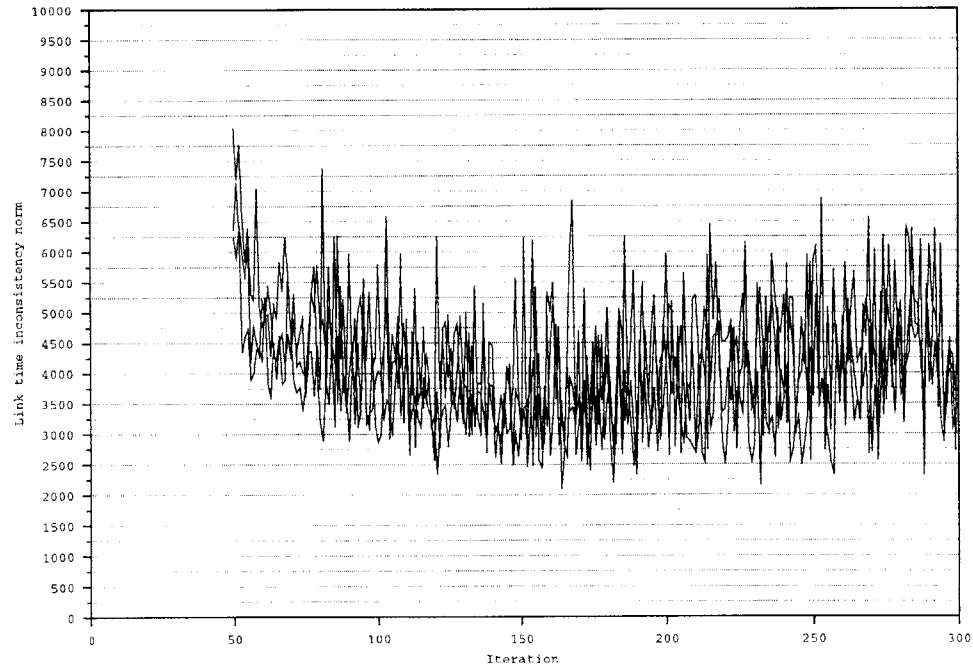


(b) IC_C convergence norm (seconds) in second pass

Figure 5.23: Run SDG-3-pol

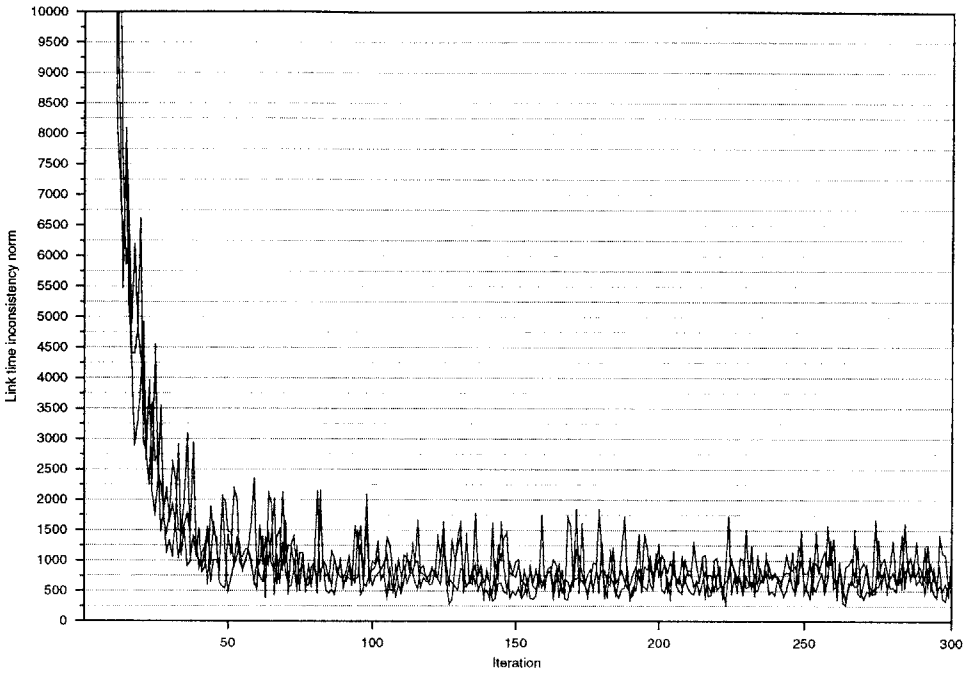


(a) IC_C convergence norm (seconds) in first pass

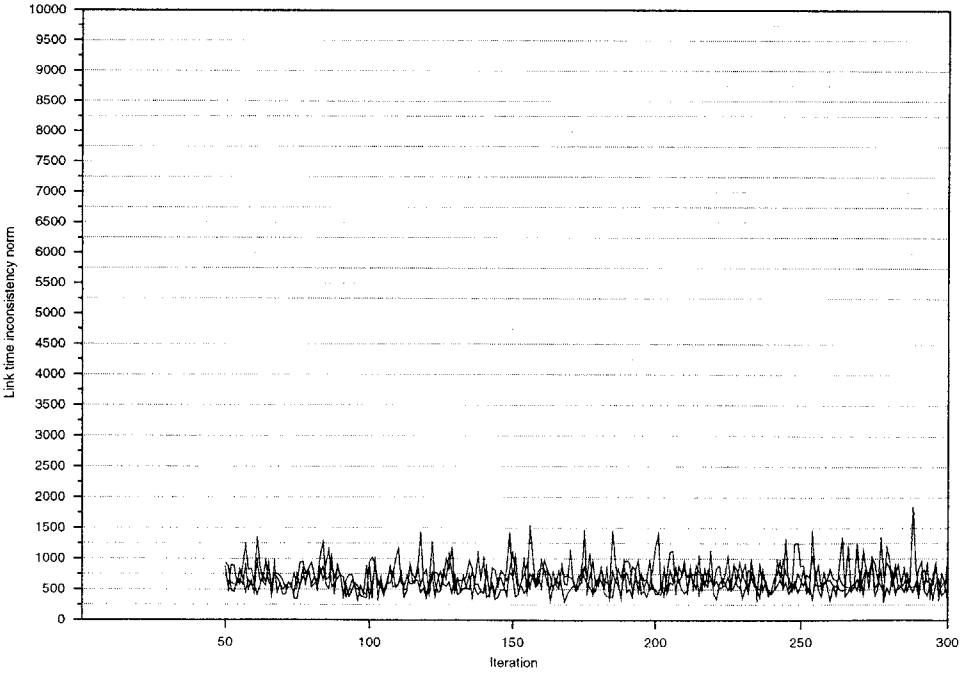


(b) IC_C convergence norm (seconds) in second pass

Figure 5.24: Run SDG-4-pol

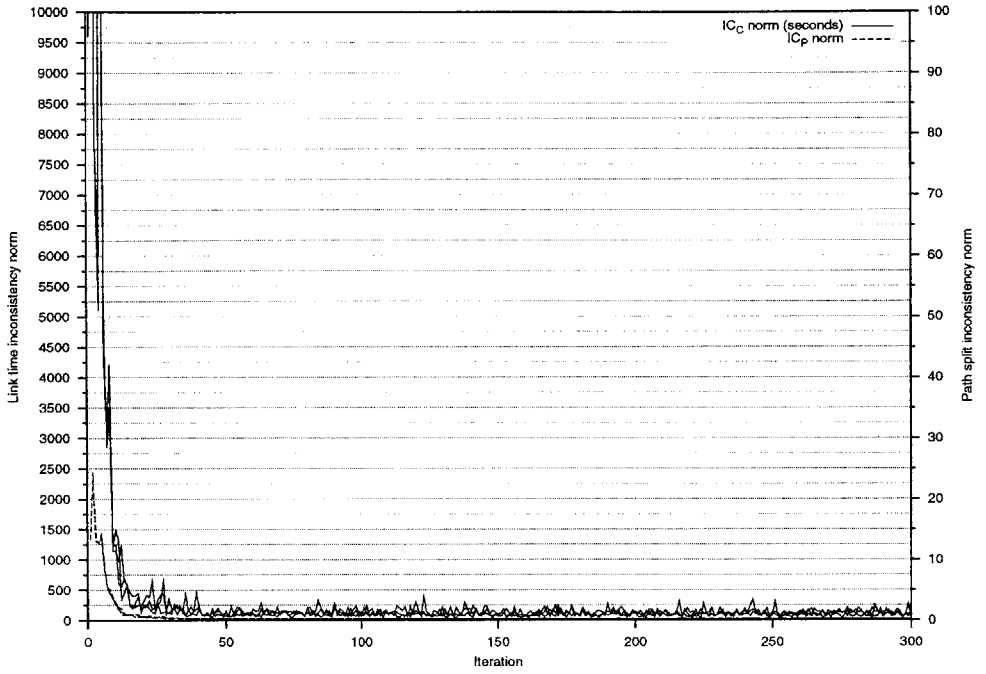


(a) IC_C convergence norm (seconds) in first pass

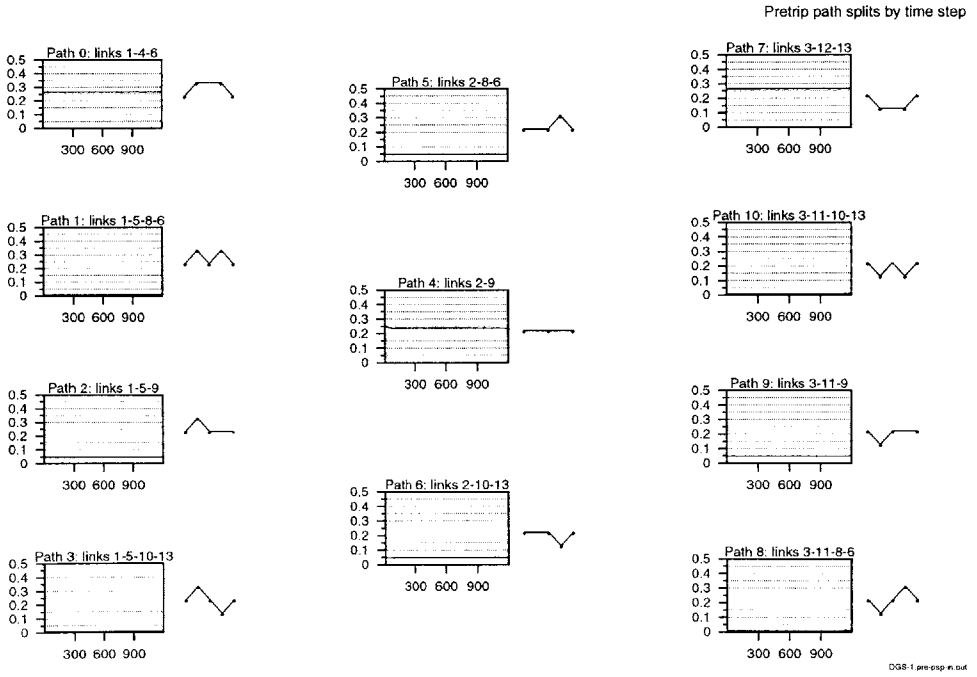


(b) IC_C convergence norm (seconds) in second pass

Figure 5.25: Run SDG-5-pol

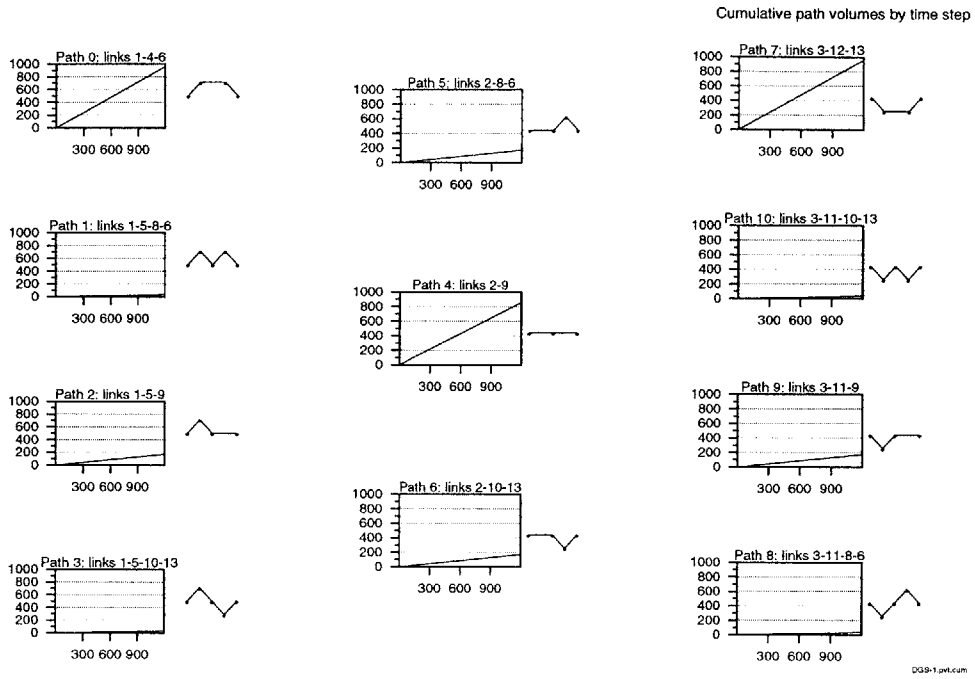


(a) IC_P convergence norm

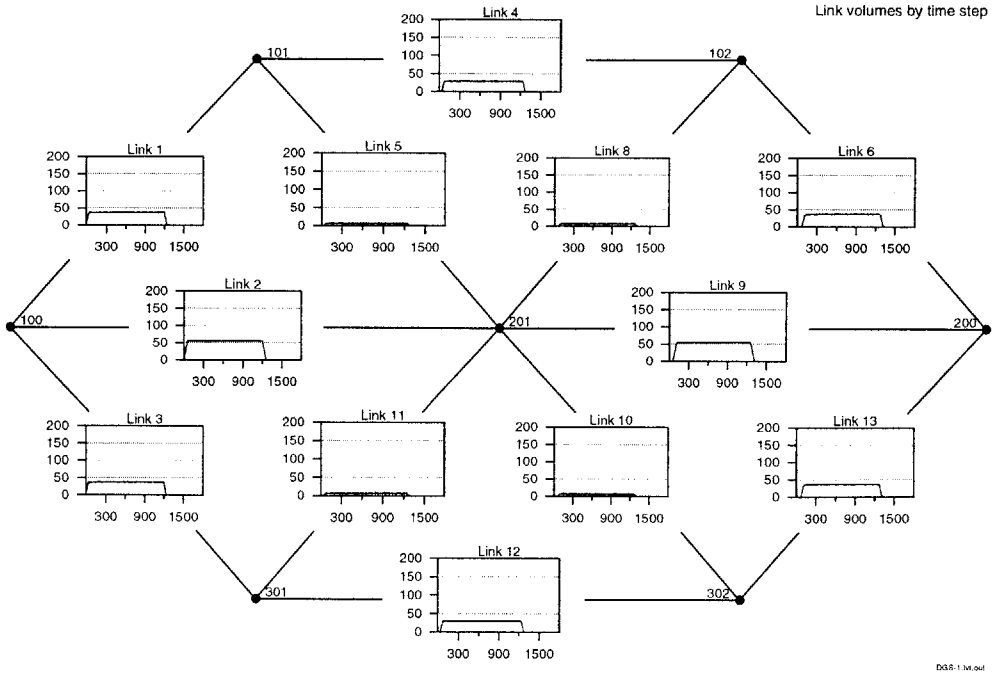


(b) Input pre-trip path split trajectories

Figure 5.26: Run DGS-1 (part 1/4)

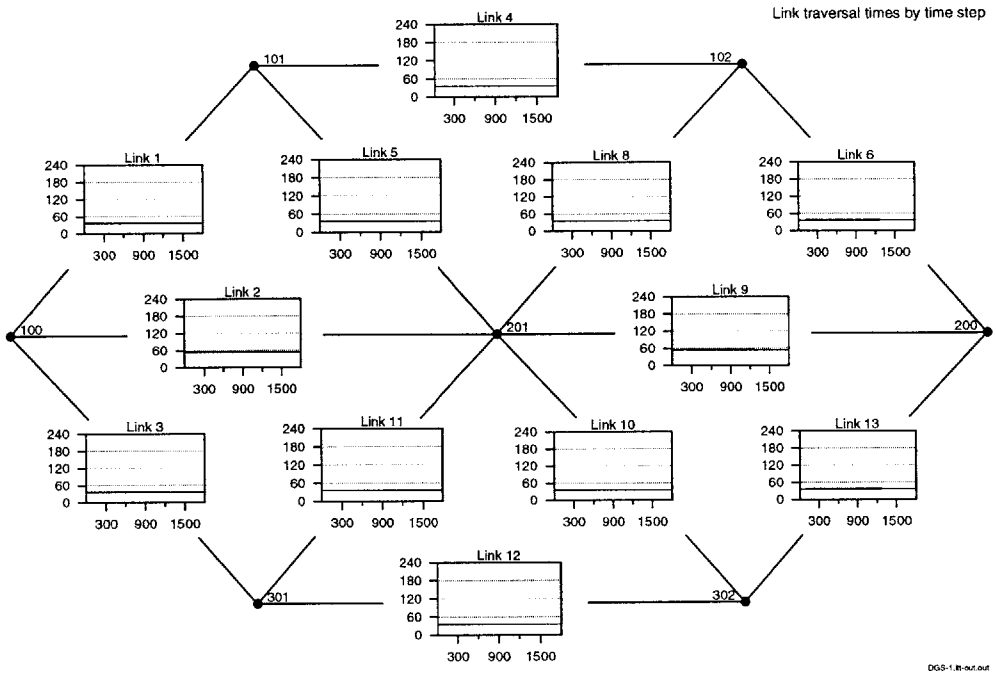


(c) Cumulative path flow trajectories (vehicles)

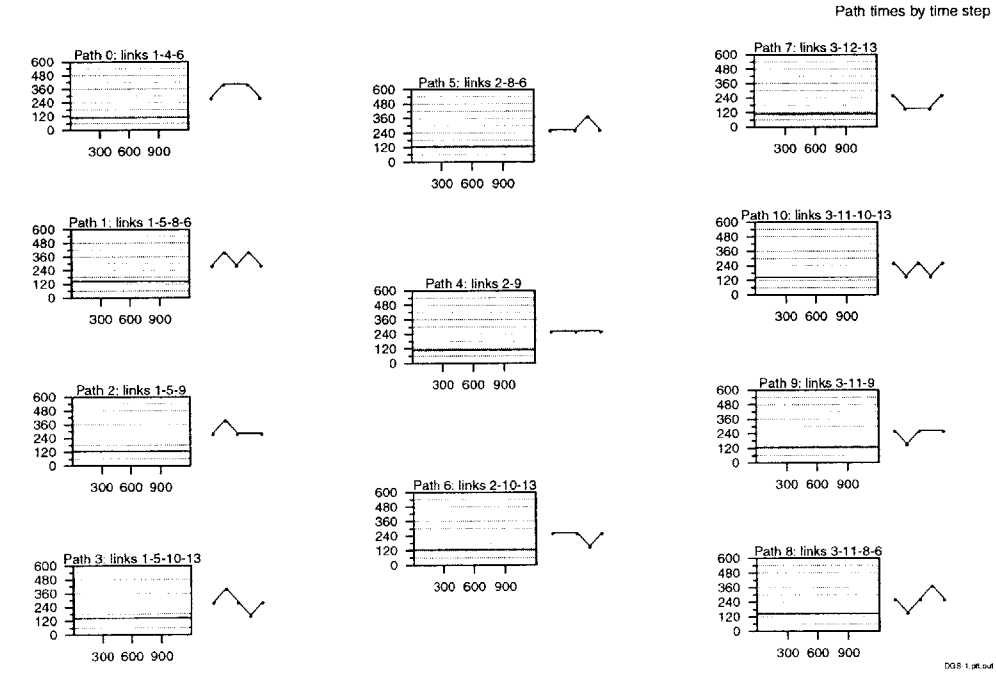


(d) Link volume trajectories (vehicles)

Figure 5.26: Run DGS-1 (part 2/4)

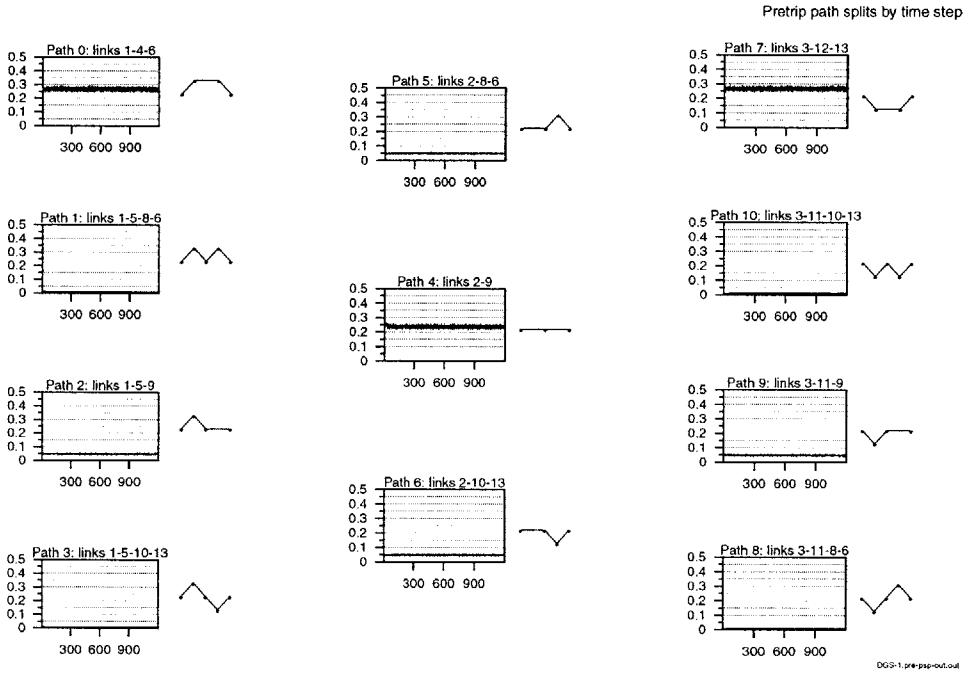


(e) Link time trajectories (seconds)



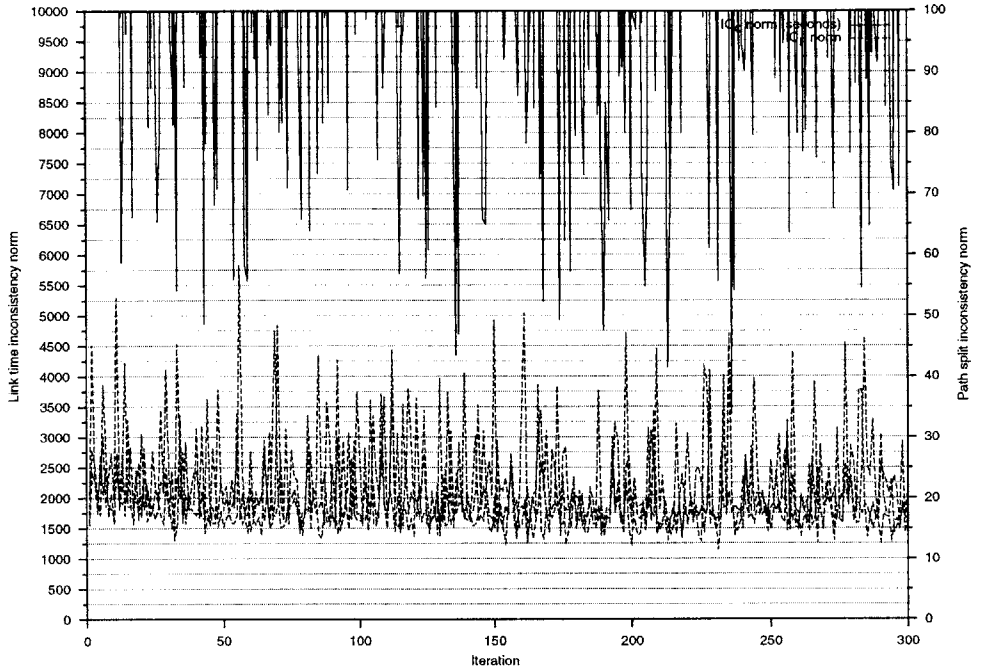
(f) Path time trajectories (seconds)

Figure 5.26: Run DGS-1 (part 3/4)

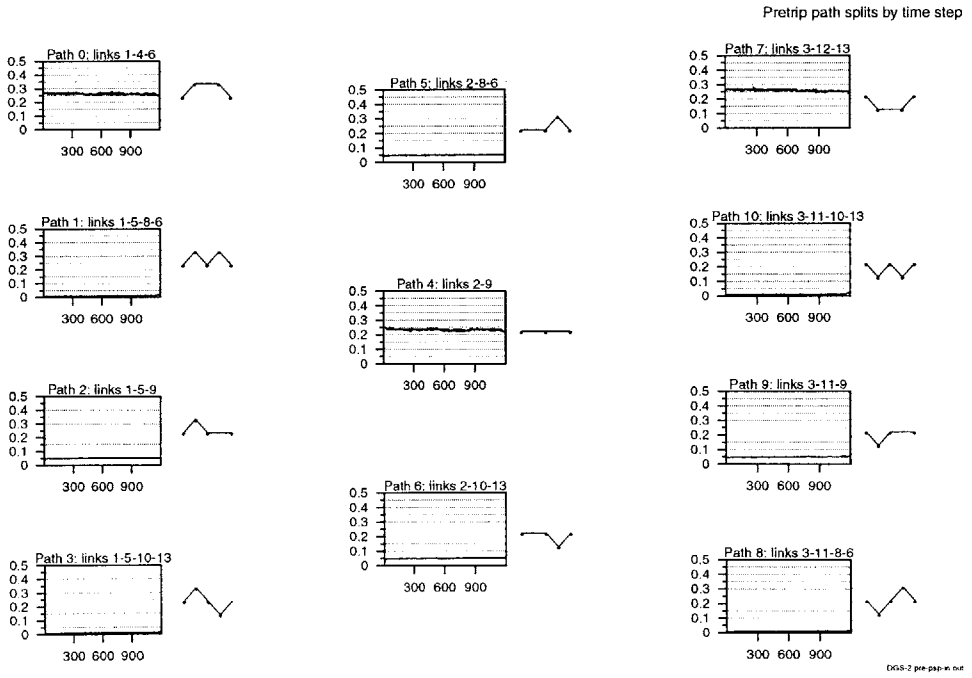


(g) Output pre-trip path split trajectories

Figure 5.26: Run DGS-1 (part 4/4)

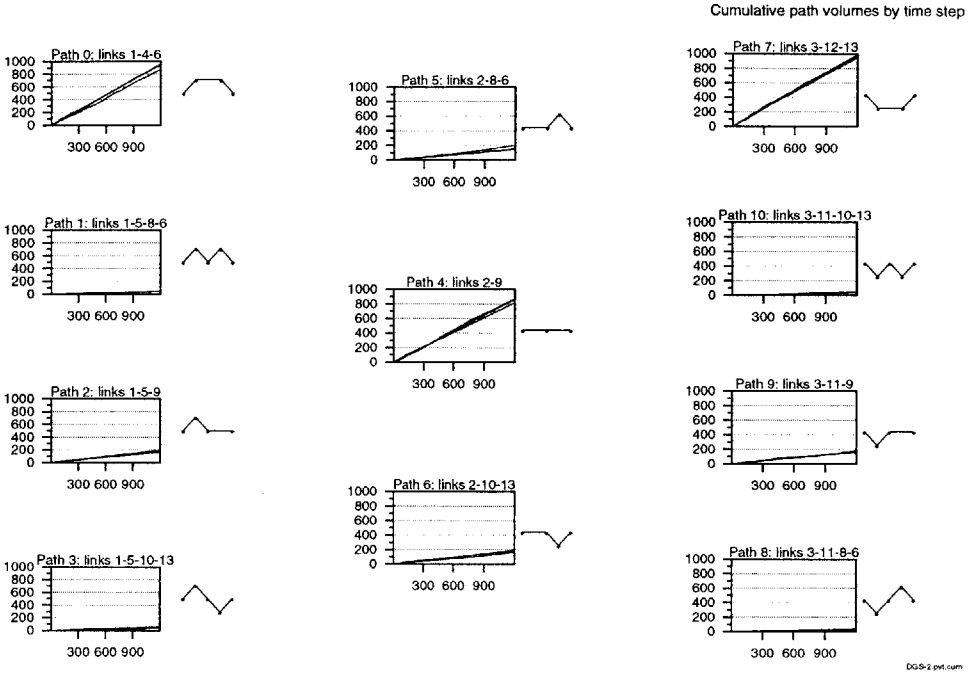


(a) IC_P convergence norm

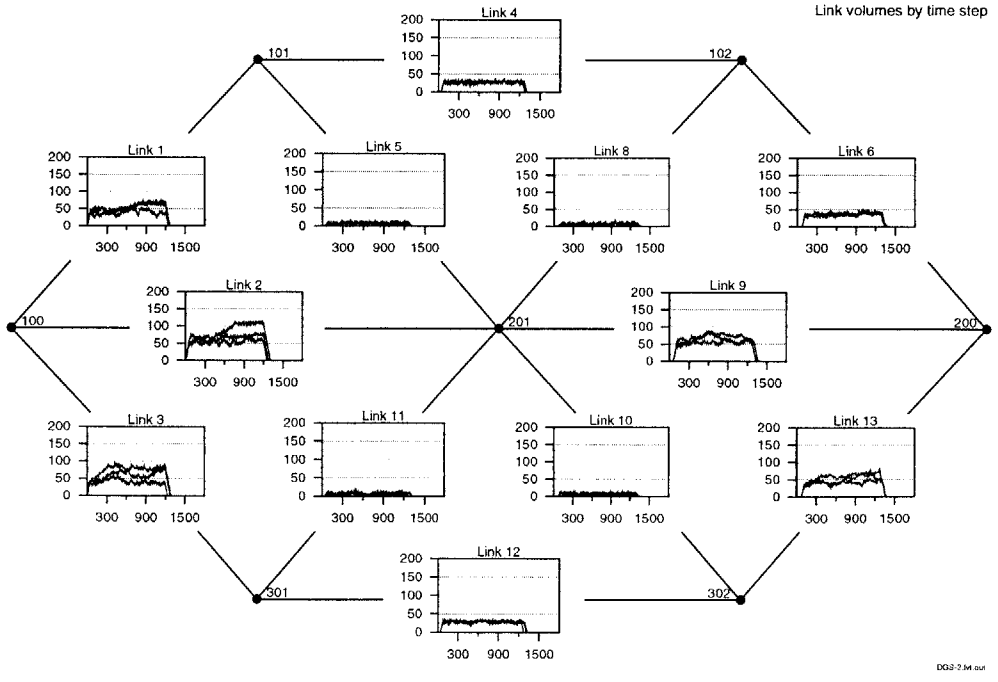


(b) Input pre-trip path split trajectories

Figure 5.27: Run DGS-2 (part 1/4)

