Modeling Railroad Terminal Operations: Supporting Real-Time Network Planning and Control

by

Andrew P. Armacost

B.S., Industrial Engineering, Northwestern University (1989)

Submitted to the Sloan School of Management
in Partial Fulfillment of the Requirements
of the Degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

at the

Massachusetts Institute of Technology

June 1995

Signature of Author______________________________
Sloan School of Management
Interdepartmental Program in Operations Research

Approved by [illegible]
Stephan E. Kolitz
The Charles Stark Draper Laboratory, Inc.
Technical Supervisor

Certified by [illegible]
Cynthia Barnhart
Assistant Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by [illegible]
Thomas L. Magnanti
Codirector, Operations Research Center

Modeling Railroad Terminal Operations: Supporting Real-Time Network Planning and Control

by

Andrew P. Armacost

B.S., Industrial Engineering, Northwestern University (1989)

Submitted to the Sloan School of Management
in Partial Fulfillment of the Requirements
of the Degree of Master of Science in Operations Research

Abstract

Since the early 1980's, the railroad industry has experienced a return to profitability. Cost-cutting measures undertaken in the wake of deregulation resulted in reductions of the labor force and the rail infrastructure. Recent studies have shown growing customer dissatisfaction with the poor reliability of rail freight transportation. This thesis proposes a hierarchical railroad planning system as a means for achieving greater reliability. In addition to the notional design of such a system, the thesis presents specific models of yard operations. These models are designed to operate within the broader context of *real-time* network planning and control. Two distinct modeling approaches are taken, a deterministic simulation to estimate yard performance and a two-machine sequencing model to plan yard activities. The real-time environment forces the use of models that rely upon fast heuristics. To consider the network effect of these models, a heuristic network coordination algorithm is developed. This heuristic derives network-wide schedules based upon outputs from the yard models. A demonstration of the models' performance is provided in a series of sample problems.

Thesis Advisor: Professor Cynthia Barnhart
Title: Assistant Professor of Civil and Environmental Engineering

Technical Supervisor: Stephan E. Kolitz, PhD.
The Charles Stark Draper Laboratory, Inc.

Acknowledgements

There are several persons to whom I owe a great deal of thanks. The first is Dr. Steve Kolitz of Draper Lab. His guidance and support during the creation of this thesis has been invaluable. I am thankful for the two years I have spent working with him and learning the subtleties of *real* systems.

Second, and an equally important influence on my research and on the work contained in this thesis, is Professor Cindy Barnhart of MIT. Her dedication to her students is remarkable. Without her help, this would not have been possible.

I am grateful to the faculty and fellow students of MIT's Operations Research Center and the members of Large Scale Optimization Group for two years of fine memories. I would specifically like to thank to Jim Christodouleas for two years of sharing in the joy and pain of our endeavors in the train business. Additionally, I would like to thank Tim Kniker for his comments and for a ceaseless passion for the world's best sport!

Next, I would like to recognize the Charles Stark Draper Laboratory and the Massachusetts Institute of Technology for providing this great educational opportunity. In addition, I would like to thank the U.S. Air Force Academy's Department of Management for their sponsorship. I hope my enthusiasm as a new faculty member will be a symbol of my gratitude.

I would like to thank my family, especially my parents for their love, support, and mentorship through this program and through life.

Finally and most importantly, I would like to thank my wife, Kathy, for her unending support during the last two years. Through it all, her love and patience were the calming influences that made this possible.

Table of Contents

Abstract ... 3

Acknowledgements ... 5

Table of Contents ... 7

List of Figures ... 10

List of Tables ... 12

1. Introduction ... 13

1.1. The Railroad Industry ... 14
1.1.1. Recent Growth Since Deregulation ... 14
1.1.2. Intermodal Service ... 17

1.2. The Reliability of Rail Freight ... 20
1.2.1. Network Delays ... 20
1.2.3. Yard Delays ... 20
1.2.2. Network Planning ... 21

1.3. Overview of this thesis ... 22

2. Tutorial on Terminal Processing ... 25

2.1 Physical Structure ... 26

2.2 Supporting Data for Freight Processing ... 27
2.2.1. The Waybill ... 28
2.2.2. The Wheel Report ... 28
2.2.3. The Consist Report ... 28
2.2.4. The Switch List ... 29

2.3 Processing Activities ... 29
2.3.1. Arrival and Receiving ... 29
2.3.2. Sorting ... 30
2.3.2.1 The Mechanism ... 31
2.3.2.2 Sorting Methodologies ... 32
2.3.3. Assembly and Departure ... 34
2.3.4. Sample Processing Rates - Radnor Yard Example ... 36

2.4 Variations ... 38
2.4.1. Physical Structure Variations ... 38
2.4.2. Traffic Variations ... 40
2.4.2.1 Local Freight ... 40
2.4.2.2 Set-off/Pick-up ... 40
2.4.2.3 Inspection-only ... 40
2.4.2.4 Pass-Through ... 41

2.5 Example Yard Layout - Selkirk Yard ... 41
2.5.1. Relationship to Network ... 41
2.5.2. Physical Structure ... 42
2.5.3. Processing of Freight ... 43

2.6 Chapter Summary ... 44

Table of Contents (cont.)

3. A Paradigm for Railroad Modeling 45

3.1. The Planning Taxonomy and Literature Review 45
3.1.1. Strategic Planning 46
3.1.1.1. Strategic Yard Models 47
3.1.1.2. Strategic Network Models 47
3.1.2. Tactical Planning 48
3.1.2.1. Tactical Yard Models 48
3.1.2.2. Tactical Network Models 50
3.1.3. Operational Planning and Control 52
3.1.3.1. Operational Yard Models 52
3.1.3.2. Operational Network Models 56
3.1.4. Literature Review Summary 58

3.2. Hierarchical Planning of a Railroad 60
3.2.1. Structuring a Sequence of Decisions 60
3.2.1.1. Scheduled Railroads 61
3.2.1.2. Demand-Driven Railroads 64
3.2.2. Operational Network Planning Hierarchy 65

3.3. Chapter Summary 70

4. Descriptive Yard Modeling: An Object Oriented Framework 71

4.1. Conceptual Design 72

4.2. Static Components - Object Oriented Design 73
4.2.1. Train Representation 74
4.2.2. Yard Representation 75

4.3. Dynamic Model - Object Oriented Deterministic Simulation 78
4.3.1. Inputs to Model 79
4.3.1.1. Car and Block List 81
4.3.1.2. Train Consist and Schedule 82
4.3.1.3. Yard Topology 83
4.3.1.4. Processing Characteristics 84
4.3.1.5. Yard Inventory 85
4.3.2. The Event List 87
4.3.3. Sequencing Algorithms 89
4.3.4. Running Time 97
4.3.5. Visualization 98

4.4. Extensions to the Framework 101
4.4.1. Pure Simulation 101
4.4.2. Algorithm Testbed 101
4.4.3. Real-Time Decision Support Tool 102

4.5. Chapter Summary 103

Table of Contents (cont.)

5. Modeling the Network 105

5.1. Decomposing the Network 106
5.1.1. Planning Nodes 107
5.1.2. Network Example 108
5.1.3. Object Oriented Representation of the Network 110
5.1.3.1. Planning Node Representation 111
5.1.3.2. Train Representation 112

5.2. The Network Planning Model 115
5.2.1. The Network Coordination Heuristic 116
5.2.2.1. 10 Node, 5 Train Example 118
5.2.2.2. 26 Node, 19 Train Example 124

5.3. Chapter Summary 130

6. A Yard Sequencing Model **131**

6.1. Notional Model 132

6.2. The Disassembly-Assembly Sequencing Model 134
6.2.1. The Basic Model 134
6.2.1.1. Step 1: Determine BlockDue Times 137
6.2.1.2. Step 2: Determine the Slack Lengths of Hump Job Sets 138
6.2.1.3. Step 3: Determine Hump Job Set Sequence by Modified EDD Rule 139
6.2.1.4. Step 4: Determine Inbound Train Hump Order 140
6.2.1.5. Step 5: Determine Assembly Start, Finish and Resulting Tardiness 141
6.2.2. Extensions to the Basic Model 146
6.2.3. High-Level Network Planning Revisited: Modifications to the Heuristic Coordinating Algorithm 154

6.3. Extensions for Yard Control Decision Support Tools 156

6.4. Chapter Summary 158

7. Summary and Future Research **161**

7.1. Summary of Thesis 161

7.2. Future Research 164

Appendix A - Selkirk Yard Layout **169**

Appendix B - Glossary of Terms for Object Oriented Modeling (OOM) **173**

Appendix C - Sample Input and Output Files for Yard Model **179**

Appendix D - Summary of Notation **187**

References **189**

List of Figures

Figure 2.1: Example Yard Layout 26
Figure 2.2: Profile of the Humping Process 31
Figure 2.3: Retarder and Group Layout in a Hump Yard 32
Figure 2.4: Layout of Bowl and Departure Yard, including pull-out leads 35
Figure 2.5: Yard Layout – Variation #1 38
Figure 2.6: Yard Layout – Variation #2 39
Figure 2.7: Selkirk Yard Relationship to Conrail Network 41
Figure 3.1: Assad's Taxonomy of Railroad Planning 46
Figure 3.2: Tactical Planning Sequence for Airlines 61
Figure 3.3: Hierarchical Planning in a Scheduled Railroad 63
Figure 3.4: Hierarchical Planning in a Demand-Driven Railroad 65
Figure 3.5: Operational Network Planning and Control 67
Figure 4.1: Data Flow Between Yard Model and Network Coordinator 73
Figure 4.2: Static Object Oriented Representation of Trains 74
Figure 4.3: Object Model for Processing Functions and Yard Resources 76
Figure 4.4: Object Model for ProcessNodes 77
Figure 4.5: Yard Model Algorithm 80
Figure 4.6: Class structure for Event 87
Figure 4.7: General Event Processing Subroutine 91
Figure 4.8: Assignment Algorithm for Receiving Track Assignment 92
Figure 4.9: Assignment Algorithm for Inspection Crew Assignment 92
Figure 4.10: Hump Assignment Algorithm 93
Figure 4.11: Single Train Assembly Algorithm 94
Figure 4.12: Multiple Train Assembly Algorithm 96
Figure 4.13: Visualization of static layout of a terminal 98
Figure 4.14a: Dynamic Model, Screen Dump #1 100
Figure 4.14b: Dynamic Model, Screen Dump #2 100
Figure 4.15: Yard Planning Algorithm Testbed-Conceptual Design 102
Figure 5.1: Network Planning Model 106
Figure 5.2: Translation of Physical Network to Planning Nodes 108
Figure 5.3: Physical Layout of Sample Class I Railroad Division 109
Figure 5.4: Planning Node Representation for Sample Class I Railroad Division 110
Figure 5.5: Object Model for Physical Network 111
Figure 5.6: Object Model for Associations between Trains and PlanningNodes 112

Figure 5.7: Object Representation—Single Train Moving Between Two Yards 113
Figure 5.8: Object Representation—Single Pass-Through Train 114
Figure 5.9: Object Representation—Two Connecting Trains 114
Figure 5.10: Implementation of the High-Level Planning Algorithm 115
Figure 5.11: Ten Node Topology 119
Figure 5.12: Initial schedule, 10 Node 5 Train Example 122
Figure 5.13: Final schedule, 10 Node 5 Train Example 123
Figure 5.14: 26 Node Network Topology 124
Figure 5.15: Initial schedule, 26 Node 19 Train Example 128
Figure 5.16: Initial schedule, 26 Node 19 Train Example 129
Figure 6.1: Generic Hump Yard Layout 133
Figure 6.2: Two-Machine Disassembly/Assembly Sequencing Model (DASM) 134
Figure 6.3: Network Planner using DASM 155
Figure 6.4: Yard Decision Support Model Using DASM 157

List of Tables

Table 1-1: Industry Overview—Class I Railroads, 1993 14
Table 1-2: Class I Railroads, 1992 - 1993, ($ in thousands) 15
Table 1-3: Class I Railroad Carloads 16
Table 1-4: US Freight Transportation Market, 1989 18
Table 1-5: Intermodal Traffic, 1992 - 1993 (# of trailers and containers) 19
Table 2-1: Inbound Arrival Schedule 33
Table 2-2: Outbound Train Consist and Blocks 34
Table 2-3: Summary of Hump Yard Processes 36
Table 2-4: Processing Times for Radnor Yard 37
Table 2-5: Summary of Yard Size 42
Table 3-1: Summary of Yard Models 58
Table 4-1: Summary of Classes derived from ProcessNode and their resources 75
Table 4-2: Summary of Event Processing for ProcessNodes 88
Table 5-1: Initial Schedule, 10 Node Network 119
Table 5-2: Arrival Time Output, 10 Node Network 120
Table 5-3: Final Schedule, 10 Node Network 121
Table 5-4: Initial Train Schedule for 26 Node Example 125
Table 5-5: Algorithm Arrival Time Output, 26 Node Example 126
Table 5-6: Final Train Schedule for 26 Node Example 127
Table 6-1: Outbound Train Schedule and Due Time Calculations (Step 1) 144
Table 6-2: Inbound Train Schedule and Slack Length Calculations (Step 2) 144
Table 6-3: Modified EDD Calculations (Step 3) 145
Table 6-4: Hump Completion Times for Each Block (Step 4) 145
Table 6-5: Block Completion Times (Step 5) and Tardiness Calculations 145
Table 6-6: Comparison of Related Objective Values (in minutes) 146
Table 6-7: Modified Block Assembly Sequence and Tardiness Calculations 150
Table 6-8: Comparison of Related Objective Values (in minutes) 151

1. Introduction

Stemming from its deregulation in 1980, the rail industry is once again profitable. Through massive cost-cutting measures, rail carriers are moving greater amounts of freight with fewer resources and making money doing so. Despite this success, rail carriers are still struggling with performance, both in terms of time and service reliability, and must improve if they are to maintain market share to ensure continued profits.

Efforts to improve service have traditionally taken a back seat to the cost-cutting measures. Recently, the focus has shifted toward service reliability. A fundamental approach for achieving this is through the development of better operating plans and schedules. Although rail planning problems have been researched for decades, it is a body of research that is sparse compared to other industries, such as the manufacturing sector or the airlines. This thesis places railroad planning research into a detailed hierarchy and develops one segment of this hierarchy, namely the operational, or day-to-day, planning of terminal (or yard) operations.

This chapter provides an introduction to the business of moving freight by rail, which includes the profitability of the rail industry since deregulation and the growth of the intermodal market. The second part of this chapter will introduce the problem of operational network planning, including the interaction between yard operations and line dispatching. Thus, having set the stage with both the business and technical motivations for the research contained herein, we will conclude with an outline of the thesis.

1.1. The Railroad Industry

The history of this industry in the United States spans nearly two centuries, but we will only highlight the last two decades. These twenty years mark the transition from a near-death industry propped up by government regulation to a fiercely competitive, yet profitable, industry vying for market share amongst itself and with other modes of transportation. Two fundamental areas provide a glimpse into the growth and outlook for future prosperity of the industry: the effect of deregulation upon the rail carriers and the growth of intermodal service.

1.1.1. Recent Growth Since Deregulation

The catalyst for dramatic change in the rail industry was the Staggers Rail Act, enacted in October, 1980. This legislation deregulated the railroad industry and forced carriers to make dramatic operating cost reductions in order to respond to competition in the free market. These consolidations took the form of labor,

Table 1-1: Industry Overview—Class I Railroads, 1993

Item	1993	1994 (Est.)*	1995 (Est.)
Operating Revenues (in billions)	$28.80	$30.30	$30.90
Year to Year % change	1.7%	5.2%	2.0%
Operating Profits	$4.31	$4.88	$5.10
Year to Year % change	42.2%	13.2%	4.5%
Operating Ratio	85.1%	83.9%	83.5%
Ton-miles (in billions)	1106.7	1155.0	1170.0
Year to Year % change	3.7%	4.4%	1.3%
Rail freight rates			
Year to Year % change	0.7%	1.0%	1.1%
Freight revenues per ton-mile (in cents)	2.53	2.55	2.57
Return on net investment	6.4%	7.8%	8.2%
Change in industrial output	4.2%	5.2%	2.2%

** 1994 actuals not available at time of publication*

Source: Standard & Poor's

Table 1-2: Class I Railroads, 1992 - 1993, ($ in thousands)

Major Railroads	1992 Revenue	1993 Revenue	% of Change	1992 NOI	1993 NOI	% of Change	1992 ROI	1993 ROI	Point Change
Union Pacific	$4,788,999	$4,856,068	+1.40%	$619,834	$564,419	-8.94%	10.53%	9.34%	-1.19
Burlington Northern	4,629,843	4,699,409	+1.50%	390,290	390,384	+0.02%	9.75%	9.36%	-0.39
CSX Transportation	4,433,719	4,380,304	-1.20%	(9,390)	323,396	NM	0.00%	4.66%	+4.66
Norfolk Southern	3,776,987	3,745,866	-0.82%	616,878	527,721	-14.45%	11.59%	9.72%	-1.87
Conrail	3,207,663	3,349,562	+4.42%	347,591	353,613	+1.73%	6.44%	6.65%	+0.21
Southern Pacific	2,878,000	2,918,600	+1.41%	55,800	(75,000)	NM	NA	0.00%	NM
Atchison, Topeka & Santa Fe	2,251,675	2,409,179	+6.99%	(64,154)	81,049	NM	0.00%	2.35%	+2.35
Totals or Averages (Major)	$25,966,866	$26,358,988	+1.51%	$1,956,849	$2,165,582	+10.67%	5.47%	6.01%	+0.54
Regional Railroads	**1992 Revenue**	**1993 Revenue**	**% of change**	**1992 NOI**	**1993 NOI**	**% of Change**	**1992 ROI**	**1993 ROI**	**Point Change**
Chicago and North Western	$816,456	$838,903	+2.75%	$97,991	$86,012	-12.22%	8.28%	7.18%	-1.10
Soo Line	576,788	593,323	+2.87%	(34,161)	12,313	NM	0.00%	1.78%	+1.78
Illinois Central Railroad	547,436	564,653	+3.15%	112,573	122,446	+8.77%	14.54%	14.74%	+0.20
Kansas City Southern	335,484	345,526	+2.99%	41,161	51,305	+24.64%	8.86%	10.53%	+1.67
Grand Trunk Western	263,745	299,490	+13.55%	(77,191)	(11,665)	NM	0.00%	0.00%	0.00
Totals or Averages (Reg'n'l)	2,539,909	$2,641,895	+4.02%	$140,373	$260,411	+85.51%	4.53%	4.89%	+0.36

NOI: Net Operating Income
ROI: Return On Investment

Source: Association of American Railroads

equipment, and infrastructure reductions, as well as fundamental shifts in the way that the railroads did business.

In 1993, railcars owned by Class I railroads numbered 1.17 million, down from the 1.8 million cars owned in 1964. The volume moved in these cars in 1994, 1.45 billion tons, remained roughly at the level of 1973, 1.53 billion tons. Yet, because of more efficient, but more expensive, truckload (TL) and less-than-truckload (LTL) service, much of the short-haul business went to motor carriers. Thus, the more telling measures are ton-miles, which increased roughly 37 percent over the same period. We have also seen increases in the total number of all carloads, which rose 4.8 percent between 1992 and 1993 and continued to rise through the first 48 weeks of 1994, at which point total carloads exceeded the 48 week total from the previous year by 5.8 percent.

Track ownership by the Class I railroads also dropped. During the period from 1980 to 1993, the carriers reduced track ownership by roughly 53,000 miles. Around 33,000 miles of track were abandoned and 20,000 converted or transitioned to short line operations. Track conversion provides two benefits, the im-

Table 1-3: Class I Railroad Carloads

Class I Railroads	1992 Total	1993 Total	92-92 % Change
Atchison, Topeka & Santa Fe	1,076,666	1,166,762	+8.4
Burlington Northem	3,079,421	3,075,494	-0.1
Chicago & North Westem	1,528,304	1,685,326	+10.3
Conrail	2,507,110	2,565,377	+2.3
CSX Transportation	4,315,132	4,209,985	-2.4
Grand Trunk Westem	408,834	433,429	+6.0
Illinois Central	783,223	758,837	-0.4
Kansas City Southem	435,134	489,574	+12.5
Norfolk Southem	3,021,536	3,028,170	+0.2
Soo Line	644,189	657,588	+2.1
Southem Pacific	1,433,938	1,464,146	+2.1
Union Pacific	3,217,121	3,322,503	+3.3
Total	22,450,608	22,857,191	+1.8

Source: Association of American Railroads

mediate being the proceeds from its sale. The longer term benefit is the increased traffic arriving on the main line from these new feeder lines. The Class I railroads began to realize the benefit during the mid to late 1980s, when the rate of track abandonment began to tail-off, bringing the total track ownership of railroads to 112,000 miles in 1993.

Reductions also hit the labor force. Over the period from 1964 to 1993, the number of train crew personnel fell 61 percent to 71,100. Much of this reduction came as the result of the reduction in line-haul crew size, including the elimination of the brakeman position. These personnel reductions were not limited to train crews. From 1980 to 1993, the total Class I work force was sheared from 460,000 to 193,000. In spite of this, the railroads keep moving larger numbers of carloads greater distances. The productivity of its employees, in the ten years prior to 1994, rose an astonishing 8.1 percent annually.

In spite of the increase in productivity and the increase in ton-miles and carloads, we have seen only a slight upswing in revenue. This has been primarily due to the lowering of shipping rates. Over the period from 1983 - 1993, rates have dropped, after adjusting for inflation, 20 percent. In 1993, the rate for rail averaged 2.5 cents per ton-mile, compared with 22 cents for LTL and 12 cents for TL. The return to profitability, therefore, stems from the post-deregulation cost-cutting measures, since net operating income has grown at a pace well beyond the minor growth in revenue. Future success in the railroad industry must arrive in the form of increased market share. Low prices will attract the shippers and reliability will keep them, as predicted in Standard and Poor's 1993 Industry Report: "Despite the cheap rates, railroads will be unable to retain customers unless they can deliver reliable, quality service." The outlook is for continued growth in the industry, with much of it from intermodal service. The next section gives a brief overview of intermodal and its impact upon the railroads, but the long-term commitment of shippers to intermodal will depend on the commitment of the railroads to increase service performance.

1.1.2. Intermodal Service

During the eighties, as the rail carriers were cutting costs and becoming profitable, motor carriers also experienced unprecedented growth. Concurrent with the Staggers Act was the Motor Carrier Act, which deregulated the trucking industry. This put the two modes of ground freight transportation industry in competition with each other. For the shipper who required fast, highly reliable, point-to-point shipping of non-bulk commodities, motor carriers provided the answer. Shipping by rail was less certain, and much of the additional volume being shipped during the 1980s found its way into the trailer of a motor carrier instead of the railcar.

Voigtlaender [1994] cites, as a long-term trend over the period 1929 to 1992, a reduction in railroad's market share from 75% to 37%, while the market share of the motor carriers rose from 3% to over 27%. This market share is expressed in terms of ton-miles. The more drastic comparison comes in terms of *revenue*. In 1989, (see Table 1-4) motor carriers captured 79% of freight transportation revenue, while rail carriers captured only 9%. This difference stems from the disparity in shipping rates, as mentioned in the previous section.

More recently, motor and rail carriers are finding themselves in more of a position of cooperation, with each other and with shippers. In an interview with *Distribution* magazine (July 1994), Susan Chapman of Walsh, Kelly and Co., states, "It's not like the old days when railroads used to be at war with the major shippers. Now, they think of shippers as partners." No better example of this spirit of cooperation can be shown than the joint venture launched by Santa Fe in the early 1990's with both J.B. Hunt and Schneider National, in an effort to pro-

Table 1-4: US Freight Transportation Market, 1989

	Revenue ($ billion)	Percentage of Total
Motor Carriers		
Public and for-hire	71.2	22
Private and for own account	81.3	25
Local freight services	101.6	31
	245.1	79
Other Carriers		
Railroads	29.6	9
Water carriers	20.2	6
Oil Pipelines	8.5	3
Air carriers	11.2	3
	69.5	21
Total Freight Revenue	323.6	100

Source: Robert V. Delaney, "Trends in Logistics and US World Competitiveness" Transportation Quarterly, Jan 91

vide seamless intermodal service to their customers. Another set of factors driving the reliance of the trucking industry upon rail carriers is the trucking industry's shortage of drivers, high-turnover of drivers, and increasing fuel cost.

Though intermodal does not register much weight in the tonnage or the ton-mile metrics of rail freight, it represented 21 percent of the revenues for the Class I railroads in 1993. The number of trailer and container shipments during the first 48 weeks of 1994 exceeded 7.5 million, an increase of 14 percent over the previous year. Freight car loadings, totaling 16.5 million, showed an increase of only 5.8 percent. The longer-term growth trend for intermodal, in terms of volume, was at a annual rate of 6.2 percent during the eighties and a more modest 5.7 percent during the nineties. Conservatively projecting the lower annual rate until the end of the century, we can expect annual shipments of over 11 million units via intermodal rail service.

The increased number of cars being moved, along with the sensitivity of intermodal shippers to service reliability, really focus the importance of on-time performance by the rail carriers. Voigtlaender cites several studies which indicate that *service* is the most important criterion for carrier selection, with price of service slightly behind. To underscore the importance of good service, he further

Table 1-5: Intermodal Traffic, 1992 - 1993 (# of trailers and containers)

Railroad	1992 Total	1993 Total	% Change
Conrail	1,230,989	1,372,787	+11.5
Union Pacific	1,269,955	1,346,450	+6.0
Atchison, Topeka & Santa Fe	1,147,536	1,218,054	+6.1
Southern Pacific	1,078,967	1,204,966	+11.7
Burlington Northern	1,045,218	1,064,331	+1.8
Norfolk Southern	910,423	992,850	+9.1
CSX Transportation*	753,915	807,698	+7.1
Chicago & North Western	682,976	729,685	+6.8
Florida East Coast	325,664	324,186	-0.5
Soo Line	200,617	209,992	+4.7
Illinois Central	82,898	87,264	+5.3
Kansas City Southern	67,353	63,113	-6.3
Grand Trunk Western	36,342	39,916	+9.8
Wisconsin Central	30,559	32,276	+5.6

** Figures do not include all of CSX Intermodal's traffic*
Source: AAR

summarizes a report which finds the elasticity of shippers' willingness to use intermodal service:

- 69% of intermodal users would increase their usage of intermodal shipping if 98% of all shipments were delivered on the scheduled day

- 65% of intermodal users would increase their usage of intermodal shipping if transit time were reduced by one day.

In summary, intermodal is becoming a larger portion of railroad revenues, second only to coal. The importance of service reliability is paramount for shippers continuing to use intermodal services. With greater numbers of railcars being used, capacity, both on the line and in the yard, soon disappears. Yet, before the railroads begin putting large sums of money in capital expansion, investors want to see better use of this existing capacity. The carriers, therefore, must balance the satisfaction of both their shippers, intermodal and general freight, and their financial backbone.

1.2. The Reliability of Rail Freight

The previous section addressed the recent trends in the general freight and intermodal rail industry. In order to lay the foundation for the remainder of this thesis, this section will briefly describe how well the carriers are actually moving the freight. A description of network planning will be presented both in Chapter 3 (A Paradigm for Railroad Modeling) and Chapter 5 (Modeling a Network). The material presented in this section is intended to help the reader comprehend the importance of yard operations in the context of network operations. This will lead to an understanding of the importance of yard modeling in the context of network planning and control.

1.2.1. Network Delays

The importance of service reliability is clear. A shipper's decision to use a service will depend upon the carrier's ability to consistently deliver on-schedule. Historically, the railroads have done poorly. In the late 1970's and early 1980's the chief cause of the performance problems was terminal operations. Then, as service design models became increasingly common, smarter decisions were being made with regard to routing of trains. Attempts to bypass yards, when possible, were successful. In addition, greater numbers of intermodal trains and unit trains meant that terminals were not involved in the processing of a larger percentage of freight. As the markets continue to grow, greater burdens will be placed upon both the lines and yards.

1.2.3. Yard Delays

The most telling figure of the effect of yard performance upon freight transit time is cited by Turnquist and Daskin [1982], describing that, on average, 77% (6.8 out of 8.8 days) of the time it takes to move a car from its loading point to its destination is spent in terminals. More recent estimates put this figure closer to 50%, though the specific causes of this reduction are not given[1]. There is no indication

[1] From the Rail Applications Special Interest Group (RASIG) Roundtable Discussion on Yard Modeling, ORSA/TIMS Joint National Meeting, October, 1994.

that these reductions are due to greater efficiency in the terminals, rather, they may be the result of better service design, smarter train routings, and increases in the number of unit trains, which pass through no intermediate terminals. In spite of this reduction, the time spent in terminals is significant, and reduction in terminal time, and controlling the variability in the terminal time, will lead to faster and more reliable movement of the freight.

In addition to viewing the terminal processing time relative to the total transit time, it is important to look at the terminal time absolutely. Martland, *et al.* [1994], compiled a set of benchmarks for hump yard performance based on over twenty-years of data collection. The authors cite *typical* car times ranging from 20-24 hours per hump yard, with scheduled connections missed roughly 15% of the time. They further recognize that the best processing times were found at Southern Railways hump yard in Knoxville, TN, over a 10-day period in 1972, during which the average processing time was 16.4 hours and the percentage of connections missed was only 8%. These figures give the reader an understanding of the slow-moving nature of terminals and the potential for improvement.

1.2.2. Network Planning

The types of decisions that concern rail freight transportation reach far beyond determining the sequence of operations on a single line or in a single terminal. A large body of research in network planning pertains to more aggregate problems, such as facility location problems, workload balancing, and seasonal/monthly train routing decisions. Rail researchers have devised a taxonomy of models, commonly attributed to the work of Assad [1980a], which places models into three levels. *Strategic* models involve long-term decisions, i.e., one or more years, involving large capital investments. *Tactical* models consider a time frame consisting of months or, perhaps, a single season. Finally, *operational* models explore the activities on a weekly or daily basis in response to specific train and car information.

To put the subject matter of this thesis in context, we will explore operational decisions regarding *specific* movements of trains. It is important to recognize interplay between the different planning levels and the need to ensure consistency between them. Stated differently, the specific decisions made at the higher, i.e., the strategic and tactical, levels influence the options for decision making at the operational level. For instance, if a decision is made at the strategic level to eliminate double track on the line between two terminals, the options for the operational planning of meets and passes along that line segment will be further constrained. Problems may arise if the aggregate demand used to determine the elimination of the double-track is inconsistent with the demand profile for that line segment at the daily or weekly level. This notion of consistency is also important to consider when developing and using models within the same planning level. A summary of specific research in network planning, terminal planning, and line planning will be related to this taxonomy in Chapter 3.

1.3. Overview of this thesis

The previous sections concisely described the business of rail, the general problem of moving freight through a rail network, and the impact of yard operations upon that movement. The primary objective of this thesis is to model the terminal and its processes in the context of operational network planning and control. There are two types of models that are considered in this thesis: descriptive models and sequencing models. This thesis treats each model separately, yet much of the material throughout pertains to both. The specific structure of this document is the following:

Chapter 2 *Tutorial on Terminal Processing*

Before any discussion on terminal modeling can be presented, it is necessary to adopt a standard nomenclature for the processing steps in the yard. This chapter provides a full description of the activities which take place in a hump yard, describes the types of decisions that are made in the terminal, and presents a real example—the Selkirk yard of Consolidated Rail (Conrail).

Chapter 3 *A Paradigm for Railroad Modeling*

Earlier, we noted the importance of treating the planning and scheduling of all components in the rail network together. This complete planning, though, presents interesting computational challenges. The rail network planning problem can be (and has been) subdivided into smaller, tractable problems. These problems are well-defined in the literature, however, some components of the decision-making process, especially at the *operational*, or day-to-day, level need refinement. The purpose of this chapter is to review planning models, as found in the last twenty years of research; to extend the notion of the hierarchy of models to a complete hierarchical planning system, which has well-defined components and linkages between them; and to pinpoint exactly within this system where the yard model developed in Chapters 4, 5, and 6 will operate.