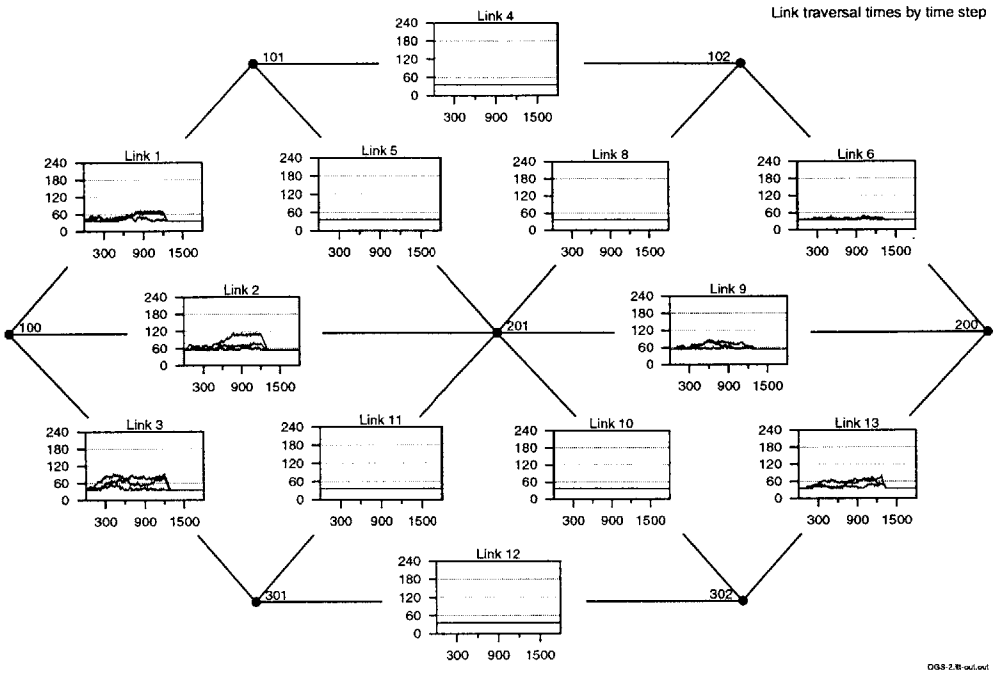


(c) Cumulative path flow trajectories (vehicles)

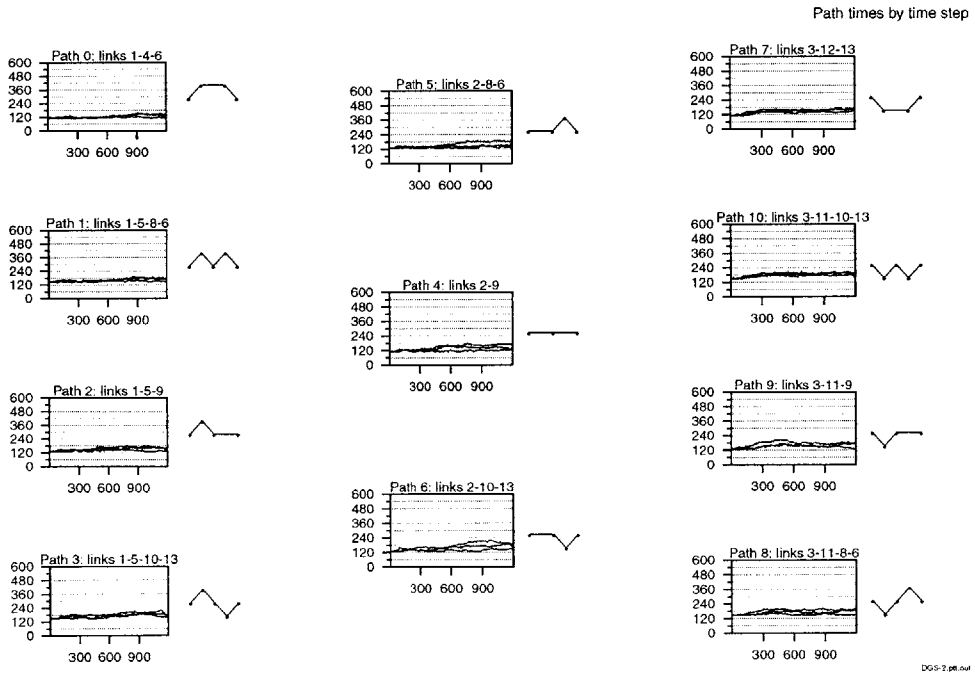


(d) Link volume trajectories (vehicles)

Figure 5.27: Run DGS-2 (part 2/4)

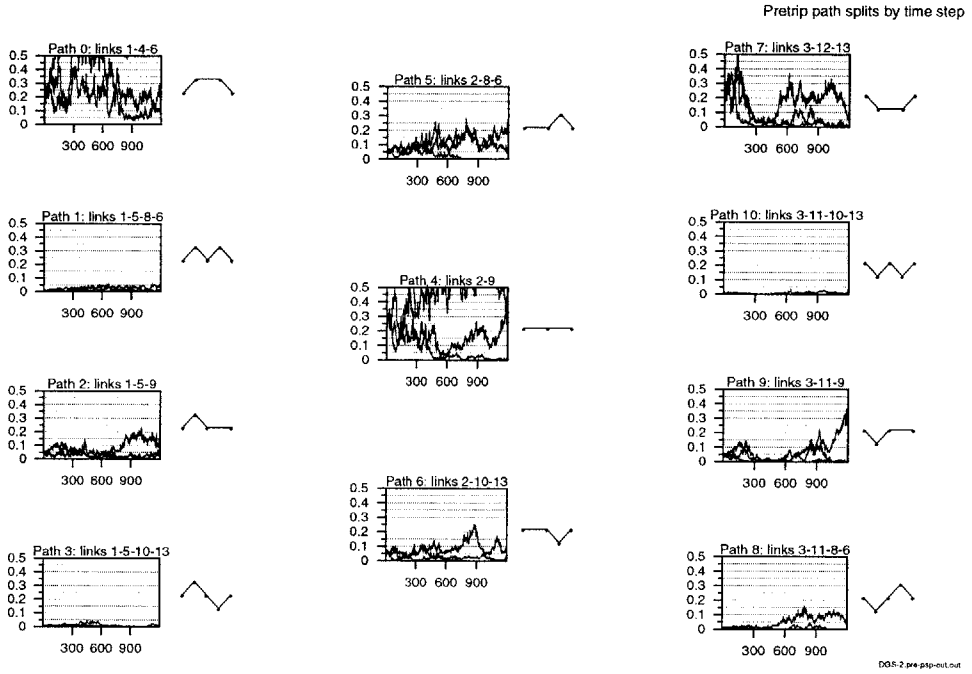


(e) Link time trajectories (seconds)



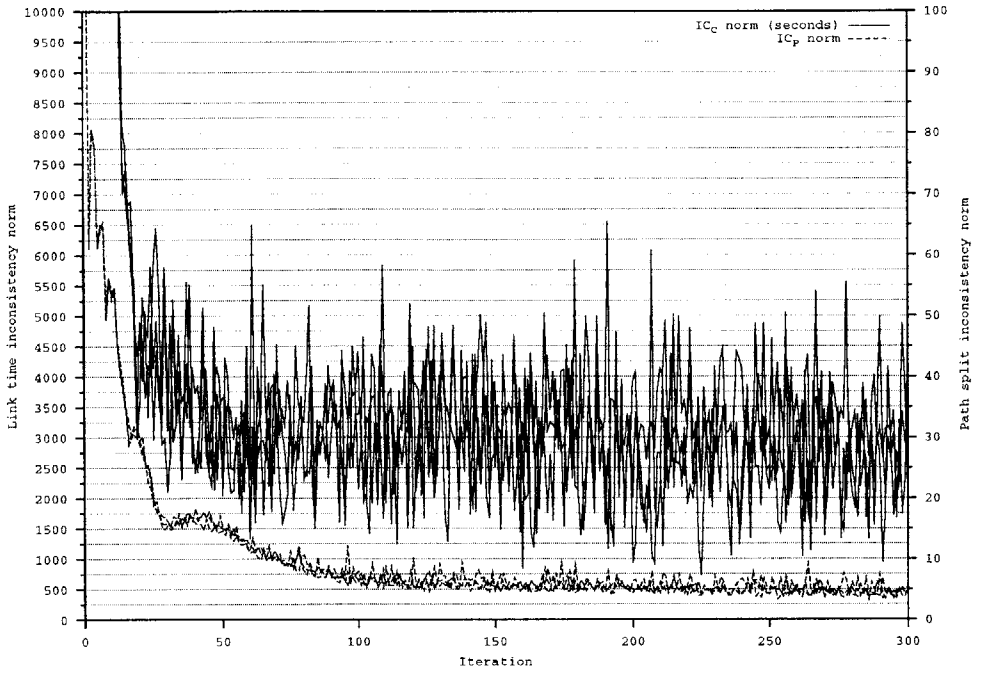
(f) Path time trajectories (seconds)

Figure 5.27: Run DGS-2 (part 3/4)

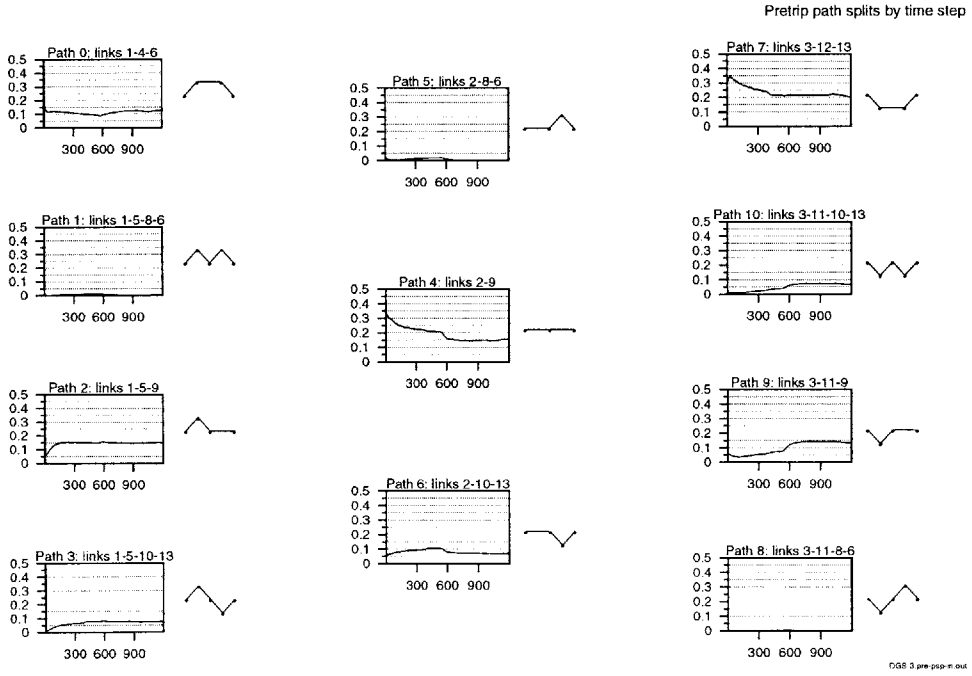


(g) Output pre-trip path split trajectories

Figure 5.27: Run DGS-2 (part 4/4)

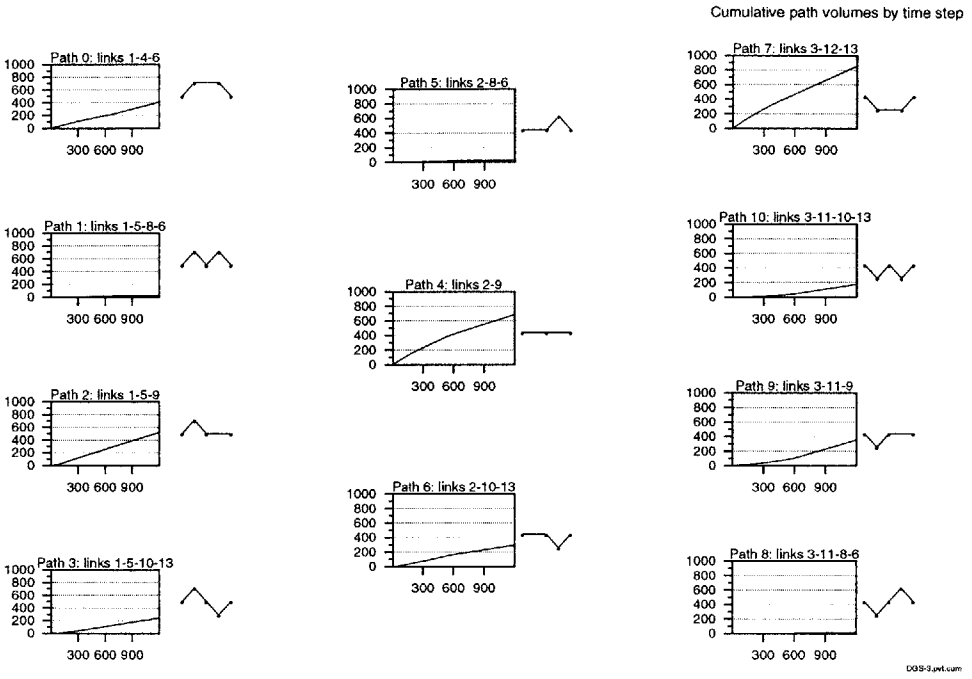


(a) IC_P convergence norm

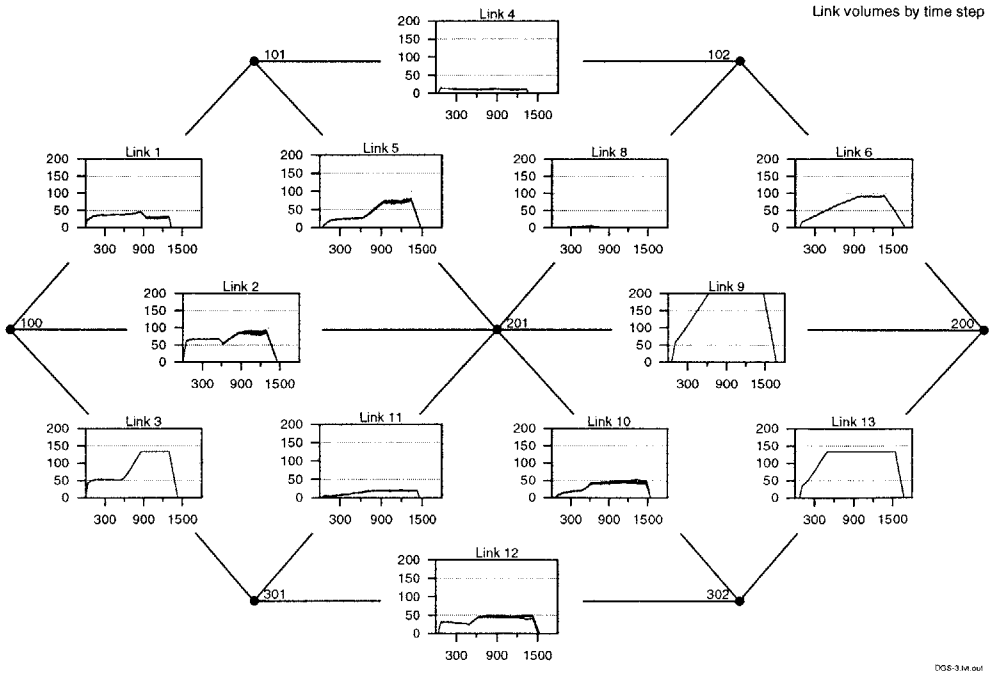


(b) Input pre-trip path split trajectories

Figure 5.28: Run DGS-3 (part 1/4)

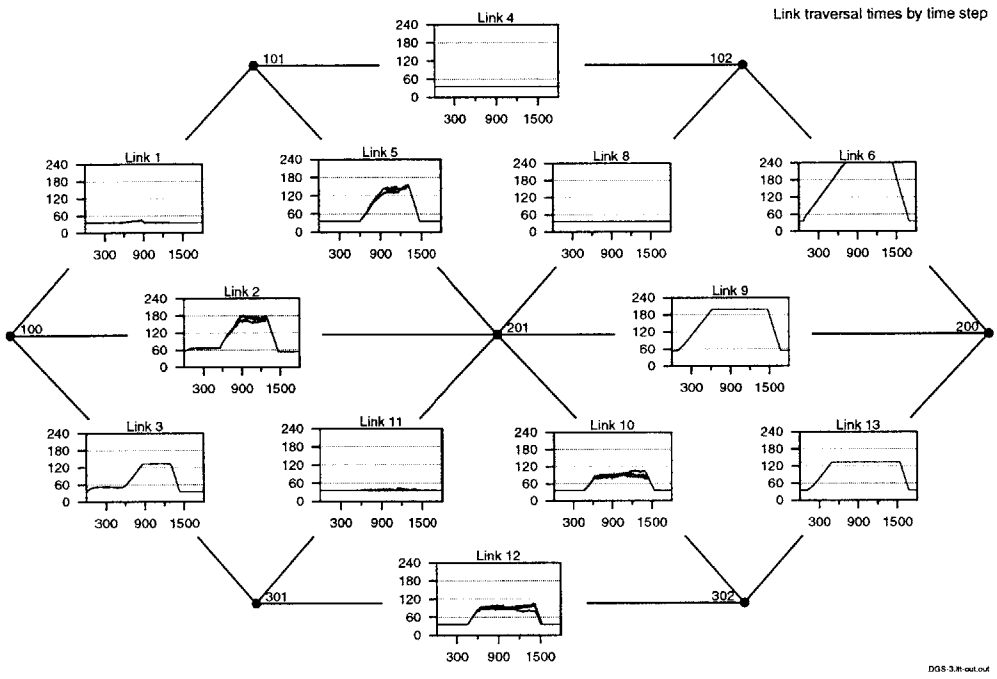


(c) Cumulative path flow trajectories (vehicles)

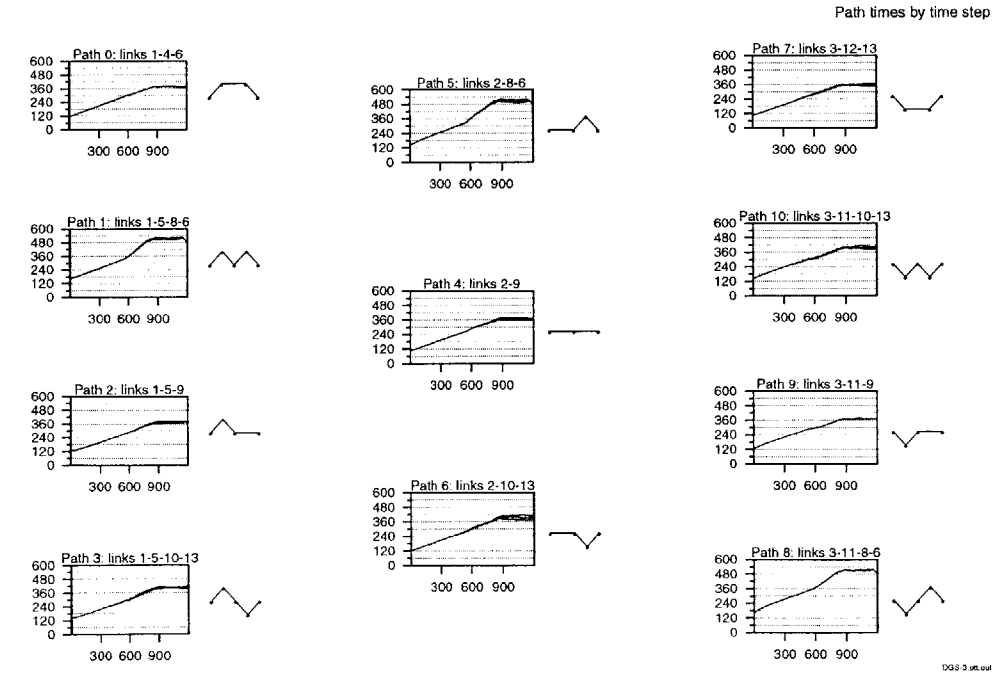


(d) Link volume trajectories (vehicles)

Figure 5.28: Run DGS-3 (part 2/4)

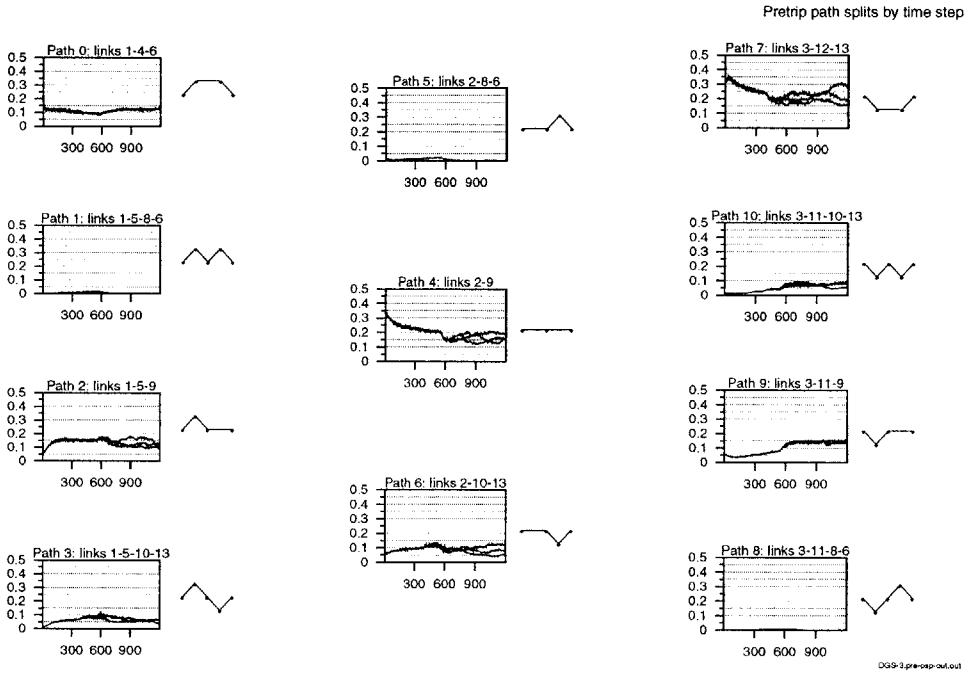


(e) Link time trajectories (seconds)



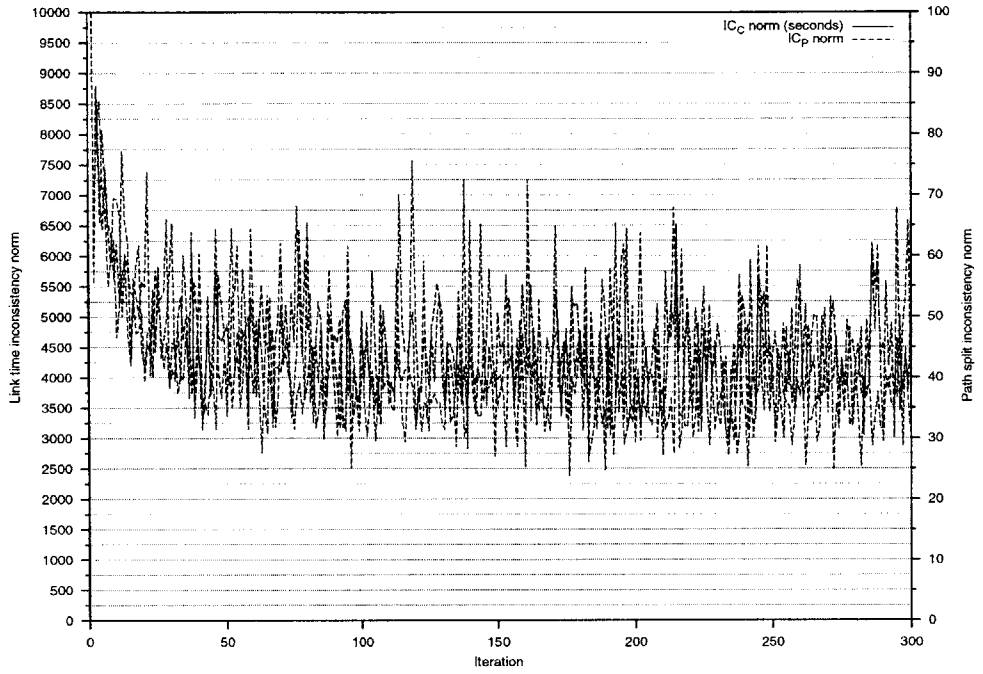
(f) Path time trajectories (seconds)

Figure 5.28: Run DGS-3 (part 3/4)

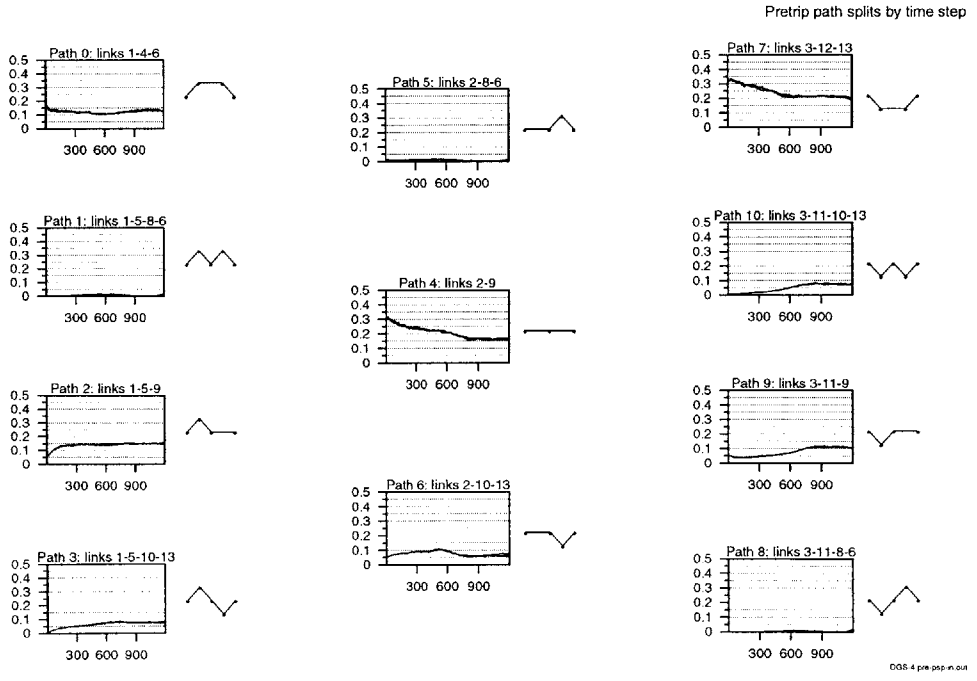


(g) Output pre-trip path split trajectories

Figure 5.28: Run DGS-3 (part 4/4)

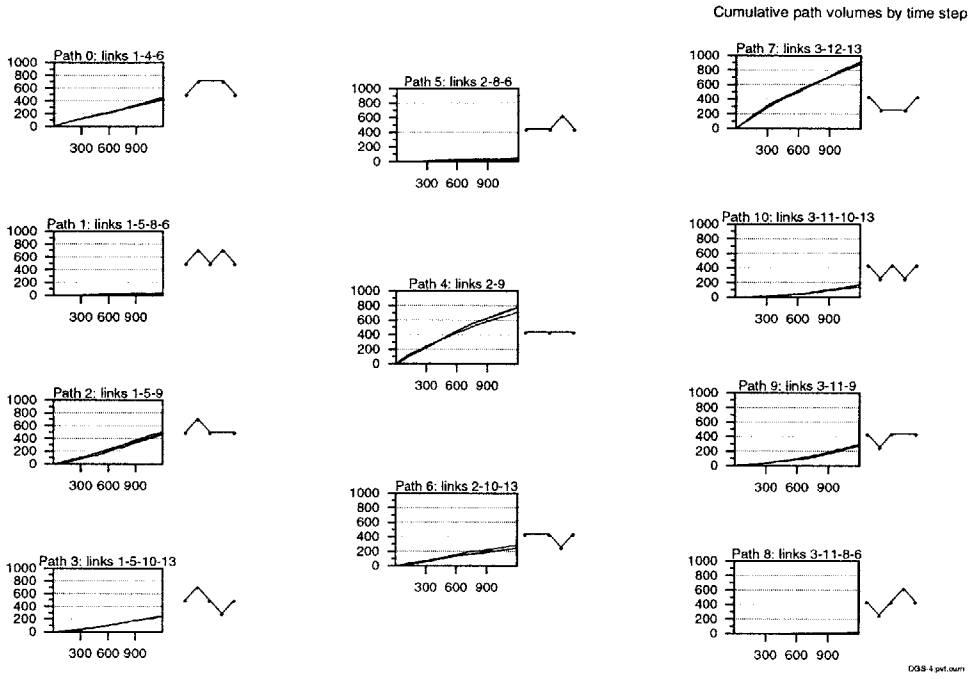


(a) IC_P convergence norm

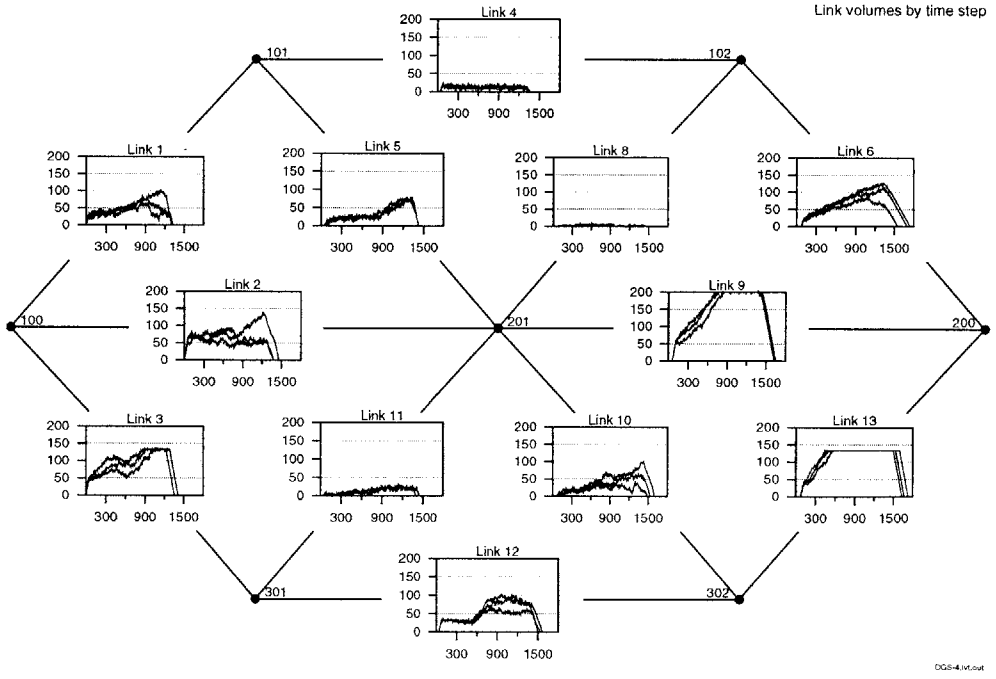


(b) Input pre-trip path split trajectories

Figure 5.29: Run DGS-4 (part 1/4)