Chapter 4 *Descriptive Yard Modeling : An Object Oriented Framework*

Keeping in mind the role of the yard model as a component within an operational network model, the first phase of this research is in the development of a purely descriptive yard model. The fundamental approach is simulation *without* stochasticity. It relies on average processing times for each of the yard activities and does not simulate a probability distribution for each process. Rather, this model allows a quick prediction of the put-through times for traffic *given* an arrival schedule, blocking plan, and car-to-block-to-train assignments. The model employs a First-In, First-Out (FIFO) processing order. The modeling structure, however, allows the incorporation of different processing rules or sequencing algorithms. Future uses of this modeling structure may include stochastic analysis of yard operating plans.

Chapter 5 *Modeling a Network of Yards*

As presented in Chapter 3, the fundamental role of a yard model is to assist the yard manager in making sequencing decisions, with consideration given to the network effects of the manager's decisions. The operational network planning and control methods described in the planning hierarchy rely upon line planning algorithms to solve the meet and pass planning problems and yard planning algorithms to solve the yard sequencing problems. This chapter presents a first step for coordinating these planning decisions across the network. The yard component used in this network planning structure is the model presented in Chapter 4. Though the yard components will not be making any "smart" decisions, other than FIFO processing, the network effects of this type of planning will be shown. Emphasis will also be placed on the development of a heuristic algorithm for coordinating the line and yard subproblems.

Chapter 6 *A Yard Sequencing Model*

The FIFO deterministic simulation may provide a suitable descriptive, or predictive, model for freight connections at yards. Ideally, a yard model used in the operational planning environment should make sequencing decisions. The model presented in Chapter 3 provides solid data structures and storage that can support such models, however, it does not incorporate algorithms to determine processing sequences. This chapter provides a formulation for a yard sequencing model that can be used at several levels of the planning hierarchy described in Chapter 3.

Chapter 7 *Summary*

This chapter summarizes the work contained in the thesis and reviews its contributions to both the academic and railroading communities. Finally, suggestions are made regarding future research and extensions to this line of research.

2. Tutorial on Terminal Processing

Before presenting the terminal modeling methods, it is important to gain an understanding of the role of the terminal from the perspective of network operations and the steps used to process the freight within the terminal. This will also provide the opportunity to introduce a standard nomenclature that will be used throughout the thesis.

As it pertains to the movement of freight from origin to destination, the role of the terminal, or yard, is much the same as a post office or an airport passenger terminal. The rail terminal is responsible for receiving inbound freight (versus letters or passengers), sorting it, and sending it to the next destination. Principally, the difference between the rail terminal and the other two examples is the existence of tight capacity constraints. Additionally, the rail terminal differs from the airport passenger terminal since the burden of moving the freight (i.e., persons) in the latter is strictly the responsibility of those persons. With rail terminals, the freight must be moved by a finite number of crews working for the terminal.

This chapter explores the steps by which freight is processed in a hump yard and the methods used by terminal managers to make sequencing decisions, given constraints on the length and number of tracks in the yard. We will begin with a description of the general structure of a hump yard, follow that with a step-by-step account of how freight is processed there, and wrap-up with a description of a real terminal—Conrail's major hump yard in Selkirk, New York.

2.1 Physical Structure

Prior to describing the processing of freight, it is important to define the physical apparatus that is used. Although each yard is different, the description presented here can be considered a typical layout for a hump yard. In addition, variations of this layout will be presented. Since the focus of this research is on the classification process as it supports movement of freight through the *network,* this discussion will highlight the parts of the yard that assist in connecting freight from inbound to outbound train.

Figure 2.1 shows the typical layout of a yard. There are three primary areas in which freight is held and processed: the *receiving* yard, the *classification* yard (or *bowl*), and the *departure* (or *forwarding*) yard. The receiving yard is used to store trains when they initially arrive at the terminal. Having been separated from their inbound train, groups of cars (or *blocks*) are assigned to and stored on tracks in the classification yard. Finally, when the freight is combined into outbound trains, it is done so in the departure yard, and the trains are then prepared for departure from the yard.

The length of track in each yard area varies. Due to the layout of switches (called "laddering", seen in Figure 2.1), the longest track in each yard tends to be near the center of the area, and the shorter tracks on the edges. And to generalize

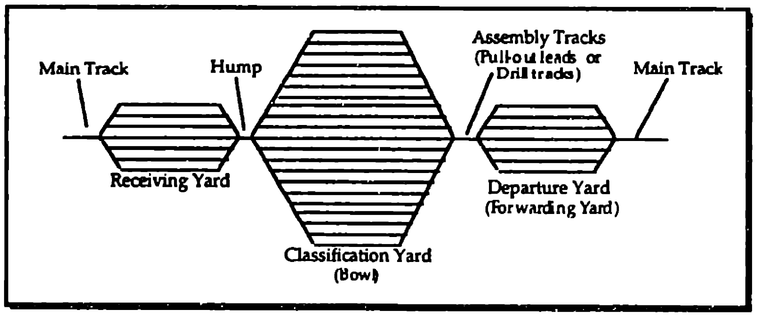

Main Track
Hump
Assembly Tracks (Pullout leads or Drill tracks)
Main Track
Receiving Yard
Departure Yard (Forwarding Yard)
Classification Yard (Bowl)

Figure 2.1: Example Yard Layout

the three areas, the tracks in both the receiving and departure yards tend to be longer than bowl tracks since they must store full trains. The bowl tracks, on the other hand, exist in greater numbers, but are much shorter because they store blocks of cars, not complete trains.

It is also important to understand the mechanism by which the freight moves *between* the areas. From the receiving and classification yards, there is a single track over the hump. The hump is a gradual, upward slope in the ground with a crest occurring at the point where the cars of a train are to be separated [The sorting process, itself, will be presented in the next section of this chapter]. On the other end, between the classification and departures yards, there exist one or more assembly tracks, also known as *pull-out leads* or *drill tracks*. These are used in the assembly of outbound trains as blocks are pulled from the classification yard to the departure yard. Additional flexibility for routing trains in the receiving and departure yards is provided by the inclusion of *cross-over* tracks (not shown in Fig 2.1), which provide additional options for moving between two tracks.

In addition to track, other finite resources include the inspection crews, the engine crews, and the engines used for moving cars through the yard. Inspections take place in both the receiving yard and the departure yard, and the inspection crews, each consisting typically of two persons, are usually assigned only to a single area for a given shift. The engines, too, are split into two groups working in two separate areas and perform two sets of tasks. The *hump engines* handle all the work in the receiving yard and the hump, while the *trim engines*, or *pullers*, work in the classification yard and the departure yard. Each engine has a crew of two persons. It is the proper coordination of the activities of the crews and tracks that allow the cars to be processed efficiently.

2.2 Supporting Data for Freight Processing

The amount of data available to the yard master prior to the arrival of trains for processing depends upon the yard's information system. However, there are some fundamental documents and sets of information that are typically part of a railroad's operations. These sets of information include waybills, wheel reports, consist reports, and switch lists.

2.2.1. The Waybill

The *waybill* is the basic document used to record the information about a shipment (i.e., a car or set of cars) moving through the railroad system. When a carrier receives an order from a shipper, a complete set of information regarding the shipment is documented in the waybill. This documentation gives the rail carrier information to track car movements through the network and it serves as part of the contract between two carriers if the freight changes hands over the course of its trip. The specific information included in the waybill is:

Car initial and number	Waybill number
Waybill date	Origin station
Name of shipper	Consignee (customer at destination)
Destination city	Route car will travel
Standard Commodity Code (7-digit code per ICC)	Physical description of articles
	Weight of shipment
Applicable weight	Total freight charges
Prepaid or collect payment	Perishability statement

At the terminal of origin, this information is fed to the carrier's car information system, which enables the shipment to be tracked as it moves through the rail network. The original waybill is kept with the conductor for the duration of the trip. When there is a change of crew, all documentation for the train is passed along to the next crew.

2.2.2. The Wheel Report

The wheel report is a consolidation of the waybill information for all the cars on a train. This listing shows destination, weight, load or empty status, hazardous materials (HAZMATS), and other relevant information for each car in the train. Wheel reports are kept by the train conductor and are handed over to the new crew when a crew change occurs. When cars are added to or dropped-off by a train, then the wheel report is modified to reflect changes in the train's composition, commonly referred to as its *consist.*

2.2.3. The Consist Report

Upon train departure, when the consist of an outbound train is verified, it is entered into the information system as a *consist report.* This report contains only time and date of departure, terminal location, and car numbers. Given the current state of automation of most railroads, this information is all that is needed, since the complete set of car data is entered with the creation of the waybill and is easily cross-referenced.

2.2.4. The Switch List

After a train enters a terminal for processing, the yard management verifies the consist to be in accordance with its consist report or its wheel report, then generates a report that describes the specific routing of cars from the hump to tracks in the classification yard. This report is called the *switch list* and is part of a bigger schedule commonly referred to as the terminal operating plan. The switch list includes not only the traffic from inbound trains, but also traffic originating locally that also needs to be sorted. In completely automated classification yards, the control of the switches is maintained by computer and executed in accordance with the track assignments shown on the switch list.

2.3 Processing Activities

Freight arrives at yards for four primary reasons, and, for each case, the processing steps differ. First, the freight may be *connecting* from one train to another. Second, the yard may be the final destination for arriving freight or the origin for departing freight. Third, the train upon which the freight is traveling may be simply passing through the yard en route to another destination. Finally, the train upon which the freight is arriving may require a mandatory safety inspection. Yard models that support network planning must account for all four cases, but the focus of this thesis is on connecting freight.

This section focuses on describing the steps required to connect cars from inbound to outbound trains (i.e., case one above), delays in which are a fundamental cause of unreliability in rail freight traffic. Processing the three other types of traffic involves subsets of these activities, so, following the complete description, the variations of these steps will be presented.

2.3.1. Arrival and Receiving

When trains arrive at a yard, they pull onto an assigned track in the receiving yard. Most yards have a video camera fixed at the entrance of the yard (the monitor is located in the hump tower) so that, as the train enters the yard, its consist is remotely verified to be consistent with the train's wheel report or its most recent consist report. Once in the receiving yard, the locomotives (or *power*) are removed from the train, leaving the consist sitting idle on the track. When the power is removed, the fault-tolerant hydraulic braking system of the train is engaged, so that the cars cannot roll freely.

Next, the cars are inspected for mechanical defects. For the purpose of safety, the track holding the train is "locked," so that no additional trains can be routed to the track during the inspection. The inspection crew, typically consisting of two members, walks the length of the train and, at each car, inspects for mechanical defects and bleeds the pressure from the braking system, allowing each car to roll freely.

If defects are found, the inspectors will fill out a *bad order* form and determine the appropriate course of action. For minor problems, defects may be repaired immediately while the cars are still on the track. More serious problems require the inspectors to tag the car so that, during the humping (or sorting) process, those cars will be sent directly to the terminal's car repair facility or to a classification track set aside for defective cars, denoted as a RIP (Repair-In-Progress) track. For the latter, the cars are pulled from the classification track to the car repair facility as soon as the repairs can be made.

Inspection represents the last receiving function. Upon the completion of the inspection, the track is unlocked and the inspection crew moves to its next train. The consist of the inspected train is ready to be classified.

2.3.2. Sorting

The inbound inspection is the final step before the train is ready to be sorted, or classified. This process is a car's transitional period between its assignment to an inbound train and its assignment to an outbound train. There are commonly accepted schemes for performing this sort, but the efficiency of each scheme depends largely upon the physical configuration of the classification yard, in terms of both number of tracks and their lengths. This section will first describe the physical apparatus used to sort the cars, followed by an introduction to the methods that are commonly used for making the assignment of cars to classification tracks.

2.3.2.1 The Mechanism

In a hump yard, *hump engines* are used to push the consist over the hump and into the bowl. A hump engine consists of two parts: a standard six-axle locomotive and a slug, which provides extra weight and traction to push the trains over the hump. Coupled to the rear of the consist, the engine pushes it from its receiving track to the hump. Moving at roughly three miles-per-hour, a speed that may vary from yard to yard, the train rolls towards the crest of the hump. As the cars pass over the hump, a member of the engine crew (the *pin-puller*) separates the cars from the train in accordance with the switch report. This is done by flipping a lever on the side of the car, which releases the *couple* between the two cars. In some yards, only single cars may pass over the hump, so the separation must occur between every car.

Upon the release of the car, gravity pulls it away from the train. As it rolls down the decline into the classification yard, the switches are aligned to route the car to its assigned track. The speed of a car is reduced by retarders, which are located in the bowl tracks (see Figure 2.3) and squeeze the wheels of the car as it passes. If the speed is gauged properly, the cars will couple with the cars already in place on the classification track. In modern hump yards, the alignment of switches and the control of the retarders are determined by computer. The

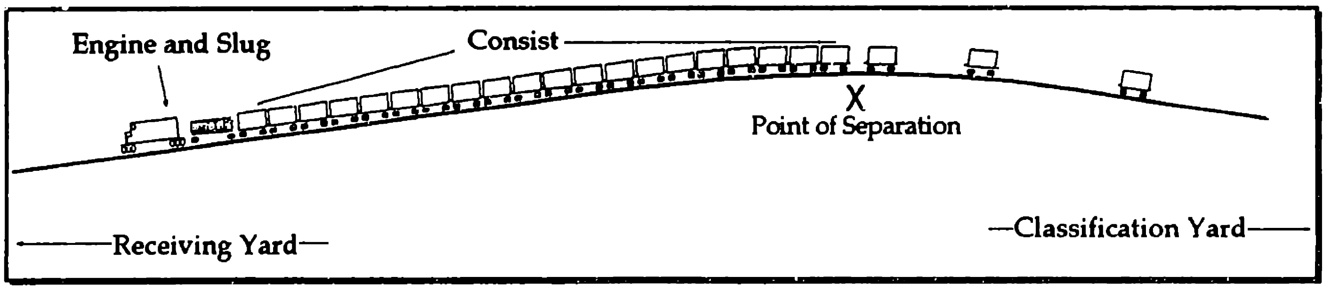


Figure 2.2: Profile of the Humping Process

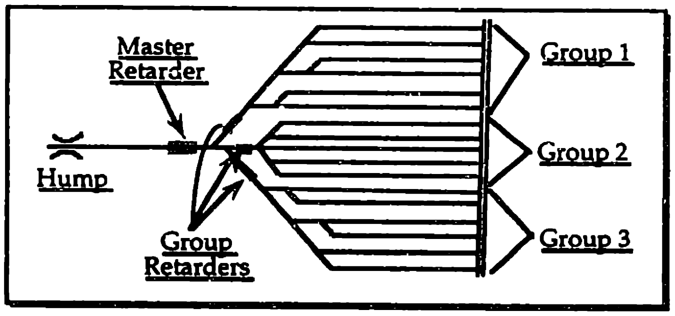


Figure 2.3: Retarder and Group Layout in a Hump Yard

amount of pressure applied by the retarders is calculated as a function of the weight of the car, wind resistance, and conditions of the track. The weight is either measured by a weigh-in-motion scale contained in the hump track or, if this does not exist, determined by information in the wheel report. At the end of each classification track are *skates,* or static retarders, which prevent cars from rolling beyond the end of the bowl track.

The layout of a typical bowl is shown in Figure 2.3, which includes a depiction of the master retarder and *group* retarders. Bowls are designed to include groups of tracks, in this case three groups. When determining the assignment of blocks to classification tracks, consideration of the groups of tracks is important, as it will affect the yard's ability to assemble trains. Blocks that are contained on tracks in the same group take less time to assemble than blocks on tracks in different groups. Thus, we see a strong interplay between the sorting process and outbound train assembly.

The humping and sorting activities are monitored and controlled by personnel in the hump tower, which is situated adjacent to the hump, i.e., "command central" for the receiving and sorting process. In addition to verification of consist, the hump tower can modify switching decisions in light of changes in current yard conditions.

2.3.2.2 Sorting Methodologies

Sorting may be done in a single stage, or in multiple stages. The number of stages refers to the number of times a car must pass over the hump or, in the case of flat switching, "kicked" by the engine into the classification yard. Yards with sufficient capacity in the classification yard will perform *single* stage sorting by

Table 2-1: Inbound Arrival Schedule

Train #	Origin	Block	Destination
1111	Milwaukee	A	Detroit
		B	Flint
		C	Indianapolis
		D	Nashville
2222	Dubuque	A	Indianapolis
		B	Kalamazoo
		C	Detroit
		D	Dayton
3333	St Louis	A	Flint
		B	Nashville
		C	Dayton
		D	Kalamazoo

block. In this case, the number of blocks can exceed the number of classification tracks, but there are limits at which the number of blocks will make single stage sorting impossible. In addition, the typical size of the blocks must ensure that a feasible assignment of cars to bowl tracks can be made. Multi-stage sorting is utilized when the capacity of the bowl is insufficient to complete single stage sorting and is more commonly used in flat yards.

Initially, we will present an example to illustrate the process of single stage sorting by block. Generically, the term *block* refers to a group of cars that are bound for a common destination. Consider the traffic on a fictitious railroad network inbound to a terminal in Chicago (Table 2-1). Three trains arrive into the terminal with connections to be made to two outbound trains, heading from Chicago to Detroit and from Chicago to Indianapolis. At the Detroit and Indianapolis terminals, freight will connect to trains bound for Flint and Kalamazoo (from Detroit) and Dayton and Nashville (from Indianapolis). The purpose of blocking at the Chicago terminal is to consolidate freight from inbound trains that is bound for the same destination. This also relieve the other two terminals (Detroit and Indianapolis) of some of the sorting burden by putting the two outbound trains into an ordering more conducive to quick processing in Detroit and Indianapolis.

When processing the freight in Chicago, we will assign each outbound block (or *classification*) to a track of sufficient length. When all the blocks for an outbound train are in place, or if some cutoff time has been reached, the outbound

Table 2-2: Outbound Train Consist and Blocks

Train #	Destination	Outbound Block	Inbound Block	Destination
8888	Detroit	[illegible]	[illegible]	[illegible]
		B	1111B	Flint
			3333A	Flint
		[illegible]	[illegible]	[illegible]
9999	Indianapolis	A	1111D	Nashville
			3333B	Nashville
		[illegible]	[illegible]	[illegible]
		C	2222D	Dayton
			3333C	Dayton

train will be assembled from its blocks. Table 2-2 shows the consist of the outbound trains and the mapping of inbound blocks to outbound blocks.

If single stage sorting is not possible, the yard master must rely upon multistage sorting methods to utilize the existing capacity in the bowl. Methods for assigning cars to classification tracks are presented in Siddiqee [1972] and Daganzo [1986], and the latter also presents the conditions under which the sorting strategies are effective. The strategies analyzed include sorting by block, sorting by train; triangular sorting; and geometric, or matrix, sorting. Though we will not go into the details of multi-stage sorting, the reader is encouraged to review the cited references as a tutorial on this subject.

2.3.3. Assembly and Departure

Assembly of outbound trains begins in one of two ways. The first is when all cars assigned to the outbound train have been sorted into the classification track corresponding to the cars' outbound blocks. Second, assembly may begin when a cut-off time, determined as some amount of time before scheduled departure of an outbound train, has been reached.

The assembly process involves the systematic movement of the blocks from the classification track to an assigned track in the departure yard. The work is performed by trim engines (or *pullers*) and their crews. The assembly process is commonly known to be the bottleneck operation in the process of connecting inbound freight to outbound trains (see Duffy [1994]).

Section 2.3.2 described the manner in which blocks are assigned to classification tracks. The impact of these assignments upon outbound train assembly is great and illustrated as follows. During the assembly process, it is not necessary for blocks to be brought from their classification track directly to the departure yard. Rather, if two blocks connecting to the same outbound train lie on adjacent bowl tracks in the same group, the trim engine may pull them to the departure yard together. This would be done by pulling the first block such that it clears the switch between the two tracks, resetting the switch, pushing the first block into the second block such that they couple together, and pulling both blocks to the departure yard simultaneously. Taking advantage of blocks that are close together allows the assembly time to be shortened, thus improving the overall efficiency of the yard.

The act of pulling the trains from the classification yard to the departure yard is further constrained by the configuration of the assembly tracks, or pull-out leads. A hypothetical example of this is shown in Figure 2.4. Here, not all classification tracks are connected with all departure yard tracks. In addition, the use of one pull-out lead may require the simultaneous use of another pullout lead. Hence, the importance of the selection of the proper assignment of blocks to classification tracks and proper use of the pull-out leads is amplified.

All of the assembly activities are controlled and monitored from the *bowl tower*, located at the end of the classification yard opposite the hump. The yard master in the bowl tower determines the block pulling sequence; the coordination of the trim engines used in train assembly; and the setting of switches in the assembly leads, the departure yard, and the trim end of the bowl. Although the humping and assembly operations are planned and executed by two separate

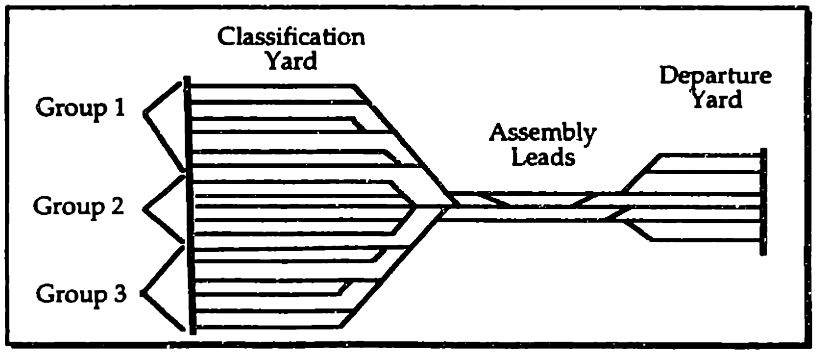


Figure 2.4: Layout of Bowl and Departure Yard, including pull-out leads

Table 2-3: Summary of Hump Yard Processes

Inbound Train Arrival
Pull-in to Receiving Yard ("*yarding*")
Removal of *power*
Inspection
Attachment of *hump* engine
Humping/classification/sorting
Assembly using *trim* engines
Attachment of *power*
Outbound inspection
Arrival of line-haul crew/approval for departure
Departure

sets of planners and crews, the coordination of these activities is paramount to the efficient operation of the yard.

When an outbound train consist has been assembled, its power must be brought from the terminal's *diesel shop*, where the locomotives are repaired, or from the locomotive *pit*, where trains are refueled and prepare for their next leg. The locomotives are attached to the outbound consist and the braking system is then re-engaged. An inspection crew walks the length of the train and inspects all cars for mechanical problems and the braking system for adequate pressure. Pending the arrival of the line-haul crew and approval from the yard master, the train departs the terminal.

The set of steps for processing inbound freight into outbound freight is summarized in Table 2-3. Slight variations of this process might occur in the interest of reducing processing time, though such variations are not common. For instance, if a train arrives late and has some high-value, tight connections to be made, the inbound inspection may be skipped in order to meet the deadline. Note that although cars on the inbound train would not be inspected as part of the inbound consist, they would be inspected upon their departure in the consist of an outbound train.

2.3.4. Sample Processing Rates - Radnor Yard Example

Chapter 1 provided aggregate figures for the average length of time required to perform the entire process of connecting freight from inbound to outbound trains. Those numbers include the time a car spends idle, such as the time waiting in the receiving yard before being shoved over the hump, or the time on a classification track until all the blocks of its outbound train are built. Comparing the idle time with the actual processing time for a car is a telling study. This section provides an example from a recent real-world study in which such times are compared. In addition, the times for the processing steps are cited.

Duffy [1994] presents a summary of data collected over a three-day period at CSX Transportation's Radnor Yard in Nashville, Tennessee. Of the 1146 minutes (~19 hours), on average, that a car spent in the yard, 434 minutes were spent processing the freight while 712 minutes had the freight sitting idle. Table 2-4 breaks down the processing times by step, as outlined in the Duffy study. The total average yard time for Radnor in the average performance range as defined in Martland, *et al.* [1994], but no comparison has been made for the rates of the individual processing steps. Furthermore, several of the processing steps, including inspections and humping, are dependent upon the length of the train, so the measures shown in the table do not accurately portray the *rate* of work on a per car basis. Nonetheless, they do provide the reader with a feel for the time it takes to perform such tasks.

The disturbing aspect of this data is the large amount of idle time at all phases of freight processing. Idle time, on average, represents 62.3 percent of the time cars spend in this yard. Assuming that the nine processes are independent, the ratio of idle time variance to total yard variance is 92.6 percent. Thus, sequencing decisions that affect cars' idle time appear to be of key importance to the

Table 2-4: Processing Times for Radnor Yard

Process Step	Average	Std Dev
Idle: Train arrival –> start of inspection	154 min.	154 min.
Inbound Inspection	123	26
Idle: Inspection complete –>Humping starts	257	142
Humping	45	16
Assembly	145	62
Idle: Assembly completes –>Inspection start	94	88
Outbound Inspection	118	29
Brake test	3	0
Idle: Train ready –> Train departure	207	135

performance of the yard. This single data point of processing times hints at a broader analysis of idle times and at potential benefits of sequencing models to reduce idle time and its variation. However, we must use caution when generally stating that reduced idle time is always beneficial. Consideration must be given to the *value* of the freight passing through the yard. Under certain circumstances, a redistribution of idle time among higher- and lower-value cars that results in a higher average idle time may attain a better result as measured by network objectives.

2.4 Variations

The previous section presented a typical structure of a rail terminal and the processing that occurs there. However, there are variations to that scheme and we will consider two types of variations. First, there are differences in the physical layout of the terminal that affect the manner in which the cars and trains are processed. Second, there are the different types of traffic, mentioned in the first part of this chapter, that require different processing steps than freight that is connecting from inbound to outbound trains.

2.4.1. Physical Structure Variations

No two yards are alike. Numbers of tracks and their lengths differ, cross-overs between tracks will occur in different places, placement of main lines through the

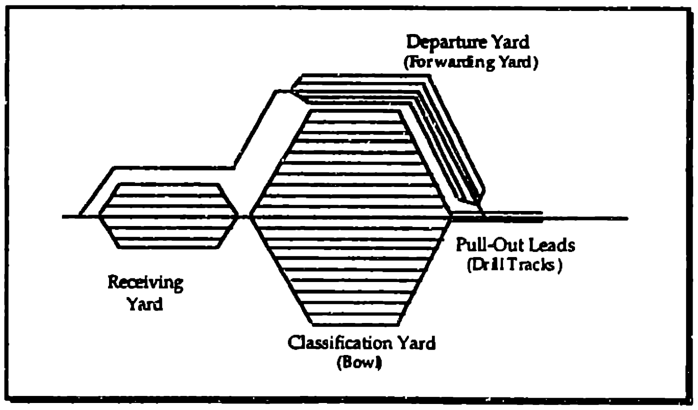


Figure 2.5: Yard Layout -- Variation #1

yard is never the same, and so forth. Each of these factors will influence the options available to yard masters for sequencing the freight processing.

Figure 2.1 depicted one possible arrangement of the *areas* of terminals[1]. In Figure 2.5, we show the case where the departure yard is situated alongside the classification yard. This type of layout will affect the *assembly* process because there is no longer a straight pull from the classification tracks to the departure yard. Instead, the trim engines must pull blocks from classification tracks into the pull-out leads, or *drill* tracks. When the end of the block clears the entry point for the departure yard, the switches are reset so as to route the cars into the departure yard. The engine reverses direction and pushes the block into the departure yard.

In some cases, terminals only have one area to handle both the receiving activities and the preparations for departure. In this case, careful coordination must occur between both activities to ensure efficient use of the limited track. Figure 2.6 portrays an example of this layout. In this case, not only would the trim engines need to pull-push the blocks to the "departure yard," but similar movement would have to be done when hump engines move freight from the "receiving yard" to the hump.

Physical variations lead to changes in the way freight is processed. Yet, the fundamental steps—receiving, classifying, and assembling trains—do not

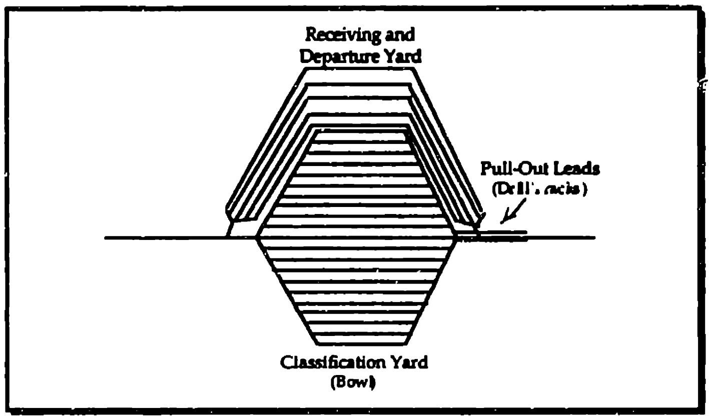

Receiving and Departure Yard
Pull-Out Leads (Drill Tracks)
Classification Yard (Bowl)

Figure 2.6: Yard Layout – Variation #2

[1]The generic layout presented there is similar to Radnor, but Radnor's layout is much more complex.

change. Instead, physical variations affect the *rate* at which the activities can be performed. Next, we will variation in the type of traffic that may enter the yard.

2.4.2. Traffic Variations

Section 2.3 focused on the processing steps required for handling freight connections between inbound and outbound trains. Yet, it is equally important to understand the other traffic that enters the yard, including pass-through trains, trains arriving for inspection, set-off/pick-up trains, and trains delivering local freight.

2.4.2.1 *Local Freight*

Freight for which the yard is the final destination is not separated from the other freight until it goes through the sorting process. One or more classification track is designated for local traffic. Instead of pulling the cars to the departure yard as part of an outbound train assembly, those cars are pulled to another area in the yard known as the local yard. Here, the traffic is sorted and prepared for delivery to local customers or shippers. Similarly, cars that originate at the terminal are consolidated in the local yard and brought to the outbound train during the assembly process.

2.4.2.2 *Set-off/Pick-up*

In certain cases, trains may arrive at a yard only to set-off (i.e., deliver) or pick-up a portion of its consist. This may be local freight, as described above, or it may be a cut of cars that is bound for further destinations. For inbound trains, the sorting process of the entire consist is foregone, and the only freight that is handled by the terminal is the cut of cars being dropped off. For outbound trains, the cars to be picked-up are assembled as usual, either through the classification and assembly processes or through the consolidation of local traffic. When the train arrives to the yard, the new cut of cars is simply added to the train's consist. This type of processing may be performed in either the receiving yard or departure yard.

2.4.2.3 *Inspection-only*

Trains are required to receive inspections every fixed number of miles, per federal regulation. In this instance, the yard will service such trains in the receiving yard or departure yard, and must account for such arrivals in the inspection crew schedule.

2.4.2.4 Pass-Through

Finally, there are pass-through trains, and if there are no tracks that bypass the terminal, these trains must travel through yard limits. The timing of such trains must be coordinated with the yard operations to avoid collisions with other trains moving to and from the main track.

2.5 Example Yard Layout - Selkirk Yard

The natural follow-on to the generic discussion of the terminal classification process is a specific example. Conrail operates a major hump yard in Selkirk, NY, just outside of Albany. The yard has a total capacity of 8500 cars and classifies between 2500 and 3500 cars during a twenty-four hour period. This is the third largest terminal in the Conrail system.

2.5.1. Relationship to Network

This section puts the Selkirk terminal in perspective with regard to freight moving through the Conrail network. This context is important because some of the activities within the terminal are dependent upon the direction of arriving or departure trains. Figure 2.7 shows how Selkirk relates to the rest of its division. Traffic arriving from the west end of the yard (control point CP "FB") is inbound

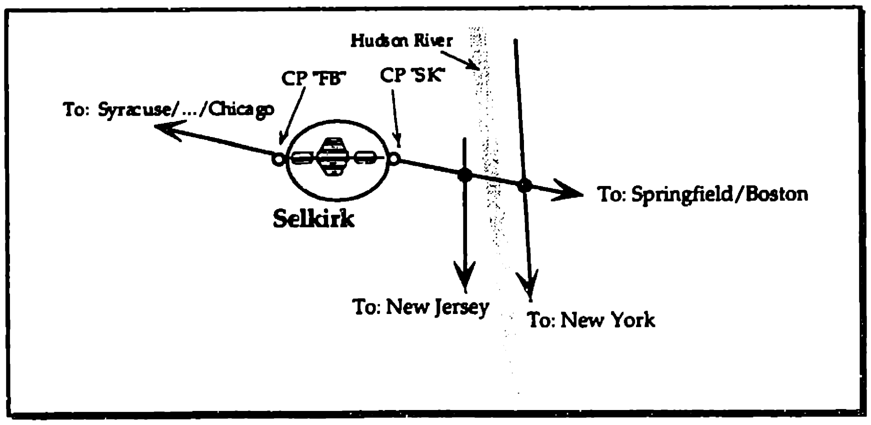


Figure 2.7: Selkirk Yard Relationship to Conrail Network

from the Syracuse/Buffalo/Chicago set of lines. The east end of the yard, CP "SK", is the boundary between the yard and the Massachusetts line (Springfield/ Worcester/ Framingham/ Boston). This line carries traffic from Massachusetts as well as traffic from New Jersey (Kearney/ Elizabethport/ Newark) and New York (New York City) that meets the Massachusetts line just east and west of the Hudson River, respectively.