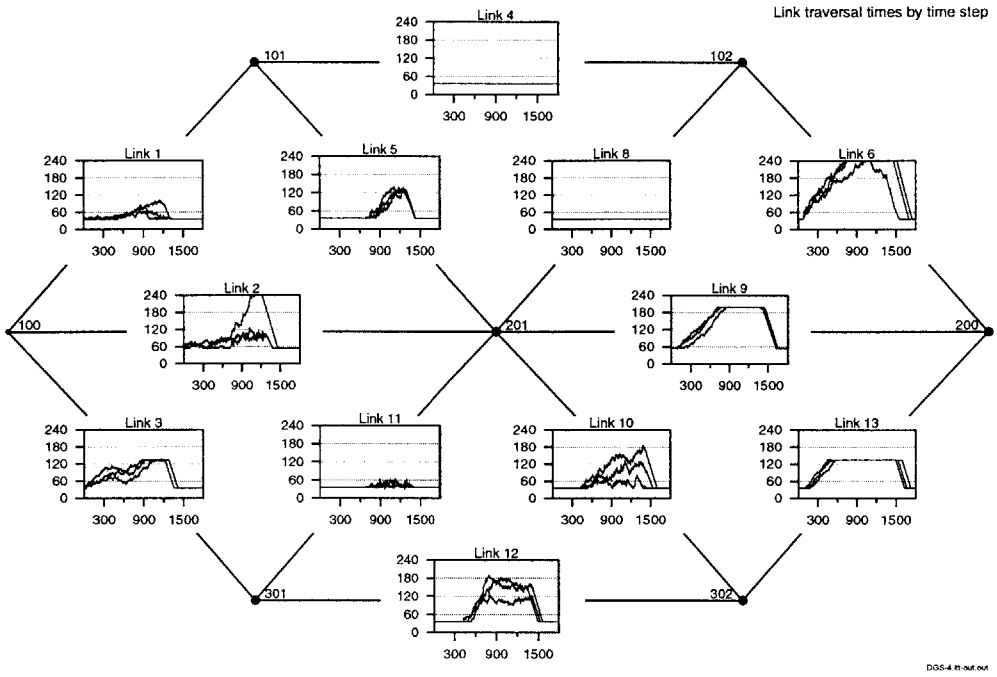


(c) Cumulative path flow trajectories (vehicles)

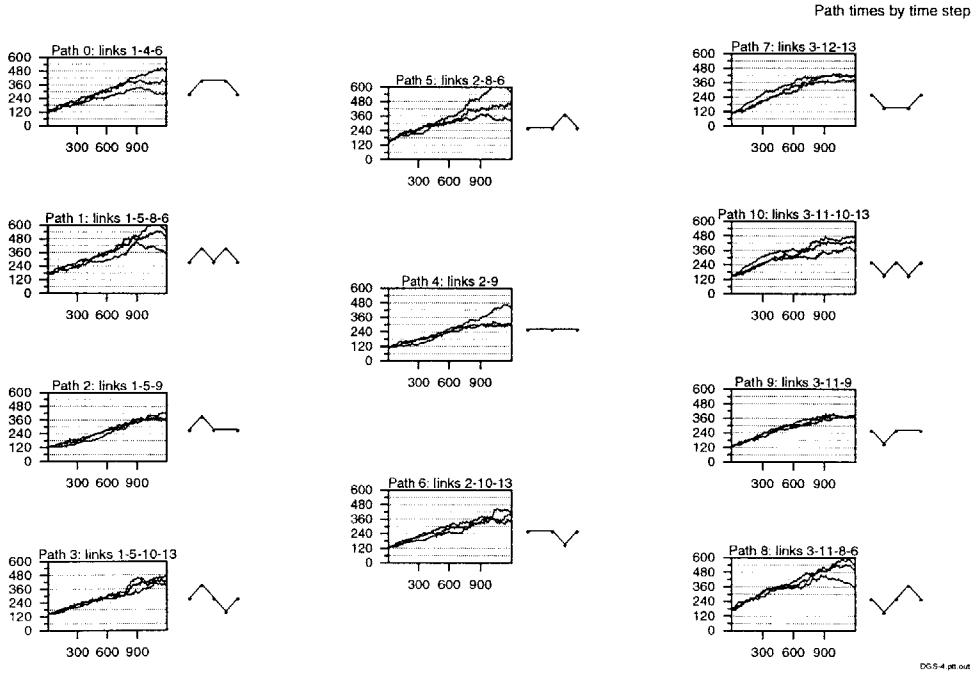


(d) Link volume trajectories (vehicles)

Figure 5.29: Run DGS-4 (part 2/4)

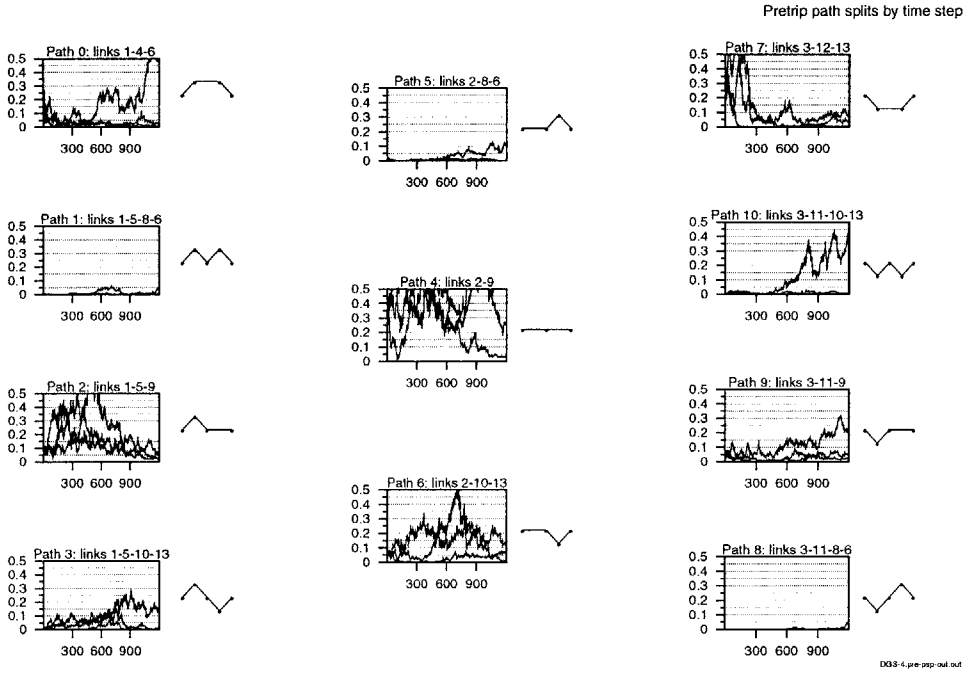


(e) Link time trajectories (seconds)



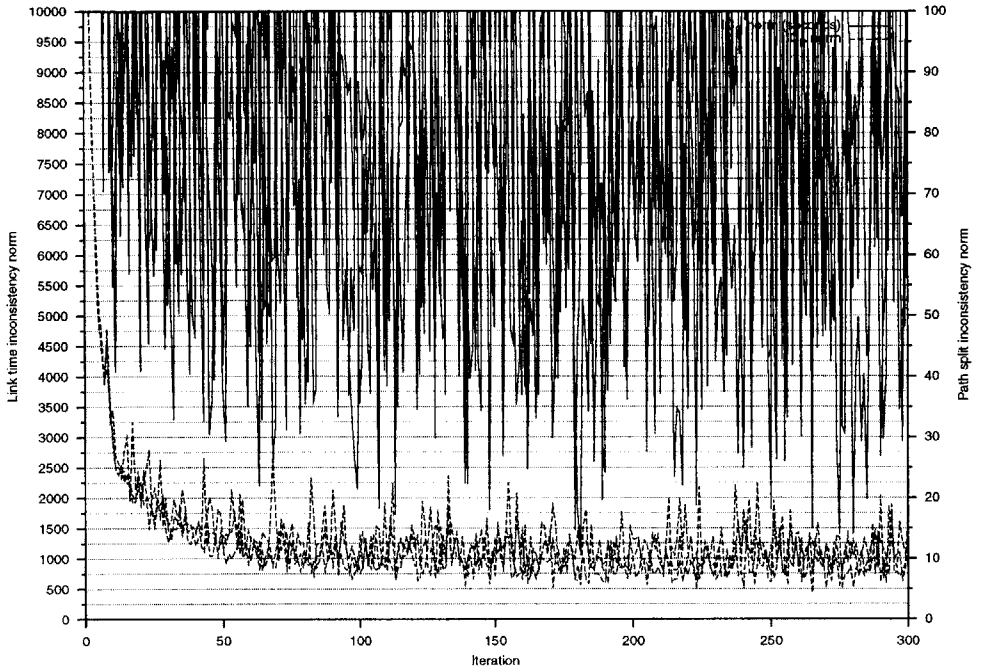
(f) Path time trajectories (seconds)

Figure 5.29: Run DGS-4 (part 3/4)

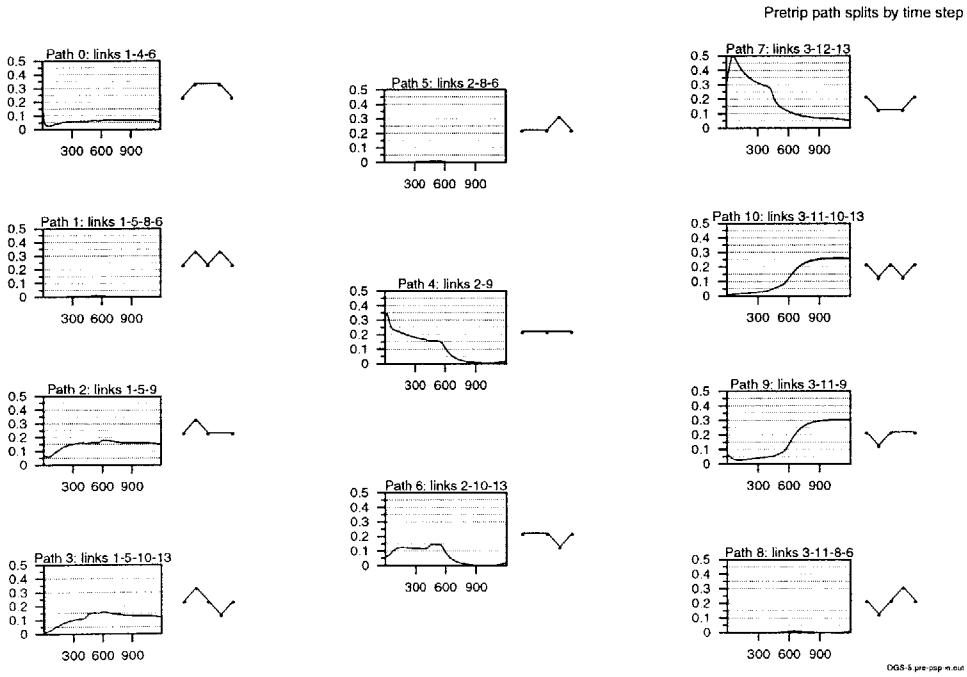


(g) Output pre-trip path split trajectories

Figure 5.29: Run DGS-4 (part 4/4)

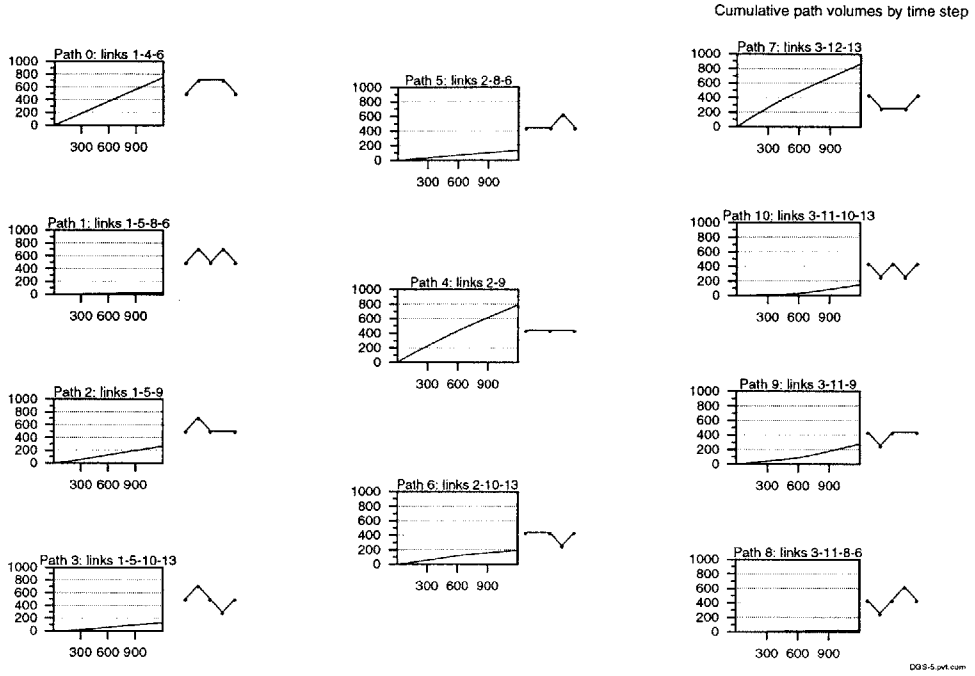


(a) IC_P convergence norm

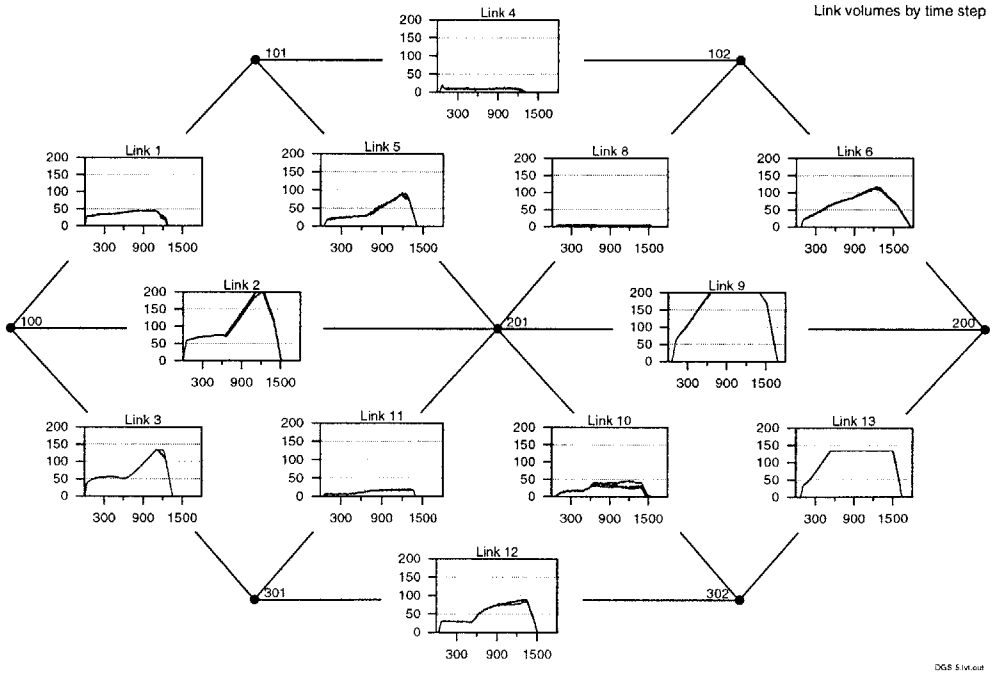


(b) Input pre-trip path split trajectories

Figure 5.30: Run DGS-5 (part 1/4)

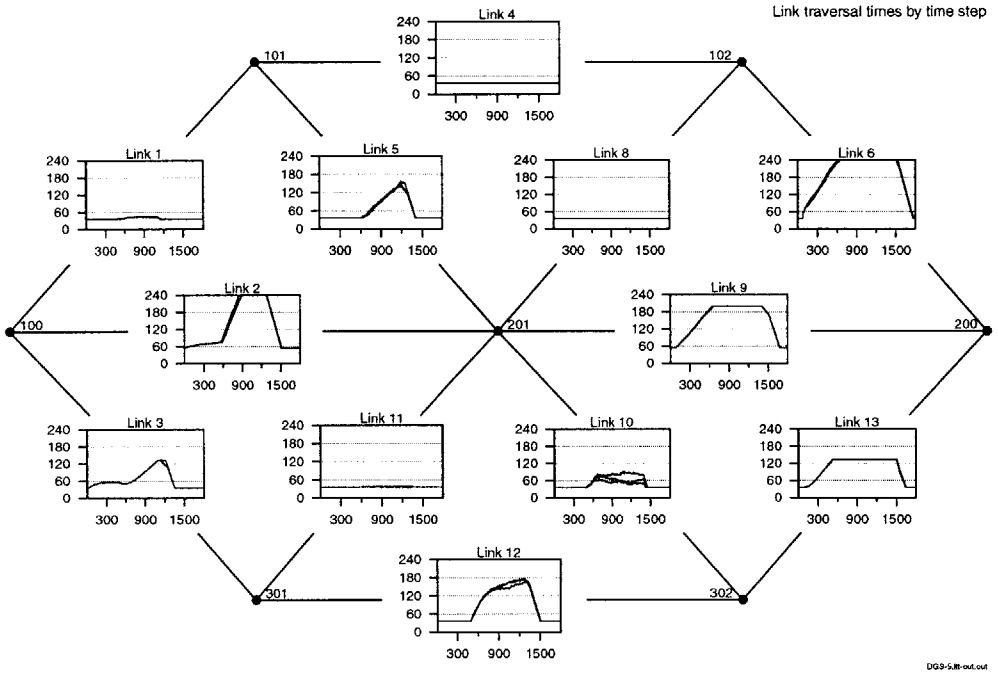


(c) Cumulative path flow trajectories (vehicles)

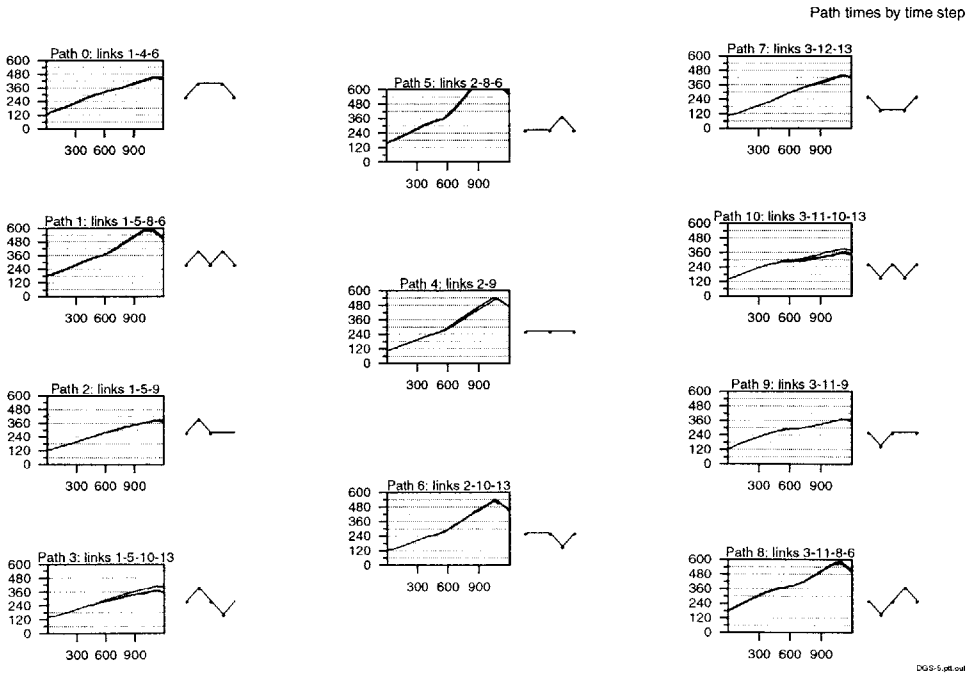


(d) Link volume trajectories (vehicles)

Figure 5.30: Run DGS-5 (part 2/4)

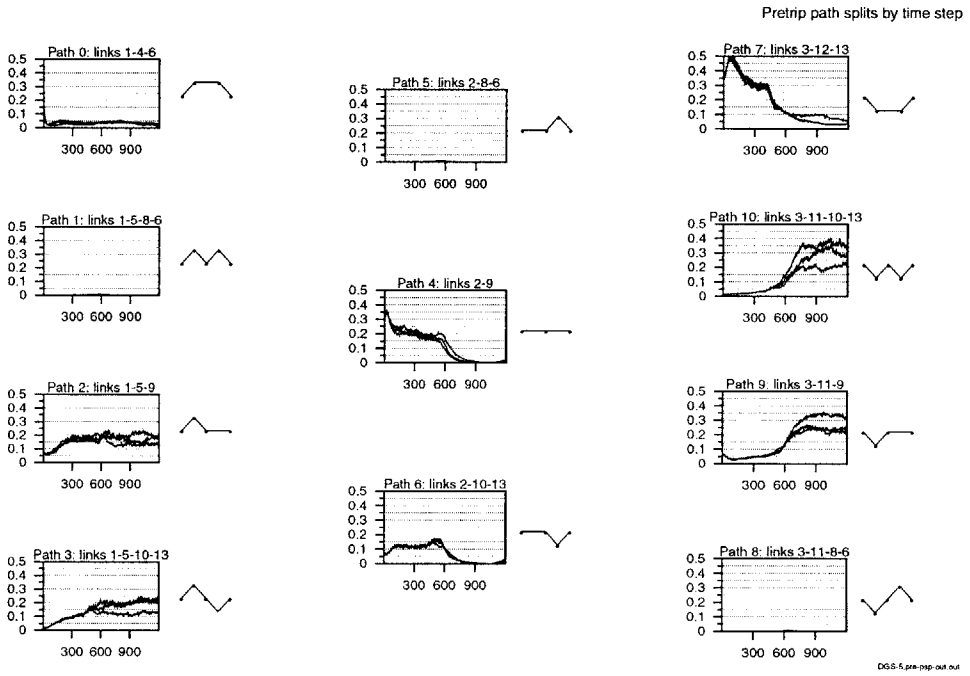


(e) Link time trajectories (seconds)



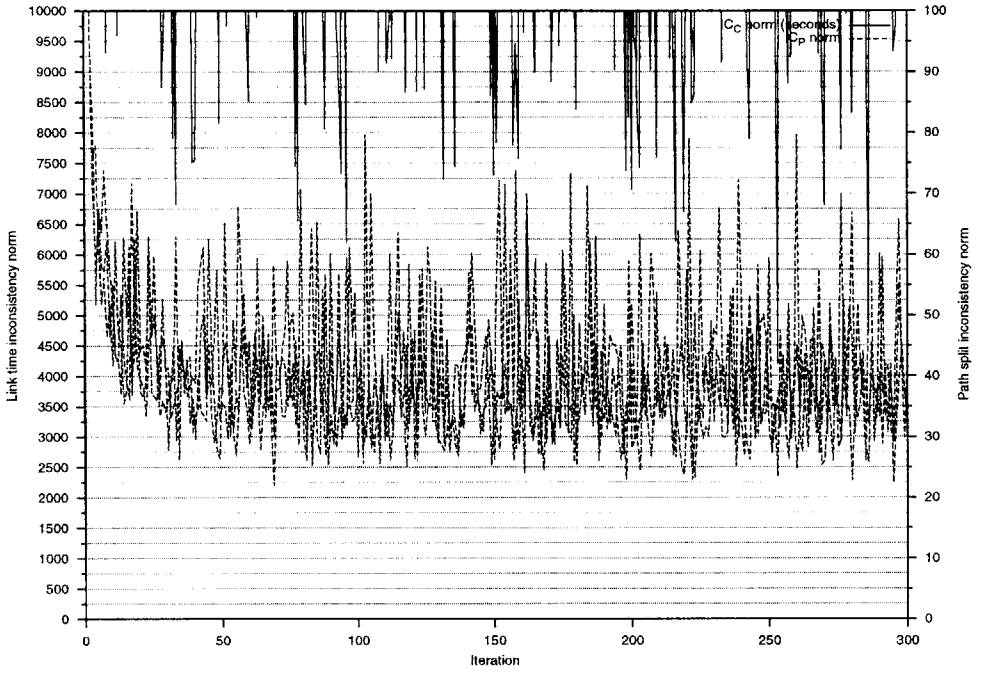
(f) Path time trajectories (seconds)

Figure 5.30: Run DGS-5 (part 3/4)

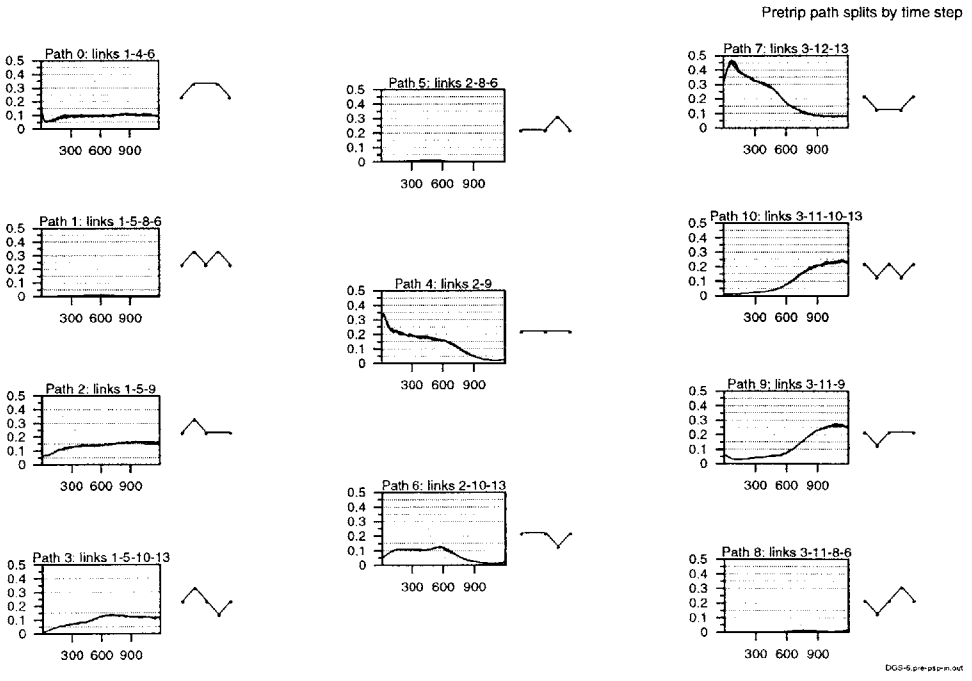


(g) Output pre-trip path split trajectories

Figure 5.30: Run DGS-5 (part 4/4)

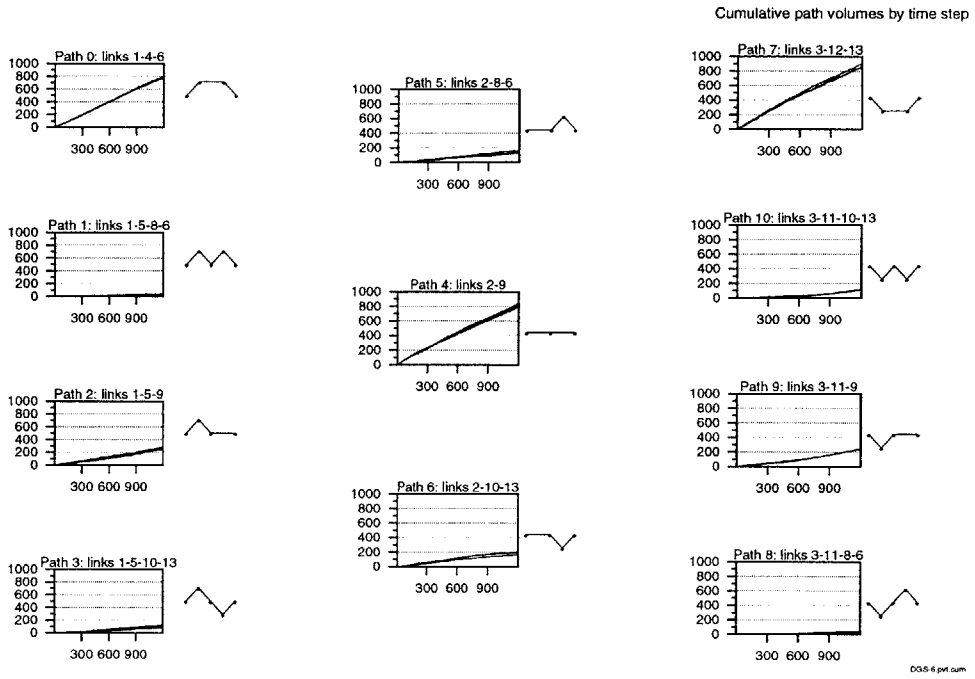


(a) IC_P convergence norm

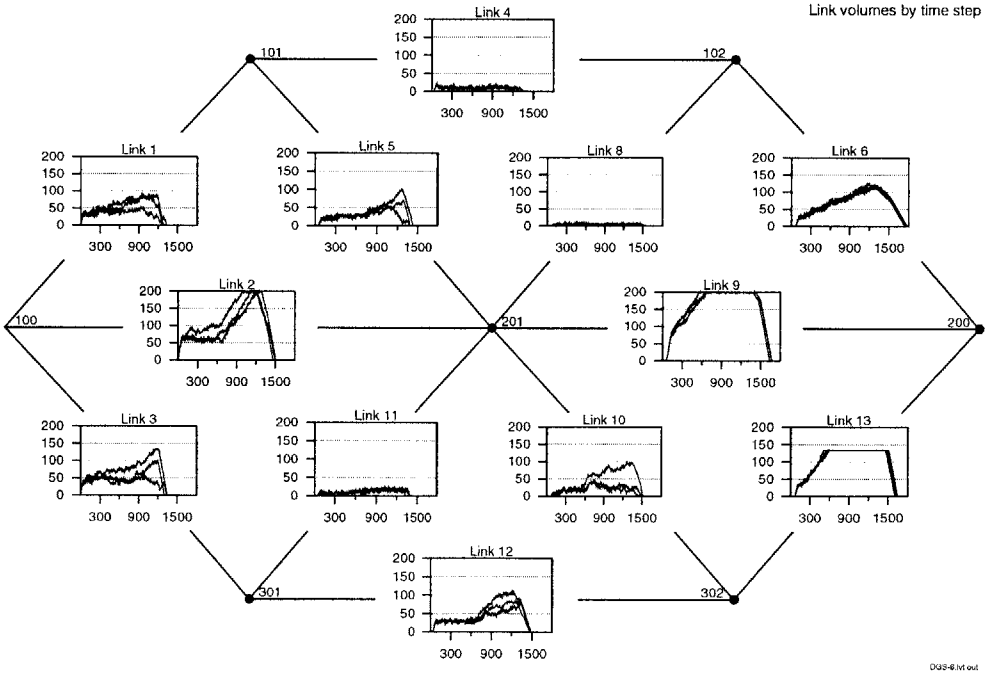


(b) Input pre-trip path split trajectories

Figure 5.31: Run DGS-6 (part 1/4)

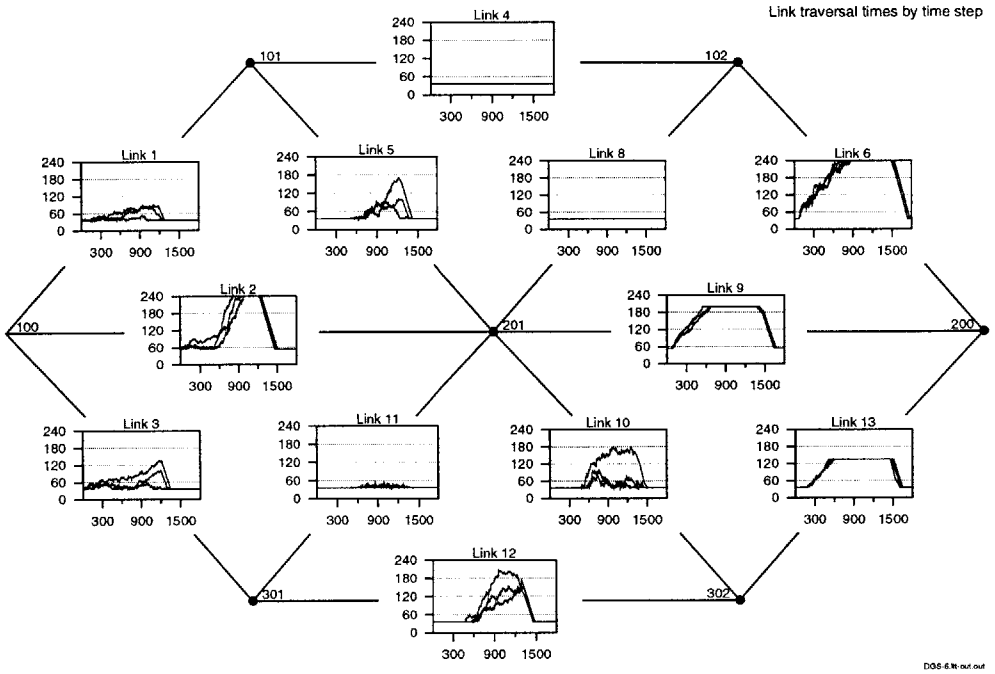


(c) Cumulative path flow trajectories (vehicles)

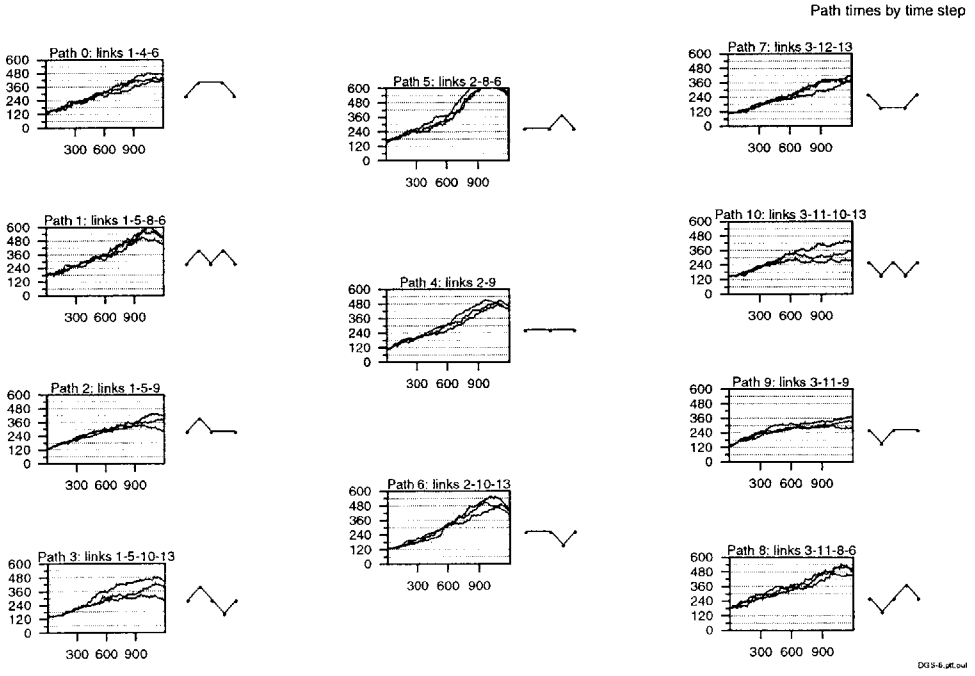


(d) Link volume trajectories (vehicles)

Figure 5.31: Run DGS-6 (part 2/4)

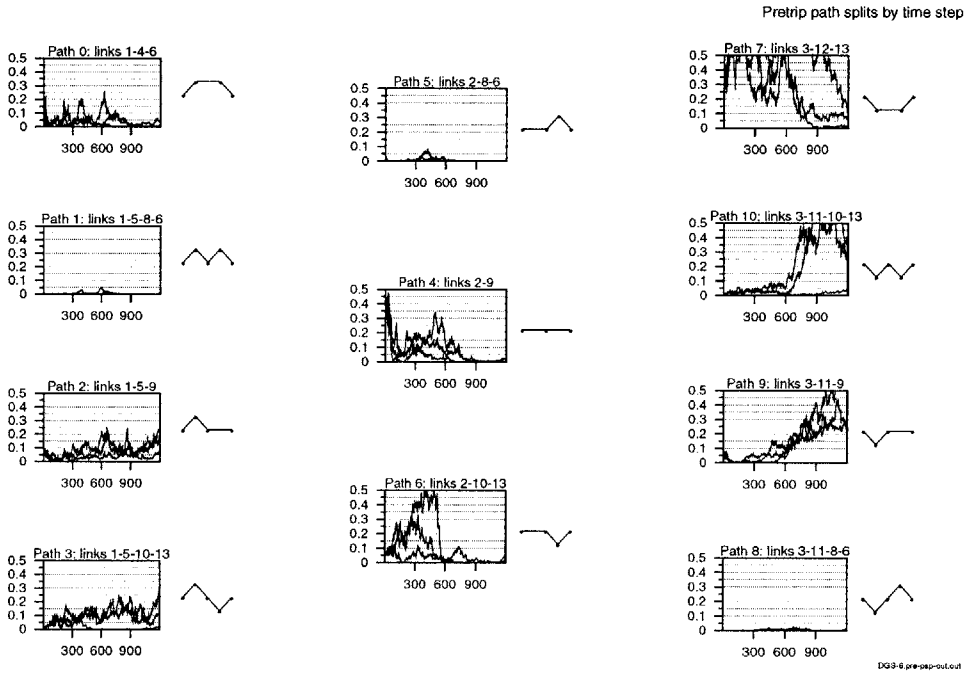


(e) Link time trajectories (seconds)



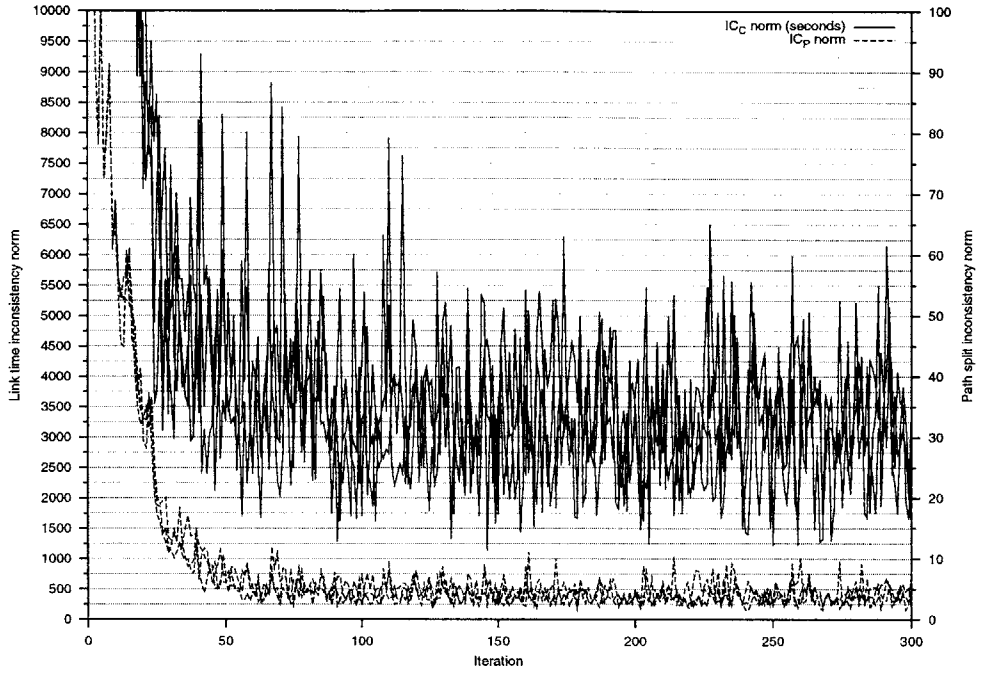
(f) Path time trajectories (seconds)

Figure 5.31: Run DGS-6 (part 3/4)

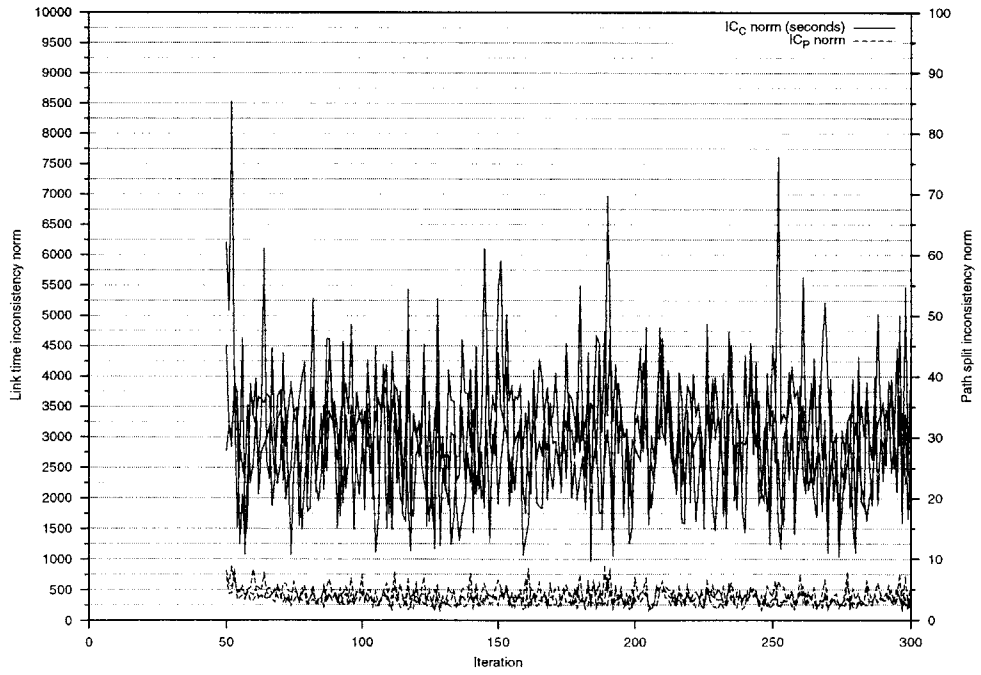


(g) Output pre-trip path split trajectories

Figure 5.31: Run DGS-6 (part 4/4)

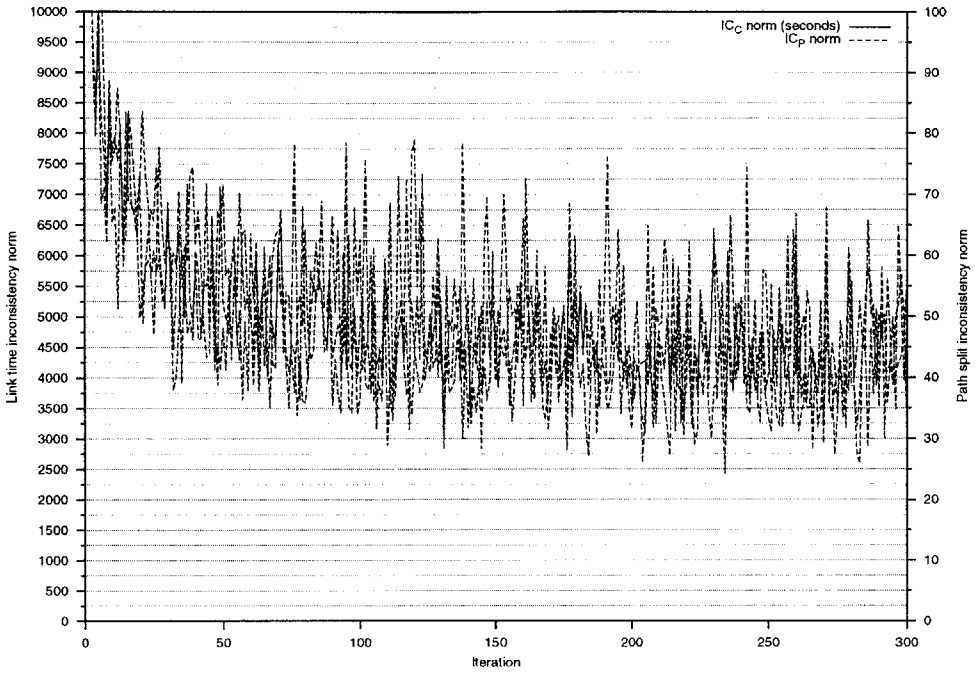


(a) IC_P convergence norm in first pass

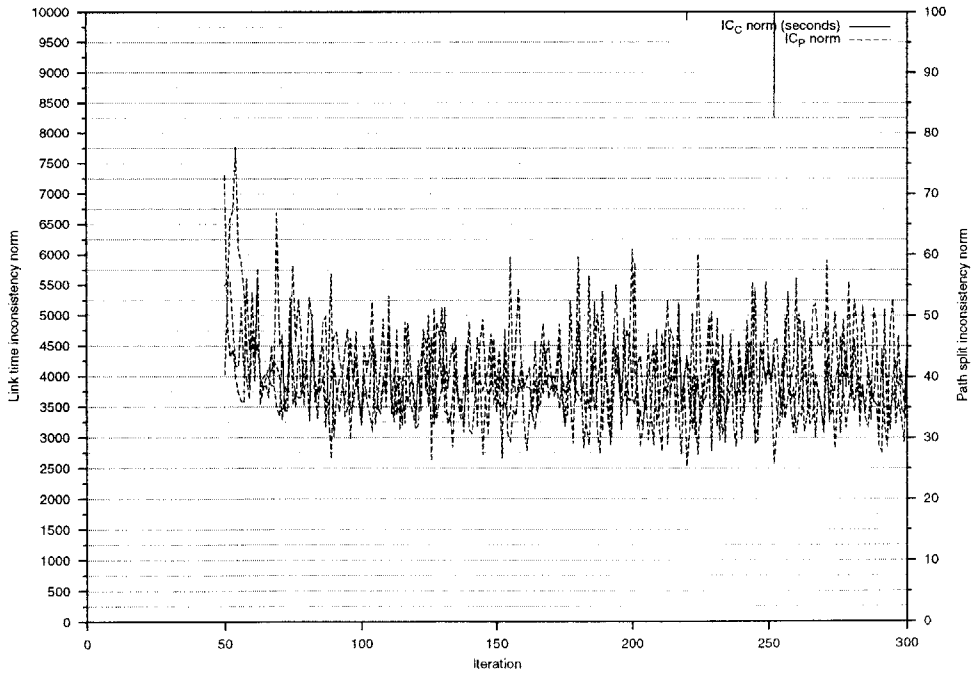


(b) IC_P convergence norm in second pass

Figure 5.32: Run DGS-3-pol

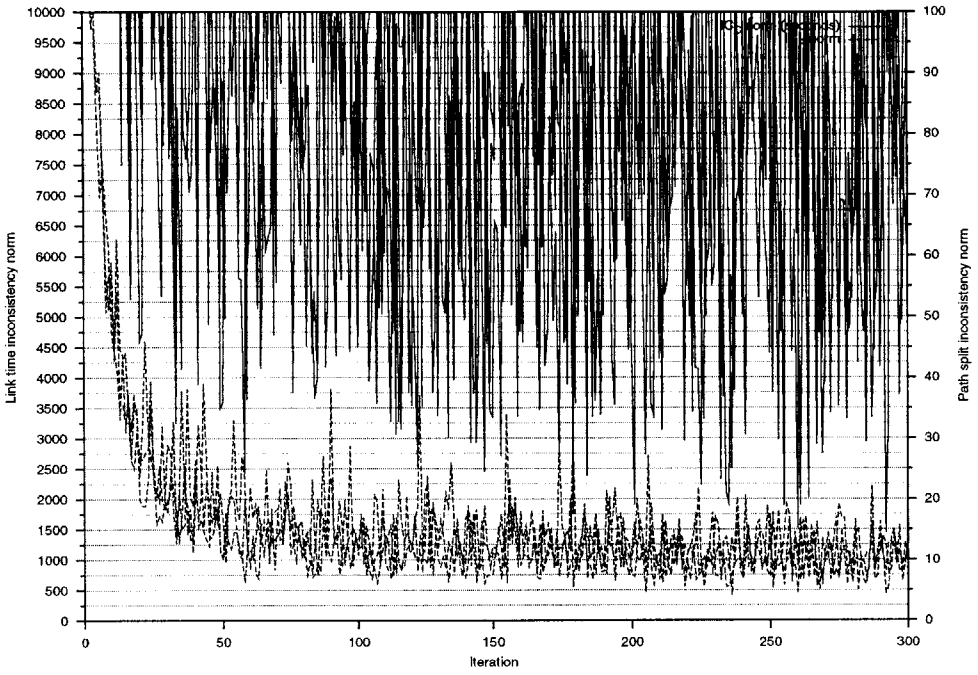


(a) IC_P convergence norm in first pass

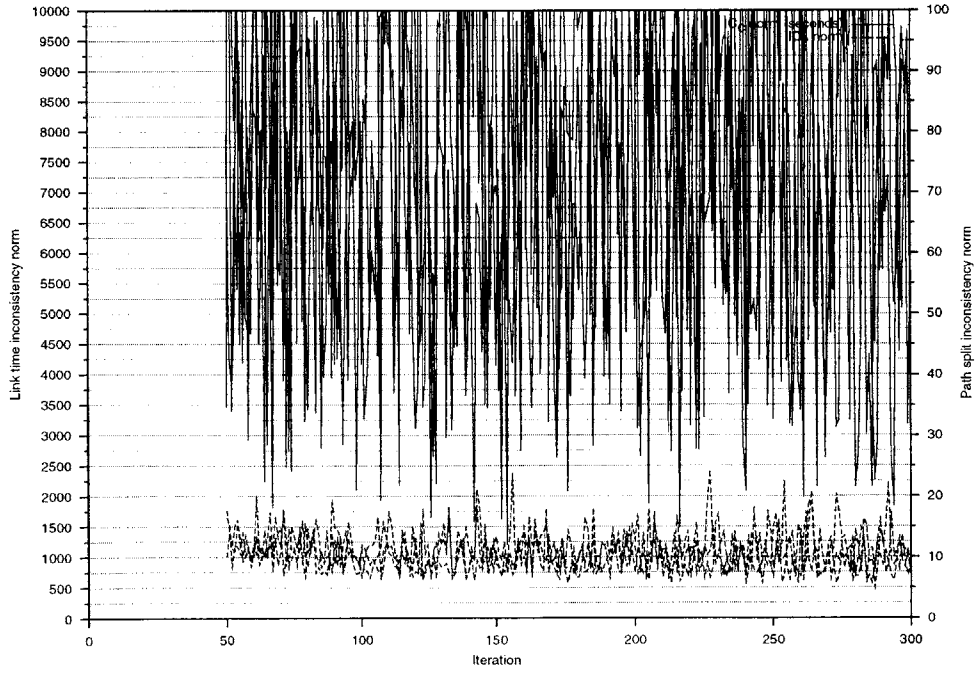


(b) IC_P convergence norm in second pass

Figure 5.33: Run DGS-4-pol

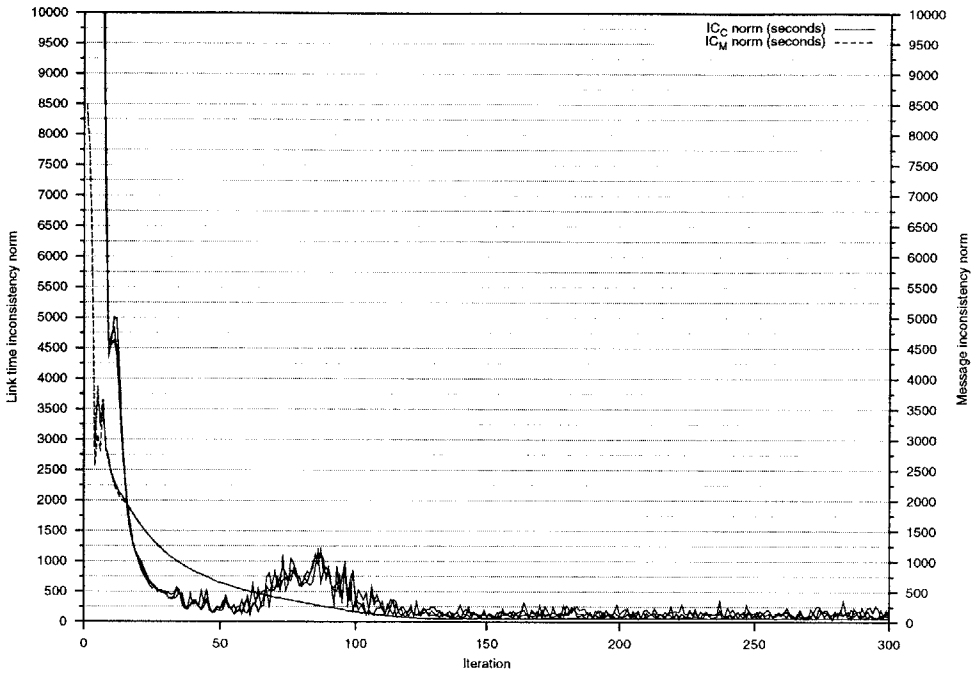


(a) IC_P convergence norm in first pass

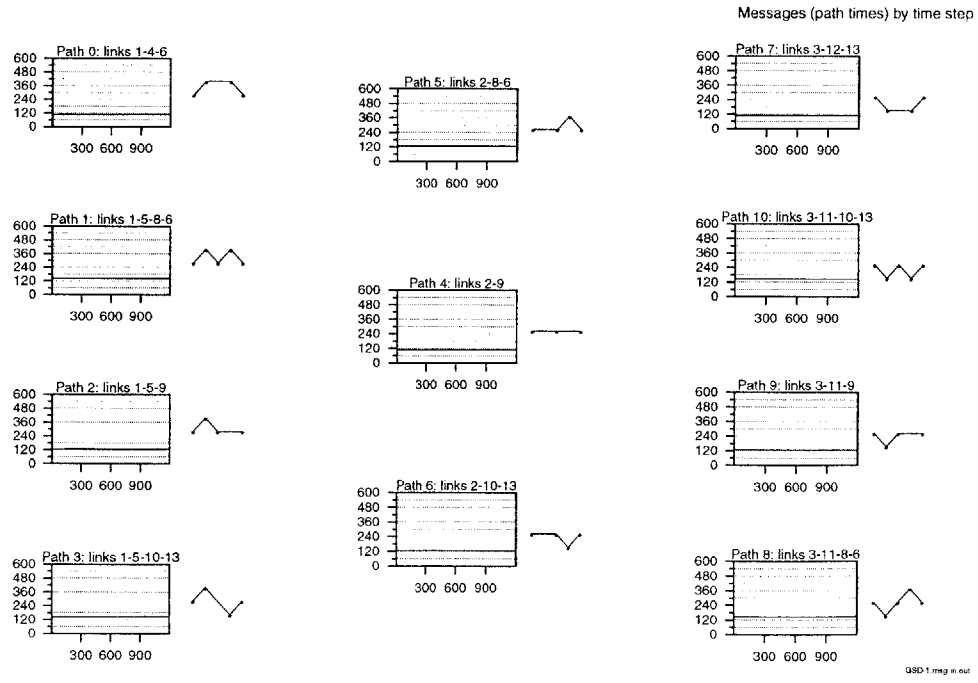


(b) IC_P convergence norm in second pass

Figure 5.34: Run DGS-5-pol

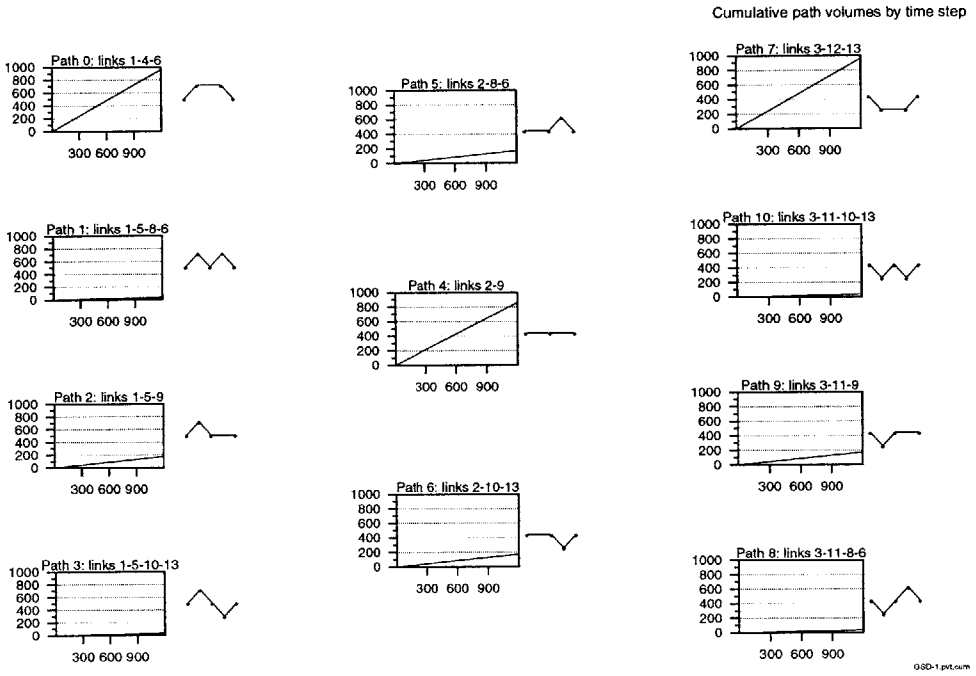


(a) IC_M convergence norm

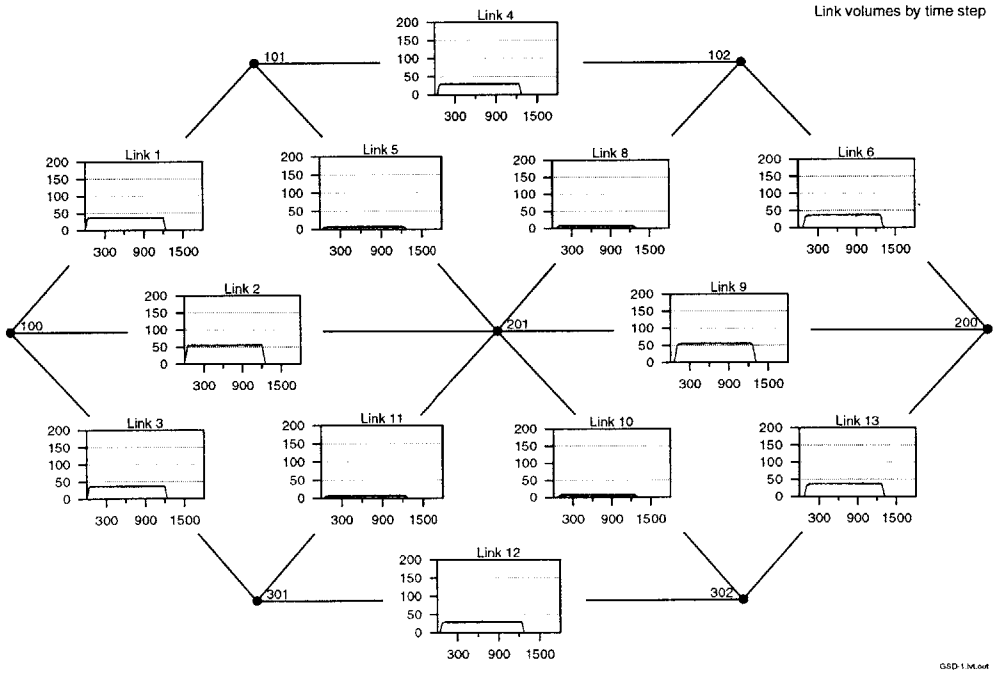


(b) Input pre-trip message trajectories (seconds)

Figure 5.35: Run GSD-1 (part 1/3)

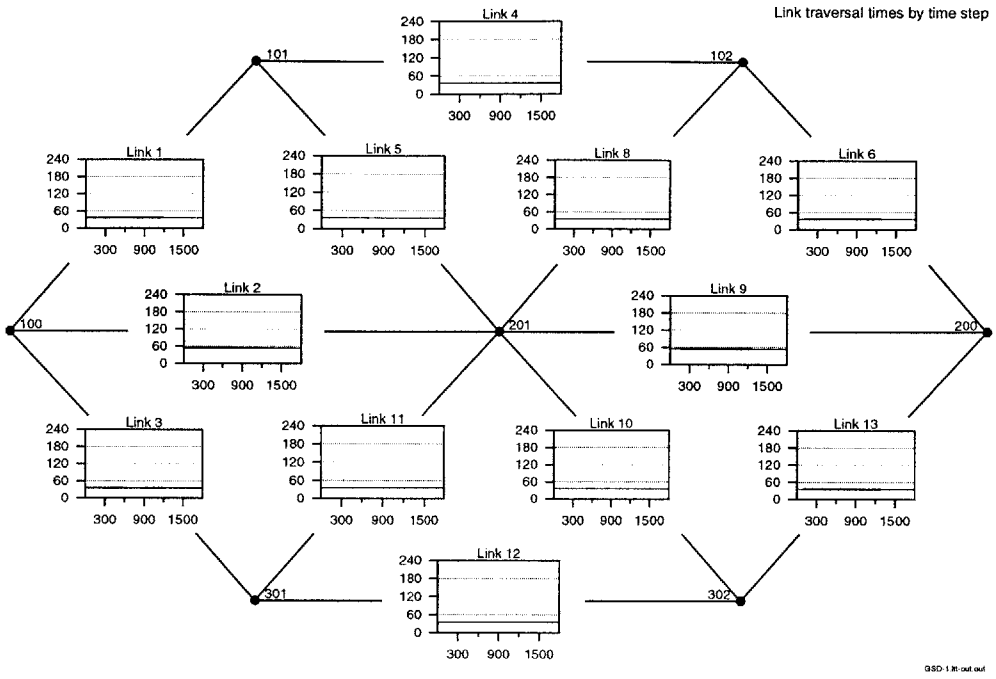


(c) Cumulative path flow trajectories (vehicles)

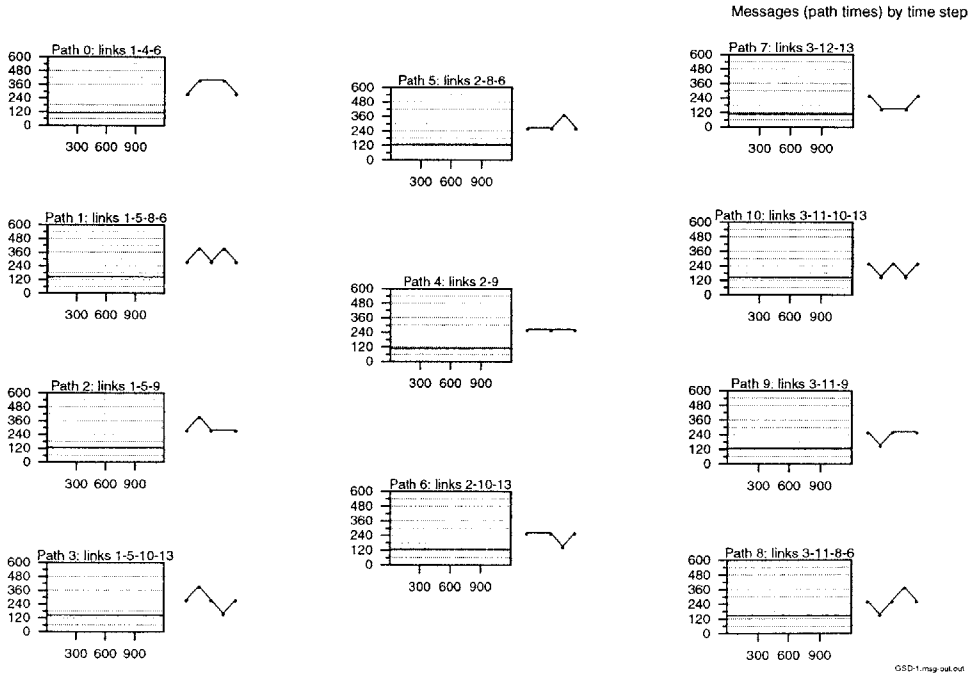


(d) Link volume trajectories (vehicles)