2.5.2. Physical Structure

The configuration described in this section is current as of October, 1994, although the terminal is undergoing several physical changes. The design, as well as the electronic control systems, date back to the late-1960's, but with its large car handling capacity and a footprint of 1250 acres, the yard is well-suited for handling current-day traffic flows.

Appendix A contains physical drawings of the terminal, which provide an idea of the size of such a yard. The sizes of the different areas, in terms of numbers of tracks, are summarized in Table 2-5. In addition to these tracks, Selkirk houses a ten-track local yard, a four-track car repair facility, a locomotive repair facility, an auto unloading site, a TrailVan ramp, and a caboose storage area. Arriving trains number in the range from 20-22 per day and the departing trains amount to 17-20 per day. Although there are 80 blocks formed at Selkirk and only 70 tracks in the classification yard, each block is assigned a "home track," on which it is built each day. The determination of these home tracks is based upon aggregate measures of seasonal demand. Occasionally, a block can not be assigned to its home track and reassignment decisions need to be made such that the impact upon the overall operations is minimized. Circumstances that would warrant a track reassignment would be either the block length exceeds the home

Table 2-5: Summary of Yard Size

Terminal Area	# Tracks
Receiving Yard	11
Classification Yard	70
Assembly Leads	3
North Departure Yard	9
South Departure Yard	5

track length or the home track is occupied. Reassignments are made with the objective of minimizing assembly time.

2.5.3. Processing of Freight

Freight processing at Selkirk occurs as described in the generic example from Section 2.3. But Selkirk does have some operating features to which attention must be drawn:

- Conrail maintains a centrally located information system that tracks the status of all trains moving through the network. The Selkirk computer system receives updates from the central computer and provides the yard personnel roughly a thirty-hour window for traffic scheduled to arrive at the yard.

- In addition to the video camera mounted at the mouth of the yard, there is an Automatic Equipment Identification (AEI) reader that scans car numbers automatically as they pass. This information is used to verify the consist report already in the central system and speeds up the creation of the train's switch list.

- The main tracks that enter and run through the yard (i.e., *fast freight* tracks) actually pass directly through the terminal. Coordination of pass-through traffic and yard operations is, therefore, done very carefully by the yard masters.

- Hump operations, with the exception of the pin-pulling, are controlled by computer. As of October, 1994, this control was being done by a circa-1968 mainframe computer, which filled a large percentage of one floor in the hump tower. During November, 1994, this system was scheduled to be replaced with a set of microcomputers, each of which would be located at various locations in the terminal and individually capable of controlling the entire hump operation.

- Assembly is performed as described under the discussion of physical layout variations. The structure of Selkirk is similar to that shown in Figure 2.5, except that there are two aeparture yards, one on each side of the classification yard.

• The south departure yard has shorter tracks, and all eastbound trains are built there. Since there is a weight limit on traffic passing through the Berkshires in western Massachusetts, this restricts the length of the trains. The south yard contains short tracks and, hence, is the ideal place for assembling such trains. Traffic heading south or west is typically assembled in the larger yard.

• Locomotive scheduling is performed centrally in the Philadelphia headquarters, and the yard abides by these central decisions when connecting locomotives to outbound trains.

2.6 Chapter Summary

This tutorial was intended to provide a top-level description of freight processing at terminals. Three fundamental areas are used to process the freight: the receiving yard, the classification yard, and the departure yard. Variations exist in any yard, and the yard management must consider many factors when planning a shift's work. One must consider the amount of capacity available in the yard, the anticipated schedule of arriving and departing trains, and the personnel and resources available to perform the tasks. Through a specific example of a major hump yard, this chapter provided some insight into the information available to the yard masters to plan the sequencing of work and the assignment of resources to that work.

3. A Paradigm for Railroad Modeling

Just as it is important to understand yard models in the context of yard operations, it is imperative to consider their relationship with other rail planning models. This chapter begins with a well-known taxonomy of railroad planning models, introduced in Assad [1980a]. Within that classification, we present the modeling advances from the last two decades, as found in the literature and in the practice of major North American rail carriers. We evolve this taxonomy by extending it to a hierarchical railroad planning system, which requires a *structured* relationship among all models in the hierarchy. Finally, we present a detailed description of the operational portion of this hierarchical planning architecture and pinpoint exactly the area at which the research in this thesis is directed.

3.1. The Planning Taxonomy and Literature Review

A landmark effort to place railroad modeling within a hierarchical structure is provided in Assad [1980a]. The author offers a three-tiered hierarchy of models, including the strategic, tactical, and operational levels (see Figure 3.1). This sec-

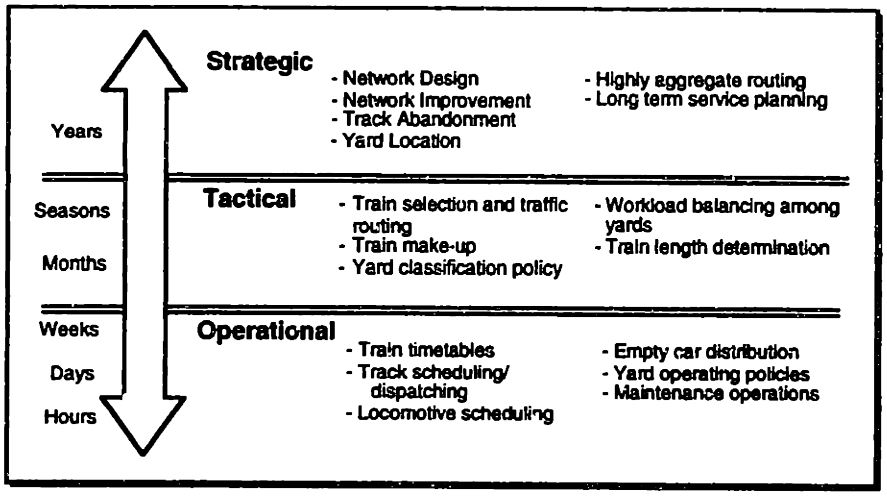

Strategic
- Network Design
- Network Improvement
- Track Abandonment
- Yard Location
- Highly aggregate routing
- Long term service planning
Years
Tactical
- Train selection and traffic routing
- Train make-up
- Yard classification policy
- Workload balancing among yards
- Train length determination
Seasons
Months
Operational
- Train timetables
- Track scheduling/ dispatching
- Locomotive scheduling
- Empty car distribution
- Yard operating policies
- Maintenance operations
Weeks
Days
Hours

Figure 3.1: Assad's Taxonomy of Railroad Planning

tion will trace through each level of the hierarchy and present recent modeling advances as they relate to moving freight through the network and to the yard operations.

3.1.1. Strategic Planning

Strategic decisions are long term in nature, involving time horizons of one or more years and generally requiring large capital expenditures. Examples include construction of new yards, yard expansion or redesign, construction or abandonment of track, and expansion of car and locomotive fleets. The class of models typically used for strategic planning includes simulations, facility location models, and network design models. Output from these models result in changes to the size and structure of the physical apparatus used to move freight.

Though most strategic decisions involve modifications to the structure of the network, some involve changes to service design on the existing network. While the capital-intensive infrastructure changes may require years to implement, service-related decisions may go into effect immediately. Such decisions include providing service to new markets or changing operating philosophy (i.e., moving from demand-driven to scheduled service). The strategic guidance, in turn, dictates service planning decisions at the tactical and operational levels.

3.1.1.1. *Strategic Yard Models*

For strategic analysis of a single yard, the predominant modeling technique is simulation. The best insight into the current state of simulation is gained by describing those created and used by the rail carriers.[1] The Union Pacific Railroad (UP), using a commercial discrete event simulation language, developed simulations for two hump yards. UP found that a primary drawback in using a generic, commercial simulation package is that modifying the initial yard simulation to fit the topology of a second yard poses a tremendous effort. A second simulation example is the Terminal Interactive Model (TRIM), a man-in-the-loop simulation developed by the Canadian National Railway (CN). This simulation is used not only by CN, but is available to other railroads through a dial-up connection. CSX Transportation (CSXT), one such user of TRIM, recently began the development of its own yard simulation, called the Yard Simulation Model (YSM) and written in the C programming language. Finally, Consolidated Rail Corporation (Conrail) has embarked on simulation efforts for several general freight and intermodal yards, relying, like UP, on commercial discrete event simulation languages. In spite of these developmental efforts, no standard tools exist for building yard simulations.

At the strategic level, yard simulations require high fidelity so that analysts can perform accurate studies of potential modifications to the structure of the yard. Simulation provides not only a decision support tool for exploring these capacity expansion issues, but can also be used to provide training to yard masters and analyze new yard operating doctrine. A simulation can also serve as a testbed to validate the results of yard sequencing models before they are put into operational use. Furthermore, repeated runs of a simulation can be used to abstract yard performance measures for use in tactical level network planning. These extensions will be discussed in later sections of this chapter.

3.1.1.2. *Strategic Network Models*

This thesis examines the issues of modeling yard operations in order to support a service of freight movements across the existing rail network. Models that perform strategic *network* planning focus on the capital-intensive problems versus

[1] From the Rail Applications Special Interest Group (RASIG) Roundtable Discussion on Yard Modeling, ORSA/TIMS Joint National Meeting, Detroit, MI, October, 1994.

the service-oriented problems. Therefore, no models are cited because they fall outside the boundaries of the research presented in the thesis. For these models, which include facility location, network design, and network improvement models, the reader is referred to Assad [1980a]. As will be discussed in Section 3.2, the strategic decisions that concern us in creating a hierarchical railroad planning model are those relating to service.

3.1.2. Tactical Planning

Tactical models are used to plan the utilization of network resources for periods of months or seasons. At this level, we are no longer making decisions dealing with large capital investment. Rather, these models determine the manner in which we will satisfy *expected* levels of customer demand through train routing, freight blocking, train make-up, train size, and workload balancing among yards. Like strategic models, those used at the tactical level represent line and yard performance abstractly. Therefore, our approach in describing these models is to first present the yard models followed by the network models. When applicable, we will highlight how yards are represented within a given network planning model.

3.1.2.1. Tactical Yard Models

The emphasis of tactical yard modeling is to provide a characterization of yard performance. These models are generally used for evaluating the effect of changing current yard operations and for representing yards within tactical network planning models. They tend to be descriptive in nature and relate the operating characteristics of the yard to an aggregate measure, or set of measures, but do not attempt to provide scheduling or sequencing decisions for activities in the yard. Descriptive yard models reviewed in this section have been developed using queuing theory, empirical models, statistical models, and simulation.

Queuing Models

Early examples of models to estimate put-through times, or delay, were developed using queuing theory. Peterson [1977a] defines a set of core processing functions in yards and analyzes them as a queuing system. Within this framework, Peterson derives probability distributions for the put-through times of different traffic classes, using historical data to verify the results of the queuing model. Peterson [1977b] further explores the relationship between the queuing model service rates and physical layout of the yard, intensity of train arrivals, and resource constraints (i.e., engines and crews).

The queuing theory approach was further explored by Turnquist and Daskin [1982], who present a queuing model based upon both the general batch arrival work of Burke [1975] and on the work of Peterson. While the Peterson model uses a train as the principal unit entering the queue, Turnquist and Daskin derive their model using cars as the principal unit arriving in batches on trains. Their M/G/1 queuing system presents a model of delay and relies on two component models: a batch-arrival delay model for trains before classification and a batch-service model to determine connection delays due to classification. Though the model includes some simplifying assumptions to make the problem tractable, the authors note that the use of this model should be as a "screening" model when considering new yard operating policies.

Empirical Models

Outside of queuing theory, efforts have been made to provide aggregate measures of yard performance using empirical methods. A notable empirical model was developed during the 1970's and early 1980's by the MIT Rail Group. Martland [1982] introduces the PMAKE model as a means to describe the probability of a car making its connection given its scheduled yard time. Calibrating a PMAKE function requires fitting a function to historical data on car connections. The function provides a means for predicting the likelihood of cars making their connections under given arrival and departure schedules. A closed-form expression such as a PMAKE function provides a component, when considered within a network model, to help evaluate service designs. In addition, the parameters used to calibrate the PMAKE function provide a method for comparing reliability among yards providing, in essence, a means for benchmarking performance. Martland, *et al.* [1994] present an anthology of benchmarks for yards studied by the MIT Rail Group over the last two decades.

Simulation

A third class of descriptive yard models are simulations. Chatlosh [1991] develops a yard simulation to investigate aggregate measures of yard performance, such as put-through times and probability of making connections. Chatlosh demonstrates that this simulation is capable of generating performance measures similar to the PMAKE model, while capturing the probabilistic nature of train arrivals and processing time and incorporating the details of yard operations. The main goal of this simulation is to provide a means to analyze the effect of various factors in yard operations upon the total car processing time and the probability of making connections. Thus, in addition to the strategic use of simulations, this work demonstrates the ability of simulation to generate aggregate performance measures that can then be used within tactical network planning models.

3.1.2.2. Tactical Network Models

As defined earlier, there are numerous critical decisions to be made across the network at the tactical level. Recently developed formulations to solve a group of the tactical planning problems within a single, comprehensive model. We will begin by presenting several models that solve individual components of the tactical planning problem, followed by composite models, that solve two or more planning components simultaneously.

Blocking Models

Bodin, *et al.* [1980] discuss an approach for determining a railroad blocking policy. A blocking policy determines how freight will be grouped together with other freight as it moves through the network. The authors recognize that solving the problem as a set of local problems will provide suboptimal decisions with respect to the network. The planning horizon for this model is six months and the traffic flows are assumed to be in steady-state. A network-based formulation is solved to find the optimal blocking plan. This model, however, has not been implemented for use in the rail industry.

A different approach is to solve the blocking problem heuristically. The Automated Blocking Model, described in Van Dyke [1986], solves the blocking problem using a heuristic approach and allows the human service planner to interact with the model to improve the service plan. This blocking model, under funding from the Association of American Railroads (AAR), has been commercialized and used by several major railroads

Routing Models

Klincewicz [1990] presents a facility location approach for determining optimal routing of freight movements. This model determines, for any freight bound from an origin to a destination, whether to ship the product directly or through what the author terms "consolidation terminals." Given that either the origin-to-terminal costs or the terminal-to-destination costs are linear, the author shows that this Freight Transportation Problem (FTP; i.e., not limited to rail) decomposes into a set of concave cost facility location problems, which are solved either optimally or through heuristic methods. The key aspect of this model is that it represents a completely different approach to solving the routing component of the tactical network planning problem.

Composite Formulations

Blocking and routing problems represent individual components of tactical planning presented in the Assad taxonomy. Models that attempt to determine the complete set of tactical planning problems *simultaneously* began to appear during the 1980's. Assad [1980b] presents a network model that incorporates a) the interplay between train routing and the activities that occur in yards, and b) the economic effects of block consolidation into single trains. This model produces decisions regarding service selection and frequency, train make-up, yard grouping, workload balancing, and train length. The network formulation represents yard delay linearly, which is used in the cost function of the problem.

Crainic, *et al.* [1984] provide a model that makes simultaneous tactical decisions for train routing, traffic routing, blocking strategies, train make-up, and allocation of classification work among the set of yards in the network. This network model endogenously determines delays along the line and within the yard. Yards are represented as steady-state queues, as in Turnquist and Daskin [1982], introducing nonlinearity in this multicommodity flow problem. As the result of this nonlinearity, the solution methodology relies upon heuristics and decomposition. The model is demonstrated on data from a Canadian rail carrier.[2]

The issue of nonlinearity was avoided in Keaton [1989], who presents a model to make decisions about direct connections between yards, frequency of train

[2] It is not known by the author whether this model is in-use by any railroads.

service, routing of individual cars on trains through the network, and blocking of cars within trains. This is the same set of problems addressed in both Crainic, *et al.* [1984] and Assad [1980b]. Keaton presents a mixed integer formulation, which is solved by either Lagrangian relaxation or by heuristic algorithms. The models expects *exogenous* determination of yard and line delays, which are incorporated into the network model as linear constraints. A later paper, Keaton [1992], reformulates the model as a pure integer program and focuses on improving the solution method. The algorithm arrives at optimal or near-optimal solutions, with good error bounds using Lagrangian relaxation and a dual adjustment method. The model is demonstrated on data collected by the AAR[3].

3.1.3. Operational Planning and Control

Operational decisions are made for horizons of weeks, days, or hours. At this level, we are concerned with the specific movement of freight through the network in response to customer demand and to obligations made by the railroad to meet that demand. General guidance provided from tactical models serves as primary input to operational models. Examples of operational models include car and locomotive scheduling, empty freight car distribution, and the train and freight sequencing decisions on lines and in yards.

3.1.3.1. Operational Yard Models

Operational yard modeling is the basis of this thesis and the literature reveals a lack of activity in this area. This section will look at models and analyses that have been reported for operational yard planning, including analyses of sorting methods, statistical process control (SPC) techniques, optimization models, project scheduling methods, and sequencing theory.

<u>Analyses of Sorting Methods</u>

Sorting refers to the manner in which inbound cars are assigned to classification tracks en route to their connections with outbound trains. The physical layout of a given yard, in conjunction with the expected consists and schedules for both inbound and outbound trains, determines the effectiveness of a given sorting

[3] It is not known by the author whether this model is in-use by any railroads.

scheme. There exist several common sorting schemes, and each will have benefits under certain conditions. The literature on sorting addresses these well-known schemes, with particular emphasis on analyzing the yard layouts for which each scheme is effective.

Siddiqee [1972] provides a complete description of the four major sorting strategies: "sort-by-train," "sort-by-block," "triangular sorting," and "geometric sorting." Siddiqee further explains under which conditions and within which types of yards these different schemes may be useful. Daganzo, *et al.* [1983] provide a more detailed investigation of the case of triangular sorting and multi-stage sorting, in which they derive equations to estimate the number of tracks required to complete the triangular sort. Such detailed analyses for all four sorting schemes are provided by Daganzo [1986], in which the author quantifies the utility for the different sorting schemes under the conditions of *known* departure times and consists for outbound trains. This notion is extended further by Daganzo for conditions of *unknown homogeneous* inbound and outbound traffic [1987a] and *unknown heterogeneous* traffic [1987b]. The key measures determined in these analyses include number of car switches performed, minimum number of tracks, and total space requirements. All of these papers recognize that there exists a strong relationship among the sorting scheme used, the physical configuration of the yard, and the expected make-up and inter-train distribution of the inbound and outbound traffic.

Statistical Process Control Methods

Duffy [1994] draws a parallel between the processing functions of the yard and the production line in a general manufacturing setting. Using data gathered at CSXT's Radnor Yard in Nashville, Duffy determines the mean and variance for each of these processing functions. Standard methods of Statistical Process Control (SPC) are used, with time-series plots providing easy identification of "jobs" gone awry. The author provides causal analyses for the points identified as "out-of-control." The bottom line of this approach is to identify when the system is not in control, determine the cause, bring the processing system under control, and influence the variables that can bring the yard's processing rates closer to their desired values. The methodology is intended as a tool for yard masters to increase the reliability of the yard operation.

Optimization Models

Yagar, *et al.* [1983] present an algorithm for sequencing the hump process. They model yard activities using simulation methods, while the hump sequencing decision relies on dynamic programming. Since the sequencing decisions are being made for *trains*, the problem remains small enough to handle with dynamic programming. Through several case examples, the authors show the effectiveness of their algorithm, called the Hump Sequencing System (HSS). They compare the average (total) yard time and show the improvement HSS brings over the actual sequencing decisions made by yard personnel. When the use of sequencing models is extended to all areas of the yard, the benefits will continue to grow.

Guignard and Kraft [1993] present a mixed integer programming model that determines the hump sequence and classification track assignments. The objective is to improve the effectiveness (i.e., meeting the time constraints) and efficiency (i.e., doing so at minimum cost) of the internal operations of a yard. The authors cite two previous efforts, one by Deloitte, Haskins, and Sells [1976] and the other by SRI International (see Wong, *et al.* [1980]). The former designed a system to optimize the hump sequence given fixed block-to-track assignments, while the latter takes a given hump sequence and determines the best use of the classification tracks.

The Guignard/Kraft model solves these two problems simultaneously by modeling the combined problem as a multicommodity flow across a space-time network. The formulation of the problem is then solved using a commercial optimization package. This method is being considered for operational use by UP and will initially be incorporated into and analyzed in one of the UP simulations described in Section 3.1.2. The drawback to this optimization approach is that, as the size of the planning horizon and number of variables grow, the problem becomes difficult or impossible to solve. It is unclear whether this approach will be practical for use as a real-time decision support tool for yards.

Project Scheduling Theory

In practice and, for that matter, in the literature, we have found only a single case of rail planning being considered in the project scheduling paradigm. CSXT has developed a scheduling tool called the Yard Planning Model[4] (YPM), developed with Microsoft Project, a commercial, PC-based project scheduling tool. It is clear that processing trains through a yard, given resource constraints and precedence relationships, has similarities to resource-constrained project scheduling and can be planned heuristically using Critical Path Method/Project Evaluation and Review Technique (CPM/PERT) tools such as Microsoft Project. YPM is, however, still under development and not used at any CSXT sites. On the academic side, Sprecher [1993] and Elmaghraby [1977] each provide a thorough treatment of models for resource constrained project scheduling. Outside of the use of basic CPM/PERT scheduling, none of the more recent developments presented in these references have been considered as means for planning freight processing in railroad yards.

<u>Scheduling and Sequencing Theory</u>

Operations Research and production journals are full of optimization models and heuristics for solving many classes of machine scheduling problems. Only recently has there been an attempt to use such models for the yard sequencing problem. Specifically, the yard processes follow closely that of a flow shop, where all jobs must be processed by the same ordering of machines. In the case of the railroad yard, cars (or blocks of cars) represent the jobs and the machines are the processes, such as receiving, inspection, humping, and assembly. Due to the subtleties of the yard sequencing problem, a direct application of flow shop scheduling techniques is not possible.

Ferguson [1993] connects the field of machine scheduling with the yard sequencing problem using a mathematical model that represents the yard problem as a two-machine sequencing problem. The objective in this formulation is to minimize the maximum tardiness of outbound trains. The particular challenge is handling inbound train disassembly (i.e., humping) with outbound train assembly, a problem for which existing machine scheduling methods do not exist. Although this research incorporates some simplifying assumptions, the general notion of exploiting traditional scheduling and sequencing algorithms is appeal-

[4] From the Rail Applications Special Interest Group (RASIG) Roundtable Discussion on Yard Modeling, ORSA/TIMS Joint National Meeting, Detroit, MI, October, 1994.

ing and shown by Ferguson to be effective. A description of the Ferguson model and proposed improvements are given in Chapter 6.

Current literature reviewed covers single machine scheduling and general flow shop scheduling literature. Background material on the general theory of scheduling and sequencing is found in Conway, *et al.* [1967], Ashour [1972], Baker [1974], Rinnooy Kan [1976], Bellman, *et al.* [1982], French [1982], Blazewicz, *et al.* [1986], and Blazewicz, *et al.* [1993]. In addition, research on single machine, flow shop, and job shop scheduling is readily available in assorted production and Operations Research journals (see Table 3-3 for a summary of recent publications).

3.1.3.2. Operational Network Models

Our primary interest in operational network planning is the control of freight movements through the network. This type of planning relies upon line dispatching models and yard sequencing models, both of which have seen relatively little research effort. The combination of line dispatching and yard sequencing into a complete operational network planning model is an *untouched* area of research. Related research areas, for which systems have been developed and used by the railroads, include real-time distribution of locomotives, freight car scheduling, and empty freight car distribution. There is, quite obviously, an interdependence between all of these planning components, a concern which is addressed in Section 3.2. This section will present research related to all types of operational network modeling, but will focus on the few models that exist for our primary interest, the real-time control of freight movements.

Haghani [1989] describes an approach to bridge the gap between the tactical planning level and the operational planning level. In this paper, the operational planning issue is empty car distribution. The model attempts to minimize the total operating cost of moving empty and loaded cars over the network. Computational experience shows that solving the combined routing problem of loaded and empty cars is preferable to solving the problems separately. Kwon [1994] presents an overview of car scheduling systems and presents a multi-commodity flow path-based formulation for the car scheduling problem. Though such scheduling systems exist in the industry, their downfall is that the real-time modeling elements have been deemed ineffective throughout the industry.[5] Locomotive scheduling systems have been greeted with more success. Recent efforts by transportation consulting groups have led to the development of locomotive distribution systems for major rail carriers.[6]

Turning now to the control of train movements through the network, the majority of effort on real-time planning has focused on lines. Jovanovic and Harker [1991] present the line Schedule Analyzer (SCAN), an interactive line planning system designed to support the daily, weekly, or monthly scheduling of train movements through a rail line segment. The key decisions in SCAN are the "meet" plans for a single line, which can be defined as the segment of track over which a single dispatcher has control. Reliability of a plan is defined as the probability that it remains feasible under conditions of uncertainty. Thus, SCAN attempts to create feasible plans with high reliability. At the time of publication in 1991, modifications were underway to incorporate the criterion for minimizing a weighted tardiness objective.

The Charles Stark Draper Laboratory initiated the development of its own line planning algorithm (Draper [1994]). This model, referred to as PLATO, incorporates detailed representations of track topology and train performance, allowing more complicated situations to be handled than can be handled using SCAN. The combinatorial nature of the meet *and* pass planning problem solved by PLATO prevents us from getting an optimal solution in any reasonable amount of time, so the algorithm attempts to solve the meet and pass problem quickly using heuristics.

Due to the explosive size of this planning problem, extending real-time train sequencing from a single line to a network of lines is not trivial. Tsitsiklis [1993] and Christodouleas [1994] explore methods of solving the problem of planning multiple lines using an approach that will initially consider a corridor and ultimately an entire rail network. The approach relies upon nonlinear optimization techniques, specifically Lagrangian Relaxation, to decompose the network into a set of single line planning problems, managed by a central coordinating mechanism. Within this notion of network control, it is critical to consider the other

[5] From the Rail Applications Special Interest Group (RASIG) Roundtable Discussion on Car Scheduling, INFORMS National Meeting, Los Angeles, CA, April, 1995.
[6] For instance, IBM Consulting (Dallas) for Burlington Northern Railroad, and Saber Decision Technologies for Conrail.

major component of the rail network—the yard. The balance of the thesis treats the yard planning problem in this context.

3.1.4. Literature Review Summary

We have placed some examples of recent modeling efforts in the context of the hierarchical scheme proposed by Assad [1980]. This section provides a model summary, grouped into three categories: yard models, network models, and sequencing models, contained in Tables 3-1, 3-2, and 3-3, respectively.

Table 3-1: Summary of Yard Models

Level	Planning Problem	Model Type	References
Strategic	Capacity Expansion	Simulation	Union Pacific* Canadian National* CSX-Transportation*
Tactical	Yard Delay	Queuing	Peterson [1977] Turnquist and Daskin [1982]
		Empirical	Martland [1982]
	Yard Delay and Operating Policy	Simulation	Chatlosh [1991]
Operational	Sorting Strategies	General Analysis	Siddiqee [1972] Daganzo, *et al.* [1983] Daganzo [1986, 1987a, 1987b]
	Performance Measurement	Statistical Process Control	Duffy [1994]
	Hump Sequencing	Network flow Dynamic Programming	Wong, *et al.* [1980] Yagar, *et al.* [1983]
	Dynamic Classification Track Assignment	Heuristic rules	Deloitte, Haskins & Sells [1978]
	Hump Sequence and Classification Track Assignment	Multi-commodity flow	Guignard and Kraft [1993]
	Yard Resource Assignment	Project Scheduling	CSX Transportation*
	Hump and Assembly Sequencing	Sequencing Theory	Ferguson [1993]

* Rail Applications Special Interest Group, ORSA/TIMS Joint National Meeting, October 1994

Table 3-2: Summary of Network Planning Models

Level	Planning Problem	Model	References
Tactical	Blocking	Multicommodity Flow Heuristic Approach	Bodin, *et al.* [1980] Van Dyke [1986]
	Routing	Network Flow Facility Location	Assad [1980b] Klincewicz [1992]
	Composite (Routing, Blocking, Make-up)	Nonlinear MCF with Heuristic Solution	Crainic, *et al.* [1984]
		MIP and IP, with Lagrangian Relaxation and dual adjust-ment	Keaton [1989, 1992]
Operational	Empty Car Distribution	Mixed Integer Program with nonlinear objective function	Haghani [1989]
	Freight Car Scheduling	Multicommodity Flow	Kwon [1994]
	Network Control	Decomposition by Lagrangian Relaxation (subproblems solved heuristically)	Christodouleas [1994] Tsitsiklis [1993]

Table 3-3: Summary of Scheduling and Sequencing References

Description	Reference
General Scheduling and Sequencing	Conway, *et al.* [1967] Ashour [1972] Baker [1974] Rinnooy Kan [1976] Bellman, *et al.* [1982] French [1982] Blazewicz, *et al.* [1986] Blazewicz , *et al.* [1993]
Resource Constrained Project Scheduling	Elmaghraby [1977] Sprecher [1993]
Problem Class	**Reference**
General Single Machine	Lee, *et al.* [1991] Feo, *et al.* [1991] Fisher [1976]
Single Machine with Tardiness Costs	Sen and Borah [1991] Holsenback and Russell [992] Fadlalla, *et al.* [1994] Panwalker, *et al.* [1993]
Single Machine with Weighted Tardiness Costs	Swarc and Liu [1993]
Single Machine with Early and Late Costs	Yano and Kim [1991] Zheng, *et al.* [1993] Davis and Kanet [1993]
Single Machine with Ready Times	Liu and MacCarthy [1991]

3.2. Hierarchical Planning of a Railroad

To contrast this section with Section 3.1, consider the previous material under the heading, "Planning Models in the Context of a Hierarchy." This implies placement of the models into a well-defined classification. However important, it does not imply that the models will constitute a hierarchical planning system. A more complete architecture must be developed. This architecture must describe the sequence in which models are used and the *specific interfaces* among these planning components.

In this section, we develop a framework for a hierarchical planning system as it relates to moving freight through the rail network. We will not consider the decision process for the capital investment decisions, such as track and yard construction. Instead, we will treat the physical infrastructure and inventory as *given* and attack the problem of creating and executing a schedule. Emphasis must be placed upon the strong linkage between tactical and operational components. First, we will describe the modeling sequence that should exist. Then, we will present a more detailed picture of hierarchical planning within the operational level and how we propose to implement the *specific* control of freight moving through the network.

3.2.1. Structuring a Sequence of Decisions

Hierarchically decomposing large scale planning problems requires the identification of discrete, tractable components of the problem. Many of these components for the rail problem were summarized in Section 3.1. The next step in creating a hierarchical planning system is to prescribe the sequence in which these components operate. This section presents sequences under two separate operating philosophies: running a scheduled railroad versus running a demand-driven railroad. In spite of the dramatic difference in operating philosophy, the planning sequences are remarkably similar. In actual operations, railroads run somewhere in between the two extremes. In fact, even "fully scheduled" systems will have some small percentage of unscheduled traffic.

3.2.1.1. Scheduled Railroads

Great emphasis has recently been placed upon the notion of running scheduled railroads[7]. Although different carriers invoke different meanings, we shall refer to a *scheduled* railroad as one that, on a monthly or seasonal basis, sets fixed train arrival and departure times that will serve as the basis for making tactical and operational decisions, much like an airline. In other words, the timetable is central to all tactical and operational planning decisions. Though the schedule may change as the result of phenomena such as unexpected weather conditions, derailments, or unplanned track maintenance, the recurring theme at all levels of planning is *adherence* to the schedule.

Planning to a fixed schedule is the modus operandi for the airline industry. Tactical and operational decisions are made *subject to* an existing flight schedule. The typical paradigm for tactical planning in the airline industry is shown in Figure 3.2 (see Barnhart, *et al.* [1994], Hane, *et al.* [1994], and Gopalan [1995]). After the flight schedule is developed, a fixed sequence of tactical planning decisions follows. Each model makes decisions *given* the decisions made earlier in the sequence. Although the specific models used by each airline may differ, the

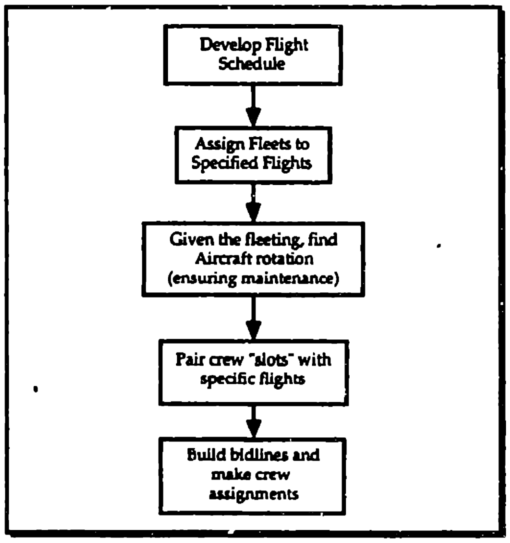

Develop Flight Schedule
Assign Fleets to Specified Flights
Given the fleeting, find Aircraft rotation (ensuring maintenance)
Pair crew "slots" with specific flights
Build bidlines and make crew assignments

Figure 3.2: Tactical Planning Sequence for Airlines

[7] See **Traffic World**, Feb. 13, 1995, pp. 24 - 26

ordering of decisions are similar among all carriers.

Although the rail industry and the commercial airline industry have major operational differences, we can highlight the similarities between the functional aspects of their planning problems. The assignment of fleets to scheduled flights would correspond to the tactical allocation of locomotives to scheduled trains. Assignment of crews to flights, given the fleet assignment, would correspond to the assignment of crews to scheduled trains. Rail crews, instead of being certified for an aircraft type, are certified to run particular topologies in the network, which in turn define the set of legs to which they can be assigned. The list of parallels can be extended, with the fundamental notion that the central input to these planning problems is the timetable, which is the fixed schedule objectives for the network. This is not to say that the problems are identical, because they are not. The underlying theme in this airline example is that they, over the course of a few years, have developed a hierarchy of tactical planning decisions and defined the data that must flow to and from each model. This type of planning needs to be addressed by the rail industry and, specifically, by each rail carrier.

Figures 3.3 depicts a notional model of hierarchical planning in a scheduled railroad. The timetables are determined tactically, not operationally as called out in Assad's taxonomy. In the boundary between the tactical and operational levels, the intent of the diagram is to depict the strong interrelationship between locomotive scheduling, freight car scheduling, empty car distribution, and control of freight movements through the network. In time, modeling experience will define these interrelationships, as well the most effective manner in which to run a "scheduled" railroad. Experience will also determine the proper sequence of tactical decisions, including proper placement of the timetable generation (i.e., either as shown in Figure 3.3 or prior to the blocking, routing and make-up models).

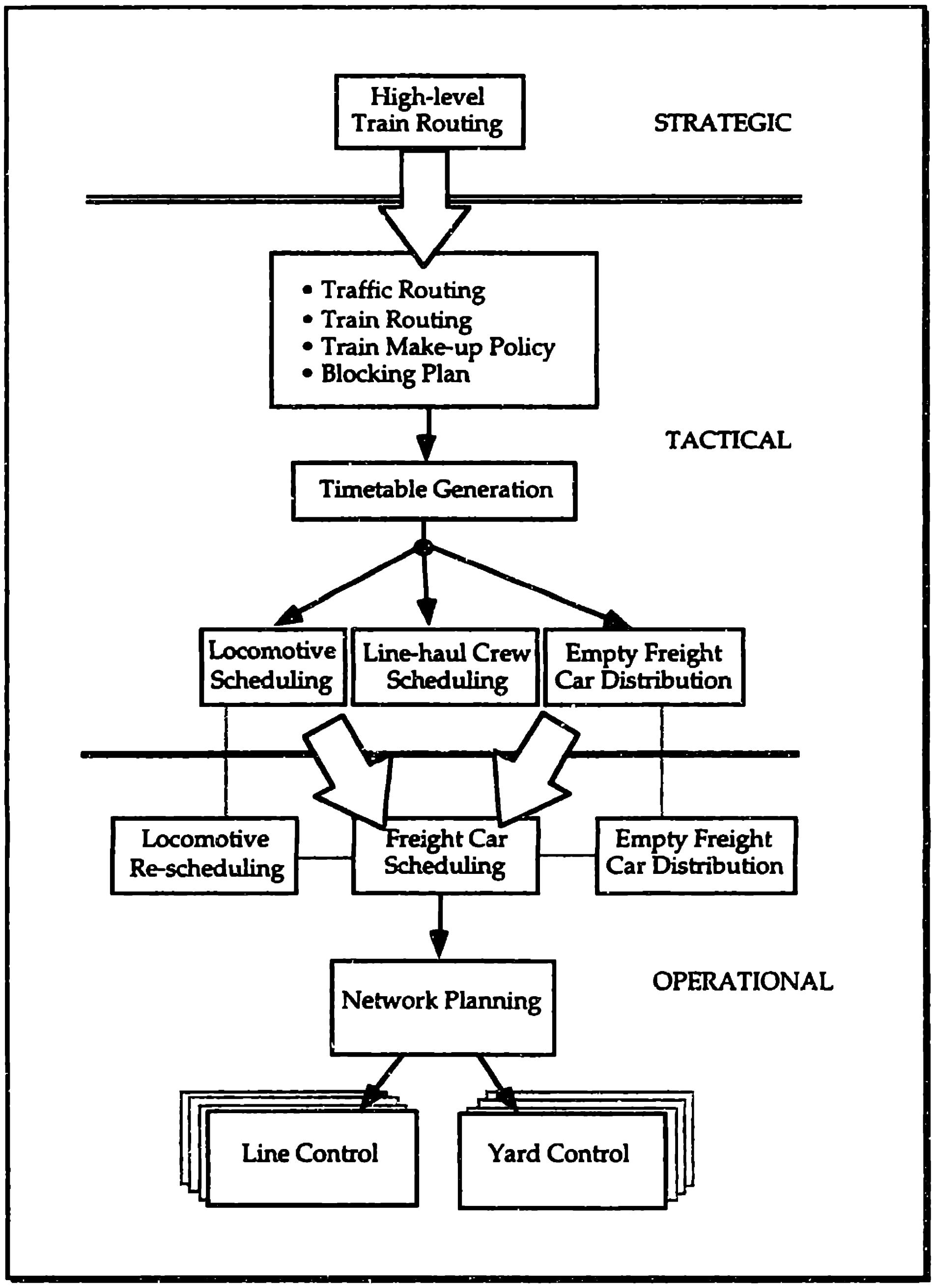


Figure 3.3: Hierarchical Planning in a Scheduled Railroad

3.2.1.2. Demand-Driven Railroads

On the other extreme from purely scheduled railroads, we have a "demand-driven" railroad, which is characterized by running trains when the freight is ready. There is no fixed train schedule. Arrival and departure times would be generated within the operational level.

The arrangement of functional modeling components is shown in Figure 3.4, differing from Figure 3.3 (the schedule railroad hierarchy) only in the subordination of timetable generation to the operational level. Though functionally these planning hierarchies look nearly identical, the specific models required to support each would be different. For example, assigning cars, either empty of full, is a more readily solved problem if train service is defined *a priori* (e.g., Kwon [1994]). If we do not have specific information about the schedules, this assignment becomes more difficult. Different models are required for the separate cases, though the *function* of the car distribution component in both hierarchies is the same.

A work-around to this issue is to generate *loose* schedule objectives at the tactical level. Then, all of the succeeding models in the hierarchy would use these loose times for planning purposes. At the operational level, instead of schedule enforcement, as in the case of scheduled railroads, we would enforce schedule refinements, significant if necessary, to support the demand-driven concept. Thus, all tactical models in this case would be the same as in the scheduled railroad hierarchy and would provide the baseline from which the operational models would plan. This brings us back to a fundamental benefit of this hierarchical approach. The decisions at higher levels of planning should provide guidance through a fixed sequence of decisions such that the lower level models can arrive at good decisions quickly. Without the help of the higher level decisions, the low level components will be ineffective.

The data flows discussed in this section were limited to the tactical level, although Figures 3.3 and 3.4 offered schematics for hierarchical planning through the operational level. The following section offers details for the operational components and provides an important backdrop for the remainder of the thesis.

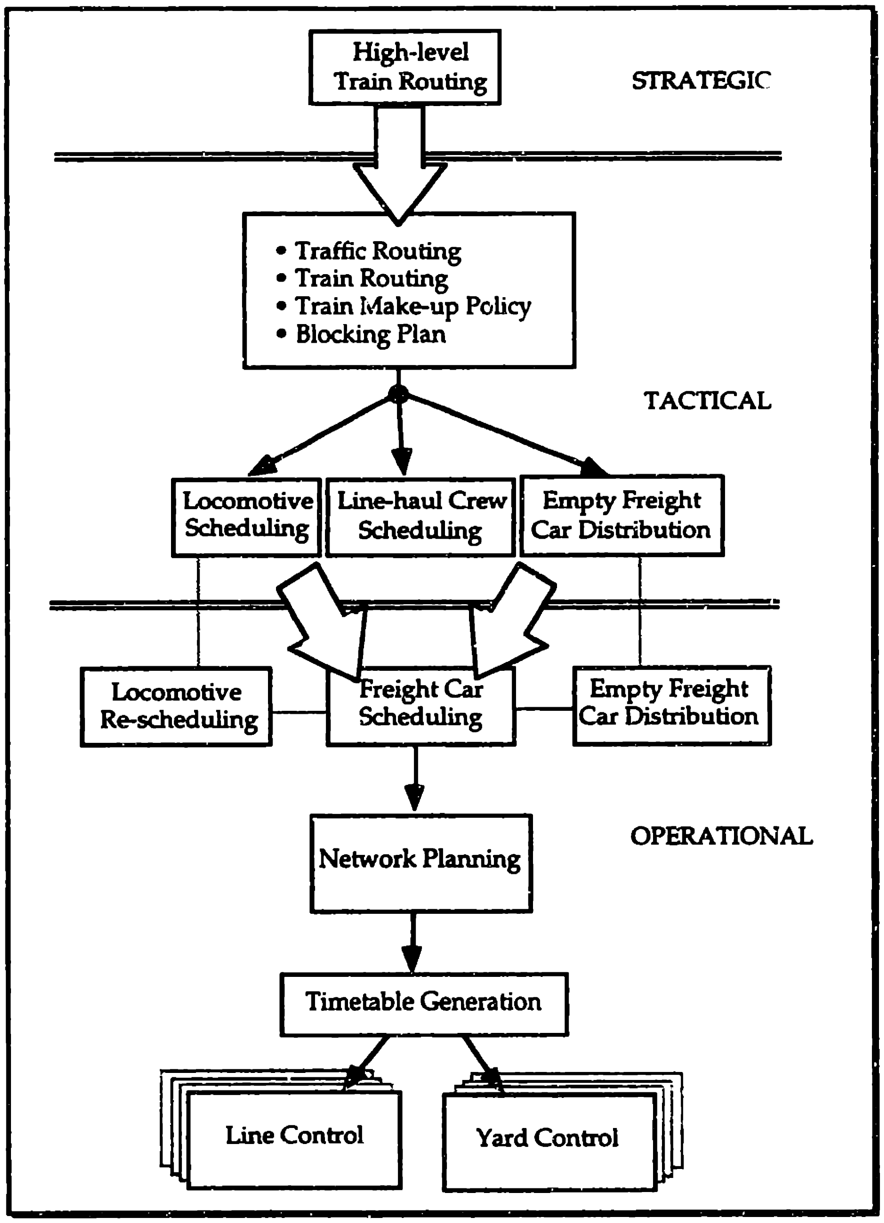


Figure 3.4: Hierarchical Planning in a Demand-Driven Railroad

3.2.2. Operational Network Planning Hierarchy

As depicted in Figures 3.3 and 3.4, operational planning involves adjustments to locomotive assignments, redistribution of empty freight cars, scheduling of loaded cars, and coordination of train movements through the network to meet the delivery obligations made to customers. There are close relationships among the components, and this section will address hierarchical planning as it relates to planning and coordination of specific train movements. This assumes that the other decisions, such as train consists and schedule objectives, have been made.

This operational planning and control problem can be subdivided into two major areas. First, there is the creation (or enforcement) of schedule objectives across the network. Next, there is the execution of line and yard control to meet the schedule objectives. We formalize the difference between schedules and plans with the following definitions:

Schedule	Times at which trains pass through selected points in the network. These serve as boundary conditions within which to create the detailed plans
Plan	A detailed specification of activities that are to be performed in order to operate within the schedule.

A simple example of the distinction between the two is the following. Consider two trains moving between Yards A and B. The *schedule* would specify the departure and arrival times at the points of interest, namely at Yards A and B. Train 1 is scheduled to depart A at 0800 and arrive at B at 1200. Train 2 is scheduled to depart B at 0830, arriving at A at 1130. The *plan* specifies the sequencing of trains on the line between A and B. In this case, the plan would specify the siding that Train 1 must occupy to allow Train 2 to pass, so that both trains will meet their scheduled arrivals. In summary, a schedule is less detailed than a plan. [Note that this is opposite from classical scheduling and sequencing usage, but consistent with terminology used in railroad operations.]

Next, we introduce the nomenclature for the individual components of this planning process. These are not models, but *functions* for which models must be used. The functional relationship of these components is depicted in Figure 3.5.

• **Operational Network Planning.** ("Network Planner") Set of components whose combined function is to create schedule objectives, within which the local controllers (see below) will create specific plans. This set consists of three components: a yard planning model, a line planning model, and a network coordinating algorithm.

•• **Yard Planning Model.** ("Yard Component") For each yard, this component determines the sequencing activities and refines

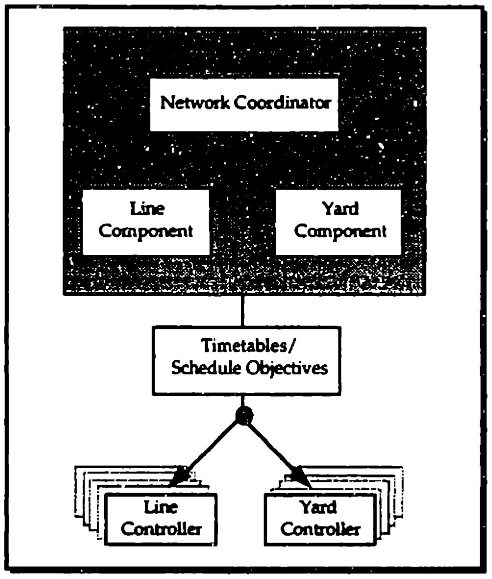


Figure 3.5: Operational Network Planning and Control

the departure schedules for trains *given* an arrival schedule and train consists.

•• **Line Planning Model.** ("Line Component") For each line, this component estimates train running times, given a set of trains moving through the line.

•• **Network Coordinating Algorithm.** ("Network Coordinator") This component reconciles differences between the line and yard plans and coordinates between them in order to achieve better *global* (i.e., network-wide) solutions.

• **Control Decision Support.** ("Local Controllers") High fidelity models that provide decision support to operations personnel for creating detailed plans and controlling line and yard activities.

•• **Line Control Decision Support.** ("Line Controller") Tool to assist dispatchers with meet and pass plans and with the generation of authorities, which are the formal instructions passed to train engineers.

•• **Yard Control Decision Support.** ("Yard Controller") Tool to assist yard masters with yard sequencing decisions and resource (i.e., crews, engines, and tracks) assignments.

In developing models to be used for these real-time planning components, there are some fundamental issues that do not arise at the tactical level. First, planning has to be done *quickly*, meaning a strong reliance upon heuristic algorithms with short run-times and very good (not necessarily optimal) solutions. Second, there must be strong *coordination* between consecutive, overlapping planning horizons. The railroads are in continuous operation, so the planning component must handle rolling planning horizons, where the decisions made in the current planning horizon will impact future horizons. Giving consideration only to the current period or to a single component of the network is myopic and leads to suboptimal decisions. Finally, real-time planning systems must help the network *recover from irregular conditions*. This includes the ability to recover as locally as possible (in terms of both time and space), resulting in minimal impact on network operations.

The network planner must be tied closely to a central information processing system that captures the *state* of the network. Though in some systems the information is updated manually after voice or radio contacts with trains, other systems automatically update information through advanced tracking tools such as Automatic Equipment Identification (AEI) tags on the side of freight cars. In either case, a complete understanding of the state of the system is required.

In addition to knowing the system state, the network planner must interface with the other operational planning components. When a current plan is no longer enforceable, the hierarchical planning system has several courses of action for recovery. There are two extremes. First, it may maintain the current car-to-train assignments and merely reschedule the arrivals and departures throughout the network such that all scheduled freight connections are preserved. Second, it may adjust the train consists while preserving the schedule. Or it may do a combination of the two. In order to take the best course of action, the network planner *must* interface with the car scheduling system if it is to consider modifying train composition. In addition, changing the composition of trains will affect train weight, implying an additional interface with the locomotive scheduling component.

We will now look inside the network planner and investigate how it creates schedule objectives. The name network planner, as opposed to network scheduler, is used because its internal mechanism creates a detailed plan of the network activities in order to arrive at a good schedule. The line and yard components are sequencing algorithms, albeit less detailed than those used in the line and yard controllers. The solution strategy taken by the network coordinator is to iterate through the entire set of lines and yards, reconcile schedule inconsistencies between plans developed by the line and yard components, and determine opportunities to improve the *network* solution beyond the local decisions created by the component planners.

When the network planner is finished, the schedule objectives, not the specific plans, are passed to the local controllers, whose job is to enforce the schedule objectives through detailed plans. These controllers utilize high-fidelity models to respond in real-time to changes in arrivals and consists. If the deviations can be resolved within the confines of the line or yard, then there is no impact upon the network. When the effects of the deviation spill beyond the boundaries of the line or yard, the local controller "notifies" the network planner. The network planner, given the current state of the system and the current schedule objectives, tries to resolve the problem as locally in the network as possible. Here, "local" refers to both the number of subordinate controllers whose plans will change *and* the time over which these changes will occur. This notion of upward communication and modification of the schedule objectives is referred to as *recovery*.

Now consider this type of planning and control vis-a-vis the operational air traffic control in the United States. The movement of aircraft through U.S. airspace is not heavily used for a continuous twenty-four period. At the start of *each* day, an air traffic plan is created using the flight schedule and the anticipated airport arrival and departure capacities. As the day progresses and unexpected events occur, such as weather changes and equipment failures, the plan must be modified. Thus, the distinction between planning and recovery is clearer than in the railroad, where the system seems to always be in a state of recovery.

In contrast, railroads run for the entire day, an operational feature that blurs the distinction between planning and recovery. There is no point at which we can hit a reset button, empty the system, and start over with a new plan for the next day. Planning occurs when the previous planning period expires or when the condition in the network warrants a new schedule. The second case is tied to recovery, which is best understood as the upward communication mechanism that allows these conditions to be handled. This not only involves invoking a new plan, but doing so, if possible, within a localized area of the network.

Having described operational planning and control of the network, consider again the issue at hand in this thesis: **the creation of a yard model to support real-time network planning**. The bulk of the work to be presented in Chapters 4, 5 and 6 relates to models that will support the yard component of the network planner. In designing the algorithms for the yard component, the two primary objectives are for the algorithms to be a) consistent with decisions that will be made by the local controllers, and b) fast. For the yard component, we propose a method (in Chapter 6) that uses a two-machine sequencing model, satisfying the "fast" requirement. This model can, in turn, be expanded to a higher level of fidelity for use in the yard controller, ensuring the "consistent" requirement.

3.3. Chapter Summary

The aim of this chapter was to provide a framework for a railroad hierarchical planning system and to develop the context for this thesis. Beginning with the railroad modeling taxonomy presented in Assad [1980a], we added to the taxonomy by describing recent modeling advances found in academic literature and in practice at major North American rail carriers. As it pertains to the operational planning and control of trains, the literature revealed that line planning research (i.e., real-time determination of meets and passes) is more advanced than yard sequencing research. There has been no work reported on real-time network planning and control, combining line and yard planning.

In addition to describing the taxonomy and the literature review, this chapter presented a notional model for hierarchical planning in the rail industry. This transcends the placement of models into a classification and implies a well-defined framework that links all planning models—from strategic train routing decisions down to line and yard control decisions made in real-time. Beyond the notional architecture presented for such as system, we explored the details of hierarchical planning at the operational level. Finally, we pinpointed the area at which this thesis is directed: the development of a yard model to support real-time railroad network planning.

4. Descriptive Yard Modeling: An Object Oriented Framework

Having presented a notional architecture for network planning and control, we can now present a specific yard model that will be used within the network planner and can be extended for use in the yard controller. To reiterate the distinction between the two components, the former is used in conjunction with a line planning model and a network coordinating model to produce schedule objectives for the network. The yard controller, on the other hand, is the decision support tool used by yard masters to determine sequencing activities and yard resource assignments within these schedule objectives.

The first step in developing a yard model is to create of the data structures for representing the yard and the activities that occur there. This begins by identifying the required input to such a model and the output it must provide back to the network coordinator. Using Object Oriented Modeling (OOM), the data needed to support the input and output is encapsulated into static data structures that represent an abstraction of the objects observed in the real system.

These static structures are then used dynamically to model the behavior of the system and estimate the connection times for freight.

Specifically, this chapter is organized as follows. The first section presents the conceptual design of the yard model and the input and output it must support. The next section describes the static object model to represent the yard and freight. The third section captures the dynamic behavior of the system using a simulation-based model that relies upon the static objects created in the preceding section. For each section, an understanding of some fundamental concepts of object oriented modeling is necessary. Appendix B is provided as a concise Object Oriented Modeling (OOM) tutorial.

4.1. Conceptual Design

As noted in Chapter 3, the network planner consists of a network coordinating model, line planning model, and yard planning model. It relies upon iterative methods to create the schedule objectives for the network. The term network planner is misleading, since its output are schedule objectives, not plans. Its internal line and yard components generate plans which the network coordinator uses to create schedule objectives. The network coordinator provides the appropriate data for the line and yard components to plan. The first step in creating a yard model is to understand that data that are needed.

The input to the yard model should help it respond to the question, "If cars of known value arrive on inbound trains at given times, make their connections to their outbound blocks, which make their connections to outbound trains, when will the outbound trains depart?" As shown in Figure 5.1, the input to the yard model will, therefore, include scheduled arrival times, schedule departure times, freight values, car-to-block assignments, and block-to-train assignments. The model prescribes internal sequencing of yard activities to determine the *actual* outbound train departures. The yard model passes the departure times back to the network coordinating algorithm, which reconciles inconsistencies between the schedules determined by all line and yard components. This process iterates until the network coordinating algorithm arrives at a good solution of network schedule objectives.

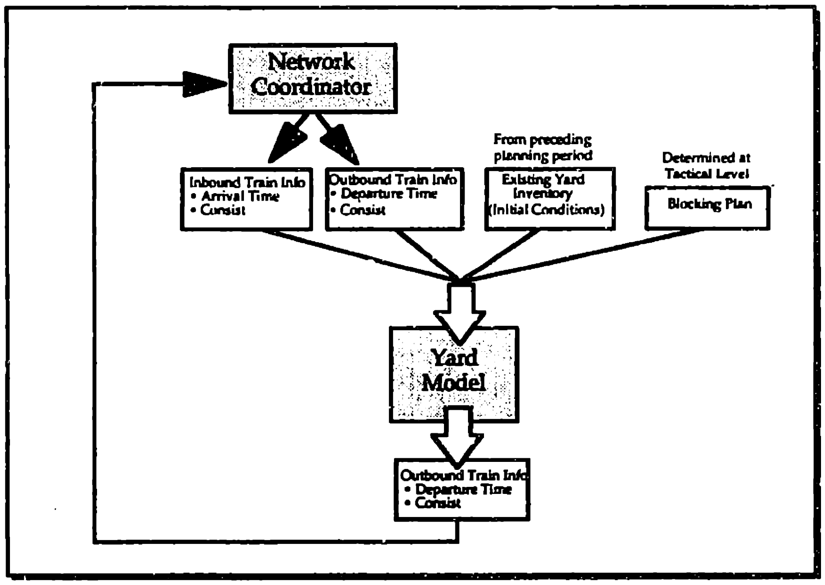


Figure 4.1: Data Flow Between Yard Model and Network Coordinator

In its second role, the yard model can be used as a yard controller, or decision support tool for yard masters. In this case, the schedule objectives have been set by the network planner and the yard controller must operate within those boundaries. The yard controller requires a higher level of fidelity. The object model described here supports a range of fidelity levels, including the level needed for the yard controller. The major modifications would be applied to the assignment algorithms that give the model its dynamic behavior.

4.2. Static Components - Object Oriented Design

The initial process of Object Oriented Modeling (OOM) involves the static representation of the physical components of the system. Dynamic behavior, which is discussed in the second half of this chapter, is simply an evolution of relationships between static objects in the system. In the case of the yard, the physical structure, namely the set of tracks, is a fixed part of the system, while the trains that move over them exhibit dynamic behavior. Using OOM, both trains and tracks first need to be defined statically, then the movement of the trains over the set of tracks in the yard is represented by updating their associations to each other. We will consider trains, tracks, engines, and crews as the fundamental objects in the system. This section is dedicated to describing the representation of these resources and the data contained in each.

4.2.1. Train Representation

Trains, in reality, comprise locomotives, cars, and cabooses, also known as ends-of-train (EOTs). In the network planning methods described in Chapter 3, a locomotive distribution system will be treated separately from the network planning and control function. So we will only consider the relationship of cars to trains, not locomotives to trains [NOTE: extending this static class design to include locomotives is a simple modification to the model]. Usage of EOTs are also not considered since their role in the railroads is becoming increasingly uncommon.

The most basic units in a general freight train are cars, which are grouped together into blocks, traveling from one yard to another. Inbound trains contain blocks that were assembled at previous yards, while outbound trains contain blocks created at the current yard or at previous yards. Single-car blocks may exist, but their occurrence is rare.

To represent cars, blocks, and trains, we define the base class **TrainRoot**[1], from which all possible components of trains are derived (see Figure 4.2). **TrainRoot** objects contain data common to all objects moving though the network, including the value, origin, destination, start date/time, and due

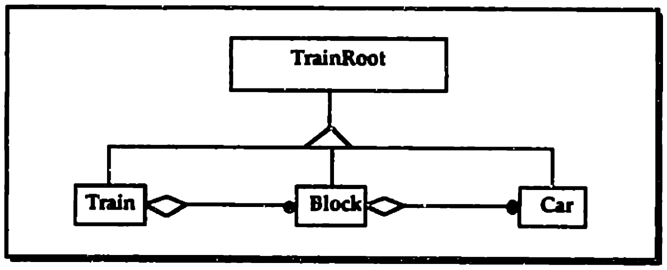


Figure 4.2: Static Object Oriented Representation cf Trains

[1] **Boldface** text indicates a class (i.e., the general data structure). Objects are the specific instances of the classes and are referred to as "**MyClass** objects."

date/time. Derived from the **TrainRoot** class[2] are **Train**, **Block**, and **Car** classes. Each contains data specific to that type of freight. We can relate each of the derived classes to each other, as depicted in Figure 4.2, by stating that **Train** objects *consist of* **Block** objects, and **Block** objects *consist of* **Car** objects. A natural extension of these relationships to intermodal transportation is the inclusion of lower-level classes, derived from **TrainRoot** and contained in **Car** objects, called **Container** or **Trailer**.

The benefit of this type of object representation is that both **Car** and **Block** objects can exist on more than one **Train**. This is important when we are dealing with connecting freight. Data associated with a **Car** or **Block** object is stored only once, yet it can appear on an inbound and an outbound **Train**. The same is true for **Car** objects contained in multiple **Block** objects. Again, the relationship among the four classes is shown in Figure 4.2.

4.2.2. Yard Representation

Yard operations can be viewed as a fixed sequence of activities for processing freight. From the description provided in Chapter 2, we understand the role of each physical area in the yard and the processes that occur there. Let us define a **ProcessNode** as any area or activity in the yard at which or during which some fundamental processing takes place upon a car, block, or train. A **ProcessNode** contains a finite list of renewable resources to perform the processing. Table 4-1 shows the list of **ProcessNodes** and the renewable resources associated with

Table 4-1: Summary of Classes derived from ProcessNode and their resources

ProcessNode	Type	Resources
MainTrack	Area	Track(s)
ReceivingYard	Area	Tracks
InboundInspection	Activity	Crews
Switch	Activity	Crews
Separator	Area	Track(s)
ClassificationYard	Area	Tracks
Trim	Activity	Crews
DepartureYard	Area	Tracks
OutboundInspection	Activity	Crews

[2] Derived is equivalent to "inherited." Recall from Appendix B that classes, *not* instances of the classes (i.e., objects), are inherited from each other. The inheritance, therefore, is of the data structure, not that data that is stored there.

each. Note that this listing corresponds to the list of activities discussed in the yard tutorial in Chapter 2.

Each **ProcessNode** is considered either a physical **Area** or a processing **Activity** (see Figure 4.3). **Activities** share the characteristic that, for a given train, the processing time of the train is known and calculated based upon the train's length. **Areas**, on the other hand, represent the physical parts of the yard where the train occupies a resource (i.e., a track) for an amount of time that is not well-known. While the train is assigned to an **Area**, it will also be assigned to an **Activity** for some period of time. For example, a train sitting in the receiving yard (an **Area**) must undergo the **Activities** of **InboundInspection** *and* the assignment of a **Switch** (or hump) engine. Thus, the amount of time the train spends in an **Area** depends upon not only the length of the train, but the idle time that results from the trains position in the **Activities'** processing sequence.

A **Train** or **Block** may be assigned to more than one **Area** at a time. For example, while a train is moving from the **ReceivingYard** to the **Hump**, the **Train** concurrently occupies its receiving track and the hump track. The receiving track will not become "free" until the train has cleared the hump.

The inheritance scheme for **ProcessNode** is to derived into two classes, **Area** and **Activity**, from which more specific classes will be derived. For the finite, renewable resources at each **ProcessNode** we introduce the class, **Resource**, which is further spilt into the specific resource types **Crew** and **Track** (as shown in Figure 4.3). Finally, we can relate a **ProcessNode** to its **Resources**, where the

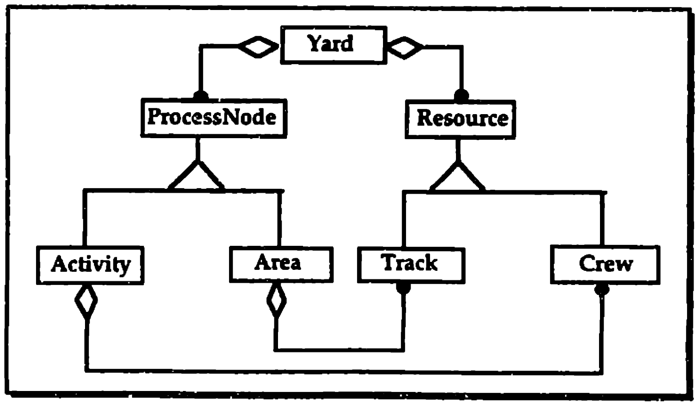


Figure 4.3: Object Model for Processing Functions and Yard Resources

type of **Resource** (i.e., **Crew** or **Track**) will depend upon the type of **ProcessNode** (i.e., **Activity** or **Area**).