Figure 5.35: Run GSD-1 (part 2/3)

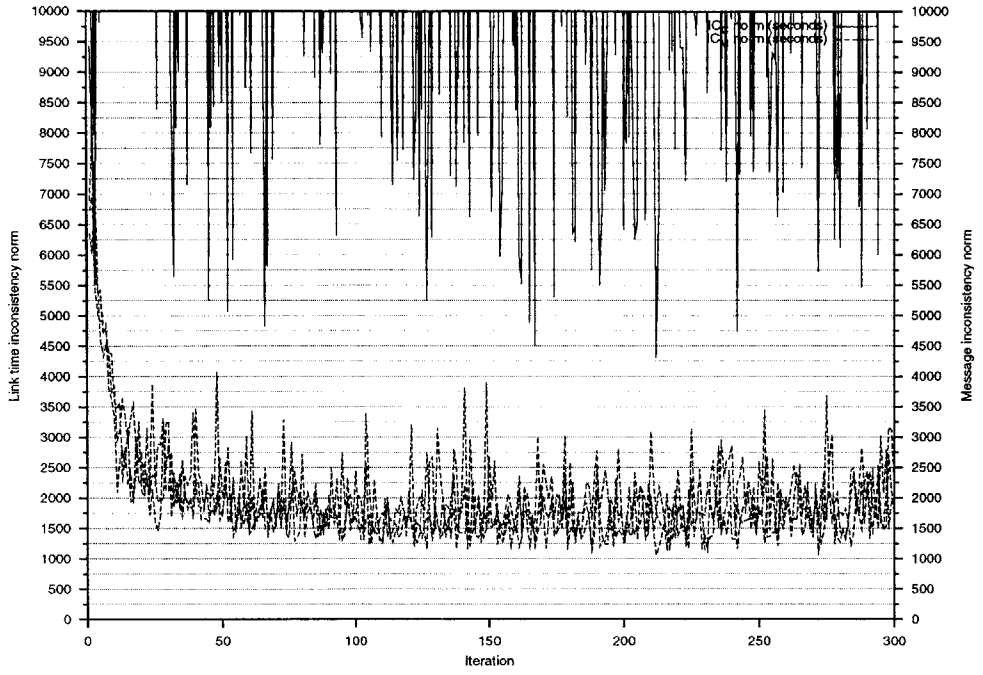


(e) Link time trajectories (seconds)

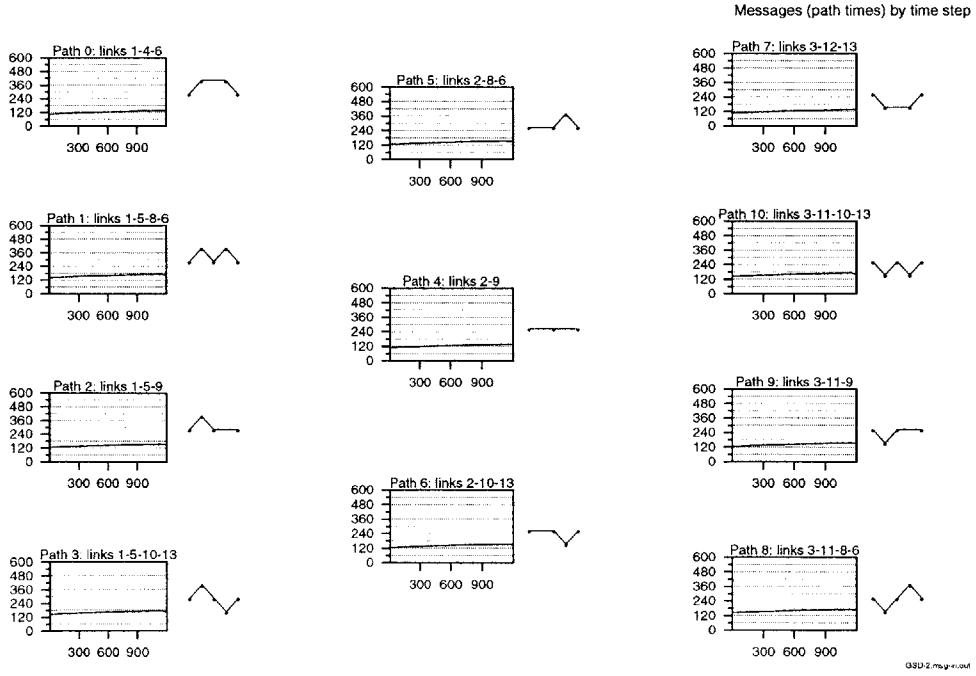


(f) Output pre-trip message trajectories (seconds)

Figure 5.35: Run GSD-1 (part 3/3)

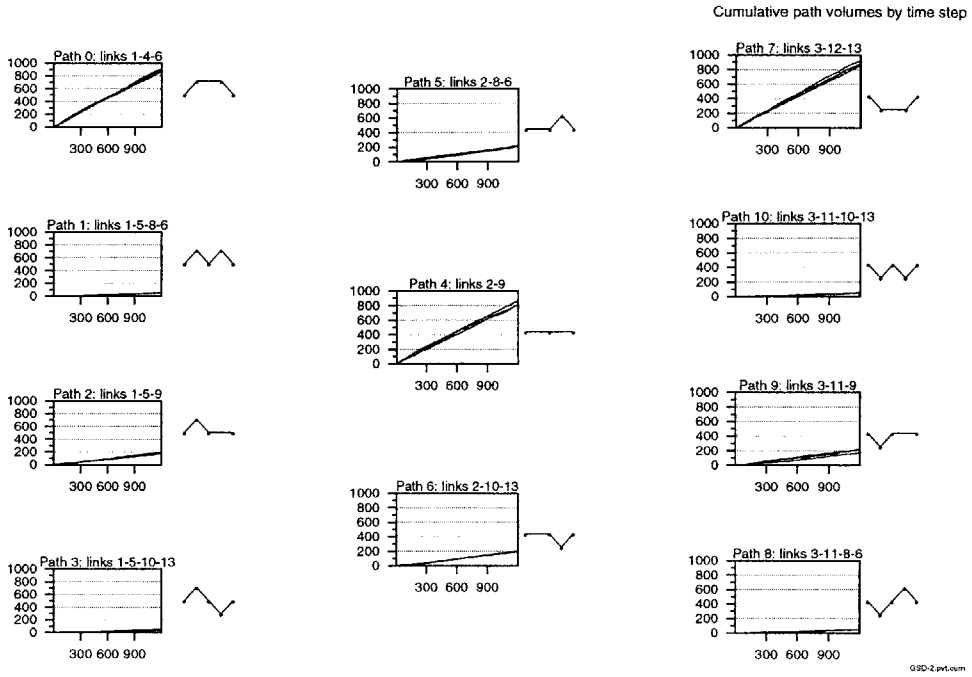


(a) IC_M convergence norm

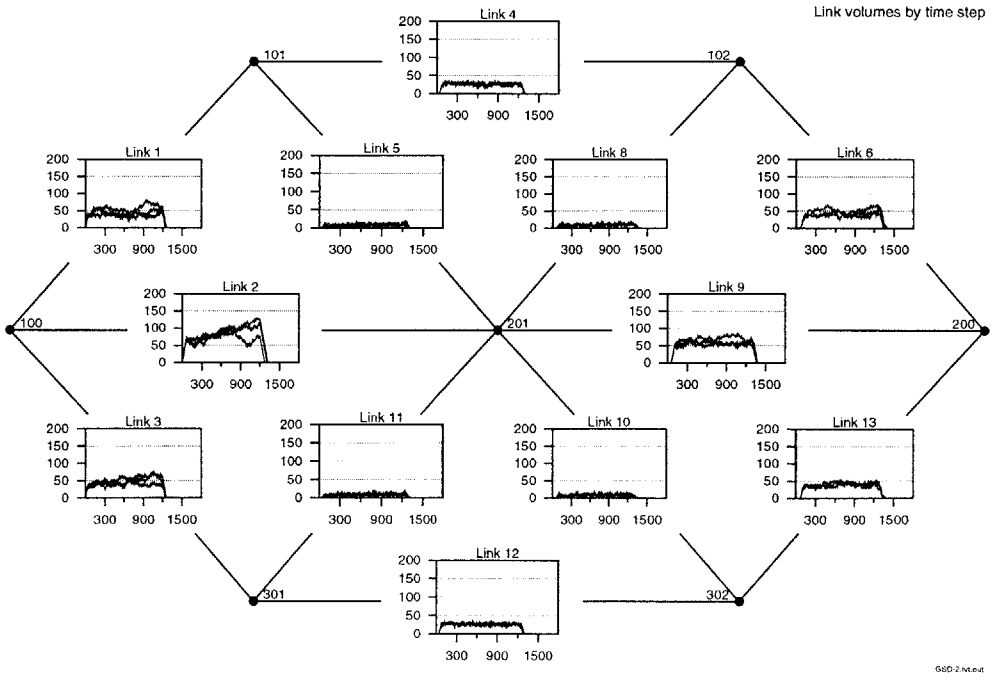


(b) Input pre-trip message trajectories (seconds)

Figure 5.36: Run GSD-2 (part 1/3)

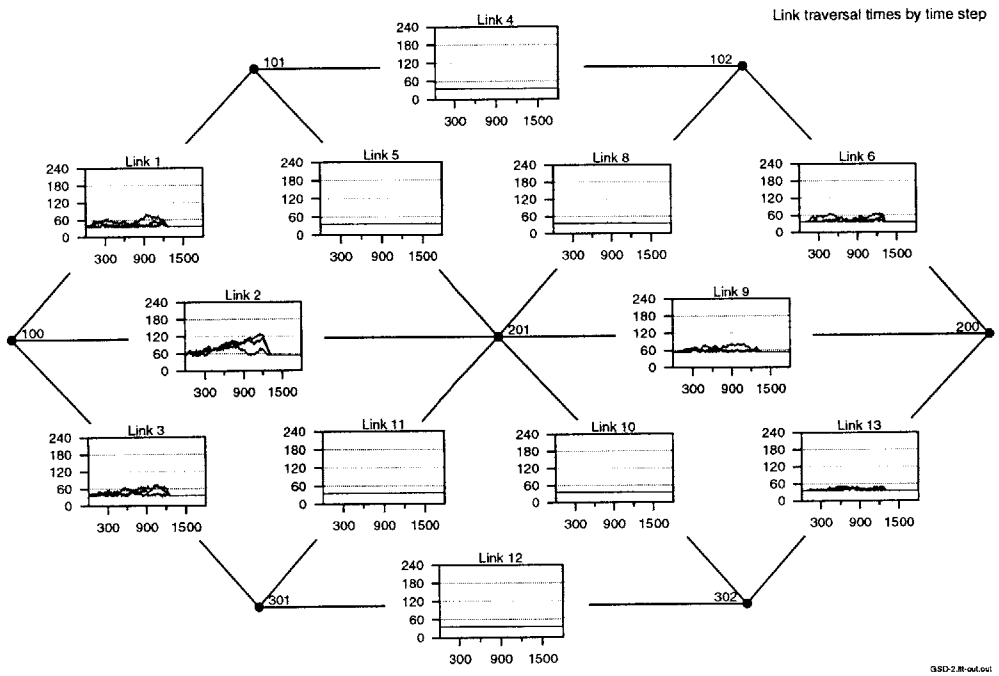


(c) Cumulative path flow trajectories (vehicles)

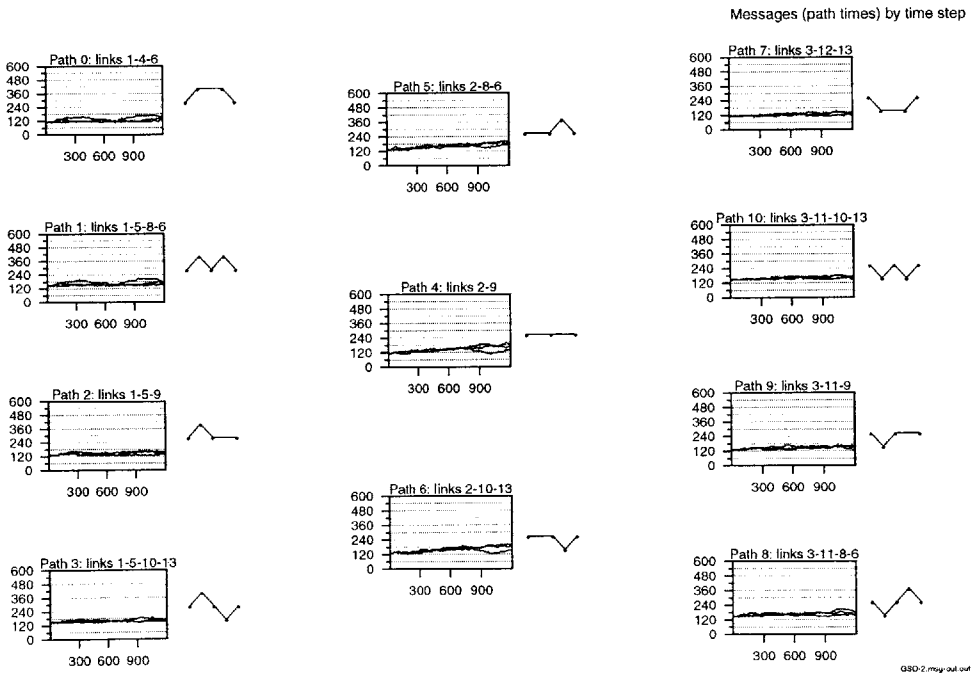


(d) Link volume trajectories (vehicles)

Figure 5.36: Run GSD-2 (part 2/3)

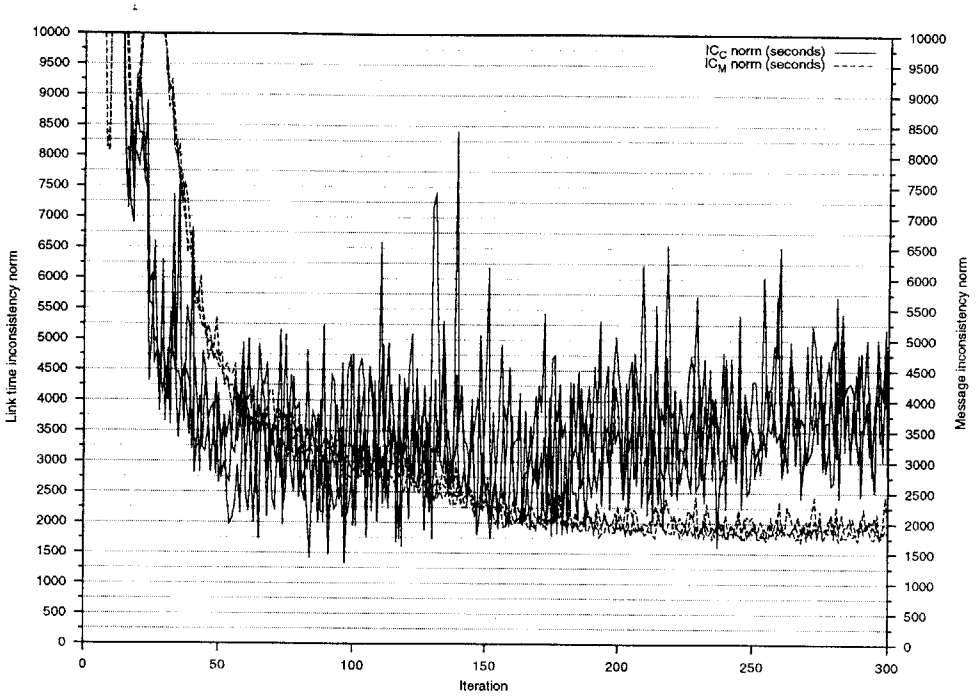


(e) Link time trajectories (seconds)

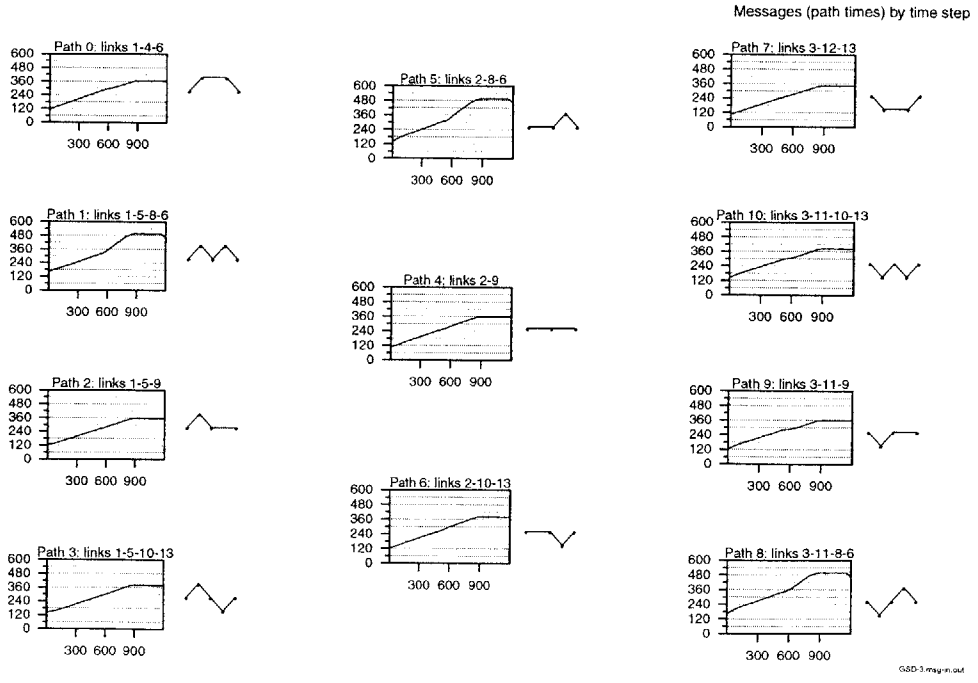


(f) Output pre-trip message trajectories (seconds)

Figure 5.36: Run GSD-2 (part 3/3)

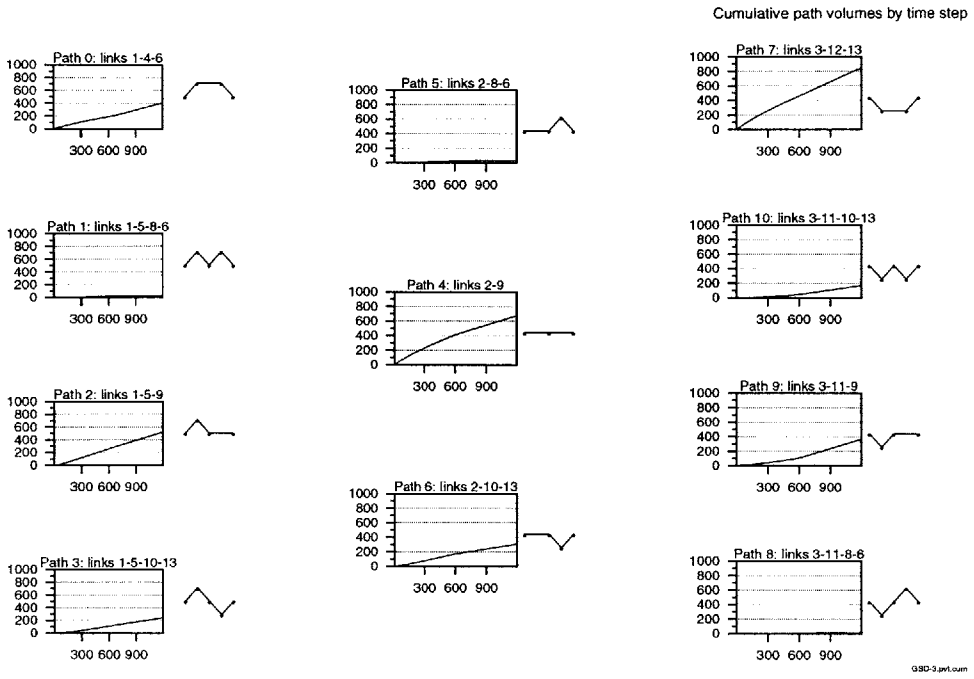


(a) IC_M convergence norm

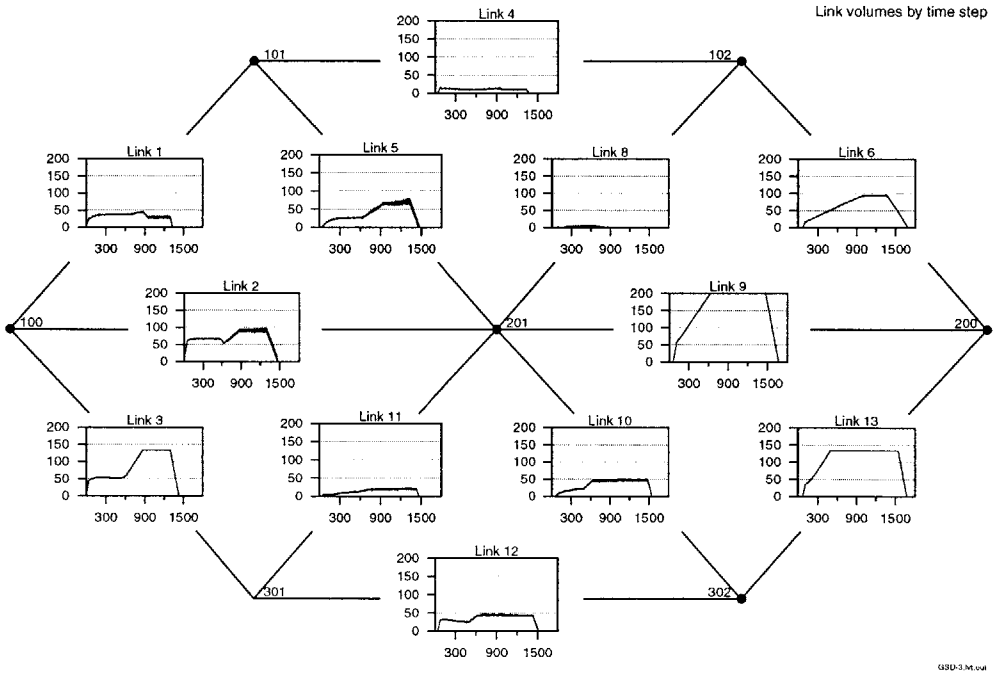


(b) Input pre-trip message trajectories (seconds)

Figure 5.37: Run GSD-3 (part 1/3)

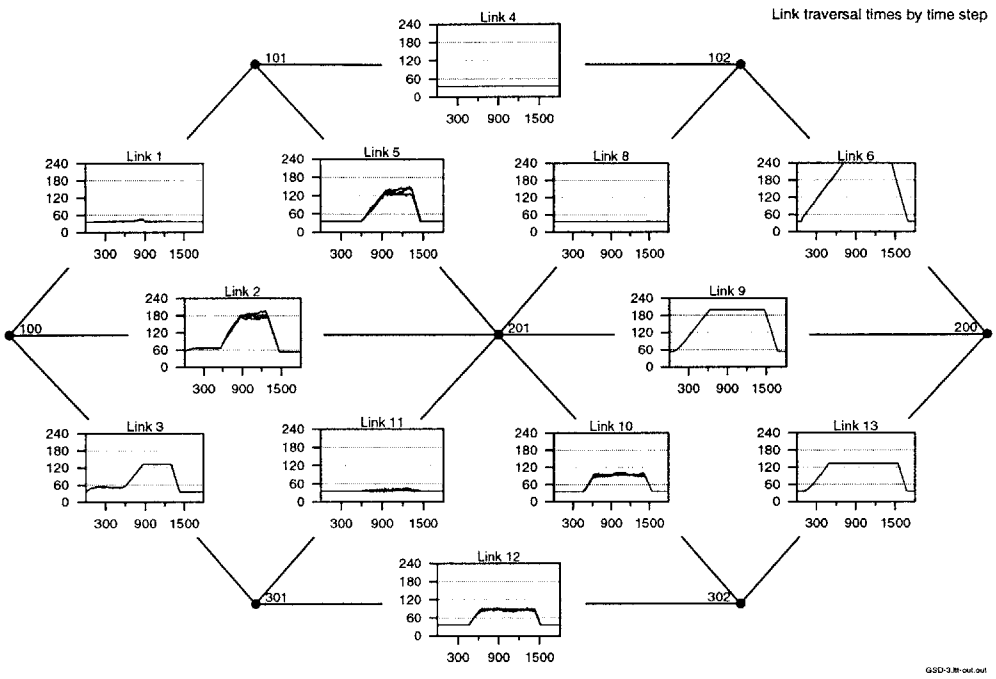


(c) Cumulative path flow trajectories (vehicles)

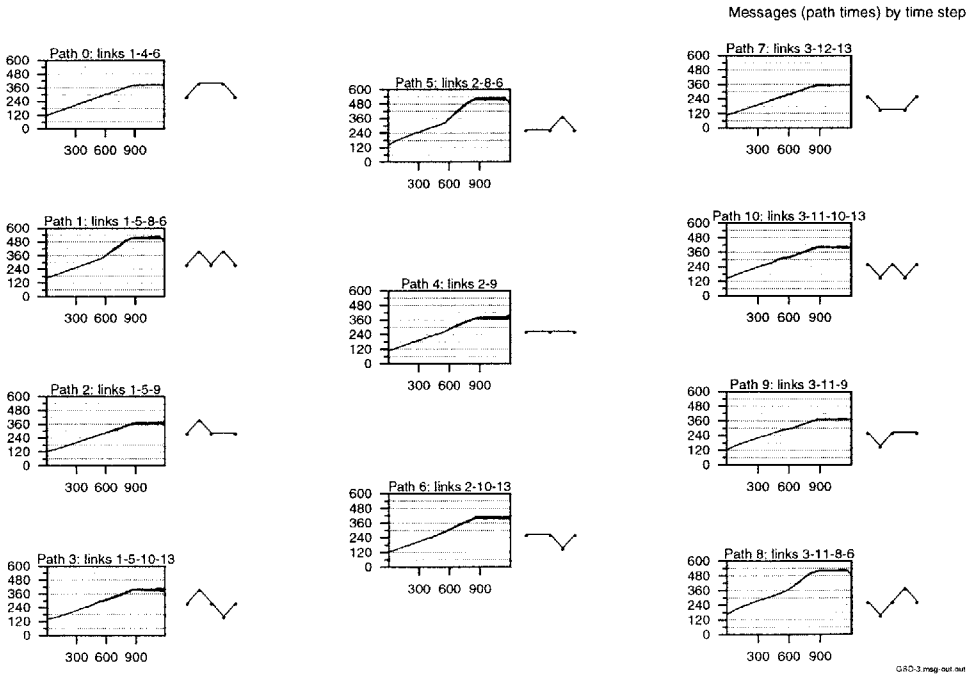


(d) Link volume trajectories (vehicles)

Figure 5.37: Run GSD-3 (part 2/3)

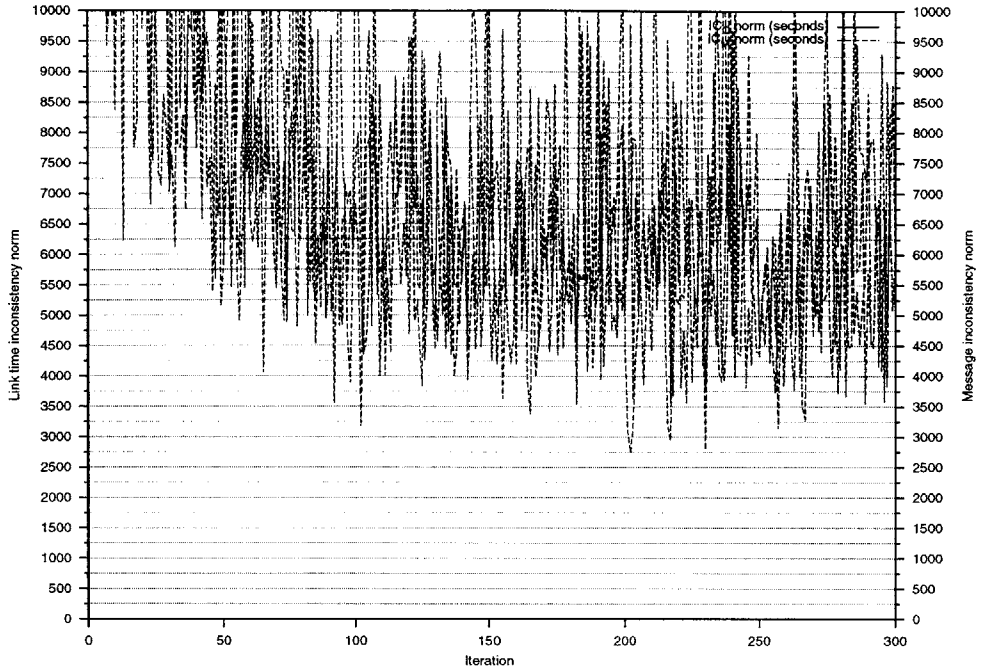


(e) Link time trajectories (seconds)

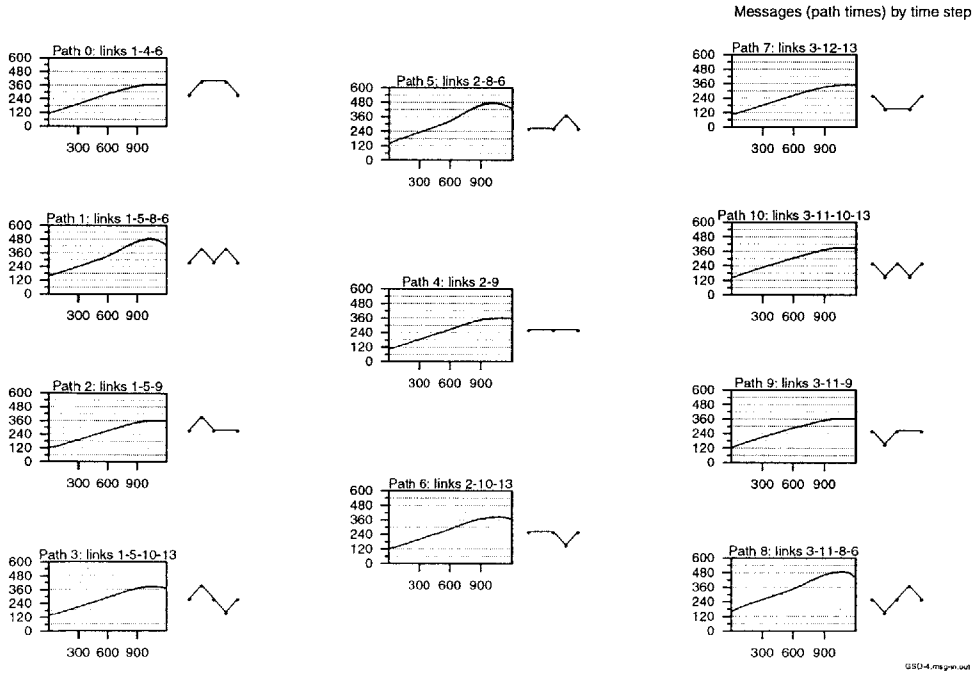


(f) Output pre-trip message trajectories (seconds)