The fundamental building blocks for the physical infrastructure of the terminal are **Track** objects. Each **Track** object belongs to a physical **Area** in the yard. **Areas** includes the derived classes **MainTrack**, **ReceivingYard**, **Hump**, **ClassificationYard**, **AssemblyTracks**, and **DepartureYard**. A **Track** object is typically contained in only a single **Area** object, though many **Track** objects may be contained in that **Area** object.

Shown in Figure 4.4 are the derived classes of **Area** and **Activity**. The **Activity** class splits into **Inspection** and **Engine** classes. An inspection's duration depends upon the length of the train. From **Inspection**, we have two derived types of inspection, **Inbound** and **Outbound**. This parallels the separation of these activities in actual yard operations. The second type of **Activity** is the class **Engine**, which refers to the process of moving and connecting an engine to its assigned block or train. There are two classes of **Engines**: **Switch** engines (i.e., hump engines), which work the hump end of the yard, and **Trim** engines, which work the assembly end. The name **Switch** is used in lieu of "**Hump**" to avoid confusion with the physical **Area** called **Hump**.

Using these static objects, we emulate a real hump yard by connecting the **ProcessNodes** according to the processing order defined in the yard tutorial of

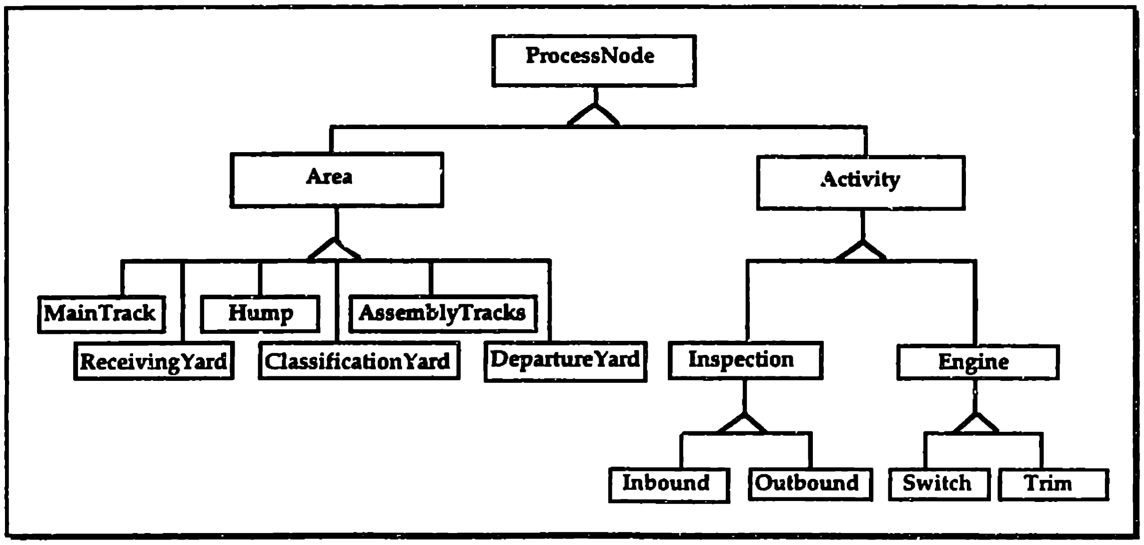


Figure 4.4: Object Model for **ProcessNodes**

Chapter 2 and revisited in Table 4-1. Straightforward yard layouts, such as the one shown in Figure 2.1, result in each **Resource** belonging to one **ProcessNode**. In more complicated yard layouts, resources may belong to more than one **ProcessNode**. For example, in the layout shown in Figure 2.6, the receiving and forwarding functions are combined into one area—a joint receiving and departure yard. In the static object model, no additional classes need to be created. Instead, the set of **Track** objects associated with the **ReceivingYard** would also be associated with the **DepartureYard**. The **ProcessNode** functions remain intact. The only difficulty comes during the dynamic behavior of the model, when both **ProcessNodes** vie for the same tracks. The next sections describes the algorithms that are used to make the freight to resource assignments.

4.3. Dynamic Model - Object Oriented Deterministic Simulation

The static representation, or object model, stores key data elements relating to the processing of connecting freight. This section presents the dynamic components for a type of modeling to which we will refer as deterministic simulation. We will first describe the general flow of the dynamic model, including the inputs for creating static objects, the control of the dynamic behavior through a central *Event List*, and the algorithms that are used to handle the First-In, First-Out (FIFO)[3] processing of the model.

This modeling approach was selected for two primary reasons. The first requirement is to have a predictive mechanism that provides put-through time estimates *quickly*. This modeling approach allows us to capture the effects of congestion in the yard, though not as quickly as using a closed-form expression for delay.

Second, having a detailed model of the interactions that occur in the yard allows the model to generate sensitivity measures for connecting freight. We can then answer questions such as, "If inbound train A arrives thirty minutes behind

[3] When possible FIFO is followed. However, when resource limitations, such as insufficient track length, prevent a train from being processed, other trains with feasible resource assignments will be processed ahead of the first train.

schedule, what will be the impact upon the connections with outbound trains C and D?" Even better, we can answer the question, "How far can we adjust the arrival of one train before its starts to affect the departure of its connections?" Since the yard model is utilized within the context of a broad network planner, answers to these questions are important. The deterministic simulation provides such insight, where the closed-form expression can not.

Even though stochastic elements are not captured, the model's object structure lends itself to conversion to a fully stochastic simulation. This includes uncertainty not only for the processing times, but for phenomena such as track failures, equipment failures, car derailments, and changing weather conditions.

To completely describe the dynamics of the model, it is important to begin with the inputs to the model and understand how the *initial state* of the static objects are created. Given these initial conditions, the evolution of the system over a specified planning period occurs through the creation and execution of *events*. The processing decisions follow a simple First In, First Out (FIFO) rule. Finally, visualization, originally designed as a means for debugging the algorithms, provides quick insight into the model's behavior.

4.3.1. Inputs to Model

The model is represented as a flowchart in Figure 4.5. Data is input in the following order: car and block information; train arrival and departure schedules, including consist information; topology of the yard; crew work schedules and processing rates; and, finally, existing physical inventory. When this input is read, the static objects are created, their data members are populated, and their associations with other objects in the system are initialized. The first four loops in the flowchart perform the static data input and the fifth loop initializes the dynamic relationships between **Train** (or **Block**) objects and the objects representing the yard. This initialization is performed by reading the train arrival list and the inventory input file. A more detailed explanation of initialization is provided in Section 4.3.2.

We will now explore the specific inputs of the model to help develop the understanding of the type of data required to support this model. For the sake of clarity, notation that has not been previous declared will be introduced

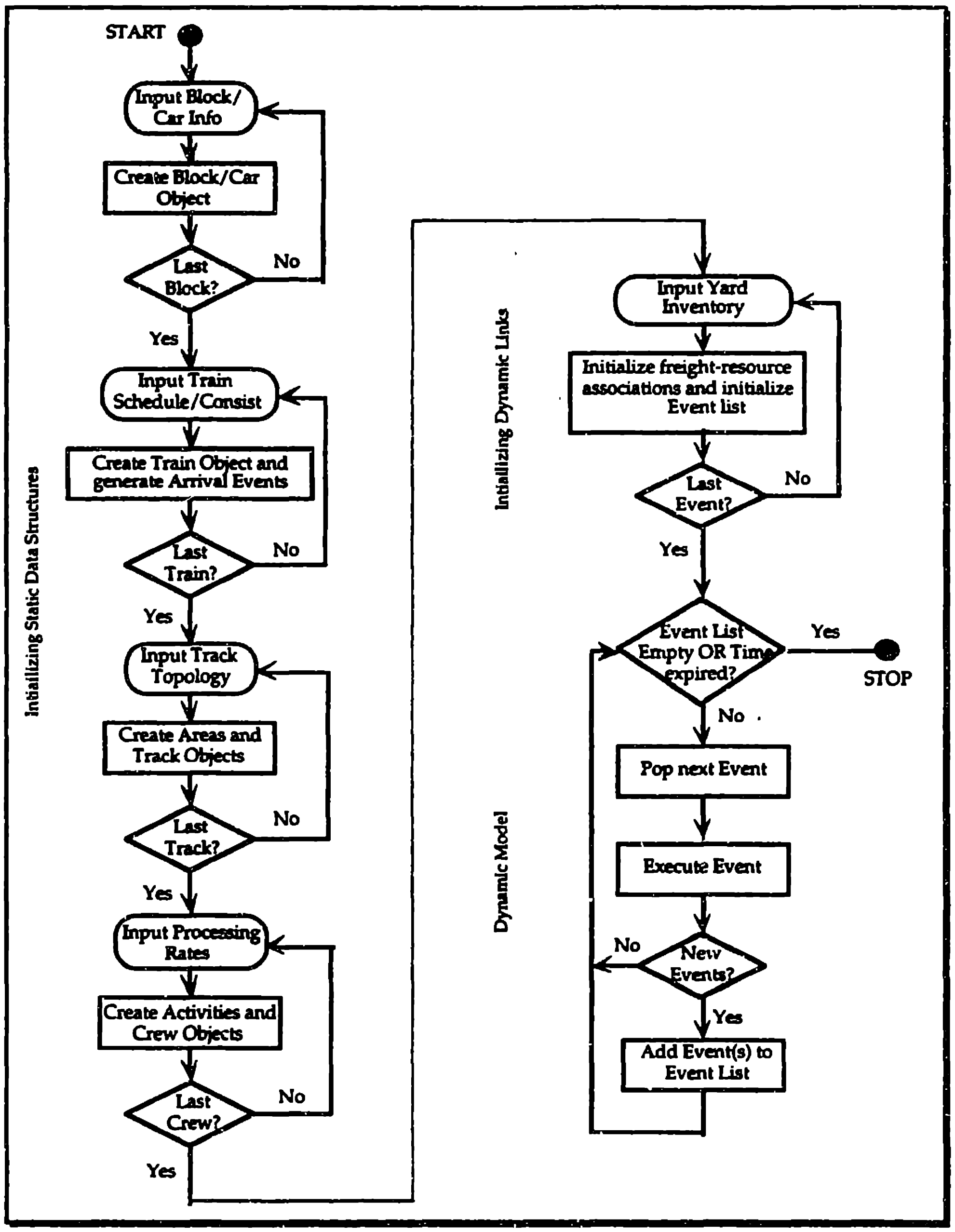

START
Input Block/ Car Info
Create Block/Car Object
Last Block?
No
Yes
Input Train Schedule/Consist
Create Train Object and generate Arrival Events
Last Train?
No
Yes
Input Track Topology
Create Areas and Track Objects
Last Track?
No
Yes
Input Processing Rates
Create Activities and Crew Objects
Last Crew?
No
Yes
Initializing Static Data Structures
Initializing Dynamic Links
Input Yard Inventory
Initialize freight-resource associations and initialize Event list
Last Event?
No
Yes
Event List Empty OR Time expired?
Yes
STOP
No
Pop next Event
Execute Event
New Events?
No
Yes
Add Event(s) to Event List
Dynamic Model

Figure 4.5: Yard Model Algorithm

immediately prior to the description of input file format. To be consistent with general notation of problems that deal with moving commodities through a network, K will denote the set of commodities in the system, namely cars, blocks and trains. The notation for these subsets will be denoted K^C, K^B, and K^T, respectively. For specific trains and blocks, we use K_i^B to denote the set of blocks included in train i, and K_j^C to denote the set of cars included in block j. When describing the physical size of an area Q in the yard, the number of tracks will be denoted n_Q. Likewise, the number of cars (blocks) in a block (train) is denoted by n_c (n_b). This list of inputs will be presented in the order in which they are read by the model (see Figure 4.5)

4.3.1.1. Car and Block List

The first data read by the model pertains to freight scheduled to move through the yard during the current planning period. Using the following notation,

K^B	Set of all blocks entering/departing the yard
ID_i	Identification number for block i
n_i	Number of cars in block i
O_i	Origin of block i
D_i	Destination of block i
K_i^C	Set of cars contained in block i, where $\lvert K_i^C\rvert = n_i$
ID_j	Car identification number
ℓ_j	Length of car j
w_j	Weight of car j
v_j	Value of car j

the format of the input file is shown in the following:

$$\boxed{\forall i \in K^B: \begin{cases} \quad ID_i \quad n_i \quad O_i \quad D_i \\ j = 1,\ldots,n_i: \begin{cases} ID_j \quad \ell_j \quad w_j \quad v_j \\ \quad \vdots \end{cases} \\ \quad \vdots \end{cases}}$$

Thus, we can create objects for all **Cars** that pass through the terminal and all specified **Blocks** that include the cars. The **Blocks** carry the same attributes as the cars, such as length, weight, and value. Each of these measures is treated as additive, so for any block, its attributes are given by

$$\ell_i = \sum_{j \in K_i^C} \ell_j \tag{4.1}$$

$$w_i = \sum_{j \in K_i^C} w_j \tag{4.2}$$

$$v_i = \sum_{j \in K_i^C} v_j \tag{4.3}$$

For length and weight, additivity makes sense. However, it is not so obvious for the measure of *value*. The more general issue of placing a value upon a block calls for additional study, but for the purpose of this model, summing its car values will suffice. Thus, small blocks consisting of low value cars will have considerably lower value than long blocks consisting of high value cars. The grey area lies in comparing the small blocks containing high-value components with long blocks carrying low value components. A separate or, perhaps, secondary measure of the block's value could be expressed as:

$$v_i = \max_{j \in K_i^C} v_j \tag{4.4}$$

but this is not employed in the model. Similar extensions about the aggregation of block values to derive a train's value need to be explored.

4.3.1.2. Train Consist and Schedule

Having created the objects that represent cars and blocks, we can now read the arriving and departing train schedule and the make-up of each train. Make-up in this model is described in terms of the blocks that a train carries. Using the following notation

K^T	Set of trains arriving/departing the yard
ID_k	Identification number for train k
n_k	Number of blocks in train k
O_k	Origin of train k
D_k	Destination of train k
d_k	Departure time of train k from origin
a_k	Arrival time of train k to destination
K_k^B	Set of Blocks contained in train k, where $\lvert K_k^B \rvert = n_k$

we describe the input file for train schedules and consists as having the format:

$$\forall k \in K^T: \begin{cases} ID_k \quad n_k \quad O_k \quad D_k \quad d_k \quad a_k \\ \qquad i=1,\ldots,n_k: \ \{ \ ID_i \\ \quad \vdots \end{cases}$$

Upon inputting this data, the **Train** objects are created and linked to the previously created **Block** and **Car** objects. The lengths, weights, and values of trains are, again, the sum of these values for the **Blocks** contained on the train.

4.3.1.3. Yard Topology

The two previous input files pertain to the creation of objects representing freight and its groupings into trains. The third input file stores the physical description of the terminal and the initial setting for the processing characteristics for each of the **ProcessNodes**. Using the notation

M	Set of main tracks, $n_M = \lvert M \rvert$
R	Set of receiving yard tracks, $n_R = \lvert R \rvert$
C	Set of classification yard tracks, $n_C = \lvert C \rvert$
P	Set of assembly (pull-out) lead tracks, $n_P = \lvert P \rvert$
D	Set of departure yard tracks, $n_D = \lvert D \rvert$
ℓ_i^Q	Length of track i in Q, where $i=1,\ldots,n_Q,\ Q \in \{M,R,C,P,D\}$

we can describe the topology input file as having the following structure

$$\begin{array}{ll} n_M & \ell_1^M \ \ell_2^M \ldots \ \ell_{n_M}^M \\ n_R & \ell_1^R \ \ell_2^R \ldots \ \ell_{n_R}^R \\ n_C & \ell_1^C \ \ell_2^C \ldots \ \ell_{n_C}^C \\ n_P & \ell_1^P \ \ell_2^P \ldots \ \ell_{n_P}^P \\ n_D & \ell_1^D \ \ell_2^D \ldots \ \ell_{n_D}^D \end{array}$$

4.3.1.4. Processing Characteristics

The rate at which processing occurs in the yard also needs to be represented. Mean processing rates, in terms of cars per minute or feet per second, are sufficient. Certain activities contain a fixed processing time component, such as the time for inspectors to "lock" receiving or departure tracks prior to inspections. These fixed times are independent of train length. Thus, we represent the processing time, t_{ij}, linearly for a job i (i.e., a train or block) at a **ProcessNode** j:

$$t_{ij} = \beta_0^j + \beta_1^j \ell_i \quad i \in \{K^T, K^B, K^C\}, j \in \{I, O, H, S, T, P\} \tag{4.5}$$

where the coefficients β_0 and β_1 are defined for each **ProcessNode**.

In addition to processing times, resources that involve crews (i.e., **Activities**) work in finite blocks of time. We can characterize this period by a start time, finish time, and break time for meals.

The structure of the input file, therefore, using the notation

t_c	Cut-off time for assembly (minutes before scheduled departure, 0 otherwise)
m_A	Assembly mode (1, 2)—See Figures 4.10, 4.11
ℓ_i	Length of track i
β_0^j, β_1^j	As defined above
I	Set of Inbound Inspection Crews, $n_I = \lvert I \rvert$
O	Set of Outbound Inspection Crews, $n_O = \lvert O \rvert$
S	Set of Hump Engine Crews, $n_S = \lvert S \rvert$
T	Set of Trim Engine Crews, $n_T = \lvert T \rvert$
H	Set of Hump Tracks, $n_H = \lvert H \rvert$
P	Set of Assembly Tracks, $n_P = \lvert P \rvert$
$(t_1, t_2, t_3, t_4)_i$	Shift start, shift end, break start, break end for crew i,

takes the following form:

$$\boxed{\begin{array}{lllll}
t_c & & & & \\
m_A & & & & \\
n_i & \beta_0^I & \beta_1^I & & \\
 & \left\{\begin{array}{l} i=1,\ldots,n_I\colon\ (t_1,t_2,t_3,t_4)_i \\ \qquad\qquad\qquad \vdots \end{array}\right. & & & \\
n_S & \beta_0^S & \beta_1^S & & \\
 & \left\{\begin{array}{l} i=1,\ldots,n_S\colon\ (t_1,t_2,t_3,t_4)_i \\ \qquad\qquad\qquad \vdots \end{array}\right. & & & \\
n_H & \beta_0^H & \beta_1^H & \ell_i^H & i=1,\ldots,n_H \\
n_T & \beta_0^T & \beta_1^T & & \\
 & \left\{\begin{array}{l} i=1,\ldots,n_T\colon\ (t_1,t_2,t_3,t_4)_i \\ \qquad\qquad\qquad \vdots \end{array}\right. & & & \\
n_P & \beta_0^P & \beta_1^P & \ell_i^P & i=1,\ldots,n_P \\
n_O & \beta_0^O & \beta_1^O & & \\
 & \left\{\begin{array}{l} i=1,\ldots,n_O\colon\ (t_1,t_2,t_3,t_4)_i \\ \qquad\qquad\qquad \vdots \end{array}\right. & & &
\end{array}}$$

Given this static representation, the objects used in this model serve as the backbone for planning, regardless of the type of planning that is done. In the version described here, the model serves as a predictive mechanism, but the static structure will support all types of yard modeling. The remaining topics in this section describe how simulation fundamentals were used to model the dynamic behavior of the system.

4.3.1.5. Yard Inventory

If the initial conditions for the planning period are such that we begin with an empty yard, then using the four previous sets of input will suffice. Given the nature of real-time planning, however, this will never be the case. For planning the current period, ending conditions of the previous period (or the current state of the system) will serve as initial input.

Initial inventory is created in three simple steps: first, a **TrainRoot** object corresponding to the freight is created in the proper form (i.e., **Train**, **Block**, or **Car**); second, associations are created between the **TrainRoot** object and

Resources to which it is assigned at the beginning of the period; and, third, either an **Event** (see Section 4.3.2) is created if the freight is initially being processed or the **TrainRoot** object is added to a queue if the freight is initially idle. The input file must capture *all* blocks and trains and their location, processing status, and time of completion.

We will use the following notation

n_{tr}	Number of trains in-process at the start of planning
n_{bl}	Number of blocks in-process at the start of planning
ID_i	Identification number for Car/Block/Train *i*
P	**ProcessNode,** $\in\{'M','R','I','S','H','T','O','P'\}$
t_i	Time of event completion for Block/Train *i*
I_i^t	Index of track holding Block/Train *i*
I_i^c	Index of crew working on Block/Train *i*
I_i^l	Index of lead track (hump or assembly track) used by Block/Train *i*
n_i	Number of cars/blocks in block/train *i*,

where the lead track is used to store a secondary track used by the train or block. For instance, if a train is being humped, it still occupies a track in the receiving yard, a switch crew, and the hump lead. The input file has the following structure:

$$\boxed{\begin{array}{l} n_{tr} \\ i=1,\dots,n_{tr}: \begin{cases} ID_i \quad t_i \quad \mathbf{P} \quad I_i^t \quad I_i^c \quad I_i^l \quad n_i \\ \qquad\quad j=1,\dots,n_i: \; \{ \; ID_j \end{cases} \\ \\ n_{bl} \\ i=1,\dots,n_{bl}: \begin{cases} ID_i \quad t_i \quad \mathbf{P} \quad I_i^t \quad I_i^c \quad I_i^l \quad n_i \\ \qquad\quad j=1,\dots,n_i: \; \{ \; ID_j \end{cases} \end{array}}$$

To reiterate the purpose of this file, it is to initialize the relationship of trains and blocks to the tracks. Neither the **Block** objects nor the **Car** objects need to be created, since they will have appeared in the block and car input file and created in the first loop of the algorithm. We *do* need to create certain **Train** objects. Since arriving **Trains** are initialized from the arrival schedule in the second loop, trains that are in-process will not be created. They need to be created when the yard inventory file is read. We then create the **Event** corresponding to the completion of processing and insert it into the **Event** list. The details of **Event** objects and the control of the dynamic model through a single list of **Events** is described in the next section.

4.3.2. The Event List

We will describe the **Event** class, the creation and maintenance of the **Event** list, and the execution of **Events**. The **Event** class, as shown in Figure 4.6, contains three primary data elements: the time of the event, the location of the event, and the trains or blocks that are involved in the event. First, we have the time, which represents the *completion* of the event. By storing the completion time, we know when resources become available for processing other freight. The second component is a pointer to the **ProcessNode**, which indicates where the event has occurred. Knowing the **ProcessNode** allows the model to reassign that node's resources and to forward the **TrainRoot** object to the correct **ProcessNode** next in the order. Finally, we have a pointer to the **TrainRoot** object itself. The **TrainRoot** object stores information about the specific **Resources** used and frees them upon completion of the **Event.**

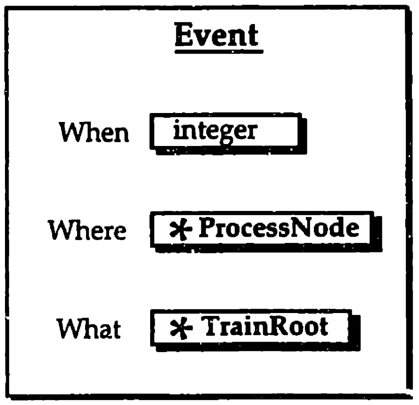


Figure 4.6: Class structure for **Event**

When an **Event** is created, we calculate its duration and, therefore, its completion time. The **Event** object stores only the completion of the **Event** since, whenever an **Event** is placed into the list, we know its duration is the time from when it was placed on the list until the completion time. At the beginning of the **Event**, all resources utilized, including track and crews, are given a *busy* status. This prevents them from being assigned to other freight until the **Event** is completed and the **Resources** are freed.

Inserting an **Event** into the list is according to its completion time. An **Event** at the top of the list indicates that the **Event** has been completed. The **Resources** consumed by the **Event** become available for reassignment to awaiting jobs. The

Table 4-2: Summary of Event Processing for **ProcessNodes**

Event	ProcessNode	TrainRoot	Event Spawned from:	
			Reassigned Resources at current ProcessNode	TrainRoot forwarded to next ProcessNode (and not queued)
Train arrives at Terminal	MainTrack (M)	Train	None	Train pulls into receiving yard
Train pulls to stop on assigned track	ReceivingYard (R)	Train	**Track** in MainTrac'	Inbound inspection
Completion of inspection	InboundInspection (I)	Train	Inspection **Crew**	Hump engine moves and hooks to train
Completion of hump engine link to train	Switch (S)	Train	None	Send train to hump for classification
Humping process complete	Hump (H)	Train	ReceivingYard **Track**; Switch engine **Crew**; Hump **Track**	Event for the arrival of cars onto Class Yard track
Block Arrives on classification track	ClassificationYard (C)	Block	None	If **Block** is the final connection for an outbound train, then **Event** for trim engine and assignment of assembly lead
Completion of trim engine link to train	Trim (T)	Block	None	Usage of assigned assembly lead
Completion of block pull through assembly track	Assembler (P)	Block	ClassificationYard **Track**; Trim engine **Crew**	If last block in outbound train, the create **Event** for Outbound Inspection
Completion of inspection	Outbound Inspection (O)	Train	Inspection **Crew**	Departure
Train clears the terminal	Departure (D)	Train	DepartureYard **Track**	None

Train, **Block**, or **Car** that was involved in the completed **Event** moves onto the next processing function, if **Resources** are available. New **Events**, corresponding to reassignment of **Resources** or the movement of the **TrainRoot** to the next **ProcessNode**, are created upon the completion of the original **Event**. The only exception is the **Event** of train departure, where no new **Events** are created because the freight will have left the system. Due to this regeneration, **Events** will be created until all trains depart the yard or until the current planning horizon has expired. For each **Event** type Table 4-2 lists the **ProcessNode** location; the resources that are assigned, freed, and reassigned; and new **Events** that generated as the result of the **Event** completion.

The initialization of the **Event** list is done in one of two ways. First, we have **Events** carried over from the previous planning period. These are either stored in memory or provided as part of the inventory input file. The model currently relies upon input from inventory files for **Event** list initialization. Second, when the train arrival schedule is read, an **Event** is generated for each arrival and added to the **Event** list. The **Event** then spawns string of **Events** that corresponds to the movement of the train, its blocks, and its cars through the set of **ProcessNodes**.

Features of OOM assist the implementation of this concept. Use of *polymorphism* allows **Events** of different types to be handled through the use of *virtual functions*. These are functions that respond differently to different types of objects. When an **Event** is removed from the list, there is no way of knowing which **ProcessNode** corresponds to that **Event**. Polymorphism provides a mechanism for case handling at *run-time*, where the behavior, or assignment of resources to freight, is determined according object types. The next section describes the algorithms that perform these assignments.

4.3.3. Sequencing Algorithms

We have now developed an understanding of the methods used for creating the dynamic portion of the model, but have not developed a notion about how the **Resources** are assigned to **TrainRoots**. The underlying decision rule is First In First Out (FIFO), also known as First Come First Served (FCFS). Under such a rule, if every job could be processed immediately by each **ProcessNode**, then the generation of **Events** would be straightforward. However, the limitations on resources require many additional cases which much be handled. The purpose of this section is to present the algorithms that are used to handle FIFO processing in light of these special cases.

FIFO sequencing is enforced both from the perspective of assigning trains to resources and from the perspective of assigning resources to trains. The first train, or block, which arrives at a **ProcessNode** is linked to the first available resource. If resources are not available, the first resource that becomes available is assigned to the first eligible train, or block, in the queue (FIFO from the perspective of the resources *and* the freight). However, if resources are available, the choice of assignment is determined by selecting the first feasible assignment found in the *list* of resources maintained for each **ProcessNode**. Since the orderings within these lists are static, selection is not based upon which resource has been idle for the greatest amount of time (which would be equivalent to a FIFO assignment).

Regardless of the algorithms used, resource assignment and re-assignment are triggered by the processing of an **Event**. The general procedure shown in Figure 4.7 is performed for **Event** processing. The first step is to free resources that were used in processing the job. If the current **ProcessNode** has freight waiting to be processed, then the free resources are re-assigned to the awaiting jobs, if such an assignment is feasible. An example of infeasibility is trying to re-assign a receiving track to the next train in the queue and train is too long for the track.

After the reassignment of **Resources**, a new **Event** is created and occurs at the current **ProcessNode**. Next, the **TrainRoot** from the current **Event** is moved to the next **ProcessNode**. If a queue exists, then the FIFO rule dominates and the **TrainRoot** is added to the queue. If the queue is empty, then the train is assigned to a resource in the next **ProcessNode**. If there is no queue and a feasible assignment exists, a new **Event** is created and added to the **Event** list. Otherwise, the **TrainRoot** is added to the queue and no **Event** is created. The completion time of the new **Event** is based on a virtual function, ProcessingTime(), which takes the form of equation (4.5).

```
CurrentTime ← CurrentEvent.when
CurrentProcessNode ← CurrentEvent.where
CurrentTrainRoot ← CurrentEvent.what

free CurrentProcessNode.resources used to process CurrentTrainRoot

if CurrentProcessNode.queue is nonempty
    remove NewTrainRoot from queue
    assign CurrentProcessNode.resources to NewTrainRoot
    create NewEvent
    NewEvent.time ← CurrentTime + ProcessingTime(CurrentProcessNode,
                                                  NewTrainRoot)
    NewEvent.where ← CurrentProcessNode
    NewEvent.what ← NewTrainRoot
    add NewEvent to the Event list
end if

if NextProcessNode.queue is nonempty
    add CurrentTrainRoot to NextProcessNode.queue
else
    if NextProcessNode.resources not available
        add CurrentTrainRoot to NextProcessNode.queue
    else
        create NewEvent
        NewEvent.time ← CurrentTime + ProcessingTime(NextProcessNode,
                                                      CurrentTrainRoot)
        NewEvent.where ← NextProcessNode
        NewEvent.when ← CurrentTrainRoot
        add NewEvent to the Event list
    end if
end if
```

Figure 4.7: General Event Processing Subroutine

Next, we will consider the algorithms that are used to assign resources to jobs. The FIFO rules still apply, but there are special cases to consider for each node. First, we will consider is the assignment of trains to receiving yard tracks, a straightforward assignment given in Figure 4.8:

```
scan list of receiving tracks
    if currentTrack is FREE
        if track length ≥ train length
            link train to currentTrack
            return flag
        else
            store currentTrack length for possible train double-over
        end if
    else
        consider next track in list
end scan

return NULL
```

Figure 4.8: Assignment Algorithm for Receiving Track Assignment

Double-over is shown in this algorithm but is not implemented. This is the practice of splitting the inbound arrivals onto multiple receiving tracks if sufficient capacity does not exist on a single track. The algorithm in Figure 4.8 shows a single scan of all tracks to see if there is a free track with sufficient capacity. The total free capacity of all tracks is calculated in the first pass and, if sufficient capacity exists on multiple tracks, would be used in a second pass to determine how to split the train, if necessary. In the current implementation, only the single pass is used. This enforces the assignment of a train to a single receiving track.

The algorithm for both inbound and outbound inspection crew assignment is equally simple and is given below in Figure 4.9:

```
scan list of inbound inspection crews
    if currentCrew is FREE
        link train to currentCrew
        return flag
    else
        consider next crew in list
end scan

return NULL
```

Figure 4.9: Assignment Algorithm for Inspection Crew Assignment

The hump assignment is made concurrently with the assignment of the inbound cars to their classification tracks. In other words, we can not begin the humping process unless there is room in the classification yard to store all the cars of the humped train. The algorithm for the assignment of trains to hump and the cars to classification tracks is shown in Figure 4.10. Here, the term inbound blocks refers to a group of cars connecting to the same outbound block.

```
if the hump is FREE
    for all blocks on inbound train
        if outbound block has not been assigned a classification track
            if outbound block can be assigned a classification track
                assign outbound block to classification track
            else
                clear all classification track assignments made for
                        blocks on this inbound train
                pick next train in queue
                return
            end if
        end if
    end for
    link inbound blocks to assigned classification tracks
    link train to hump
end if
```

Figure 4.10: Hump Assignment Algorithm

The assignment for outbound train assembly is more difficult than the hump assignment. This is because several assembly options exist, including:

1) Assemble one train at a time, using all available trim engines;
2) Assemble several trains at once, using a single engine per train;

Combinations of the two may exist, but the model treats only these cases. As noted in the discussion of the processing characteristics file, an input flag indicates which assembly mode is being used.

Depending on the presence of cut-off times, the assembly process is triggered in one of two ways. First, there is the one-hundred percent rule. This method assembles an outbound train only if all cars are in-place on their assigned classification tracks. When the train is assembled, it is prepared for departure immediately and departs when the preparations have been completed.

The second rule implemented is assembly at a fixed cut-off time, which is commonly used to enforce adherence to the departure schedules. Only the cars in the classification yard by the cut-off time will be assembled into the outbound train. If all cars are situated in the classification yard before the cut-off time, assembly begins immediately, just as in the case of one-hundred percent connection. The only difference is that in the case of cut-off times, the train is held in the yard until its scheduled departure time. This allows tracks in the classification yard to be available to store other blocks. Given these triggers, we can introduce the algorithms for the two modes of assembly.

Figure 4.11 shows the first case of assembly, in which a single train is built using all available trim engines. The first requirement for starting assembly is for a departure track to be available. If not, the train is added to a queue for the next available departure track. If one is available, the algorithm cycles through all blocks connecting to the outbound train. The blocks are selected according to *station order*, which is defined exogenously and specifies the exact ordering of blocks on the outbound train. Each pass through the loop assigns a trim engine to pull the block to the departure yard and an assembly track (or lead) upon which to pull it. If either of these resources are unavailable, the block is placed into the proper queue to await resource assignment. When resources are finished with their current job, they look to the queue to see if any jobs are waiting. This enforces the FIFO processing order. The main loop cycles until no blocks for the current outbound train exist, at which time the train is passed to the outbound inspection **ProcessNode**.

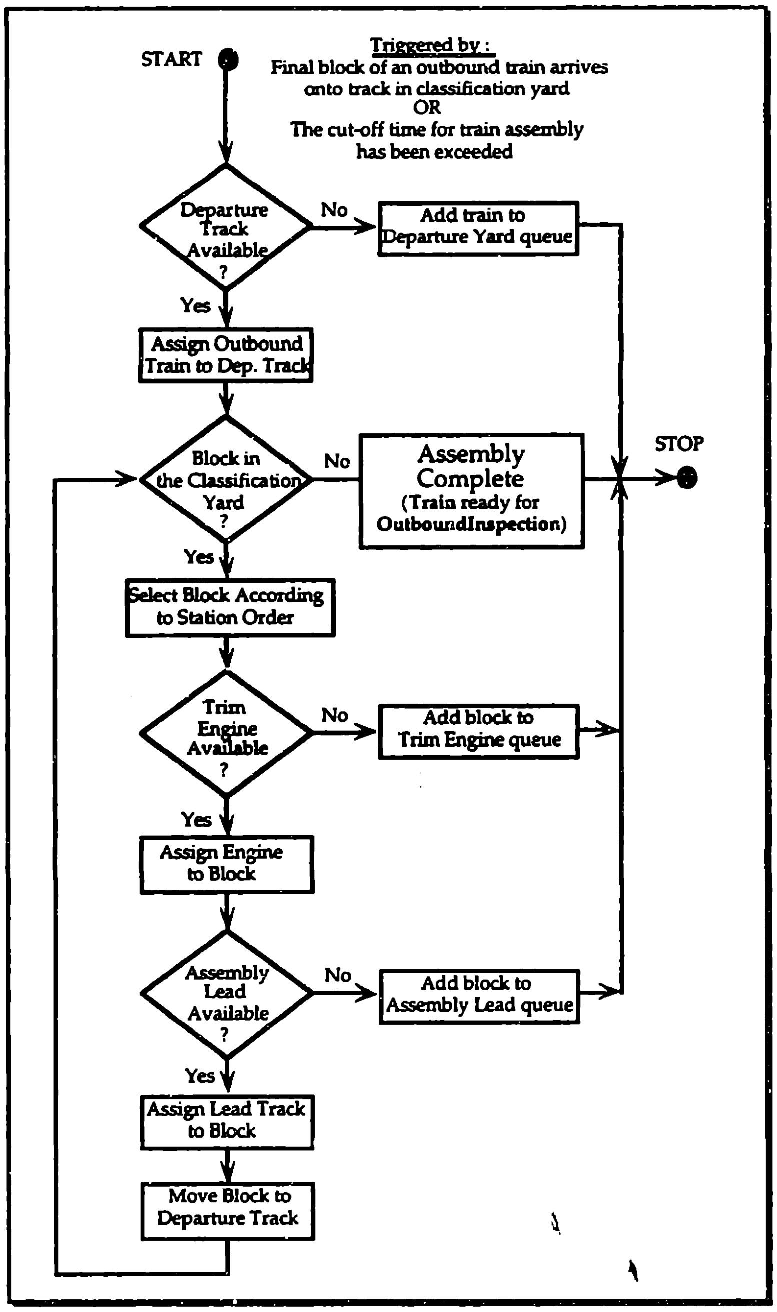

START
Triggered by :
Final block of an outbound train arrives onto track in classification yard
OR
The cut-off time for train assembly has been exceeded
Departure Track Available ?
No
Add train to Departure Yard queue
Yes
Assign Outbound Train to Dep. Track
Block in the Classification Yard ?
No
Assembly Complete (Train ready for OutboundInspection)
STOP
Yes
Select Block According to Station Order
Trim Engine Available ?
No
Add block to Trim Engine queue
Yes
Assign Engine to Block
Assembly Lead Available ?
No
Add block to Assembly Lead queue
Yes
Assign Lead Track to Block
Move Block to Departure Track

Figure 4.11: Single Train Assembly Algorithm

For the second case, Figure 4.12 depicts the general flow of the algorithm, which is similar to the first case. The primary difference is that the trim engine is assigned to process an entire train. This assignment lies outside the loop that cycles through all blocks. Each time a block is ready to be pulled, an assembly lead must be assigned for the engine to pull it from the classification yard to the departure yard. To illustrate this, consider having two outbound trains to be built, two trim engines, two departure tracks, and a single assembly track. Under Case II assembly, each train receives its assignment to a departure track and an assignment to a trim engine. As the two engines begin to pull blocks, they must compete for the assembly track. This algorithm assigns the assembly track in FIFO order by block.

To this point in the chapter, the processing described has related to freight connecting from inbound trains to outbound trains. Although this is our primary interest, recall from Chapter 2 that other types of traffic may enter the yard, including pass-through trains and inspection trains. The **ProcessNode** structure allows these cases to be easily handled. Each type of train will be routed through a subset of the **ProcessNodes** that were described for connecting freight.

For instance, trains passing through a yard with a by-pass track will be handled by the **MainTrack** node. If there is congestion in the yard, the train is added to the queue in the usual FIFO to await a track assignment. The processing time would be dependent upon the fact that this is a pass-through train. For trains making connections, this time is simply the time it takes for the train to move from the main track into the receiving yard. Pass-through trains, on the other hand, will occupy the main track for the train's entire passage through the yard. Since the type of train is not known until its arrival **Event** reaches the top of the **Event** list, the model determines processing times at run-time using virtual functions.

The example using a by-pass track is simple in that it only involves trains passing through a single **ProcessNode**. In the case where no by-pass track exists and the main tracks are routed through the yard, more **ProcessNodes** are involved in coordinating the pass-through train with other trains and blocks in the yard. The ordering of **ProcessNodes** for each train type is a well-defined portion of the model. The same is true for inspection trains, which must be routed to either the departure yard or the receiving yard to have an inspection performed. This involves the assignment of tracks and inspection crews. Finally, if these other train types are to receive higher priority than connecting trains, they will be assigned to the **ProcessNode** queues with priority, while the connecting trains are inserted into the same queues according to the FIFO discipline.

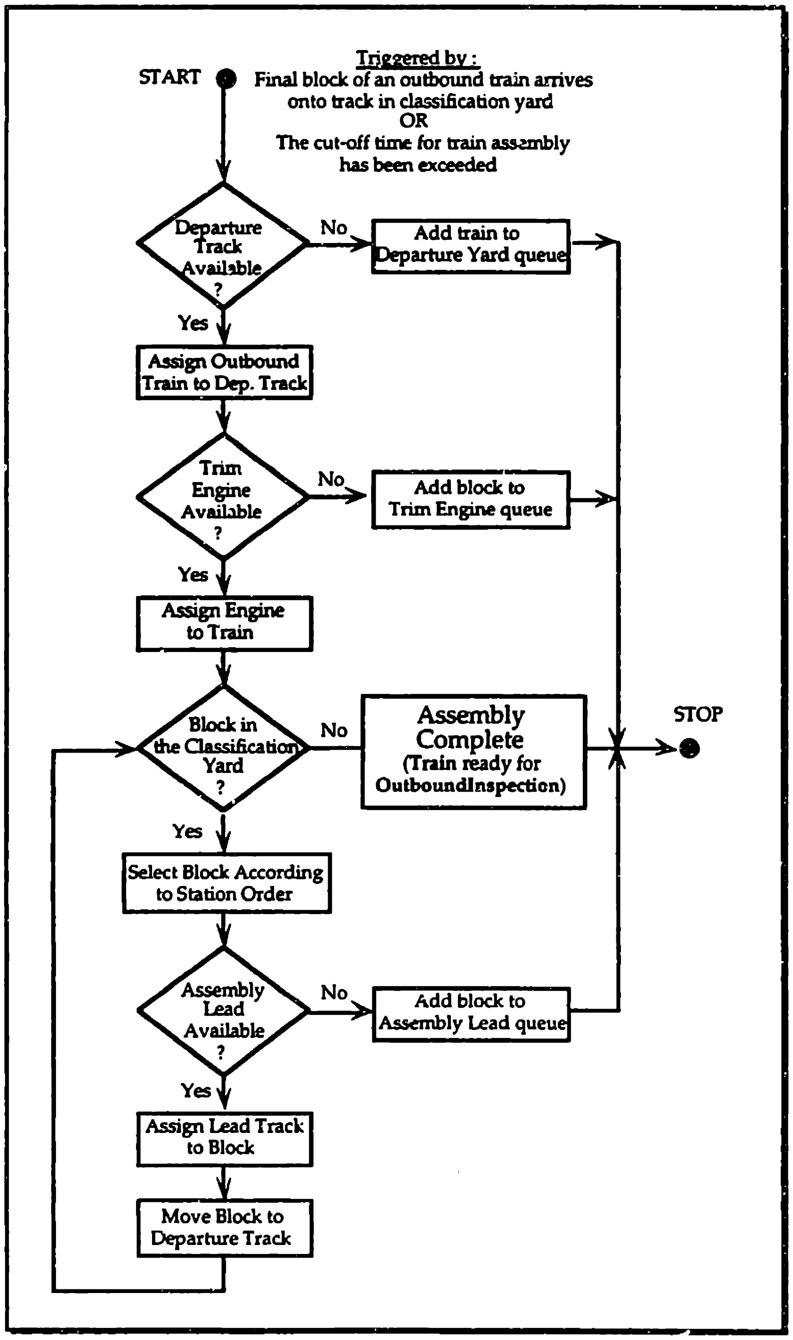

Triggered by :
Final block of an outbound train arrives onto track in classification yard
OR
The cut-off time for train assembly has been exceeded
START
Departure Track Available ?
No
Add train to Departure Yard queue
Yes
Assign Outbound Train to Dep. Track
Trim Engine Available ?
No
Add block to Trim Engine queue
Yes
Assign Engine to Train
Block in the Classification Yard ?
No
Assembly Complete (Train ready for OutboundInspection)
STOP
Yes
Select Block According to Station Order
Assembly Lead Available ?
No
Add block to Assembly Lead queue
Yes
Assign Lead Track to Block
Move Block to Departure Track

Figure 4.12: Multiple Train Assembly Algorithm

4.3.4. Running Time

It is important to understand that the implementation of the yard model is not tuned for efficiency. The main goal of the research was to create a mechanism giving quick predictions of put-through time, based upon arrival and departure schedules of trains and their consists. Supporting an object-oriented programming paradigm requires additional overhead that detracts from the efficiency of the model. The benefits of using object orientation outweigh the burden imposed by this overhead because of the convenient encapsulation of the data associated with the trains and yard resources.

Getting a grasp on the complexity of the model involves counting the **Events** that are executed and the assignments that are made to generate those **Events**. Processing each **Event** requires a nearly identical amount of computation and data structure updates. Though some processing is deferred into queues, this deferral does not add any additional computation to the model. From Table 4-2, five **ProcessNodes** generate **Events** for inbound trains, three for outbound blocks, and two for outbound trains. The total number of events generated will be linear in these input sizes. Since all assignments are FIFO, searching for resource-to-freight assignments is, in the worst case, $O(n_C n_B)$, where n_C is the number of classification tracks and n_B is the number outbound blocks.

In other words, as the number of arriving and departing trains and blocks increase, the model run time does not explode. To provide a subjective description of running time, the model will arrive at a prediction of put-through times in under one second for any problem of reasonable size run on a PC or a workstation-class computer. Thus, the use of this model, if it can be shown to be an effective *predictive* mechanism, would be a viable component of the network planner described in Chapter 3. This yard model would provide connection times used by the network planner to create schedule objectives for the network.

4.3.5. Visualization

A model that describes activities in a tabular format does not present the user with an understandable picture of the system behavior. Originally designed to assist debugging efforts, the visual layer of this model provides a means for "seeing" the activities. Screen snapshots of the model's visual layer are shown below. The running time of the visualized version has been slowed to allow the user to see the animation. Instead of taking under a second to simulate the entire day, the visualized version takes several minutes.

The visual representation of the static structure of the yard is shown in Figure 4.13. The upper portion shows the physical relationship of the different areas of the yard, while the bottom portion of the screen shows that length of each track in the three primary areas. This particular run of the model is based upon a layout similar to the one shown in Figure 2.1. Here, there is double track entering the yard, eight tracks in the receiving yard, a single track over the hump, twenty-five classification tracks, a single assembly lead track, and six departure yard tracks.

At the bottom edge of the top drawing, there is a single track shown for through-traffic. This represents a by-pass track that does not interfere with the other activities in the yard. For cases where such a track does not exist, the through-traffic would pass through designated high-speed tracks in the yard, but their movement would be coordinated and sequenced with the other freight vying for those tracks.

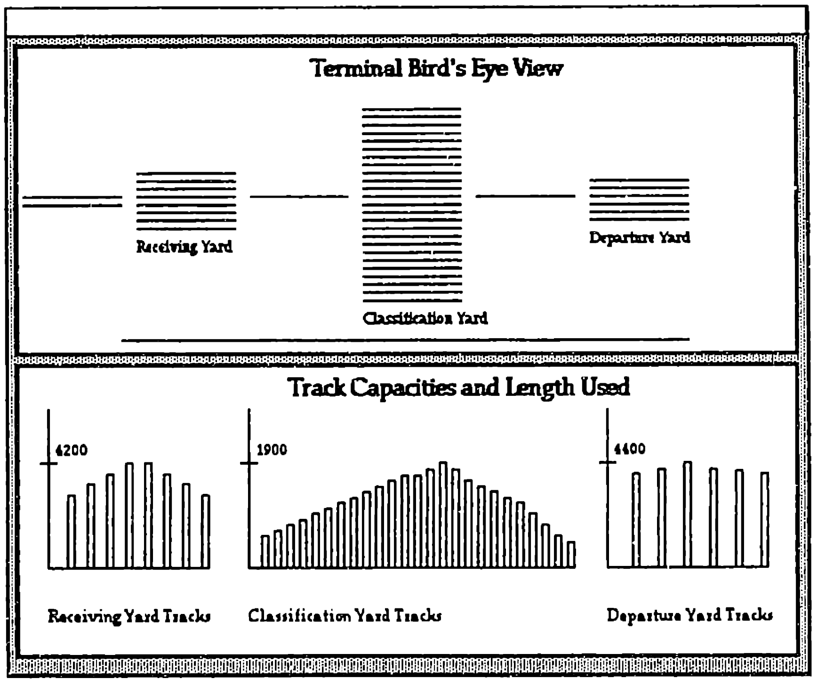


Figure 4.13: Visualization of static layout of a terminal

Having established this static representation of the yard, we begin to move freight through the yard. In the run shown in Figures 4.14a and 4.14b, we have started with an empty yard. As trains enter the yard, the visual layer animates their movement onto the assigned track. Each rectangle on the train represents a block of cars. As the trains take their position in the yard, the bars in the lower half of the window serve as moving gauges of used track capacity. When a train departs from a track, the gauge returns to its original "empty" setting.

The current time, which is driven by the occurrence of **Events**, is shown in the upper left corner of the screen. Recall that the underlying model is a discrete event representation of a continuous-time system. Times between consecutive **Events** will generally not be equally spaced. One challenge in creating the visual layer was to interpret the inter-event times properly and translate them to ensure a smooth animation. Colors are also utilized in the model to indicate when a train or block is going through different processing steps, where blue indicates inspection, green indicates that an engine is moving to the freight for hook-up, black is used when the freight is moving, and red is used when the freight has been queued for its next processing function.

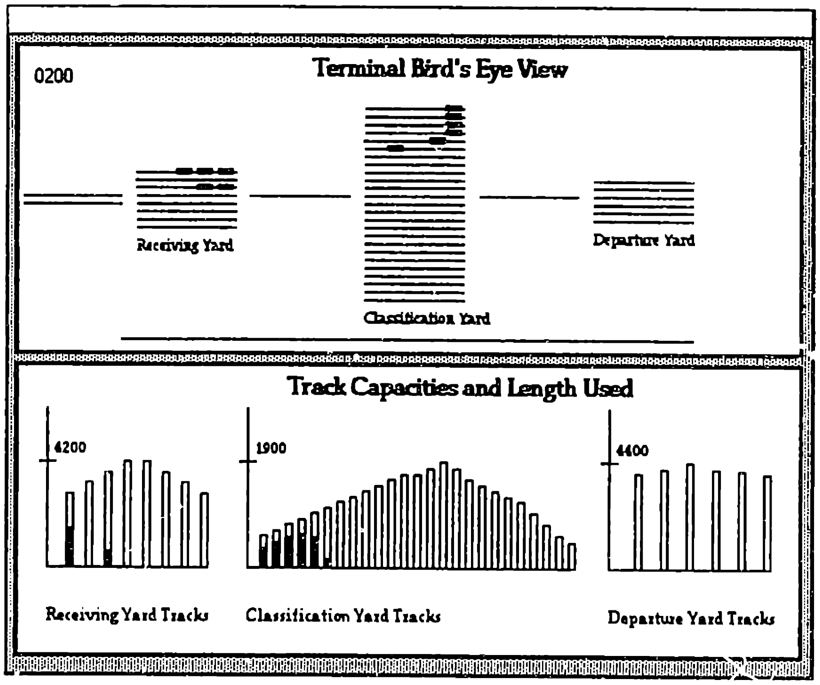


Figure 4.14a: Dynamic Model, Screen Dump #1

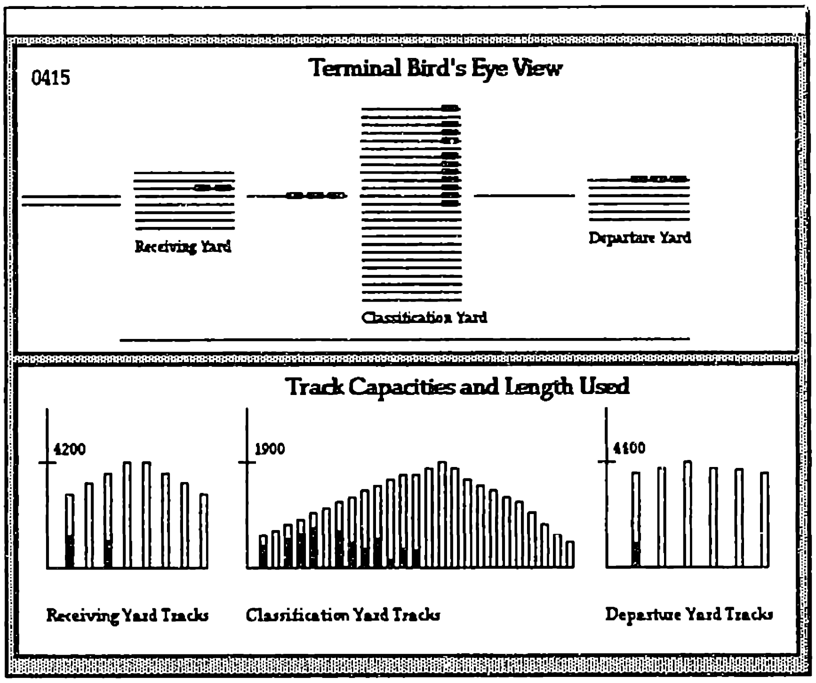


Figure 4.14b: Dynamic Model, Screen Dump #2

4.4. Extensions to the Framework

Using an object oriented design provides a convenient encapsulation of the data associated with terminal operations. The static structures could be utilized in network planning models (see Chapter 5) and as the basis for sequencing algorithms (see Chapter 6). The dynamic structure of the model provides a foundation for a complete simulation, an algorithm testbed, and a decision support tool for yard managers, each of which are discussed below.

4.4.1. Pure Simulation

The same dynamic structure presented in Section 4.3 can be used as the basis for a full-blown simulation. Stochasticity can be introduced in several areas. The first is in the processing times of the **ProcessNodes**. Rather than using only deterministic coefficients, we could also specify the variance and make assumptions about the distribution of processing times (e.g., Gaussian, exponential, uniform). In addition, uncertainty could be introduced into the

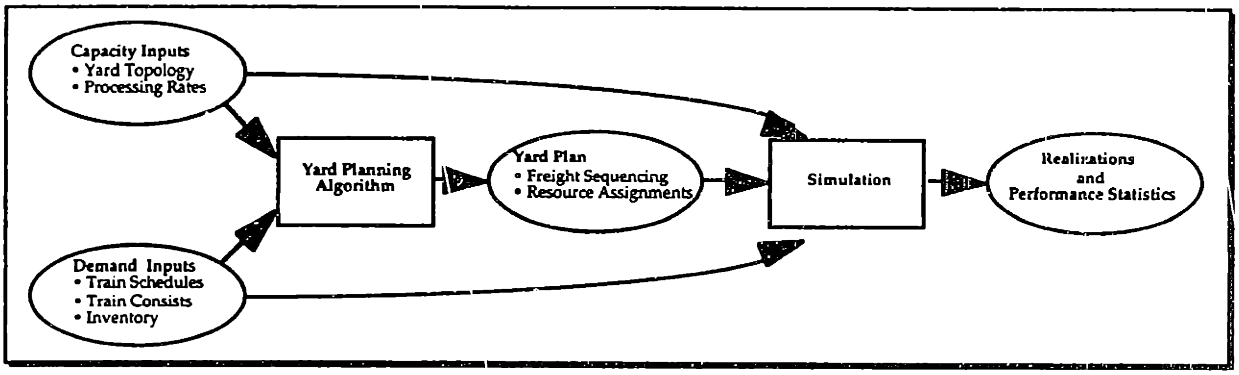


Figure 4.15: Yard Planning Algorithm Testbed-Conceptual Design

arrival process. The times specified in the arrival schedule would represent the *expected* time with some possibility of variation. Other types of random events that could be generated include changes in processing times due to weather or the random elimination of resources due to events like in-yard derailments or crew illness. The degree to which stochasticity should be included would depend upon the analysis required to be performed by the simulation.

4.4.2. Algorithm Testbed

In the development of yard planning algorithms, a testbed would serve as a useful device to evaluate the algorithms under various operating conditions. This would allow a head-to-head comparison of the algorithms, which may include simple heuristics used by yard masters, heuristic planning algorithms, and optimization models.

The testbed would provide a repeatable environment in which to test yard planning algorithms. It should consist of two primary phases: a planning phase and a realization phase (see Figure 4.15). The planning phase involves using the planning algorithms to determine processing sequences based on a given scenario of arrivals, departures, and consists. The realization phase is a simulation that tests the sequences under conditions of uncertainty. Both phases rely on two primary inputs: the schedule of train arrivals and departures, or *demand*, and the physical configuration of the terminal, or *capacity*. Balancing the anticipated demand with available capacity is the role of any planning algorithm for the yard. Testing these algorithms under conditions of uncertainty is the role of the testbed. An example of this type of testbed has been implemented for analyzing ground holding algorithms in the context of an Air Traffic Flow Management system. Design of this testbed is provided in Robinson [1992] and Hocker [1994].

4.4.3. Real-Time Decision Support Tool

The object oriented framework has been designed to provide the static structures that can be used as part of a decision support tool at terminals. This tool is equivalent to the *yard controller* described in Chapter 3. The structures provide an easy means for encapsulating the data from any central information management system, and a convenient structure upon which a graphical layer can be developed. This will provide the capability for yard management to visualize the current state of the yard and the output of any planning algorithm.

4.5. Chapter Summary

This chapter offers a general design for developing yard models to be used within a broader, real-time network planning and control system of models. Such a design must begin with an understanding of the data required to support the decision making. Using Object Oriented Modeling (OOM), we created a representation of yards and trains that will support a variety of modeling methods, such as heuristic dispatching rules, heuristic planning algorithms, and optimization models. Upon these static structures, we developed a dynamic model using a deterministic simulation approach. Finally, possible extensions to this modeling technique were discussed.

5. Modeling the Network

The previous chapter presented a model for a single yard. Since the overall goal of this thesis is to explore yard models in the context of real-time network planning, this chapter extends the single yard modeling structure to a network. This chapter begins with a notional decomposition of the network into smaller, tractable subproblems. To gain some understanding of the size of a real-world problem using this decomposition, we will present an example from one division of a Class I railroad. Next, the object model structure presented in Chapter 4 is tailored to represent the network and the trains scheduled to move through it. Finally, as an initial step towards planning the activities in the network, a *coordination heuristic* is introduced and demonstrated with two examples.

The network planning presented in this chapter can be related back to the hierarchical planning framework described in Chapter 3. At the operational level, the network planning model develops system-wide schedule objectives and passes them to the lower level decision support tools used for line dispatching and yard sequencing. The network planner consists of three components: a line planning component, a yard planning component, and a network coordination algorithm (see Figure 5.1). It is the third of these components that is responsible

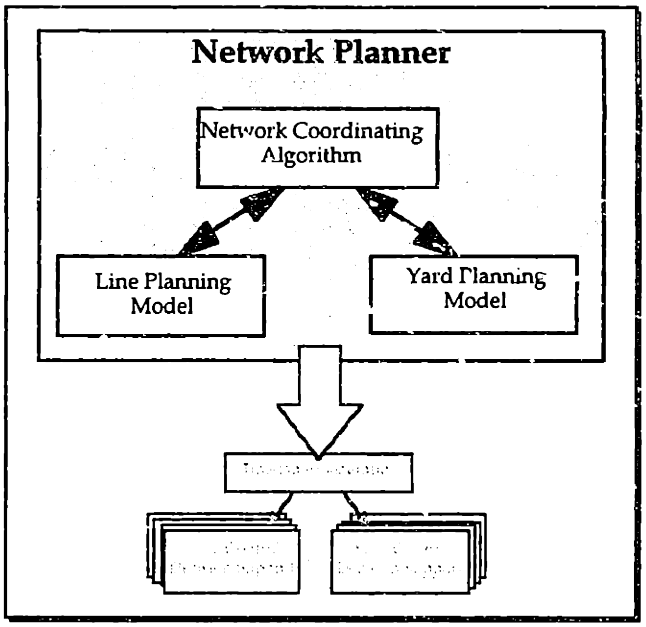


Figure 5.1: Network Planning Model

for "overseeing" the creation of the network-wide schedule objectives, relying upon the input provided by the other two components.

The object oriented software architecture described in this chapter enables input and output between the model components, without regard to the specific algorithms that are used. Thus, the structure can serve as a testbed, allowing different planning algorithms to be tested in the network environment. In the examples presented at the end of the chapter, the testbed uses the coordination heuristic (from Section 5.2.1), the deterministic yard simulation (from Chapter 4), and fixed line-haul times in lieu of line planning. Two examples are presented using this set of models.

5.1. Decomposing the Network

Real-time operational network planning requires higher fidelity modeling than tactical or strategic planning, and the formulation of this problem becomes intractable as the network size grows. Dividing the planning problem into smaller problems is the only hope to solve this problem quickly. This section presents a notional decomposition and an example of a realistic problem size.

5.1.1. Planning Nodes

The logical breakdown into subproblems is to consider lines and yards separately. This is true for two reasons. First, historically, the railroads have managed networks based upon the separate functionality of these two components. Second, the *de facto* development of real-time planning tools has been along the same lines and the network planning model should be decomposed to exploit the existing models.

We will give each planning subproblem the generic label *planning node*, where the planning algorithm invoked for a node will depend upon whether it is a yard or a line. Currently, real-time planning algorithms exist for the meet and pass decisions on the line (see Jovanovic and Harker [1991] and Draper [1994]). A limited number of operational yard sequencing models exist, with heuristic planning methods (as developed in Ferguson [1993]) seeming to be an attractive candidate for real-time yard planning in this network context. Initially, the yard model from Chapter 4 will be utilized in this network planning structure. As part of future research it will replaced by an adaptation of the Ferguson model.

With this definition of planning nodes, the network representation takes an atypical form. The traditional means for representing a rail network is to consider yards as nodes (or vertices) and lines as arcs (or edges) and to develop planning models using network optimization or multicommodity flow techniques (e.g., Bodin, *et al.* [1980], Assad [1980b], Crainic, *et al.* [1984], Keaton [1989, 1992]). With the planning node approach, the network will be described using nodes to represent the planning subproblems and arcs to simply represent the adjacency relationships between nodes.

To illustrate the planning node representation, consider the rail subnetwork in Figure 5.2a, consisting of six line segments and a single yard. The node adjacencies, represented by the undirected (grey) arcs in Figure 5.2b, are based on the possible flows from line to line, line to yard, or yard to line. For example, the junction of Lines 1 and 2 occurs at the west boundary of the yard. Northbound traffic moving on Line 2 may turn westward onto Line 1 or eastward into the yard. As a result the adjacency of planning nodes must have Line 2 adjacent to both Yard 3 and Line 1. Similarly, at the east junction of Lines 4, 5, 6, and 7, pairwise adjacency must be established if traffic is allowed to flow in any direc-

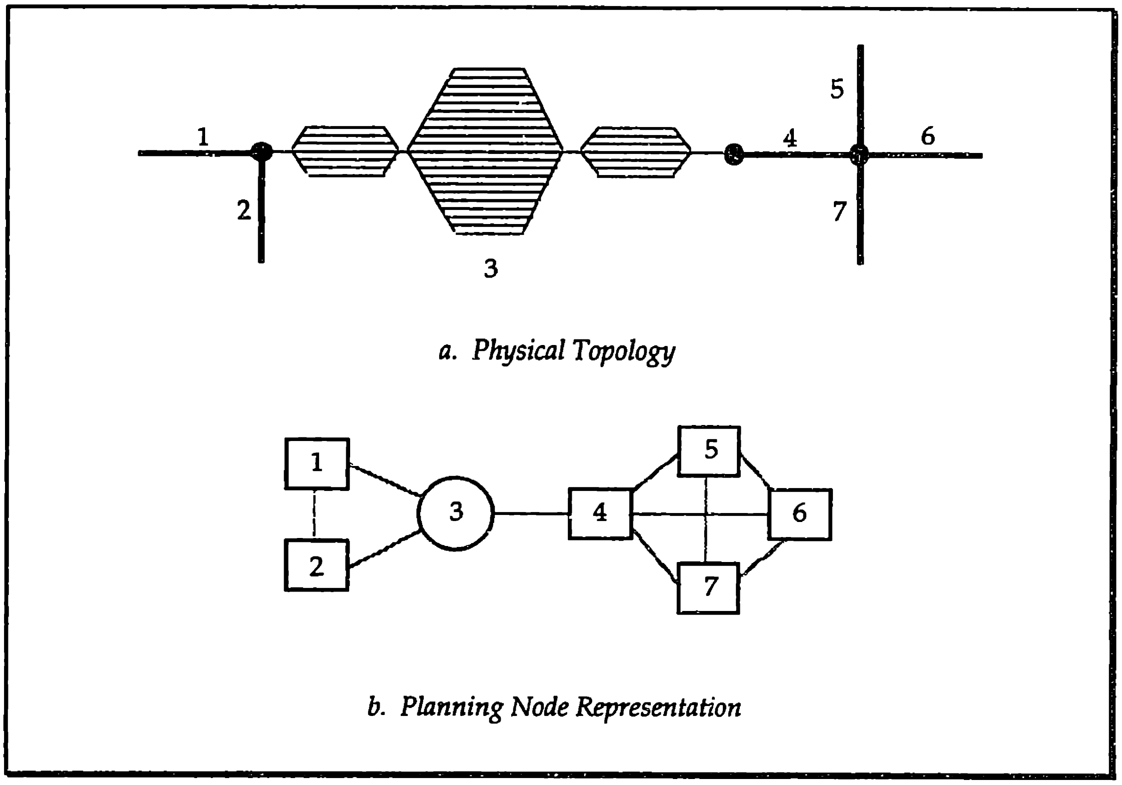


Figure 5.2: Translation of Physical Network to Planning Nodes

tion through that junction. The resulting planning node adjacency is seen in Figure 5.2b.