Figure 5.37: Run GSD-3 (part 3/3)

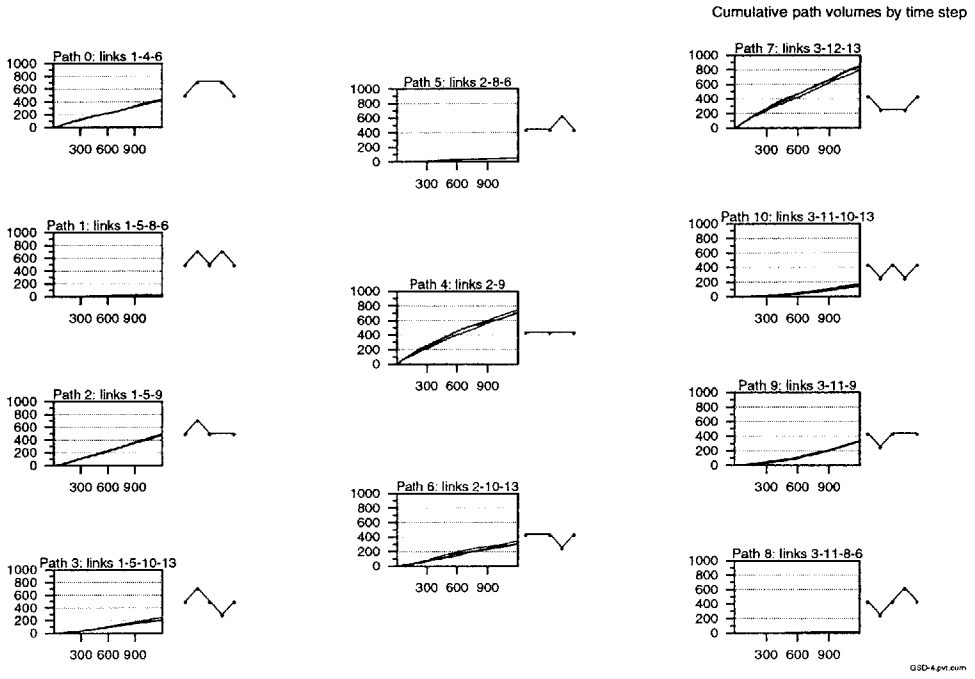


(a) IC_M convergence norm

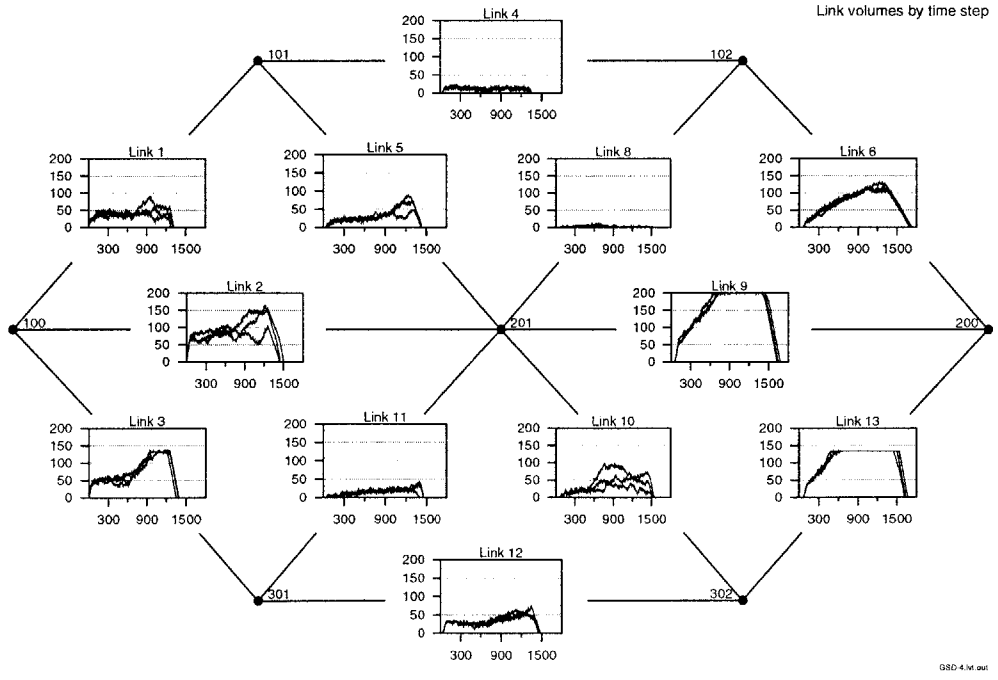


(b) Input pre-trip message trajectories (seconds)

Figure 5.38: Run GSD-4 (part 1/3)

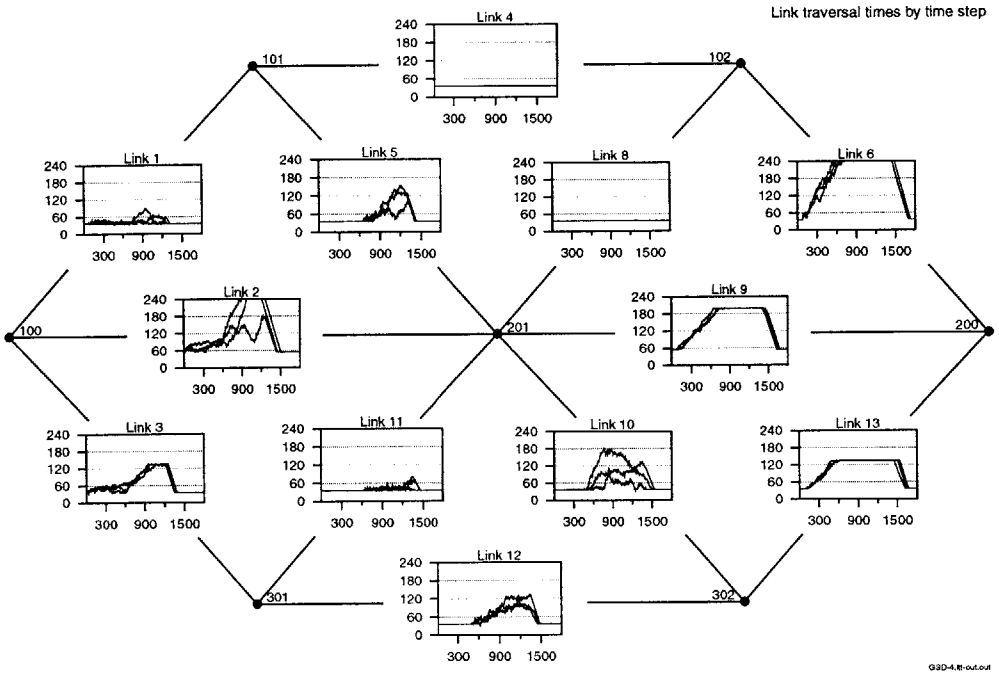


(c) Cumulative path flow trajectories (vehicles)

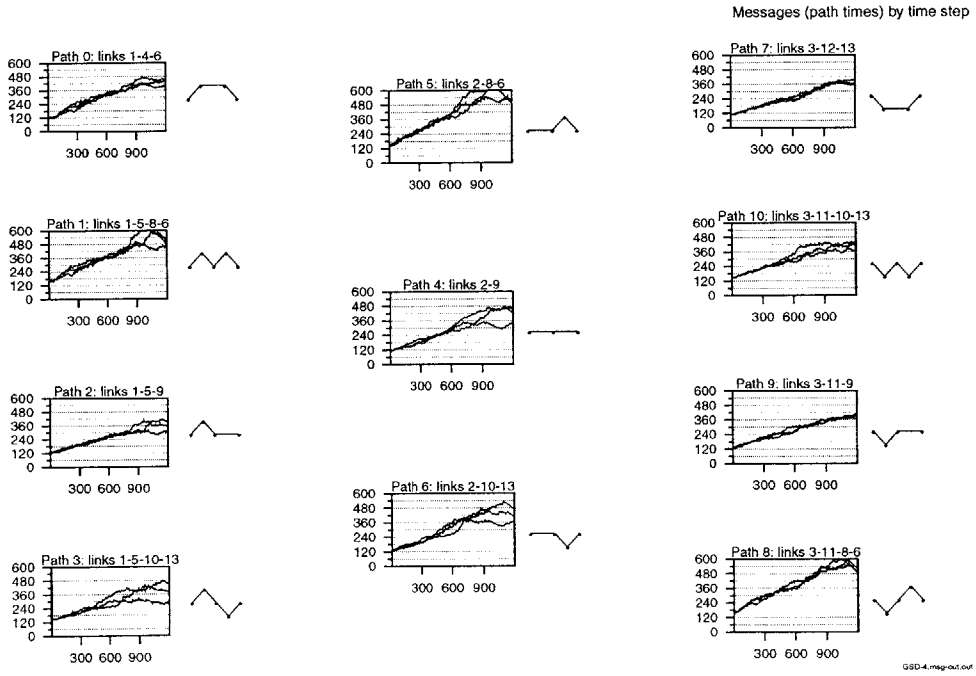


(d) Link volume trajectories (vehicles)

Figure 5.38: Run GSD-4 (part 2/3)

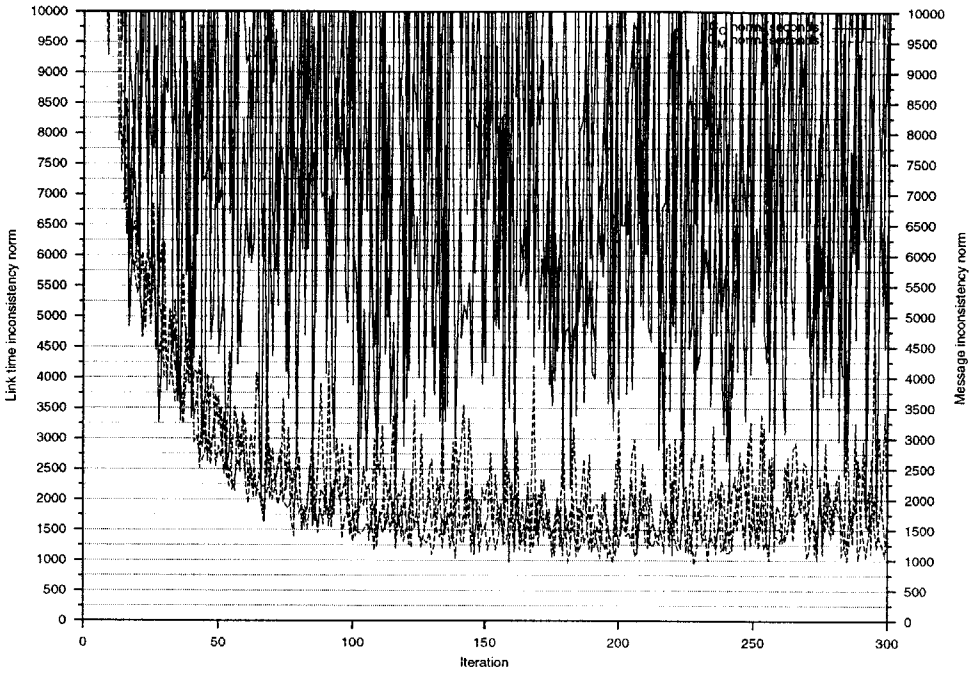


(e) Link time trajectories (seconds)

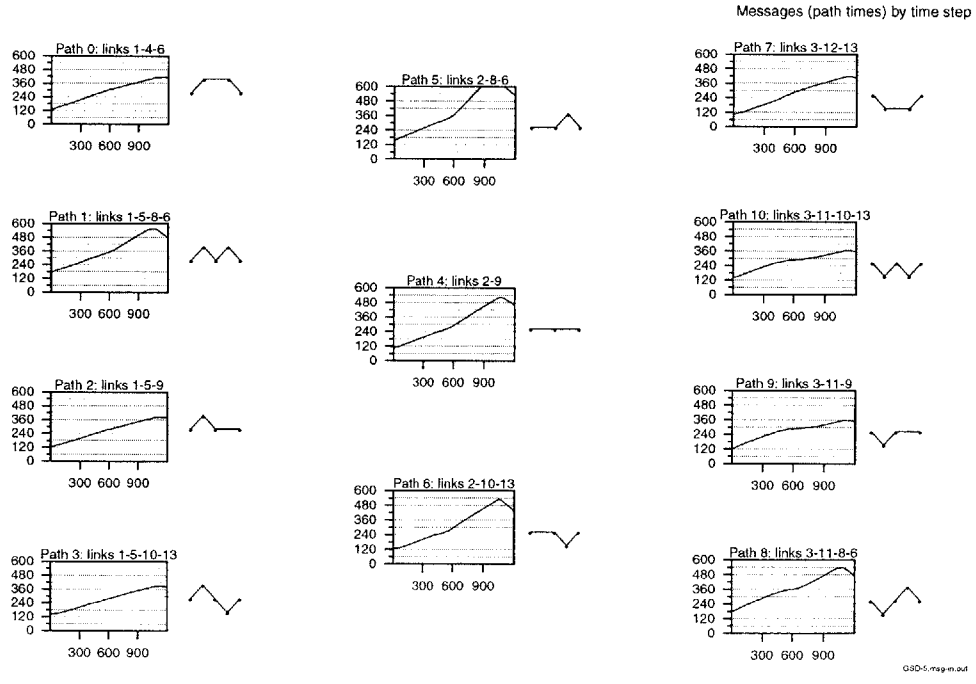


(f) Output pre-trip message trajectories (seconds)

Figure 5.38: Run GSD-4 (part 3/3)

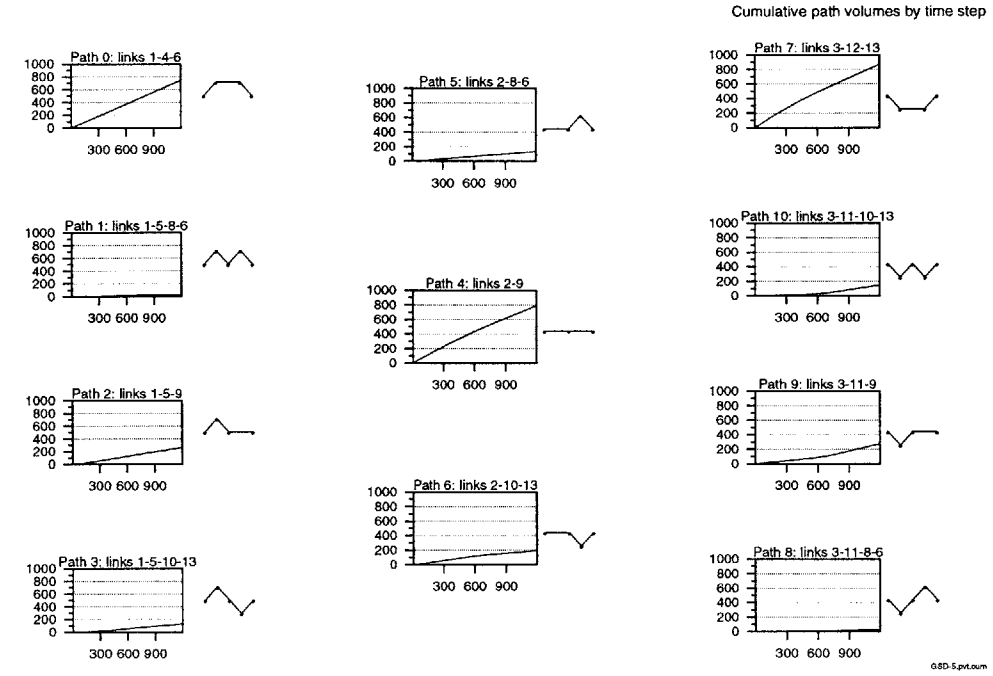


(a) IC_M convergence norm

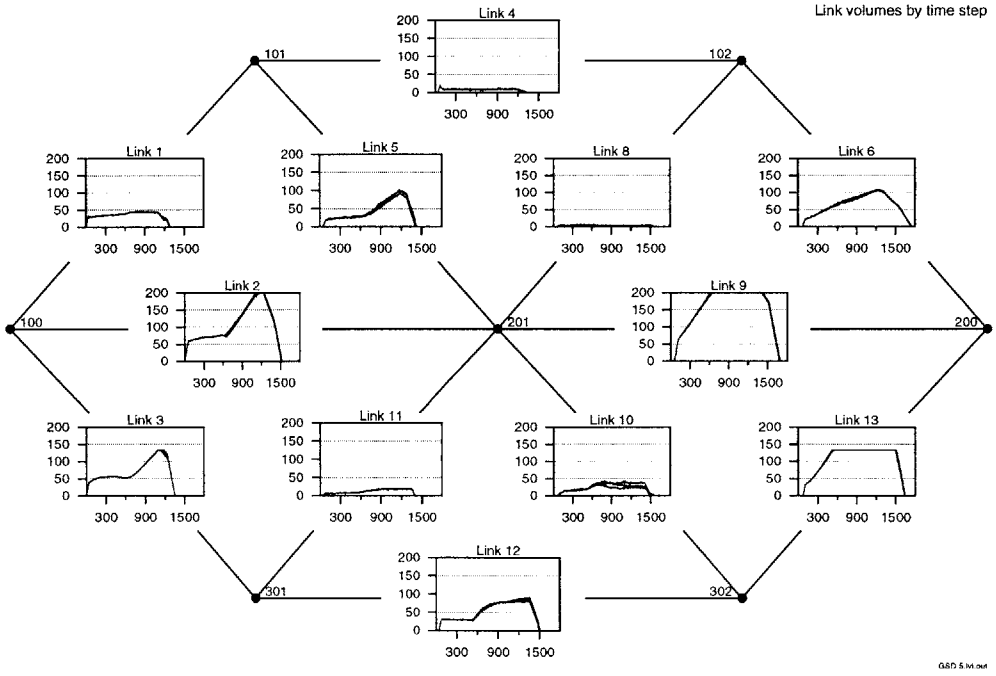


(b) Input pre-trip message trajectories (seconds)

Figure 5.39: Run GSD-5 (part 1/3)

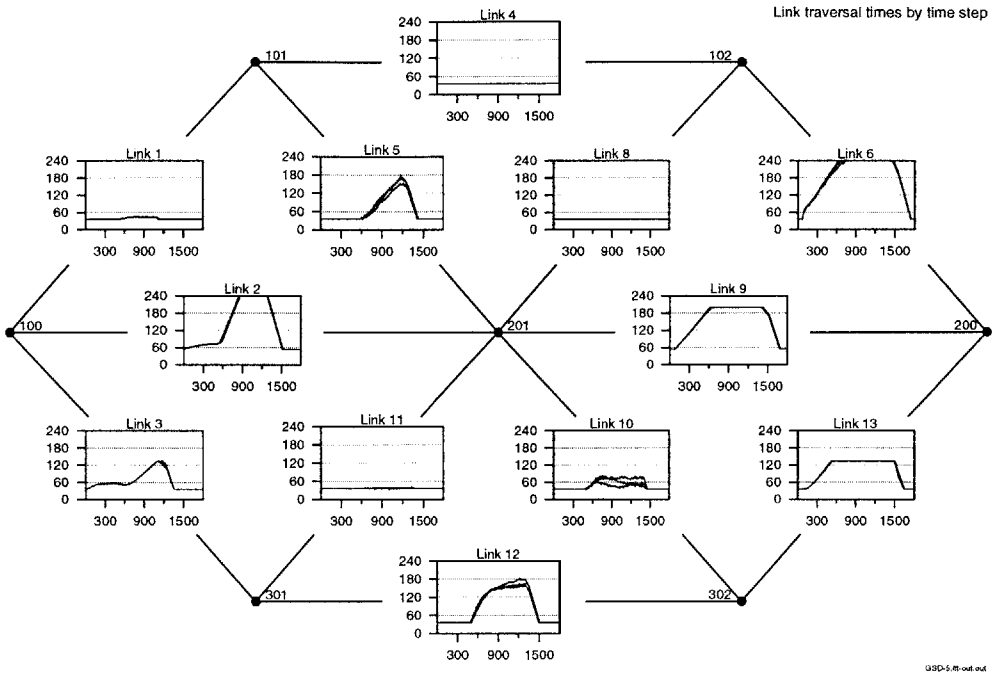


(c) Cumulative path flow trajectories (vehicles)

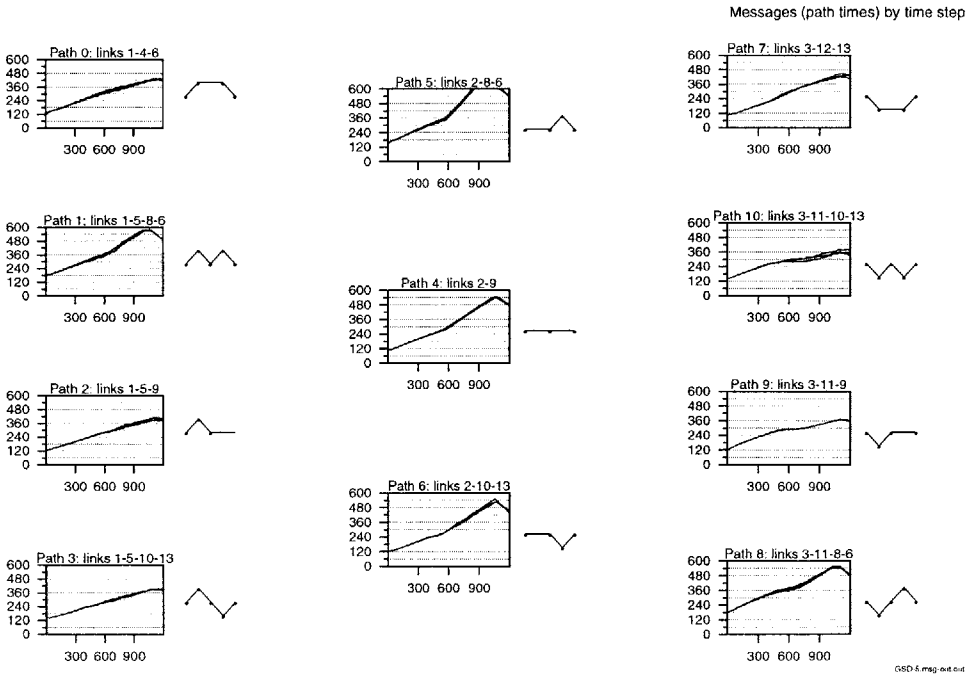


(d) Link volume trajectories (vehicles)

Figure 5.39: Run GSD-5 (part 2/3)

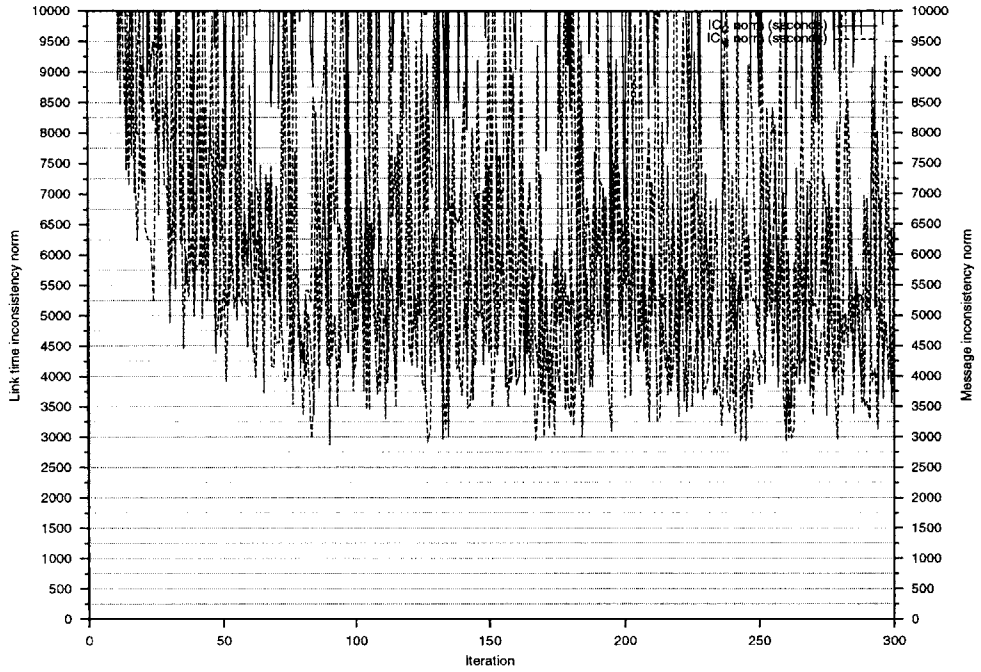


(e) Link time trajectories (seconds)

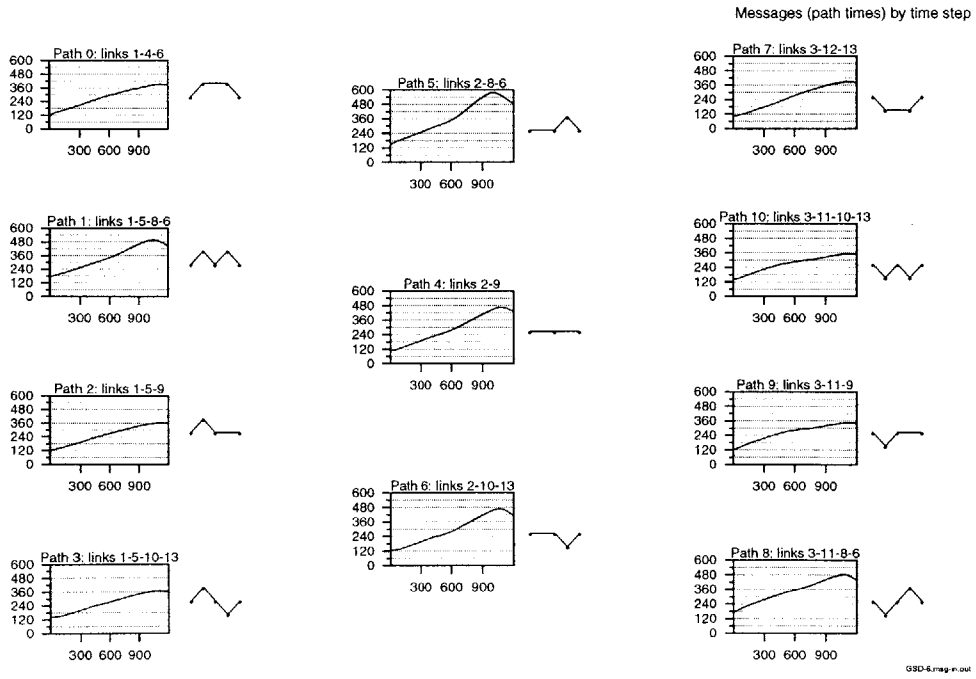


(f) Output pre-trip message trajectories (seconds)

Figure 5.39: Run GSD-5 (part 3/3)

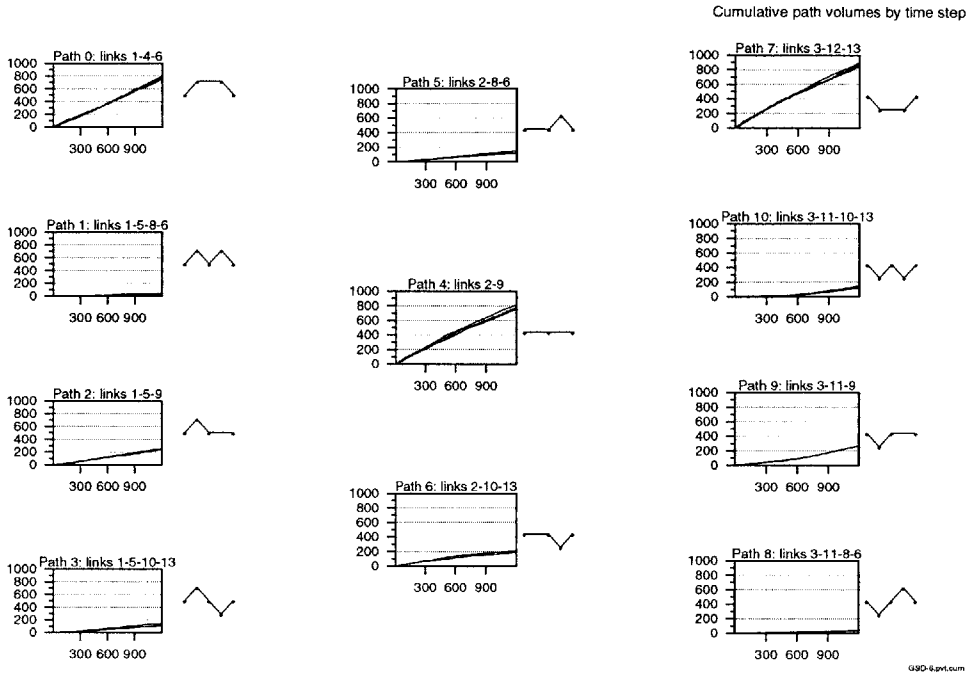


(a) IC_M convergence norm

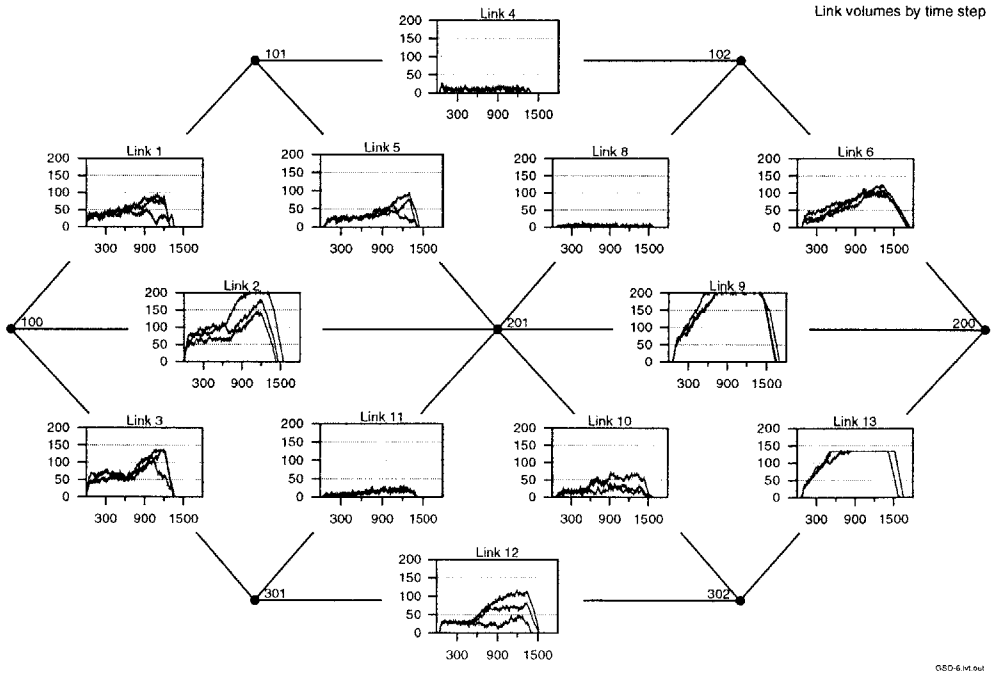


(b) Input pre-trip message trajectories (seconds)

Figure 5.40: Run GSD-6 (part 1/3)