The adjacencies provide information for determining traffic and train routing. Since the network planner takes such routing information as given, the adjacency information will *not* be used in the planning described in this chapter. However, an important function of real-time network planning is recovering from unplanned changes in the system. Part of that recovery involves coordinating with car scheduling systems to determine re-routing of freight in a manner different from the original routing. Including the adjacency lists in the network representation enables this type of recovery planning to be implemented.

5.1.2. Network Example

As a simple example in demonstrating the size of a rail network, one division of a major Class I railroad is placed into this planning node structure. From network maps and published timetables[1], the physical layout of the division is reconstructed and shown in Figure 5.3. It is not drawn to scale but is intended to illustrate the size of the planning problem. The division consists of 31 yards[2] and 47 line segments. Line and yard sizes will vary and a planning node must exist for each.

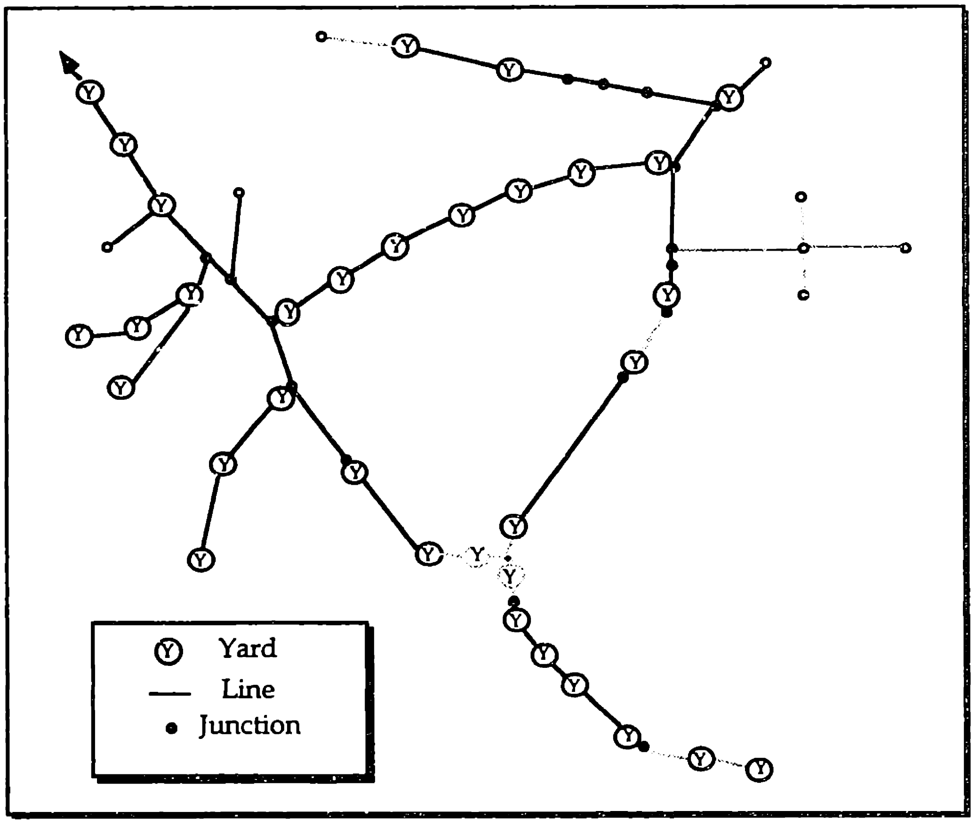


Figure 5.3: Physical Layout of Sample Class I Railroad Division

[1] The term "timetable" is a misnomer. These documents are the published track topologies, descriptions of speed limits, operating restrictions, etc., for a set of lines. Also included in the timetables is information regarding siding locations and lengths, and yard limits.
[2] The specific breakdown of large, hump yards versus smaller, flat switching yards is not given. It is assumed the ratio of flat yards to hump yards is high, as typical in most railroads. Work at the flat yards is characterized primarily by consolidation and distribution of local traffic.

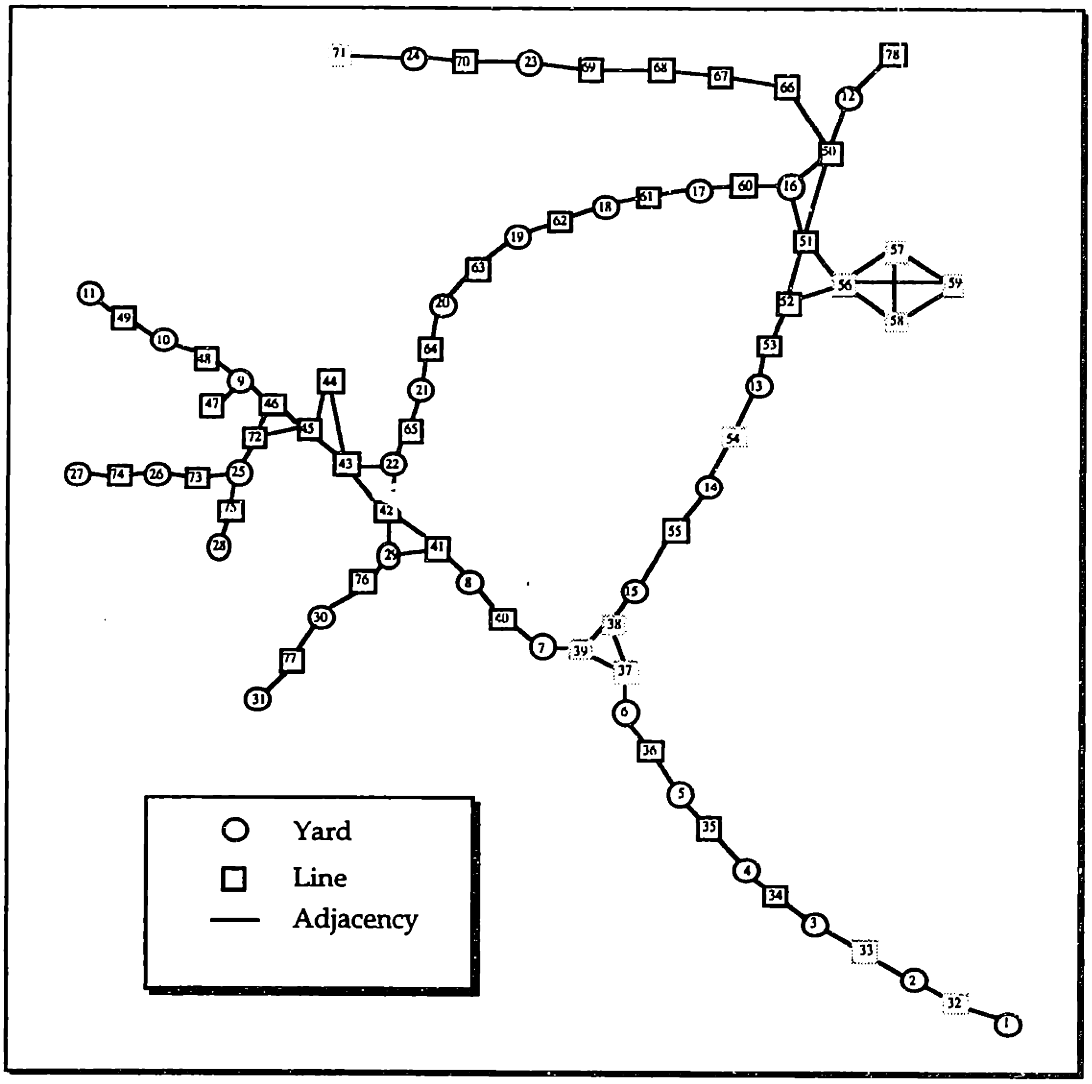


Figure 5.4: Planning Node Representation for Sample Class I Railroad Division

The corresponding network of planning nodes is shown in Figure 5.4 with a somewhat arbitrary numbering of nodes. Network components that are not part of this carrier's network are shown in grey. Activities for those nodes are determined exogenously, yet the impact of trains flowing from these nodes must be considered.

5.1.3. Object Oriented Representation of the Network

Like the descriptive yard model presented in Chapter 4, the network model has both static and dynamic components. Understanding the data and creating the appropriate structures to support network planning is a primary concern. This section describes the implementation of the planning node concept and the representation of trains moving through the network.

5.1.3.1. Planning Node Representation

The network is simply a collection of lines and yards. Thus, the static object model of the network begins with a single class, **Network**, for which there is only one instance, or object. **Network**, in turn, contains a complete listing of **PlanningNode** objects which comprise the network. The adjacency relationships between **PlanningNodes** are stored using linked lists. **PlanningNode** is a parent class for two derived classes, **Yard** and **Line**, each of which stores the physical information required to support yard and line planning, respectively. No specific topological information regarding the network is stored in the class **Network**. It is all distributed among the **PlanningNodes**. The relationships are depicted in Figure 5.5.

The network coordinating algorithm is incorporated as a member function[3] of the class **Network**. When invoked, it cycles through the list of **PlanningNodes**, each of which represents one subproblem of the network planning problem. At each **PlanningNode**, the appropriate node planning algorithm, implemented using virtual functions[4], will be called. The network coordinator must resolve

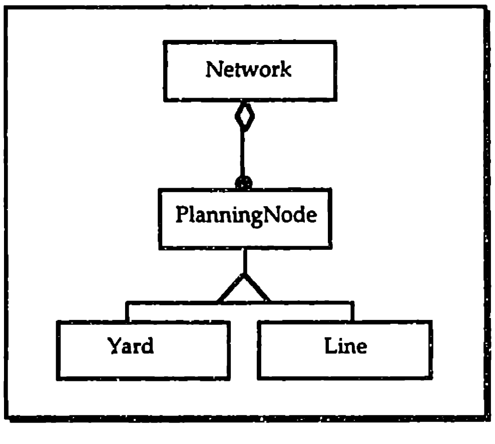


Figure 5.5: Object Model for Physical Network

[3] See Appendix B for definition.
[4] See Appendix B for definition.

the inconsistencies between the plans determined at each node and iterate through the list of **PlanningNode** objects until it converges upon a solution.

5.1.3.2. Train Representation

Having established the representation of the physical network, it is necessary to describe the objects used to represent trains. A train will pass through several planning nodes en route to its destination. At each node, the train must be represented differently, even though it represents the same set of cars and locomotives. Recall from Chapter 4 that a train remains the same object in the yard model if it does not change its consist. The same is true for representing a train in the network. A train will be considered the same train as long as there is no consist change. However, that train will be represented differently within each planning node.

The models for the line component and the yard component represent trains differently. As an example, the train class for the line model requires line-haul performance data but no specific consist data. The train class for the yard model requires consist data but no line-haul performance data. In general, the data sets required for each are essentially non-overlapping. Thus, for each train in the network model, we have a single class called **MasterTrain**. Contained in each **MasterTrain** object is a list of the specific **NodeTrains**. Derived from the **NodeTrain** class are two types of **NodeTrains, LineTrains** and **YardTrains**. These relationships are shown in Figure 5.6.

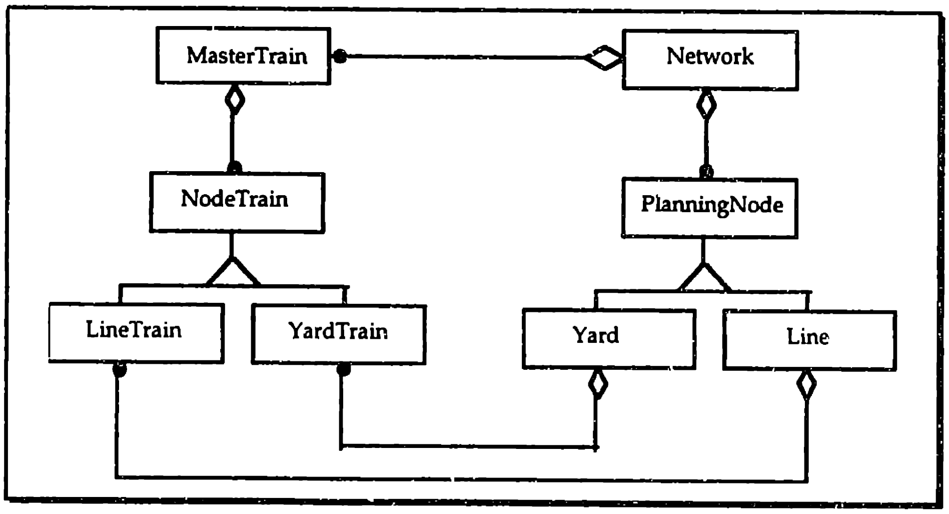


Figure 5.6: Object Model for Associations between Trains and PlanningNodes

Each train, therefore, is represented by multiple objects, where each train object is specific to the node through which the train will pass. For each train, we store one **MasterTrain** object at the network level. A list of all **MasterTrain** objects is stored in the **Network** object. Each **MasterTrain** object contains a list of node-specific train objects, which are stored in the order that the train traverses the nodes. Each of the **NodeTrains** are contained in the train arrival or departure (or both) lists for that node. This is described more specifically in the following three examples.

First, consider a single train passing from left to right over the topology shown in Figure 5.7.a. The single **MasterTrain** contains an ordered list of **NodeTrain** objects, each of which is a either a **LineTrain** or a **YardTrain**, as shown in Figure 5.7.b.

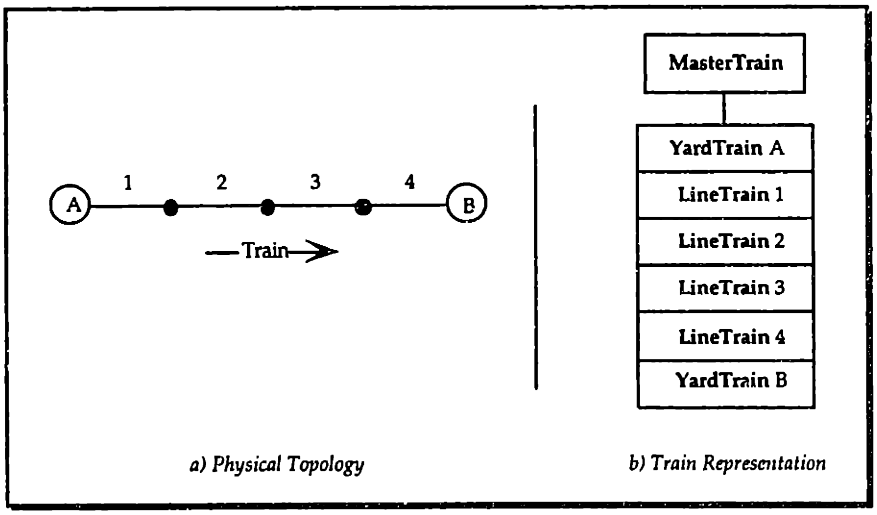


Figure 5.7: Object Representation—Single Train Moving Between Two Yards

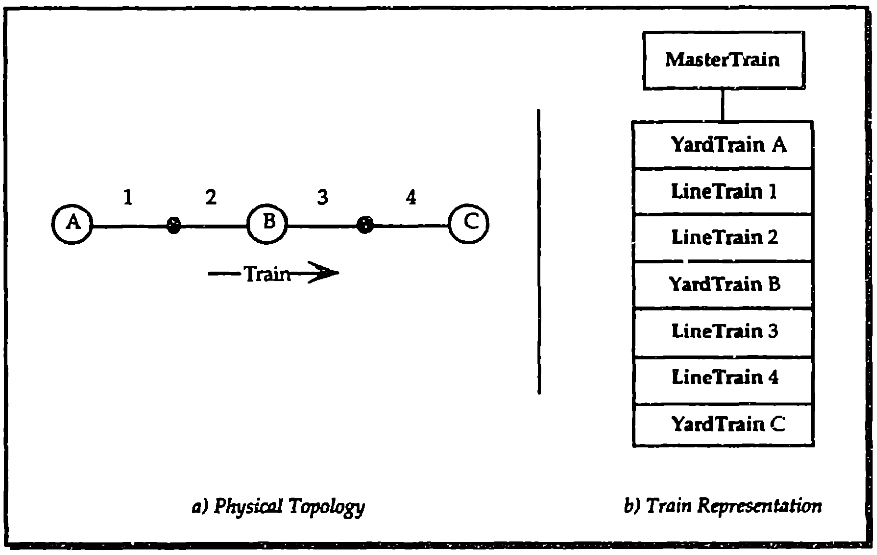


Figure 5.8: Object Representation—Single Pass-Through Train

Next, consider the case of a train moving through two yards and passing through a third, as shown in Figure 5.8.a. We add an additional **YardTrain** to the train representation. The yard model for B treats the train as a single object since there is no consist change. This is consistent with the design of the yard model structure presented in Chapter 4.

Finally, consider the same topology and the case of two trains with connecting freight, as shown in Figure 5.9. Train x passes from A to B and has connections to Train y. Train y moves from B to C. We define two **MasterTrains** in accordance with our train definition, which states that trains do not change their

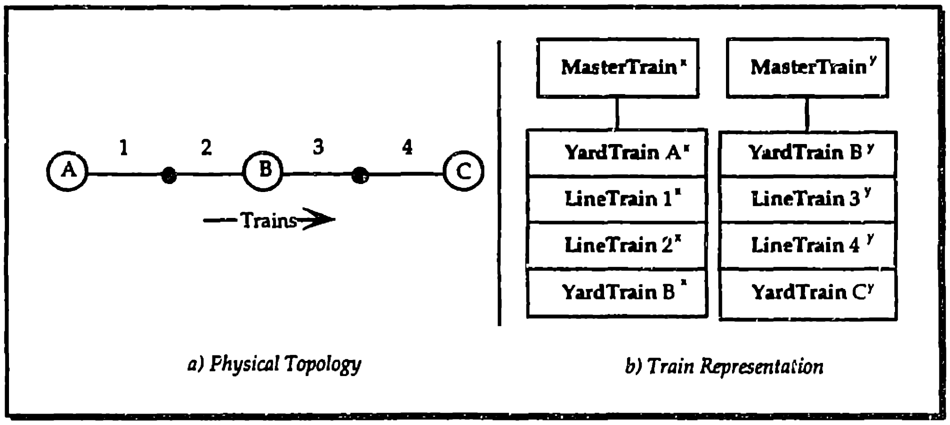


Figure 5.9: Object Representation—Two Connecting Trains

consist. For Train x, the **MasterTrain** contains the list of four **NodeTrains** ordered according to the train's route. The same is true for Train y. In the figure, node identifiers A, B, C, 1, 2, 3, and 4 are superscripted with the train indices x and y. At Yard B, Train x arrives and ceases to exist. Since its elements are ordered according to the route of the train, the **NodeTrain** list finishes with the **YardTrain** object for Yard B. Train y originates at B so its **NodeTrain** list begins with Train y's **YardTrain** object at Yard B. Looking at Yard B, we see that this representation is consistent with the train object structure defined in Chapter 4. Though the memory requirements are increased by storing multiple objects per train, this representation has the benefit of providing a structure that is well-suited for distributed computing.

5.2. The Network Planning Model

The network planner creates schedule objectives that are passed down to the line and yard controllers. To support the development of the network planner, a software framework was built around the concepts presented in Section 5.1. This framework serves as a testbed for algorithms being considered for use as the components of the network planner. Its design allows the component models to be treated as separate software modules and easily inserted into the testbed. This allows the creation of a prototype network planner.

This section describes research related to the development of the coordinating algorithm used in the network planner. As an initial coordinating mechanism, a heuristic algorithm is developed, encoded, and inserted into the testbed. Using two sample problems, the algorithm's performance is demonstrated using the deterministic yard simulation and fixed line-haul times. Figure 5.10 revisits the in-

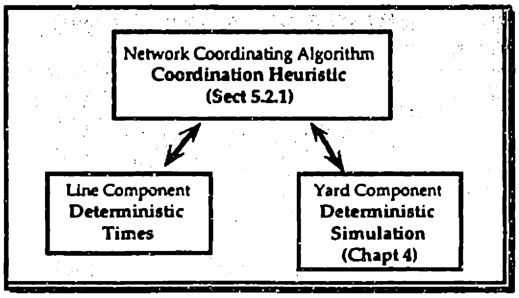


Figure 5.10: Implementation of the High-Level Planning Algorithm

ternal structure of the network planner and highlights the specific models that are used for each of the components.

5.2.1. The Network Coordination Heuristic

The network coordinator is the component of the network planner that resolves differences between plans developed by the line and yard planning components[5]. It ensures that the boundaries between the subproblems are consistent. In addition, the coordinator should enforce network objectives so that it does not simply create a consistent set of locally optimal solutions to the subproblems.

An explanation of boundary consistency is necessary. For each subproblem (i.e., *planning node*), we denote an arrival and a departure time. Though arrivals and departures are typically used to describe activities at terminals, we will also use these terms for the line subproblems. For a given train, the departure time from one planning node must equal the arrival time at its next node. Consistency for the entire planning problem implies this condition is met for all trains at all nodes.

Current research in this area (see Christodouleas [1994]) involves the use of nonlinear programming methods to decompose the network into the planning node structure described in Section 5.1. Using Lagrangian relaxation, this approach dualizes the constraints that specify boundary consistency. The algorithm iterates through the entire set of subproblems, calculates new dual values using an ascent method, updates the arrival times, and continues to iterate. The algorithm stops when the dual variables converge, signifying that the maximization of the Lagrangian dual problem has been attained and that the boundary constraints have been satisfied. The corresponding primal problem provides a network planning solution—consistent network schedule objectives that would be passed to the line and yard controllers.

This focus of this section is to create a coordination *heuristic.* The aim is to satisfy boundary consistency *without* using dual variables. The input to the subproblems at each iteration will be the train arrival times. At the end of each iteration, instead of updating the dual variables, the algorithm simply copies the *de-*

[5] Recall that although the internal components of the network planner create plans, they do so only in order to develop schedule objectives that are consistent with decisions that will be made by the controllers.

parture times for each train as the input arrival time for the train's adjacent node. Thus, at the *start* of each iteration of the coordination heuristic, we ensure that the boundary constraints are met. The algorithm iterates until these constraints are also satisfied at the *end* of an iteration.

To portray this mathematically, the following notation will be used:

T	Number of trains
N	Number of planning nodes
i	Index for trains, $i = 1,\ldots,T$
j	Index for planning nodes, $j = 1,\ldots,N$
j_i^{next}	Next planning node visited by train i after node j
T_j	Set of trains at planning node j
t	Iteration index for algorithm
a_{ij}^t	Arrival time of train i at node j for iteration t
$\mathbf{a}_j^t$	Vector of train arrival times at node j for iteration t
d_{ij}^t	Departure time of train i at node j for iteration t
$\mathbf{d}_j^t$	Vector of train departure times at node j for iteration t
$NPA_j(\ )$	Node planning algorithm for node j
δ	Flag variable for convergence

The coordination heuristic is expressed as:

```
initialize $a_{ij}^0$   $\forall j \in N;\ i \in T(j)$
$t \leftarrow 1$
repeat
    $\delta \leftarrow 0$
    for $j = 1,\ldots,N$
        $\mathbf{d}_j^t \leftarrow NPA_j(\mathbf{a}_j^t)$
        for $i \in T_j$
            $k \leftarrow j_i^{next}$
            $\mathbf{a}_{ik}^t \leftarrow \mathbf{d}_{ij}^t$
            if $\left|\mathbf{a}_{ik}^t - \mathbf{a}_{ik}^{t-1}\right| > \delta$ then
                $\delta \leftarrow \left|\mathbf{a}_{ik}^t - \mathbf{a}_{ik}^{t-1}\right|$
            end if
        end for
    end for
    $t \leftarrow t+1$
until $(\delta = 0)$
```

In essence, this simple algorithm starts with an initial value for train arrival times at each node. At each iteration, new estimates for departure times are created and are used as the arrival times at the next iteration. When all train arrivals in two consecutive iterations do not change, the algorithm stops. The resulting set of schedule objectives meet the boundary consistency requirement.

The initial arrival times are not required for all trains. Some trains will receive connections from earlier trains. The schedule times for these later trains, regardless of how they are initialized, will depend upon the earlier trains. Thus, the only initial times that are required are for trains that do not depend on connections. Furthermore, the algorithm only requires the arrival time for the initial planning node for those trains. The rest of the arrival and departure times will be determined as the effects of these initial times ripple through the network.

The algorithm converges on a wide range of sample problems and is always expected to converge when using fixed times for lines and the deterministic FIFO simulation for yards. The general idea is the following. If a planning node modifies its schedule at each iteration, these changes will only affect *other* planning nodes at *later* times. Earlier activity in the schedule will not be affected. As the algorithm iterates through the entire set of planning nodes, the changes will be seen later in the planning period as the changes made at earlier iterations ripple through the network. Again, ripple effects are only forward looking in time. We expect convergence to be proportional to both the number of trains and the number of planning nodes. A future area of study will be the convergence properties of this algorithm when sequencing models are used for the line and yard components.

We will illustrate the performance and convergence of this algorithm with two examples. The times for trains on all lines are deterministic and not dependent upon congestion (i.e., the presence of multiple trains simultaneously on the same line). To simplify things, for the purpose of demonstration, the deterministic times for lines are identical and set to three hours. In addition, all yards are clones of the example presented in Chapter 4.

5.2.2.1. 10 Node, 5 Train Example

The first example utilizes the ten node network shown below in Figure 5.11. Over this network five trains are run according to the input schedule shown in

Table 5-1. This initial input was hand-generated with insufficient slack time for connections in the yard and for haul times on the line.

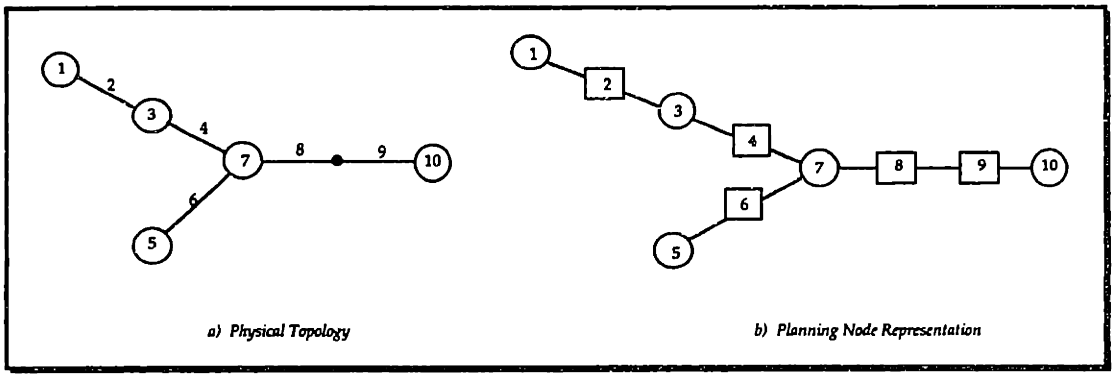


Figure 5.11: Ten Node Topology

Table 5-1: Initial Schedule, 10 Node Network
(Times shown in HHMM format)

Train #	Origin	Scheduled Departure	Destination	Schedule Arrival	Blocks
1111	1	0800	3	1000	101, 102
1112	3	1300	7	1500	101, 103
1113	5	1000	7	1200	104, 105, 106
1114	7	1900	10	2100	101, 109
1115	7	1400	10	1600	106, 107, 108

The coordination heuristic is used to develop the new schedule. We illustrate the convergence of the algorithm by showing the output of consecutive iterations, as shown in Table 5-2. This table shows the departure time from the first node in a train's path and the arrival times for all other nodes in that path.

The ripple effect of schedule changes in one planning node can be seen by tracking a series of connections through the network. For instance, the traffic (in this case, a single train) moving from Yard 1 to Yard 3 affects connections made for the train moving from Yard 3 to Yard 7, which, in turn, affects connections made on a train moving from Yard 7 to Yard 10. Thus, after the first iteration of the algorithm, the schedule is adjusted for the changes in initial train movements. These initial changes then ripple through the rest of the network, with changes not being felt at the later nodes until several iterations into the algorithm. Since the *longest* string of connections is between Yard 1 and Yard 10 (through Yard 3), Yard 10's schedule is the last to be adjusted in response to the change to Yard 3's arrival schedule from the first iteration.

Table 5-2: Arrival Time Output, 10 Node Network

(Time in minutes)

			ITERATION							
Train Number	**Departs from Node**	**Arrives at Node**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1111	1	2	480	480	480	480	480	480	480	480
		3	600	660	660	660	660	660	660	660
1112	3	4	780	781	841	841	841	841	841	841
		7	900	960	961	1021	1021	1021	1021	1021
1113	5	6	600	600	600	600	600	600	600	600
		7	720	780	780	780	780	780	780	780
1114	7	8	1140	1081	1141	1142	1202	1202	1202	1202
		9	1140	1320	1261	1321	1322	1382	1382	1382
		10	1260	1320	1500	1441	1501	1502	1562	1562
1115	7	8	840	940	1000	1000	1000	1000	1000	1000
		9	840	1020	1120	1180	1180	1180	1180	1180
		10	960	1020	1200	1300	1360	1360	1360	1360

Table 5-3: Final Schedule, 10 Node Network
(Times shown in HHMM format)

Train #	Origin	Departure	Destination	Arrival	Blocks
1111	1	0800	3	1100	101, 102
1112	**3**	**1401**	**7**	**1701**	**101, 103**
1113	5	1000	7	1300	104, 105, 106
1114	**7**	**2002**	**10**	**2602**	**101, 109**
1115	7	1640	10	2240	106, 107, 108

We express the final schedule in the same format as the initial schedule (see Table 5-3). Seeing the difference between the initial schedule and the final schedule is difficult when only looking at the data in tabular format. Taking the times from these schedules, we graphically depict the arrival times and the connections between trains in Figure 5.12 (for the initial schedule) and 5.13 (for the final schedule).

The format of these diagrams requires some explanation. On the x-axis, we have the planning node index numbers. Though putting the nodes in order along a line implies a linear adjacency relationship, keep in mind that this is not the case. When reading these diagrams, some mental manipulation is necessary to relate the linear depiction of Figures 5.12 and 5.13 to the true network topology shown in Figure 5.11. Circles (yards) and squares (lines) have been added to help with this mental transformation.

The times reflected in these diagrams are *arrivals*, not departures. Recall from an earlier discussion that a departure from one planning node *always* corresponds to an arrival at another node. In the diagrams, the small triangles represent train arrivals at either lines or yards. Departures are not shown as they simply clutter the picture, especially when displaying higher traffic densities. The thick black lines represent train movements through and between nodes. The thin, dotted lines represent connections between inbound and outbound trains made at yards.