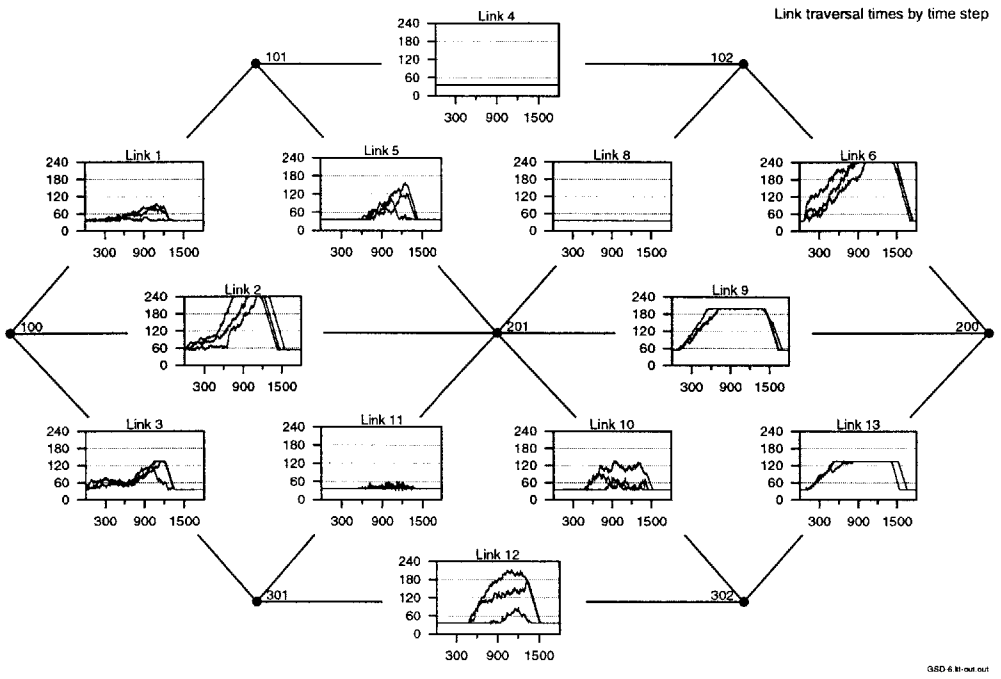


(c) Cumulative path flow trajectories (vehicles)

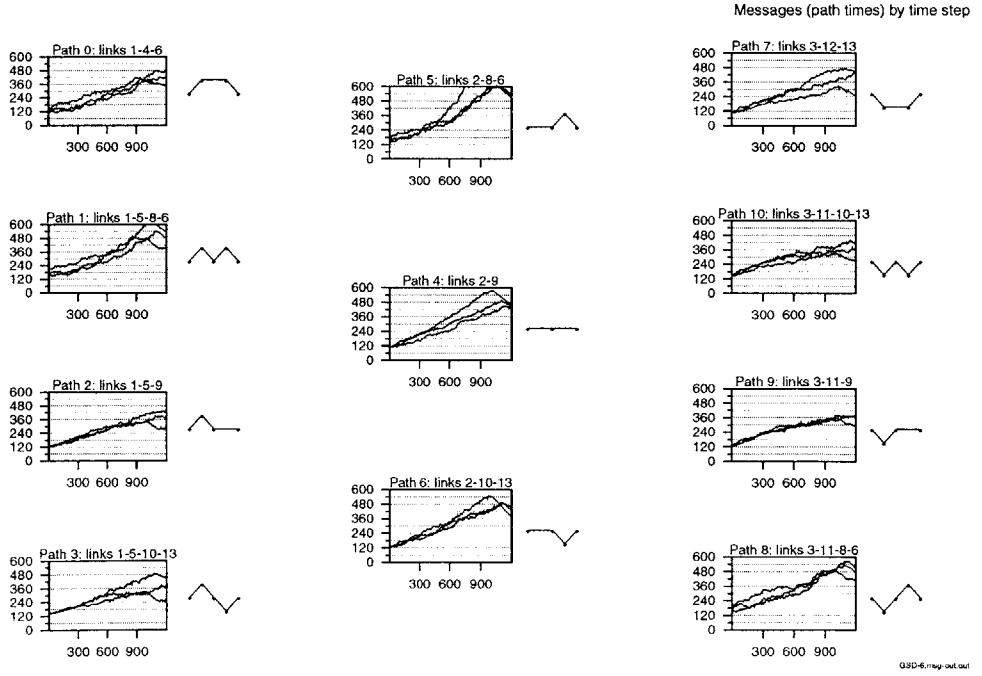


(d) Link volume trajectories (vehicles)

Figure 5.40: Run GSD-6 (part 2/3)

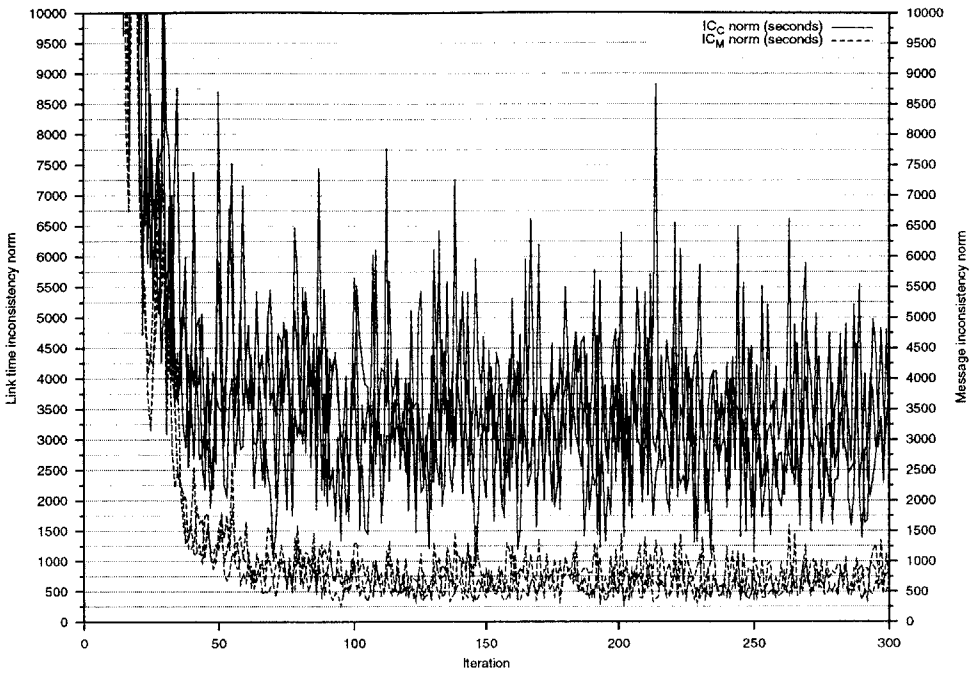


(e) Link time trajectories (seconds)

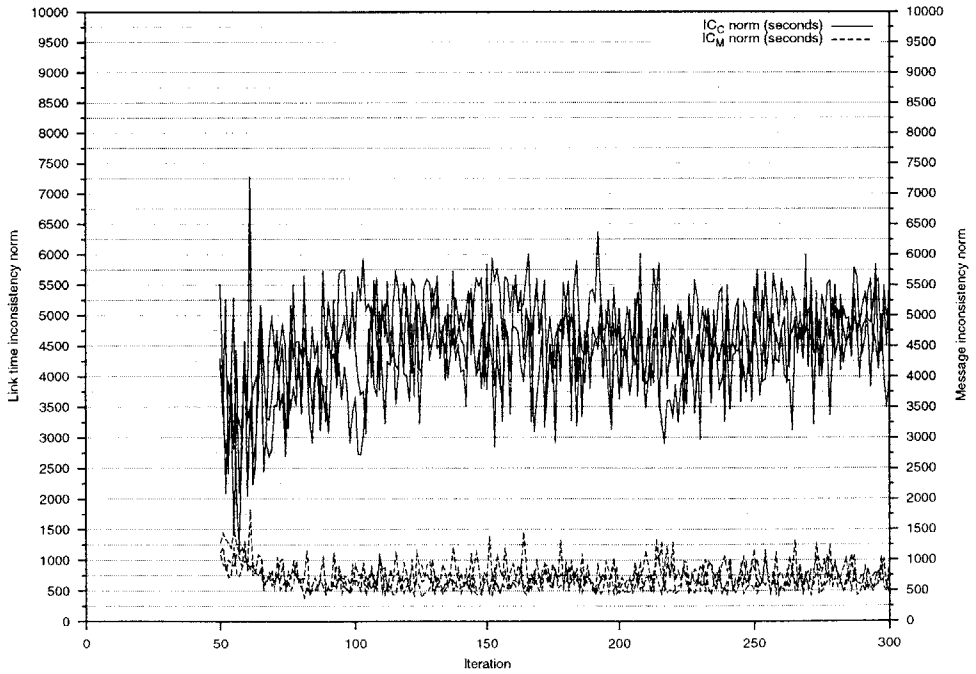


(f) Output pre-trip message trajectories (seconds)

Figure 5.40: Run GSD-6 (part 3/3)

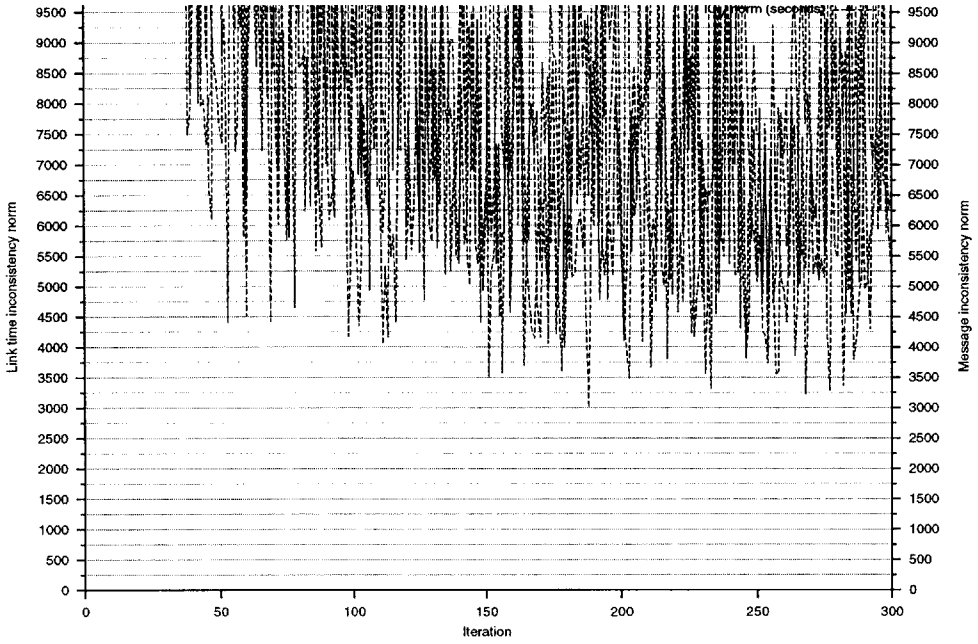


(a) IC_M convergence norm in first pass

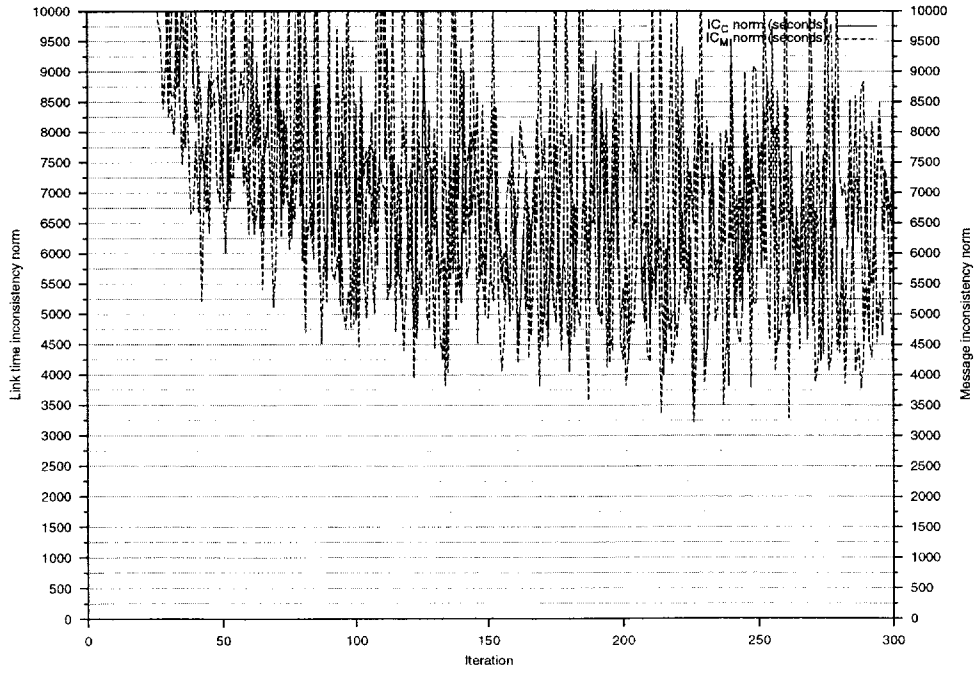


(b) IC_M convergence norm in second pass

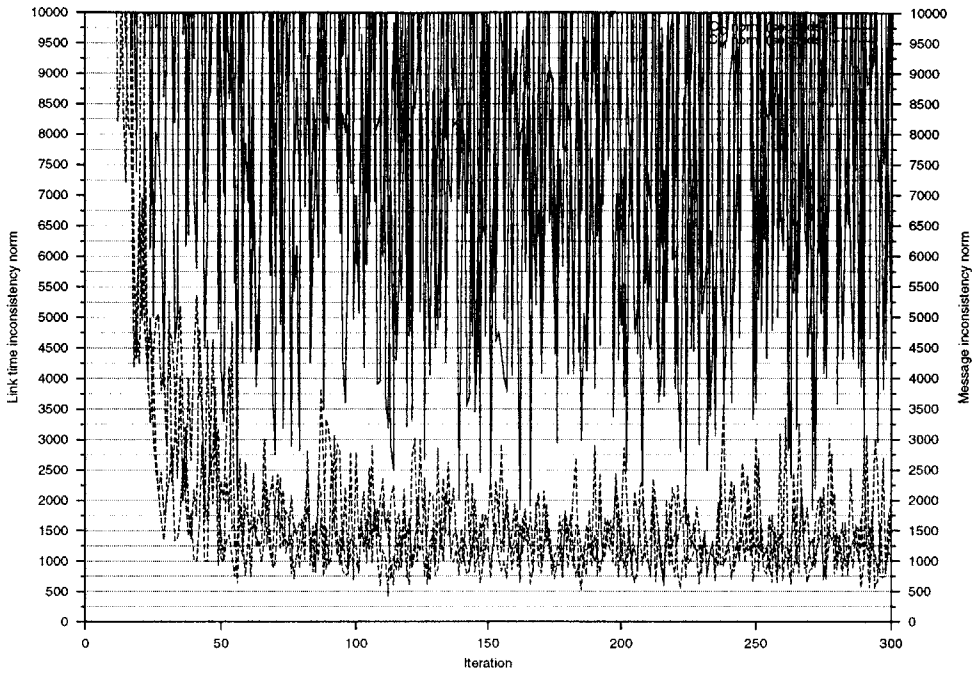
Figure 5.41: Run GSD-3-pol



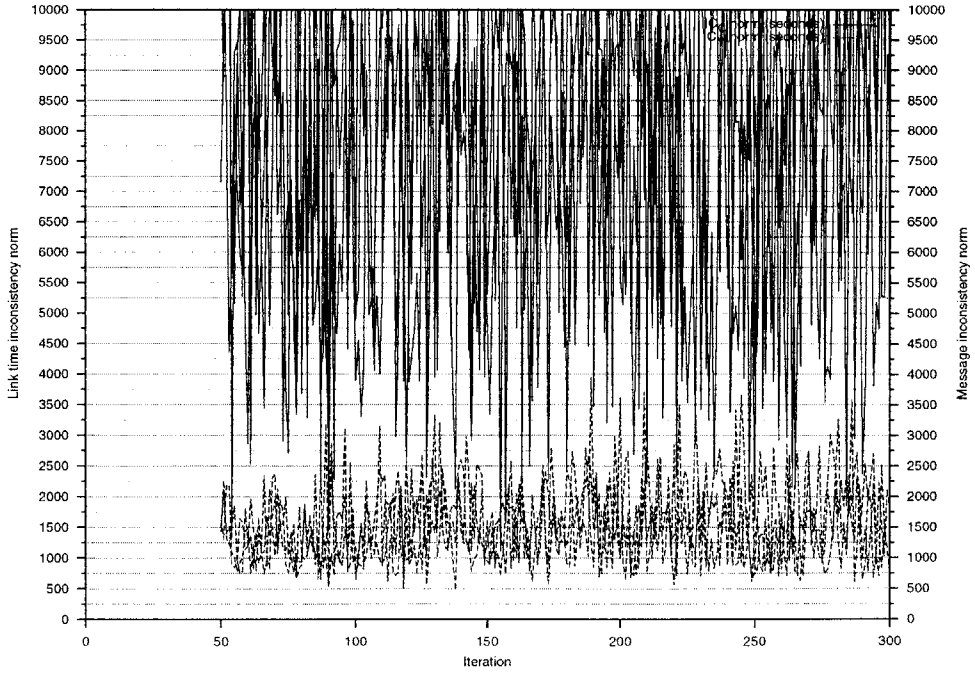
(a) IC_M convergence norm in first pass



(b) IC_M convergence norm in second pass



(a) IC_M convergence norm in first pass



(b) IC_M convergence norm in second pass

Figure 5.43: Run GSD-5-pol

Chapter 6

Conclusions and Directions for Further Work

The traffic lights, they turn blue tomorrow

Hendrix [1967]

6.1 Recapitulation of the work and conclusions

This thesis has considered the formulation and solution of the consistent anticipatory route guidance generation (RGG) problem. Anticipatory route guidance consists of messages that attempt to influence drivers' path choices before or during their trip, where the messages are based on forecasts of what network conditions will be during the trip. The basic problem in generating anticipatory guidance is that drivers are likely to react in some way to the guidance messages they receive, and if this reaction is not properly taken into account when carrying out the forecasts and generating the guidance messages, then most likely the forecasts will be wrong and the guidance inappropriate.

It was argued that the RGG problem is more general than the conventional full information dynamic traffic prediction problem because RGG requires explicit consideration of the specific properties (e.g. geographic and temporal availability, precision and accuracy) of route guidance messages, whereas the full information problem implicitly assumes that perfect information is available always and everywhere. It was shown, in fact, that the conventional full information problem can be considered as a special case of the general RGG problem.

Anticipatory guidance requires a predictive model to forecast future dynamic traffic conditions from which guidance messages are derived. The model must take into account drivers' pre-trip and en route responses to disseminated guidance messages, and must be able to predict the network conditions resulting from these responses. The fundamental notion of *consistency* was defined: in the framework of a particular predictive model, anticipatory guidance is said to be consistent if the assumptions on which it is based prove to be realized, within the logic of the predictive model, after the guidance is disseminated and drivers react to it. Consistency is a model-based concept: it is simply the requirement that model inputs and outputs not be contradictory. Consistency as an issue in guidance generation is unique to anticipatory guidance; other forms of guidance (e.g. guidance based on prevailing or historical conditions) do not confront it.

The thesis proposed a general analysis framework for the RGG problem. This framework identifies a set of variables and basic causal relationships between them as fundamental to guidance modeling and computation. Variables are all time-dependent and consist of network conditions, path splits and guidance messages. Basic relationships between the fundamental variables are the

network loading map, which maps path splits into network conditions; the guidance map, which maps network conditions into the corresponding guidance messages; and the routing map, which maps guidance messages into path splits.

In this framework, consistency corresponds to a fixed point of a composite map of the basic relationships representing an RGG problem. Three different composite map problem formulations were derived. Each combines the basic relationships in a different order to map one of the framework variables from its domain into itself. The composite network condition map, for example, begins with a time trajectory of network condition variables, applies the guidance map to obtain messages, from which the routing map determines path splits, which are used by the network loader to obtain a new network condition trajectory. The composite path split and guidance maps are defined in similar fashion. The three composite map formulations are equivalent with respect to the existence or not of a guidance fixed point, but have different functional properties and provide different opportunities for designing solution algorithms. The thesis relates this framework to a number of other approaches to the guidance generation and similar problems that have been presented in the literature.

Stochasticity is present in many models used for dynamic traffic prediction and route guidance generation. Stochasticity may arise from the supply side (for example, randomness in flow propagation relationships) or the demand side (for example, probabilistic path choice behavior), or both. Stochasticity can have a very significant impact on the RGG problem analysis and solution approach, and it was shown that in general the outputs of an RGG model run may need to be treated as realizations of stochastic processes. Under mild conditions, the notion of consistency as a fixed point of a composite map retains its validity, where in this case the fixed point is a stationary process of the random variables that define the RGG problem.

This framework formalizes and generalizes the *ad hoc* and often more limited approaches that have been used to define, analyze and compute consistent anticipatory guidance in the past, and opens the RGG problem to systematic investigation of solution methods.

Accordingly, numerical methods for the computation of fixed points were examined, focusing on approaches that are both applicable to large-scale problems such as RGG and also mathematically rigorous. Three classes of methods were considered, according to the amount of stochasticity in the composite map whose fixed point is to be determined.

For highly stochastic problems, where the fixed point solutions must be considered to be general stochastic processes, Gibbs sampling can be rigorously used to generate draws from the stationary stochastic process solution. Gibbs sampling provides considerable insight into the nature of stochastic process solutions of dynamic network problems; however, the method is computationally very demanding since it involves a large number of dynamic network loadings.

A number of methods for fully deterministic problems were reviewed, including triangulation (simplicial decomposition), functional iteration and a variety of its small step-size generalizations. Gradient-based methods were not considered because, in many models (most notably simulation-based models), the gradients would require numerical rather than analytical evaluation, and the typically large size of RGG problems makes this computationally impractical.

Quasi-deterministic problems are those in which function evaluations return deterministic values affected by zero-mean noise. Techniques for performing standard numerical analysis procedures (e.g. root, fixed point or optimum finding) on such functions are known as stochastic approximation methods. To the transportation community, the method of successive averages (MSA), a special

case of the multidimensional generalization by Blum [1954] of the one-dimensional Robbins-Monro procedure [Robbins and Monro, 1951], is the best-known of these techniques. However, many other stochastic approximation procedures have been developed since the MSA. Iterate averaging methods, including a method due to Polyak [1990], which have asymptotically optimal convergence properties yet are very simple to implement, were identified in this research as particularly promising for route guidance (and other stochastic network) applications.

This research also developed a guidance-oriented dynamic traffic simulation software system that allows convenient experimentation with RGG problem formulations and solution methods. The simulator is vehicle-based and implements a store-and-forward propagation protocol with spillback. It provides facilities for representing geographically specific guidance sources (e.g. VMS or guidance beacons) disseminating arbitrary guidance messages derived from predicted network conditions. Pre-trip and en route driver responses to descriptive guidance are represented by a logit-based path choice model; a simple compliance model is used for prescriptive guidance. The system provides a number of ways to influence the stochasticity of the simulation relationships and outputs, so that outputs ranging from fully stochastic to almost deterministic can be obtained at the user's option. Each of the three distinct composite map guidance formulations is implemented, as is a variety of solution techniques. The software is written in C++ and is designed to be easy to modify and to enhance. It writes ASCII files of selected time-dependent system variables that can be plotted or post-processed in a variety of ways according to the particular needs of an analysis.

The simple simulator was applied in a number of computational tests. Initial tests investigated the properties of the simulator itself: in particular, the "degree" of stochasticity introduced by different simulator components, and the extent to which this could be controlled. It was found that by applying path splits in an aggregate fashion to homogeneous groups of travelers, by using bucket rounding in all instances of fraction-to-integer conversions, and by subdividing individual vehicles into sets of independently-processed fractional flow units, stochasticity in simulator outputs could be substantially suppressed. Conversely, without these measures simulator outputs are highly stochastic, in the sense for example that multiple replications of a composite map evaluation with identical input data lead to quite different simulation variable output time trajectories.

Further tests investigated the suitability of Gibbs sampling as a means of computing consistent stochastic process solutions to models with high stochasticity; the tests took place in a full information problem setting. It was found that Gibbs sampling could effectively generate multiple realizations from the solution process. Examination of the outputs suggested that, for the given problem data, output distributions in each time period could be treated as unimodal and were generally concentrated around the process mean value trajectory.

Finally, an extensive series of tests was carried out to determine the performance of different solution methods applied to the various formulations in qualitatively different problem settings. All runs involved a simple 15-link network that was initially empty, a time step of one second, and an analysis period generally between one-half and one hour, depending on the time needed to clear the network. Three problem settings were considered: full information without incident; full information with an incident on one link; and limited information (e.g. short-range guidance such as a VMS) with an incident on one link. Each of these situations was separately modeled using the composite network condition, composite path split and composite message formulations, and solutions were obtained for each formulation using both the MSA and Polyak averaging. As a final factor in the computational tests, the simple simulator was configured to evaluate the

composite maps in both fully stochastic and quasi-deterministic implementations, and solutions were computed for these two cases as well.

A convergence norm was defined for each composite map formulation, measuring the vector 2-norm between "input" and "output" map variables in successive iterations. At a fixed point of a deterministic map, this norm value would of course be zero; at the fixed point of a noisy map the norm would vary somewhat near zero; and at the fixed point stationary distribution of a highly stochastic model, the norm would vary, perhaps considerably, within a range strictly separated (w.p. 1) from zero. Because of the inherent map stochasticity, it did not prove possible to identify a "universal" convergence measure allowing comparison on a common basis of the convergence properties of the different composite map formulations. However, the number of iterations to convergence of each particular map's norm, when applied to the same problem, was a useful indicator of the relative performance of the different formulations in the various situations considered. No attempt was made to measure or analyze the computer resources used (execution time, memory usage, etc.) in the different runs because the simulation program code was designed and coded pursuing objectives other than computational efficiency, and included computationally expensive features for tracking convergence that would not be present in a production version.

A few general conclusions emerged from analysis of the run results. The MSA appeared to successfully solve all problems that it was presented, but it exhibited for RGG problems the same slow asymptotic convergence that has been observed in its applications to other problems. For a given quasi-deterministic problem map, Polyak averaging generally outperformed the MSA as a fixed point solution method by factors (i.e. ratio of the number of iterations to convergence) ranging from around two to more than four. Since Polyak averaging can be rigorously applied whenever the MSA can be, the method merits very serious consideration as a solution method for the RGG problem. It also appears very likely that its advantages would carry over to other transportation problems that utilize the MSA as a solution method, such as the static SUE flow prediction problem.

The advantages of Polyak averaging over the MSA were less clear for highly stochastic problems. In both cases, a stationary range of convergence norm values was reached in comparable numbers of iterations. More fundamental, however, is the question of how to utilize the outputs of a highly stochastic model: for a given set of input data, two computed solutions, even though drawn from the same stationary stochastic process, could differ significantly. Guidance messages would be comparably different and, to the extent that the model is a good representation of reality, could be quite at odds with the particular stochastic process realization that is the actual traffic conditions on the network. The effectiveness of guidance in such situations, and ultimately its acceptance by road users, would be questionable. This point is pursued in Section 6.2 below.

While it is difficult to be equally conclusive about the relative advantages of the different composite map formulations in computing guidance, it appeared that path-based formulations (the path split formulation or the path time message formulation) converged to their minimum norm range more quickly than the other formulations in most cases. The advantages appeared to be less in situations of highly perturbed traffic flow. However, increased confidence in conclusions of this type can only come from the development of better-optimized solution codes.

As implied above, problems involving incident situations generally required more computational time to solve. This is not a surprising conclusion, given that all solution procedures were initialized with an empty network. It may be that this effect can be reduced by more appropriate initialization.

In a rolling horizon framework, for example, it would make sense to use the solution computed in the previous roll stage to initialize the computations.

It was interesting to note that limited information guidance did not take noticeably more iterations to converge than full information guidance applied to the same problem setting. On this basis, there does not appear to be any reason, from a computational point of view, to prefer a full information approach to the more conceptually correct limited information approach.

6.2 Directions for further work

Route guidance modeling and computation are relatively new fields, and there are many issues that will need considerable investigation before the RGG problem can be considered a mature research topic. The work carried out in this thesis has served to formalize the definition of the problem, and in so doing has also identified a number of areas requiring further work.

Data and basic modeling The first area is basic data collection and traffic modeling. By appropriate specification of its processing steps, the simple simulator's outputs can be made to range from nearly deterministic to highly stochastic. What degree of stochasticity corresponds best to the realities of driver path choice and traffic flow on the network? In fact currently available data do not provide a reliable answer to this question. There are only limited data on stochasticity in traffic propagation along links, and very little data indeed on probabilistic path choice and path switching behavior by drivers. Until more reliable data are available on the random aspects of traffic propagation and driver path choice, stochastic network models will lack solid empirical basis.

Modeling of driver response to guidance messages is a new field. Examples of current approaches include simple compliance probabilities with prescriptive guidance [Papageorgiou, 1990]; boundedly rational comparisons of the travel times of the currently chosen and the recommended alternative route [Mahmassani et al., 1994]; and switching penalties applied to travel times of alternative routes, as done in the simple simulator. The level of sophistication exhibited by these approaches falls far short of that typical in models of other kinds of traveler behavior. No doubt as more detailed data on path choice and switching behavior become available, the sophistication and accuracy of driver response to guidance models will also increase. In any case, the framework developed in this thesis shows how such driver models, when available, can be readily incorporated in an overall network modeling system.

Mathematical issues A number of mathematical issues were identified but not resolved during the course of this research. Little is currently known about the properties of stochastic, time-dependent traffic network models. It was argued in this thesis that, in general, the outputs of such models must be considered as general stochastic processes. Under what conditions can such processes be acceptably approximated by processes that are simpler and that offer greater structure to exploit? In particular, under what conditions can a model's general stochastic process outputs be approximated by a deterministic process perturbed by noise. This question obviously relates to the more practical question of when is it valid to apply stochastic approximation methods, rather than Gibbs sampling or related methods, to compute a fixed point trajectory. Note that this issue is not limited to the RGG problem, but extends to the solution of dynamic traffic simulation models in general.

Another mathematical question relates to the existence and approximation of fixed points in RGG models. Many realistic guidance technologies lead to discontinuous models; however, the best-known fixed point existence theorems (e.g. Brouwer, Kakutani) place some form of continuity requirement on the involved maps. Nonetheless, some progress has been made in establishing conditions for the existence of fixed points in discontinuous maps. For example, Dasgupta and Maskin [1986a,b] have looked at a number of cases of discontinuous games having Nash equilibria. Bernstein and Smith [1994] considered the case of lower semi-continuous link cost functions and established conditions for the existence of traffic equilibria. Another investigation by de Luca [1995] applied generalized quasi-variational inequalities to examine the general question of traffic equilibria in the presence of discontinuous cost functions. A very general result, due to Tarski [1955], asserts the existence of a fixed point of monotone (but not necessarily continuous) functions defined on lattices (such as $[0,1]^N$). All of these results suggest that somewhat general fixed point existence condition results for discontinuous RGG models might be possible.

However, it is possible to construct simple RGG models that do not have any fixed points, and this property doubtlessly carries over in some cases to more realistic models. Is there a meaningful notion of fixed point approximation in such cases? It may be that no approximation is acceptable (in the sense that guidance generated from it will inevitably worsen traffic conditions compared to a no-guidance situation). How can such cases be detected and what should then be done?

Algorithmic and formulation issues It was seen that Polyak averaging provides considerable improvement in convergence speed in some situations. Polyak averaging is one example of accelerated stochastic approximation methods that have been developed in the approximately 50 years since the work of Robbins, Monro and Blum; Bather's method is another. These improved methods do not appear to be widely known to the transportation community, yet they offer a simple and mathematically rigorous way to improve the notoriously poor "tail" properties of the MSA. In addition to further investigating the application of accelerated methods to the RGG problem, it would be of great interest to compare the performance of these methods with that of the MSA in standard transportation applications such as the static SUE problem and the full information dynamic traffic assignment problem.

Reoptimization of the dynamic network loading (DNL) step was identified as a particular technique that holds the possibility of significantly improving the computational efficiency of the fixed point solution process and Gibbs sampling. In effect, many invocations of DNL are carried out as part of an iterative process that modifies the DNL inputs by only a small amount from one iteration to the next. It may be possible to exploit the similarity of inputs so as to compute a loading with less effort than, say, starting from an empty network.

The framework developed in this thesis was based on the identification of modeling variables (guidance messages, path splits and network conditions—more particularly link times) that can be considered as fundamental to the RGG problem. A number of other problem variables can be derived from these fundamental variables; the path times that were used as a proxy for guidance in the $G \circ S \circ D$ computational runs is an example of such a derived variable. It is possible that formulations based on certain derived variables can be solved more efficiently than those based on the basic framework variables. Computational experiments will be required to address this question.

The possible multiplicity of guidance fixed points was not an issue in this dissertation. Note,

however, that if multiple fixed points could be found, it would be possible to rank them on the basis of exogenous criteria (e.g. total travel time minimization or pollution reduction).

Computational issues Another set of issues relates to the practical solution of RGG models. It is clear that solution methods for these models are computationally intensive; yet guidance, to be useful, must be timely. While there are many time-related tradeoffs that must be made in implementing an operational system (see Section 2.3.6, for example), faster computation of guidance solutions will always be advantageous. In this regard, parallel or distributed computer architectures are of considerable interest. A number of deterministic fixed point solution methods have been developed specifically for such architectures [Bertsekas and Tsitsiklis, 1997], but methods applicable to quasi-deterministic problems appear to have received less attention in the literature. The Gibbs sampling method lends itself directly to parallel processing.

Stochasticity The appropriate handling of fully stochastic simulator outputs is a major question raised by this research. It was seen that, in the presence of significant model stochasticity, replications of the solution process can vary so substantially that guidance based on one replication is useless (or worse) for conditions obtained in another. At least two distinct approaches may be effective in addressing this problem.

The first approach utilizes information about the time-dependent probability distribution of problem variables (link traversal times, for example) to compute adaptive guidance rules (guidance strategies). The flexibility made possible by adaptive guidance results in superior performance compared to fixed guidance. Adaptive time-minimizing prescriptive guidance, for example, would compute a set of conditional path recommendations based on the time of arrival at intermediate nodes rather than a fixed path, and would result in lower average travel time. Algorithms to compute minimum expected time strategies (time-dependent decision rules at each node) in stochastic time-dependent networks have been proposed by Hall [1986], Fu and Rilett [1998], Miller-Hooks and Mahmassani [2000] and Chabini [2000] and are applicable to this problem, provided that the link traversal time stochastic process can be approximated. This is feasible with the simple simulator's dynamic network loader, but computationally very expensive.

The second approach is to use local feedback control to stabilize the traffic system in a desired way. Local feedback regulation, using set points determined by a higher-level consistent guidance computation, provides the possibility of rapid reaction to stochastic fluctuations. Papageorgiou [1990] and his co-workers have extensively investigated approaches based on this and similar ideas.

Extensions Although the thesis has considered route guidance only, the ideas extend naturally to guidance on the full range of travel decisions (e.g. whether or not to travel, at what time, to what destinations and by what modes). The basic difference is that the time-dependent OD matrix, which was here considered fixed, becomes variable in these other problems.

The discussion above has indicated how the computation of consistent anticipatory guidance can account for the particular guidance system in place. This capability then makes it possible to consider the problem of designing an anticipatory guidance system subject to a budget constraint. The guidance system design problem involves many interrelated parameters: the nature of the guidance information to be disseminated; the communications technology (e.g. public or restricted

access, long-range or short-range, etc.); locations where short-range information sources should be positioned; system operations characteristics; etc.

There are numerous similarities between this problem and the network design problem. Just as practitioners use traffic assignment models in the evaluation of network investment options, guidance generation methods could be of use in evaluating alternative guidance system designs. And just as the rigorous analysis of the network equilibrium problem led to comparably rigorous investigation of the equilibrium-based network design problem, better understanding of the route guidance generation problem may eventually lead to a more precise treatment of the guidance system design problem.

The thesis has utilized a within-day problem setting. An extension of the work would involve the generalization of this setting to incorporate day-to-day learning. Current models (e.g. Cascetta [1989]) postulate that drivers adapt their behavior based on differences observed in prior days between expected and actual network flows and costs. When guidance is included in the model, a driver's adaptation would also be based on an assessment of the value of guidance on prior trips, and would lead to a decision about how much weight to accord to guidance messages. Such an extension is a major research topic, but would ultimately lead to a very general framework for incorporating information and experience into traffic prediction and guidance generation models.

Bibliography

Heydayat Z. Aashtiani. *The Multi-Modal Traffic Assignment Problem.* PhD thesis, Sloan School of Management, Massachusetts Institute of Technology, May 1979.

Sami Naguib Abdelhamid. Transformation of observations in stochastic approximation. *Annals of Statistics*, 1(6):1158–1174, November 1973.

Dan Anbar. On optimal estimation methods using stochastic approximation procedures. *Annals of Statistics*, 1(6):1175–1184, November 1973.

M. Bacharach. Matrix rounding problems. *Management Science*, 9:732–742, 1966.

Jean-Bernard Baillon. Un théorème de type ergodique pour les contractions non linéaires dans un espace de hilbert. *Computes rendus hebdomadaires des séances de l'Académie des Sciences*, 280–Série A(22):1511–1514, June 1975.

J. A. Bather. Stochastic approximation: A generalisation of the Robbins-Monro procedure. In P. Mandl and M. Hušková, editors, *Proceedings of the Fourth Prague Symposium on Asymptotic Statistics*, pages 13–27. Charles University, 1989.

Moshe Ben-Akiva, Michel Bierlaire, Jon Bottom, Haris Koutsopoulos, and Rabi Mishalani. Development of a route guidance generation system for real-time application. In Papageorgiou and Pouliezos [1997], pages 433–438.

Moshe Ben-Akiva, André de Palma, and Isam Kaysi. Dynamic network models and driver information systems. *Transportation Research A*, 25A(5):251–266, 1991.

Moshe Ben-Akiva, André de Palma, and Isam Kaysi. The impact of predictive information on guidance efficiency: An analytical approach. In Lucio Bianco and Paolo Toth, editors, *Advanced Methods in Transportation Analysis*, pages 413–432. Springer-Verlag, 1996.

David Bernstein and Tony E. Smith. Equilibria for networks with lower semi-continuous costs: With an application to congestion pricing. *Transportation Science*, 28(3):221–235, August 1994.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Athena Scientific, 1997.

Julian Besag. Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal of the Royal Statistical Society*, 36(2):192–236, 1974.

Julius R. Blum. Multidimensional stochastic approximation methods. *Annals of Mathematical Statistics*, 25(4):737–744, December 1954.

A. Bolelli, V. Mauro, and E. Perono. Models and strategies for dynamic route guidance. Part B: A decentralized, fully dynamic, infrastructure supported route guidance. In *Advanced Telematics in Road Transport. Proceedings of the DRIVE Conference, Brussels*, pages 99–105, 1991.

A. Bolelli and K. Rutley. The VAMOS White Book for variable message signs applications. In *Advanced Telematics in Road Transport. Proceedings of the DRIVE Conference, Brussels.*, pages 134–147, 1991.

Kim C. Border. *Fixed Point Theorems with Applications to Economics and Game Theory.* Cambridge University Press, 1985.

Jon Bottom, Moshe Ben-Akiva, Michel Bierlaire, Ismail Chabini, Haris Koutsopoulos, and Qi Yang. Investigation of route guidance generation issues by simulation with DynaMIT. In Avishai Ceder, editor, *Proceedings of the 14th International Symposium on Transportation and Traffic Theory*, pages 577–600. Pergamon, 1999.

Piet H. L. Bovy and Nanne J. van der Zijpp. Advanced traveler information services in real-time traffic prediction models. *Tijdschrift Vervoerwetenschap*, 4 (WCTR issue):381–394, 1995.

Piet H. L. Bovy and Nanne J. van der Zijpp. The close connection between dynamic traffic management and driver information systems. In Richard Emmerink and Peter Nijkamp, editors, *Behavioural and Network Impacts of Driver Information Systems*, pages 355–370. Ashgate Publishing Company, 1999.

D. Braess and G. Koch. On the existence of equilibria in asymmetrical multiclass-user transportation networks. *Transportation Science*, 13(1):56–63, February 1979.

George W. Brown. Iterative solution of games by fictitious play. In Tjalling C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 374–376. Cowles Commission for Research in Economics, John Wiley & Sons, Inc., 1951.

J. E. Burrell. Multipath route assignment and its applications to capacity restraint. In *Proceedings of the 4th International Symposium on the Theory of Road Traffic Flow*, 1968.

G. E. Cantarella. A general fixed-point approach to multi-mode multi-user equilibrium assignment with elastic demand. *Transportation Science*, 31(2):107–128, May 1997.

G. E. Cantarella and E. Cascetta. Dynamic processes and equilibrium in transportation networks: Towards a unifying theory. *Transportation Science*, 29(4):305–329, November 1995.

Malachy Carey. Nonconvexity of the dynamic traffic assignment problem. *Transportation Research B*, 26B(2):127–133, 1992.

E. Cascetta, G. E. Cantarella, and M. di Gangi. Evaluation of control strategies through a doubly dynamic assignment model. *Transportation Research Record*, 1306:1–13, 1991.

Ennio Cascetta. A stochastic process approach to the analysis of temporal dynamics in transportation networks. *Transportation Research B*, 23B(1):1–17, 1989.

Ennio Cascetta and Giulio Erberto Cantarella. A day-to-day and within-day dynamic stochastic assignment model. *Transportation Research A*, 25A(5):277–291, 1991.

Ennio Cascetta, Agostino Nuzzolo, Francesco Russo, and Antonino Vitetta. A modified logit route model overcoming path overlapping problems: Specification and some calibration results for interurban networks. In Lesort [1996], pages 697–711.

Ennio Cascetta and Maria Nadia Postorino. Fixed point models for the estimation of OD matrices using traffic counts on congested networks. submitted to *Transportation Science*, 1998.

Ennio Cascetta, Francesco Russo, and Antonino Vitetta. Stochastic user equilibrium assignment with explicit path enumeration: Comparison of models and algorithms. In Papageorgiou and Pouliezos [1997], pages 1078–1084.

Ismail Chabini. A new algorithm for shortest paths in discrete dynamic networks. In Papageorgiou and Pouliezos [1997], pages 551–556.

Ismail Chabini. Minimum expected travel times in stochastic time-dependent networks, revisited. submitted to *Transportation Science*, July 2000.

Kan Chen and Steven E. Underwood. Research on anticipatory route guidance. In VNIS91 VNIS91, pages 427–439.

K. L. Chung. On a stochastic approximation method. *Annals of Mathematical Statistics*, 25(3): 463–483, September 1954.

Lawrence H. Cox. A constructive procedure for unbiased controlled rounding. *Journal of the American Statistical Association*, 82(398):520–524, June 1987.

Lawrence H. Cox and Lawrence R. Ernst. Controlled rounding. *INFOR*, 20(4):423–432, November 1982.

Stella C. Dafermos. An extended traffic assignment model with applications to two-way traffic. *Transportation Science*, 5(4):366–389, November 1971.

Stella C. Dafermos and Frederick T. Sparrow. The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards - B. Mathematical Sciences*, 73B (2):91–118, April-June 1969.

Carlos F. Daganzo. Stochastic network equilibrium with multiple vehicle types and asymmetric, indefinite link cost Jacobians. *Transportation Science*, 17(3):282–300, August 1983.

Carlos F. Daganzo and Yosef Sheffi. On stochastic models of traffic assignment. *Transportation Science*, 11(3):253–274, August 1977.

Partha Dasgupta and Eric Maskin. The existence of equilibrium in discontinuous economic games, I: Theory. *Review of Economic Studies*, 53:1–26, 1986a.

Partha Dasgupta and Eric Maskin. The existence of equilibrium in discontinuous economic games, II: Applications. *Review of Economic Studies*, 53:27–41, 1986b.

Gary A. Davis and Nancy L. Nihan. Large population approximations of a general stochastic traffic assignment model. *Operations Research*, 41(1):169–178, 1993.

Marino de Luca. Generalized quasi-variational inequalities and traffic equilibrium problem. In F. Giannessi and A. Maugeri, editors, *Variational Inequalities and Network Equilibrium Problems*, pages 45–54. Plenum Press, 1995. Proceedings of the International School of Mathematics "G. Stampacchia" 19th course on Variational Inequalities and Network Equilibrium Problems, held June 19–25, 1994 in Erice, Italy.

P. R. de Waal, A. G. Steenbeek, and J. H. van Schuppen. Annex J: Simulation of a routing control algorithm for motorway networks. In *DACCORD Deliverable D06.3: Off-line Simulation Results*. Hague Consulting Group, The Hague, March 1999. Version 2.0.

Shantayanan Devarajan. A note on network equilibrium and noncooperative games. *Transportation Research B*, 15B(6):421–426, 1981.

Friedrich Dürrenmatt. *The Physicists*. Grove Press, New York, 1964. James Kirkup, translator.

Leonid Engelson. Self-fulfilling and recursive forecasts—an analytical perspective for driver information systems. In *Proceedings of the 8th IATBR Meeting*, Austin, Texas, 1997.

Vaclav Fabian. On asymptotic normality in stochastic approximation. *Annals of Mathematical Statistics*, 39(4):1327–1332, August 1968.

Vaclav Fabian. Asymptotically efficient stochastic approximation; the RM case. *Annals of Statistics*, 1(3):486–495, May 1973.

Caroline Fisk. A nonlinear equation framework for solving network equilibrium problems. *Environment and Planning A*, 16:67–80, 1984.

Caroline Fisk and Sang Nguyen. Existence and uniqueness properties of an asymmetric two-mode equilibrium model. *Transportation Science*, 15(4):318–328, November 1981.

Michael Florian. A traffic equilibrium model of travel by car and public transit modes. *Transportation Science*, 11(2):166–179, May 1977.

Edward W. Frees and David Ruppert. Estimation following a sequentially designed experiment. *Journal of the American Statistical Association*, 85(412):1123–1129, December 1990.

Liping Fu and L. R. Rilett. Expected shortest paths in dynamic and stochastic traffic networks. *Transportation Research B*, 32(7):499–516, September 1998.

Robert G. Gallager. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, COM-25(1):73–85, January 1977.

Alfredo Garcia, Daniel Reaume, and Robert L. Smith. Fictitious play for finding system optimal routings in dynamic traffic networks. *Transportation Research B*, 34:147–156, 2000.

Christian Gawron. An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. *International Journal of Modern Physics C*, 9(3):393–407, 1998.

D. C. Gazis. Modeling and optimal control of congested transportation systems. *Networks*, 4: 113–124, 1974.

Alan E. Gelfand and Adrian F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, June 1990.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6: 721–741, 1984.

John Geweke. Monte Carlo simulation and numerical integration. In Hans M. Amman, David A. Kendrick, and John Rust, editors, *Handbook of Computational Economics*, number 13 in Handbooks in Economics, chapter 15. Elsevier, 1996.

Glenn Gould. *Bach: The Six Partitas.* Columbia Masterworks stereophonic record M2S693, 1963. Liner notes include "A Conversation with Glenn Gould", an interview by David Johnson.

Randolph W. Hall. The fastest path through a network with random time-dependent travel times. *Transportation Science*, 20(3):182–188, August 1986.

Patrick T. Harker and Jong-Shi Pang. Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications. *Mathematical Programming*, 48:161–220, 1990.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(2): 100–107, July 1968.

Alain Haurie and Patrice Marcotte. On the relationship between Nash-Cournot and Wardrop equilibria. *Networks*, 15:295–308, 1985.

Alain Haurie and Patrice Marcotte. A game-theoretic approach to network equilibrium. *Mathematical Programming Study*, 26:252–255, 1986.

Yaser E. Hawas and Hani S. Mahmassani. A decentralized scheme for real-time route guidance in vehicular traffic networks. In *Proceedings of the Second World Conference on ITS, Volume IV*, pages 1956–1963, 1995.

Martin L. Hazelton. Some remarks on stochastic user equilibrium. *Transportation Research B*, 32(2):101–108, 1998.

Martin L. Hazelton, Seungjae Lee, and John W. Polak. Stationary states in stochastic process models of traffic assignment: A Markov chain Monte Carlo approach. In Lesort [1996], pages 341–357.

Jimi Hendrix. *The Wind Cries Mary.* Foxy Lady Music Company, 1967.

Serge P. Hoogendoorn and Piet H. L. Bovy. Genetic optimization of the routing control problem. In Piet H. L. Bovy, editor, *Motorway Traffic Flow Analysis—New Methodologies and Recent Empirical Findings*, pages 287–311. Delft University Press, 1998.

David E. Kaufman, Robert L. Smith, and Karl E. Wunderlich. An iterative routing/assignment method for anticipatory real-time route guidance. In VNIS91 VNIS91, pages 693–700.

David E. Kaufman, Robert L. Smith, and Karl E. Wunderlich. User-equilibrium properties of fixed points in iterative dynamic routing/assignment methods. *Transportation Research C*, 6:1–16, 1998.

Isam Kaysi, Moshe Ben-Akiva, and André de Palma. Design aspects of advanced traveler information systems. In Nathan H. Gartner and Gennaro Improta, editors, *Urban Traffic Networks: Dynamic Flow Modeling and Control*, pages 59–81. Springer-Verlag, 1995.

Isam Kaysi, Moshe Ben-Akiva, and Haris Koutsopoulos. Integrated approach to vehicle routing and congestion prediction for real-time driver guidance. *Transportation Research Record*, 1408: 66–74, 1993.

Isam Adnan Kaysi. *Framework and Models for the Provision of Real-time Driver Information.* PhD thesis, Department of Civil Engineering, Massachusetts Institute of Technology, February 1992.

Terence Kelly and Kai Nagel. Relaxation criteria for iterated traffic simulations. Unclassified Report 97-4453, Los Alamos National Laboratory, 1997.

Gotz Kersting. Almost sure approximation of the Robbins-Monro process by sums of independent random variables. *Annals of Probability*, 5(6):954–965, December 1977.

J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23(3):462–466, September 1952.

Kiyoshi Kobayashi. Incomplete information and logistical network equilibria. In Å. E. Andersson, D. F. Batten, K. Kobayashi, and K. Yoshikawa, editors, *The Cosmo-Creative Society: Logistical Networks in a Dynamic Economy*, pages 95–119. Springer-Verlag, 1993.

Kiyoshi Kobayashi. Information, rational expectations and network equilibria—an analytical perspective for route guidance systems. *Annals of Regional Science*, 28:369–393, 1994.

Kiyoshi Kobayashi and Hirokazu Tatano. Information and rational expectations in modelling driver information systems: A welfare measurement. In Richard Emmerink and Peter Nijkamp, editors, *Behavioural and Network Impacts of Driver Information Systems*, chapter 4, pages 69–92. Ashgate Publishing Company, 1999.

Harold W. Kuhn. How to compute economic equilibria by pivotal methods. In Jerzy Łoś and Maria W. Łoś, editors, *Computing Equilibria: How and Why*, pages 21–38. North-Holland Publishing Company, 1976.

Harold W. Kuhn. Pathfix: An algorithm for computing traffic equilibria. In *Proceedings of the 1977 IEEE Conference on Decision & Control*, volume 1, pages 831–834. IEEE Control Systems Society, 1977.

Harold J. Kushner. General convergence results for stochastic approximations via weak convergence theory. *Journal of Mathematical Analysis and Applications*, 61:490–503, 1977.

Harold J. Kushner and Dean S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems.* Number 26 in Applied Mathematical Sciences. Springer-Verlag, 1978.

Harold J. Kushner and Jichuan Yang. Stochastic approximation with averaging of the iterates: Optimal asymptotic rate of convergence for general processes. *SIAM Journal of Control and Optimization*, 31(4):1045–1062, July 1993.

Harold J. Kushner and Jichuan Yang. Stochastic approximation with averaging and feedback: Rapidly convergent "on line" algorithms. *IEEE Transactions on Automatic Control*, AC-40 (1):24–34, January 1995.

Harold J. Kushner and G. George Yin. *Stochastic Approximation Algorithms and Applications.* Number 35 in Applications of Mathematics—Stochastic Modeling and Applied Probability. Springer-Verlag, 1997.

T. L. Lai and Herbert Robbins. Adaptive design and stochastic approximation. *Annals of Statistics*, 7(6):1196–1221, November 1979.

Jean-Baptiste Lesort, editor. *Proceedings of the 13th International Symposium on Transportation and Traffic Theory*, 1996.

Fabien M. Leurent. Path-storing equilibration algorithms for several traffic assignment models. In Yorgos J. Stephanedes and Franceso Filippi, editors, *Proceedings of the 4th International Conference on Applications of Advanced Technologies in Transportation Engineering*, pages 633–638, June 1995.

J. Lindley. Urban freeway congestion: Quantification of the problem and effectiveness of potential solutions. *ITE Journal*, 1987.

Lennart Ljung. Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, AC-22(4):551–575, August 1977.

Lennart Ljung. Strong convergence of a stochastic approximation algorithm. *Annals of Statistics*, 6(3):680–696, May 1978.

Thomas L. Magnanti and Georgia Perakis. Averaging schemes for variational inequalities and systems of equations. *Mathematics of Operations Research*, 22(3):568–587, 1997a.

Thomas L. Magnanti and Georgia Perakis. Solving variational inequality and fixed point problems by averaging and optimizing potentials. Report OR 324-97, Operations Research Center, M.I.T., 1997b.

M. J. Maher. SAM—a stochastic assignment model. In *Mathematics in Transport Planning and Control*, volume 38 of *The Institute of Mathematics and its Applications Conference Series*, pages 121–132. Clarendon Press, 1992.

M. J. Maher and P. C. Hughes. A probit-based stochastic user equilibrium assignment model. *Transportation Research B*, 31(4):341–355, 1997.

Hani S. Mahmassani, Ta-Yin Hu, Srinivas Peeta, and Athanasios Ziliaskopoulos. Development and testing of dynamic traffic assignment and simulation procedures for ATIS/ATMS applications. Technical Report DTFH61-90-R–0074-FG, Center for Transportation Research, University of Texas at Austin, 1994.

Hani S. Mahmassani and R. Jayakrishnan. System performance and user response under real-time information in a congested traffic corridor. *Transportation Research A*, 25A(5):293–307, 1991.

Said Mammar, Albert Messmer, Peder Jensen, Markos Papageorgiou, Habib Hadj-Salem, and Lone Jensen. Automatic control of variable message signs in Aalborg. *Transportation Research C*, 4 (3):131–150, 1996.

Vito Mauro. Advanced traffic management and guidance: Experimental results from the Torino 5T scheme. In *Proceedings of the Tristan-III Conference*, San Juan, Puerto Rico, 1998.

Albert Messmer and Markos Papageorgiou. METANET: A macroscopic simulation program for motorway networks. *Traffic Engineering and Control*, pages 466–470, September 1990.

Albert Messmer and Markos Papageorgiou. Automatic control methods applied to freeway network traffic. *Automatica*, 30(4):691–702, 1994.

Albert Messmer, Markos Papageorgiou, and Neil Mackenzie. Automatic control of variable message signs in the interurban Scottish highway network. *Transportation Research C*, 6:173–187, 1998.

Elise D. Miller-Hooks and Hani S. Mahmassani. Least expected time paths in stochastic, time-varying transportation networks. *Transportation Science*, 34(2):198–215, May 2000.

Dov Monderer and Lloyd S. Shapley. Fictitious play property for games with identical interests. *Journal of Economic Theory*, 68(1):258–265, January 1996.

John F. Muth. Rational expectations and the theory of price movements. *Econometrica*, 29(3): 315–335, July 1961.

Kai Nagel. Experiences with iterated traffic microsimulations in Dallas. In M. Schreckenberg and D. E. Wolf, editors, *Proceedings of the Workshop on Traffic and Granular Flow*. Gerhard-Mercator-Universität, Duisburg, Germany, Springer-Verlag, 1997.

Kai Nagel and Christopher L. Barrett. Using microsimulation feedback for trip adaptation for realistic traffic in Dallas. Unclassified Report 97-1334, Los Alamos National Laboratory, March 1997.

Kai Nagel, Martin Pieck, Patrice M. Simon, and Marcus Rickert. Comparison between three different traffic micro-simulations and reality in Dallas. Unclassified Report 98-2944, Los Alamos National Laboratory, 1998.

M. S. Nargundkar and Walt Saveland. Random-rounding to prevent statistical disclosures. In *Proceedings of the Social Statistics Section*, pages 382–385. American Statistical Association, 1972.

Radford M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, September 1993.

M. B. Nevelson and R. Z. Hasminskii. *Stochastic Approximation and Recursive Estimation*, volume 47 of *Translations of Mathematical Monographs*. American Mathematical Society, 1973. Translated from the Russian.

OECD. Dynamic traffic management in urban and suburban road systems. Technical report, Organization for Economic Cooperation and Development, 1987.

M. Papageorgiou and A. Pouliezos, editors. *Proceedings of the 8th IFAC Symposium on Transportation Systems*, Chania, Greece, June 1997.

Markos Papageorgiou. Dynamic modeling, assignment, and route guidance in traffic networks. *Transportation Research B*, 24B(6):471–495, 1990.

Markos Papageorgiou and Albert Messmer. Dynamic network traffic assignment and route guidance via feedback regulation. *Transportation Research Record*, 1306:49–58, 1991.

Michael Patriksson. *The Traffic Assignment Problem: Models and Methods*. VSP BV, The Netherlands, 1994.

Yannis Pavlis and Markos Papageorgiou. Simple decentralized feedback strategies for route guidance in traffic networks. *Transportation Science*, 33(3):264–278, August 1999.

B. T. Polyak. New method of stochastic approximation type. *Automation and Remote Control*, 51(7):937–946, July 1990.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal of Control and Optimization*, 30(4):838–855, July 1992.

Warren B. Powell, Patrick Jaillet, and Amedeo Odoni. Stochastic and dynamic networks and routing. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, chapter 3, pages 141–295. Elsevier, 1995.

Roy Radner. Equilibrium under uncertainty. In Kenneth J. Arrow and Michael D. Intriligator, editors, *Handbook of Mathematical Economics*, volume 2, chapter 20, pages 923–1006. North-Holland Publishing Company, 1982.

Marcus Rickert and Kai Nagel. Issues of simulation-based route assignment. Unclassified Report 98-4601, Los Alamos National Laboratory, December 1998.

H. Robbins and D. Siegmund. A convergence theorem for non-negative almost positive supermartingales and some applications. In J. S. Rustagi, editor, *Optimizing Methods in Statistics*, pages 237–257. Academic Press, 1971.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Julia Robinson. An iterative method of solving a game. *Annals of Mathematics*, 54(2):296–301, September 1951.

R. W. Rosenthal. The network equilibrium problem in integers. *Networks*, 3:53–59, 1973.

David Ruppert. Almost sure approximations to the Robbins-Monro and Kiefer-Wolfowitz process with dependent noise. *Annals of Probability*, 10(1):178–187, February 1982.

David Ruppert. Efficient estimators from a slowly-convergent Robbins-Monro process. Technical Report 781, School of Operations Research and Industrial Engineering, Cornell University, 1988.

David Ruppert. Stochastic approximation. In B. K. Ghosh and P. K. Sen, editors, *Handbook of Sequential Analysis*, chapter 22, pages 503–529. Marcel Dekker, Inc., 1991.

Jerome Sacks. Asymptotic distribution of stochastic approximation procedures. *Annals of Mathematical Statistics*, 29(2):373–405, June 1958.

Herbert Scarf. The approximation of fixed points of a continuous mapping. *SIAM Journal of Applied Mathematics*, 15(5):1328–1343, September 1967.

Rainer Schwabe and Harro Walk. On a stochastic approximation procedure based on averaging. *Metrika*, 44(2):165–180, 1996.

Yosef Sheffi and Warren B. Powell. An algorithm for the equilibrium assignment problem with random link times. *Networks*, 12:191–207, 1982.

Patrice M. Simon and Kai Nagel. Simple queuing model applied to the city of Portland. Unclassified Report 98-3903, Los Alamos National Laboratory, 1998.

S. A. Smulders. *Control of Freeway Traffic*. Number 80 in CWI Tracts. CWI, 1996.

Eliahu Stern and David Leiser. Levels of spatial knowledge and urban travel modeling. *Geographic Analysis*, 20(2):140–155, April 1988.

Martin A. Tanner and Wing Hung Wong. The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association*, 82(398):528–550, June 1987.

Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

Michael J. Todd. *The Computation of Fixed Points and Applications*. Number 124 in Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, 1976.

J. H. van Schuppen. Annex D: Routing control of motorway networks. In *DACCORD Deliverable D06.1: Co-ordinated Control Strategies*. Hague Consulting Group, The Hague, 1997.

J.A.C. van Toorenburg and R.J.P. van der Linden. Predictive control in traffic management. Technical report, Adviesdienst Verkeer en Vervoer (AVV), Ministry of Transport, Public Works and Water Management, Rotterdam, The Netherlands, March 1996.

J. H. Venter. An extension of the Robbins-Monro procedure. *Annals of Mathematical Statistics*, 38(1):181–190, February 1967.

VNIS91. *Vehicle Navigation & Information Systems Conference Proceedings (VNIS '91)*, Dearborn, Michigan, U.S.A., 1991.

R. von Tomkewitsch. ALI-SCOUT—a universal guidance and information system for road traffic. In *Proc. 2nd Int. Conf. On Road Traffic Control, 1986*, pages 22–25, 1986.

P. T. R. Wang, W. P. Niedringhaus, D. K. Codelli, and M. F. McGurrin. A comparison of travel time reduction using current *vs.* partially predictive travel times. In Lise Olaussen and Erik Helli, editors, *Conference Record of Papers presented at the 3rd Vehicle Navigation & Information Systems Conference (VNIS '92)*, pages 476–482, Oslo, Norway, 1992.

M. T. Wasan. *Stochastic Approximation.* Number 58 in Cambridge Tracts in Mathematics and Mathematical Physics. Cambridge University Press, 1969.

David Watling. Asymmetric problems and stochastic process models of traffic assignment. *Transportation Research B*, 30(5):339–357, 1996.

David Watling. A stochastic process model of day-to-day traffic assignment and information. In Richard Emmerink and Peter Nijkamp, editors, *Behavioural and Network Impacts of Driver Information Systems*, chapter 6, pages 115–139. Ashgate Publishing Company, 1999.

M. B. Wisten and M. J. Smith. Distributed computation of dynamic traffic equilibria. *Transportation Research C*, 5(2):77–93, 1997.

Karl Eric Wunderlich. *Link time prediction for dynamic route guidance in vehicular traffic networks.* PhD thesis, Department of Industrial and Operations Engineering, University of Michigan, 1994.

Samuel Yagar. Dynamic traffic assignment by individual path minimization and queuing. *Transportation Research*, 5(3):179–196, August 1971.

Zaifu Yang. *Computing Equilibria and Fixed Points: The Solution of Nonlinear Inequalities.* Kluwer Academic Publishers, 1999.