Figure 5.9: Initial Schedule, 10 Node 5 Train Example

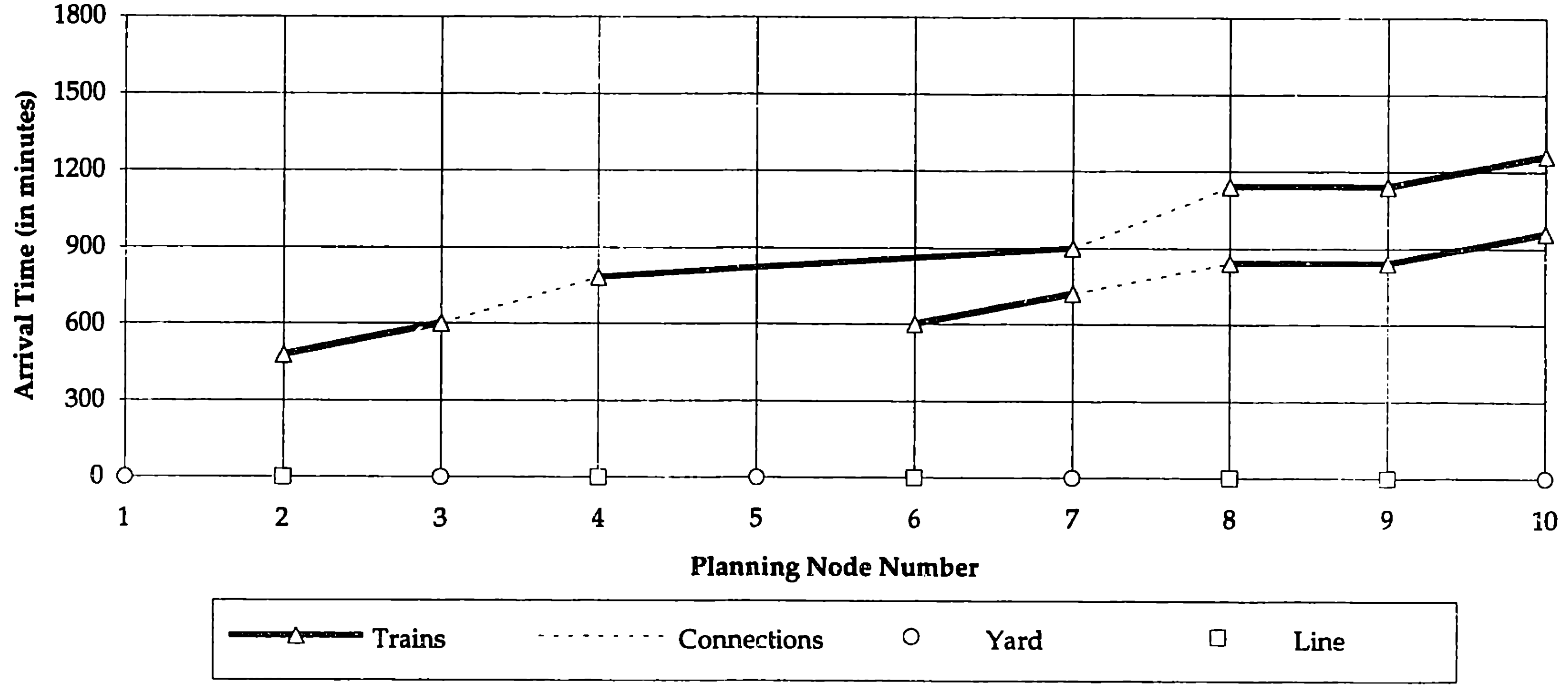
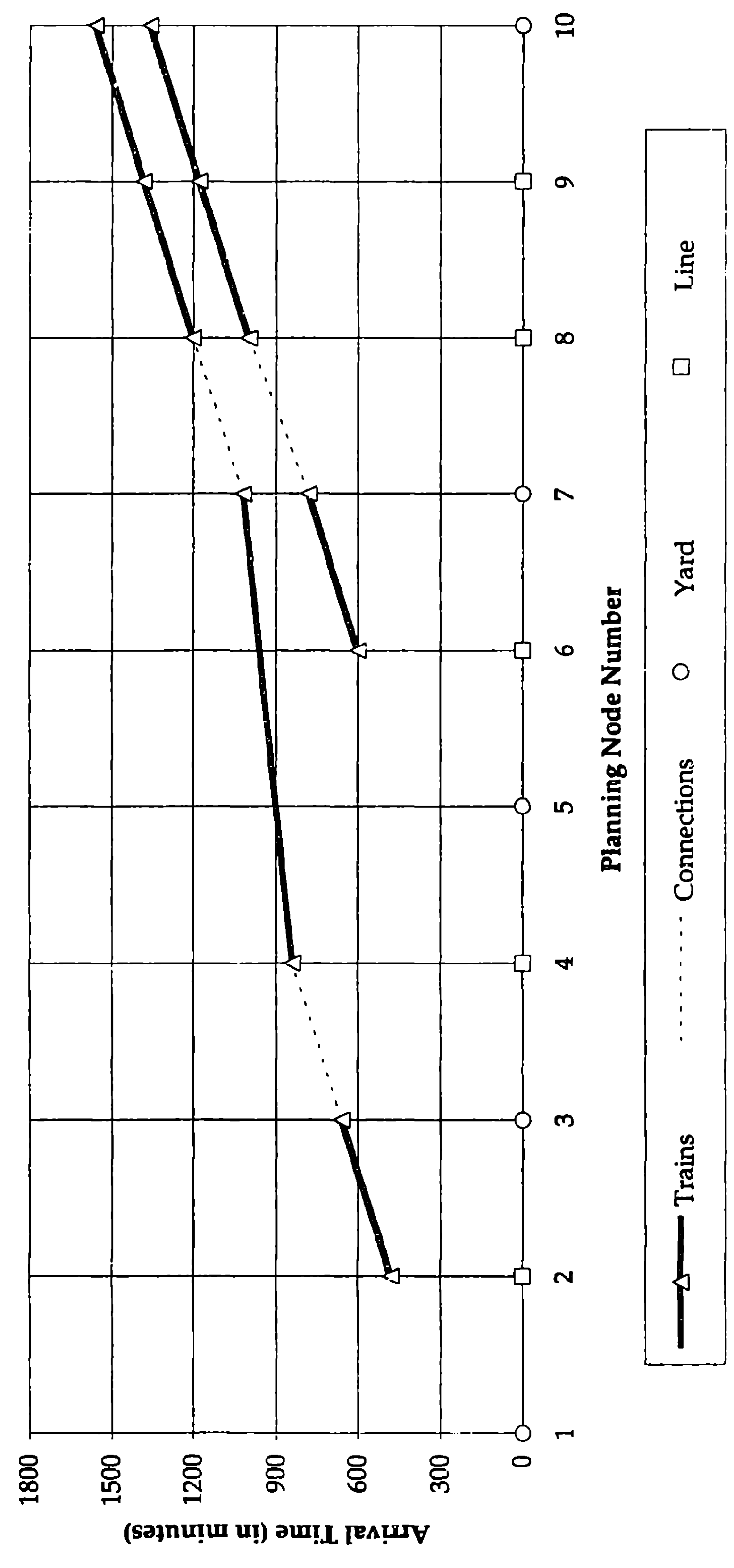


Figure 5.10: Final Schedule, 10 Node 5 Train Example



Reading the diagram is best demonstrated through examples. First, consider the arrival shown for Line 2. This arrival is equivalent to the departure from Yard 1, which is adjacent to Line 2. This train continues to Yard 3. The difference between the arrival time at Yard 3 and the arrival time at Line 2 is simply the total running time on Line 2 (i.e., three hours). At Yard 3, there is freight on the arriving train that must connect to the train bound for Yard 7 via Line 4. Due to the "non-linearity" of the network, Nodes 4 and 7, though they do not appear to be adjacent in this representation, are, in fact, adjacent.

If we are interested in the activity of a single node, say Yard 7, we simply need to look along the vertical line corresponding to the yard. The triangles represent the train arrivals. Train departures are found by looking at the lines adjacent to the yard. Arrivals at the adjacent lines are equivalent to departures from the yard, as specified in the boundary consistency requirement.

In this 10 Node example, the traffic patterns were intentionally kept simple. The predicted yard times in this example are small due to the small number of train arrivals and departures and, hence, the small number of connections, In addition, the inventory of the yard models were initialized only with the freight scheduled to move on the five trains. Increasing work-in-progress in the yards and the amount of traffic arriving and departing will surely amount to an increase in the processing time for freight. This would also influence the "ripple" effect that occurs at each iteration of the coordination heuristic.

5.2.2.2. 26 Node, 19 Train Example

We will now take a look at a much larger example, both in terms of the network size and the traffic load. The network is depicted in Figure 5.14 and consists of

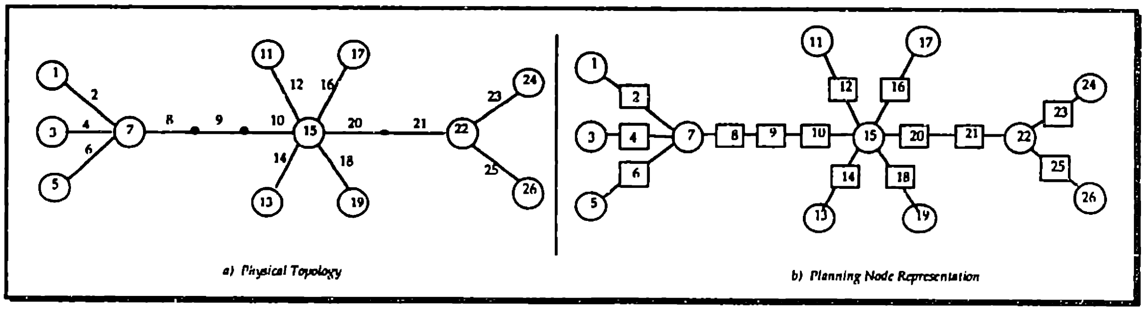


Figure 5.14: 26 Node Network Topology

eleven yards and fifteen lines. On this network are scheduled 19 trains, as shown in Table 5-4.

Given this initial schedule, the heuristic coordination algorithm converges in eleven iterations. The increase in the number of iterations is caused by an increase in the number of planning nodes and an increase in the number of con-

Table 5-4: Initial Train Schedule for 26 Node Example

(Times shown in HHMM format)

Train #	Origin	Departure	Destination	Arrival	Blocks
1111	1	0800	7	1000	101, 102, 103
1112	3	1300	7	1500	104, 105
1113	5	1000	7	1200	106, 107, 108, 109
1114	7	1900	15	2100	103, 105, 109, 110, 111
1115	11	1400	10	1600	112, 113, 114, 115, 116
1116	13	0800	15	1000	117, 118
1117	19	0930	15	1130	119, 120, 121, 122
1118	17	1000	15	1200	123, 124, 125
1119	15	1700	7	1900	112, 115, 116, 123
1128	15	1200	7	1400	126, 127, 128, 129
1120	15	1500	22	1700	118, 122, 124
1129	15	2600	22	2800	103, 109, 113
1121	15	1530	11	1730	117, 121, 125
1122	15	1630	13	1830	114, 120
1123	22	2900	24	3100	113, 124
1124	22	2930	26	3130	122, 103
1125	7	1645	1	1845	106, 112, 126
1126	7	1715	3	1915	101, 127
1127	7	1830	5	2030	102, 104, 116, 128

nections made between trains. The output arrival times following each iteration are shown in Table 5-5.

Table 5-5: Algorithm Arrival Time Output, 26 Node Example
(Time in minutes)

			ITERATION											
Train Number	**Depart Node Number**	**Arrive Node Number**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1111	1	2	480	480	480	480	480	480	480	480	480	480	480	480
		7	600	660	660	660	660	660	660	660	660	660	660	660
1112	3	4	720	720	780	780	780	780	780	780	780	780	780	780
		7	840	960	960	960	960	960	960	960	960	960	960	960
1113	5	6	600	600	600	600	600	600	600	600	600	600	600	600
		7	720	780	780	780	780	780	780	780	780	780	780	780
1114	7	8	1040	1153	1213	1213	1213	1233	1213	1213	1213	1213	1213	1213
		9	1140	1320	1333	[illegible]	1393	1393	1393	1393	1393	1393	1393	1393
		10	1260	1380	1500	1513	1573	1575	1573	1573	1573	1573	1573	1573
		15	1500	1320	1560	1680	1693	1753	1753	1753	1753	1753	1753	1753
1115	11	12	840	840	840	840	840	840	840	840	840	840	840	840
		15	960	1020	1020	1020	1020	1020	1020	1020	1020	1020	1020	1020
1116	13	14	480	480	480	480	480	480	480	480	480	480	480	480
		15	600	660	660	660	660	660	660	660	660	660	660	660
1117	19	18	570	570	570	570	570	570	570	570	570	570	570	570
		15	690	750	750	750	750	750	750	750	750	750	750	750
1118	17	16	600	600	600	600	600	600	600	600	600	600	600	600
		15	720	720	780	780	780	780	780	780	780	780	780	780
1119	15	10	1020	1225	1285	1285	1285	1285	1285	1285	1285	1285	1285	1285
		9	1020	1200	1405	1465	1465	1465	1465	1465	1465	1465	1465	1465
		8	1020	1200	1380	1585	1645	1645	1645	1645	1645	1645	1645	1645
		7	1140	1200	1380	1560	1765	1825	1825	1825	1825	1825	1825	1825
1128	15	10	720	720	720	720	720	720	720	720	720	720	720	720
		9	720	900	900	900	900	900	900	900	900	900	900	900
		8	720	900	1080	1080	1080	1080	1080	1080	1080	1080	1080	1080
		7	840	900	1080	1260	1260	1260	1260	1260	1260	1260	1260	1260
1120	15	20	900	938	998	998	998	998	998	998	998	998	998	998
		21	900	1080	1118	1178	1178	1178	1178	1178	1178	1178	1178	1178
		22	1020	1080	1260	1298	1358	1358	1358	1358	1358	1358	1358	1358
1129	15	20	1560	1720	1540	1720	1900	1913	1973	1973	1973	1973	1973	1973
		21	1560	1740	1900	1720	1900	2080	2093	2153	2153	2153	2153	2153
		22	1680	1740	1920	2080	1900	2080	2260	2273	2333	2333	2333	2333
1121	15	12	930	983	1043	1043	1043	1043	1043	1043	1043	1043	1043	1043
		11	1050	1110	1163	1223	1223	1223	1223	1223	1223	1223	1223	1223
1122	15	14	990	1145	1205	1205	1305	1205	1305	1205	1205	1205	1205	1205
		13	1110	1170	1325	1385	1385	1385	1385	1385	1385	1385	1385	1385
1123	22	23	1740	1906	1966	2146	2306	2126	2306	2486	2499	2559	2559	2559
		24	1860	1920	2086	2146	2326	2486	2306	2486	2666	2679	2739	2739
1124	22	25	1770	1861	1921	2101	2261	2081	2261	2441	2454	2514	2514	2514
		26	1890	1950	2041	2101	2281	2441	2261	2441	2621	2634	2694	2694
1125	7	2	1005	1356	1416	1596	1776	1981	2041	2041	2041	2041	2041	2041
		1	1125	1185	1536	1596	1776	1956	2161	2221	2221	2221	2221	2221
1126	7	4	1035	1023	1083	1263	1443	1443	1443	1443	1443	1443	1443	1443
		3	1155	1215	1203	1263	1443	1623	1623	1623	1623	1623	1623	1623
1127	7	6	1110	1426	1486	1666	1846	2051	2111	2111	2111	2111	2111	2111
		5	1230	1290	1606	1666	1846	2026	2231	2291	2291	2291	2291	2291

The final train schedule is shown in Table 5-6, along with the graphical representation of the initial and final schedules shown in Figures 5.15 and 5.16, respectively.

Table 5-6: Final Train Schedule for 26 Node Example
(Times shown in HHMM format)

Train #	Origin	Departure	Destination	Arrival	Blocks
1111	1	0800	7	1100	101, 102, 103
1112	3	1300	7	1600	104, 105
1113	5	1000	7	1300	106, 107, 108, 109
1114	7	1900	15	2913	103, 105, 109, 110, 111
1115	11	1400	10	1700	112, 113, 114, 115, 116
1116	13	0800	15	1100	117, 118
1117	19	0930	15	1230	119, 120, 121, 122
1118	17	1000	15	1300	123, 124, 125
1119	15	1700	7	3025	112, 115, 116, 123
1128	15	1200	7	2100	126, 127, 128, 129
1120	15	1500	22	2238	118, 122, 124
1129	15	2600	22	3853	103, 109, 113
1121	15	1530	11	2023	117, 121, 125
1122	15	1630	13	2305	114, 120
1123	22	2900	24	4539	113, 124
1124	22	2930	26	4454	122, 103
1125	7	1645	1	3701	106, 112, 126
1126	7	1715	3	2703	101, 127
1127	7	1830	5	3811	102, 104, 116, 128

Figure 5.12: Initial Schedule, 26 Node 19 Train Example

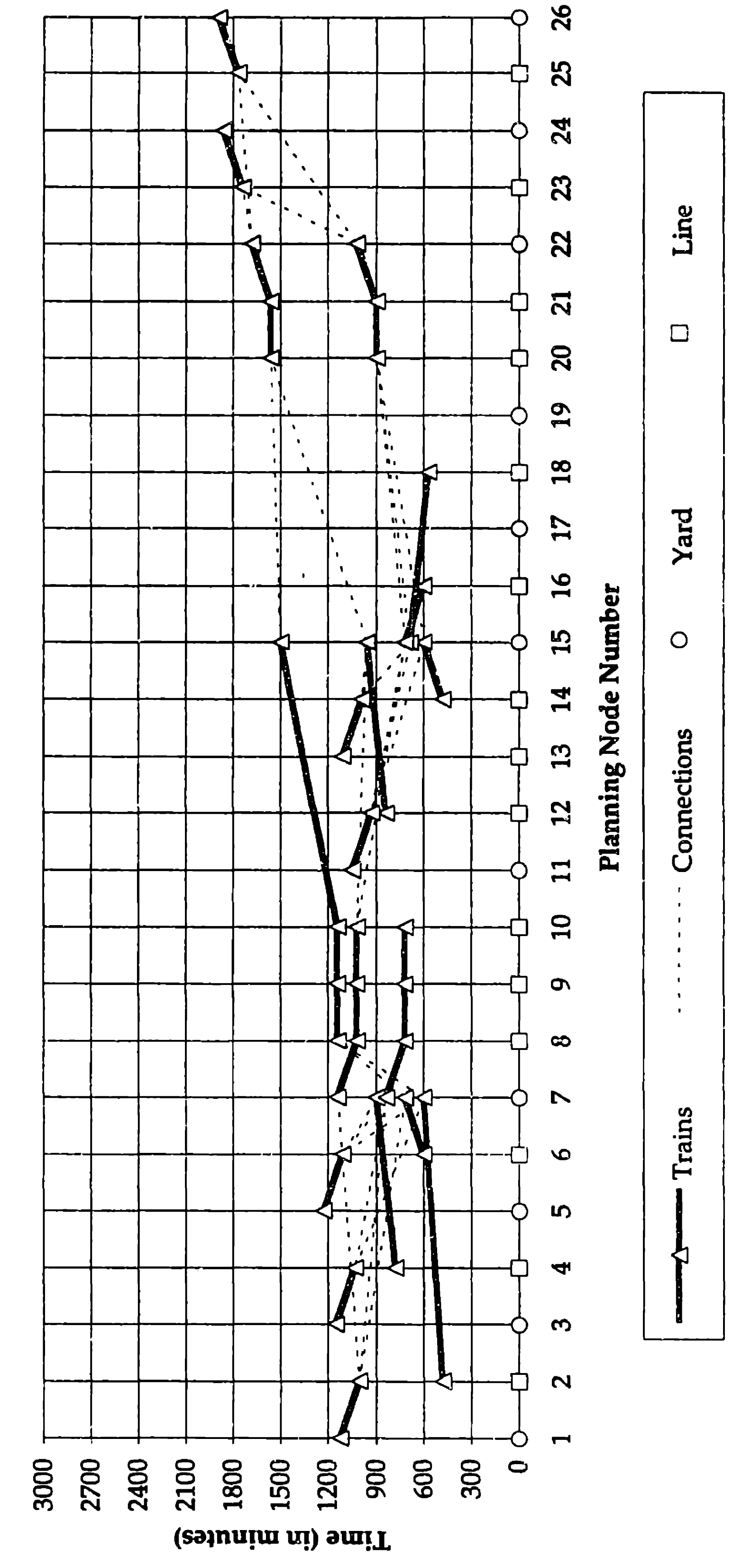
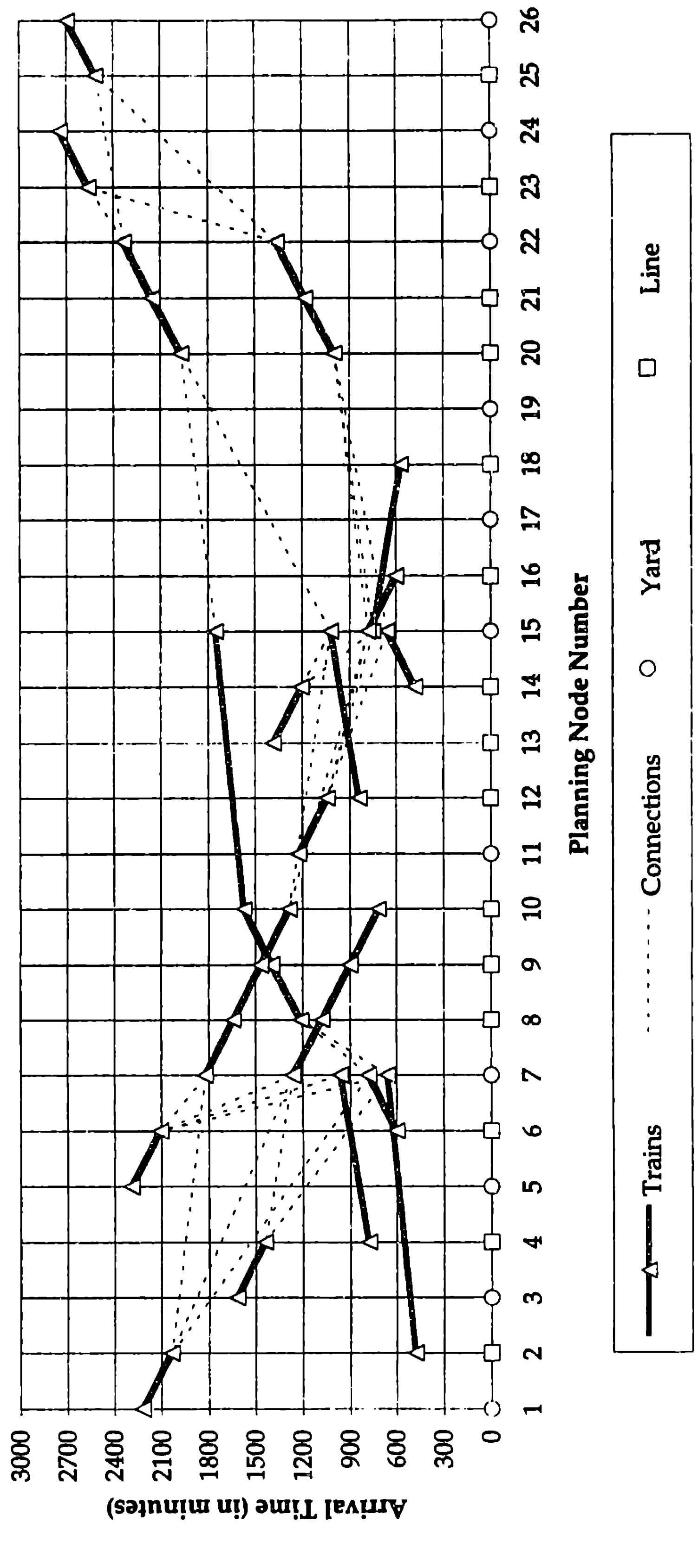


Figure 5.13: Final Schedule, 26 Node 19 Train Example



5.3. Chapter Summary

The chapter presented a representation of the network planning problem that is consistent with the object oriented yard model presented in Chapter 4. A testbed has been created based on this object oriented network design. It allows algorithms to be easily inserted and tested with other algorithms as part of the operational network planner. For the purpose of demonstration, the deterministic simulation created in Chapter 4 was incorporated as the yard planning component. Furthermore, a coordination heuristic was derived to serve as the network coordinating algorithm. For the final component, fixed line-haul times were used. The combination of the three components was illustrated on two examples.

The main purpose of the demonstration was to show the performance of the coordination heuristic. Further work is required to test its effectiveness when different algorithms are used for the line and yard planning components. Chapter 6 presents the formulation of a yard sequencing model which will be incorporated into the testbed as part of a future effort. The convergence properties of the heuristic can be explored on problems for which sequencing decisions are being made for the subproblems, versus the purely *descriptive* models as used in this chapter.

This heuristic is a first step toward creating a true network coordinating algorithm. Recall that such an algorithm should have two primary characteristics. First, it should enforce consistency among the boundaries of the subproblems. Second, it should enforce network objectives so that the final solution is better than one based on a set of consistent, locally optimal solutions. The coordination heuristic satisfies the consistency requirement. Future work should be aimed at modifying the heuristic to include the second requirement.

6. A Yard Sequencing Model

The network planning framework presented in Chapter 5 relied upon a descriptive model for the yards and fixed travel times for the lines. The primary purpose of developing that model was to create a coordination heuristic and, equally important, the structure to support the data flows between the heuristic and the network planner's line and yard components. The next step in developing the network planner, in which *planning decisions* are made by the line and yard components, is to incorporate line planning algorithms and yard sequencing algorithms instead of the descriptive models used in Chapter 5.

Sequencing models exist for the line planning problem and include such models as SCAN (Jovanovic and Harker [1991]) and PLATO (Draper [1994]). Though sequencing models have been developed for the yard, they typically rely upon enumerative methods (e.g., Yagar, *et al.* [1983]) or optimization approaches (e.g., Guignard and Kraft [1993]). In real-time planning, however, we can *not* rely on such models, as they are too computationally burdensome. Our approach must rely upon fast heuristics.

We will turn our attention, therefore, to general scheduling and sequencing theory, a field with key results that will guide us toward an approach for generating yard sequencing decisions quickly. Though many classes of problems addressed in this field have been proven NP-Hard (Rinnooy Kan [1976]), considerable effort has been devoted to the development of heuristics. Recently, Ferguson [1993] connected machine scheduling theory with rail terminal scheduling and the results are encouraging.

It is this approach, summarized in the first part of the chapter, that serves as our starting point. The second part of the chapter is concerned with refining the model for use within the network planner. The final part of this chapter presents the issues associated with using this model as the foundation for a yard control decision support tool (or yard *controller*), for which additional layers of modeling fidelity must be added.

6.1. Notional Model

Recall from Chapters 3 and 5 that the purpose of operational network planning is to create schedule objectives that are passed down to the local decision support tools (i.e., the line and yard controllers). The role of the network planner will depend upon whether the railroad is running a fully scheduled operation; an unscheduled, or demand-driven, operation; or any combination of the two. If operating in a *scheduled* railroad, the network planner will receive schedule objectives from the higher levels of planning. Its primary function is schedule enforcement. In the demand-driven context, the network planner receives "loose" schedule objectives from the higher levels and creates firm schedule objectives that are passed down to the local controllers. Whether the network model is intended to operate within a scheduled or unscheduled railroad, the yard sequencing model that will be used will not change, although modifications would be required for the network coordinating algorithm.

In describing the yard model, we first present a notional design for the model and the decisions that we expect it to make. Consider a typical layout of a hump yard, as shown in Figure 6.1. Arriving trains have pre-defined schedules and consists. The goal of the sequencing model is to develop a

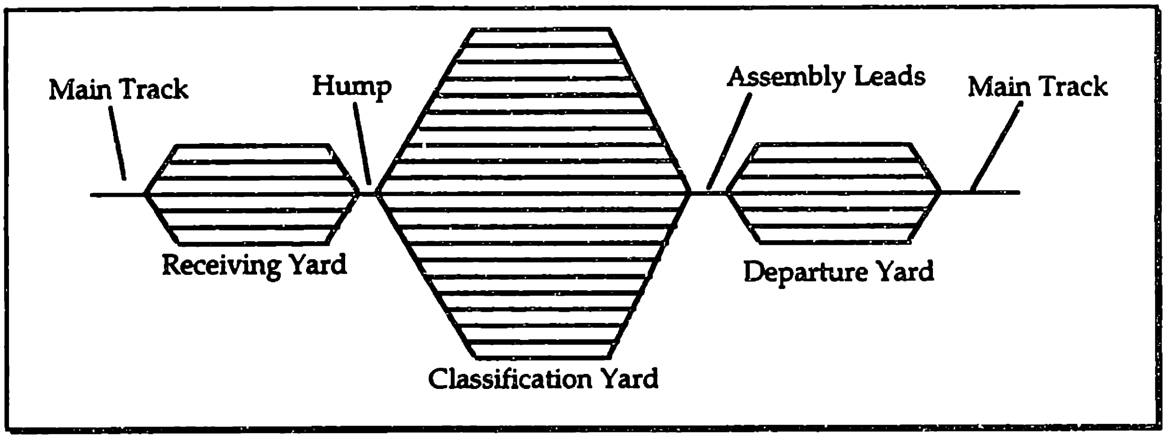


Figure 6.1: Generic Hump Yard Layout

schedule internal to the yard such that departure schedules and connections are met. Choosing the appropriate level of fidelity for the yard model in this environment is important. It is proposed that only the hump and assembly sequencing decisions be considered, as they are the primary determinants for freight connection times. Allocation of freight to specific crews and tracks represents a set of second-order decisions, which are made *given* the primary sequencing decisions. These second-order decisions are made in the yard controller (i.e., the lowest level), while the yard model used in the network planner only considers the hump and assembly sequencing decisions.

The general approach is to treat the problem as a serial, two-machine sequencing problem, where the first machine represents the humping (or disassembly) process and the second machine represents the assembly process. Since many yards operate in a manner that allows multiple, simultaneous assemblies, the second machine may, in fact, be represented by several machines functioning in parallel. The two cases are shown in Figure 6.2, in which figure a) shows the second machine as truly a single machine and in figure b) shows the second machine consisting of three parallel machines. The stages in the figure refer to the status of the freight, whether in the form of an inbound train, outbound train, or block of cars. The outputs are the key decisions that are made at each of the machines. Inputs required to generate these decisions are not shown since they are too numerous. They are fully

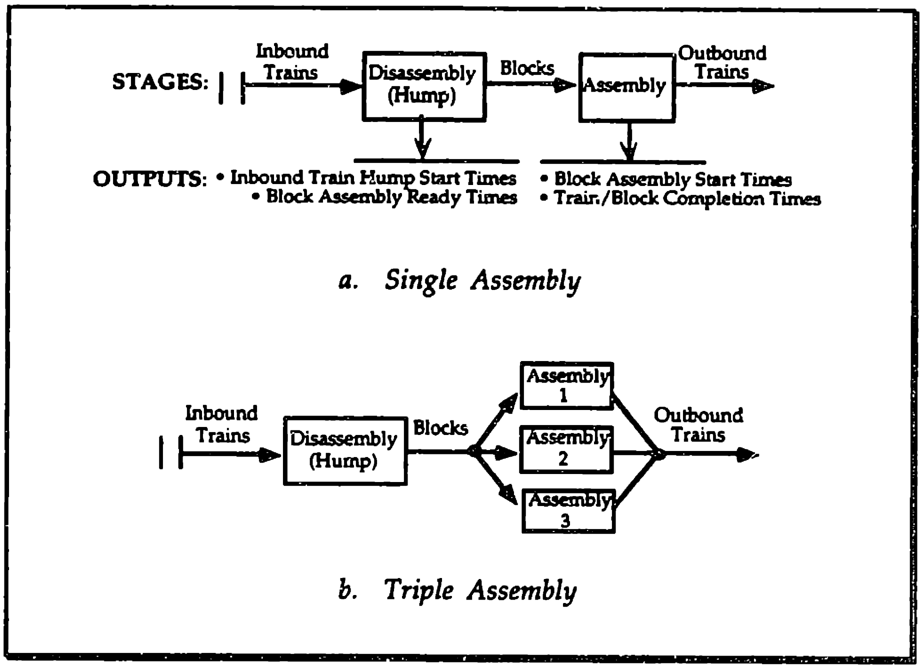


Figure 6.2: Two-Machine Disassembly/ Assembly Sequencing Model (DASM)

described in the detailed presentation that follows. The basic model from Ferguson [1993] deals with the case shown in Figure 6.2a.

6.2. The Disassembly-Assembly Sequencing Model

The sequencing model presented in Ferguson [1993] (referred to herein as the *basic* model) serves as the starting point for developing a sequencing model as the yard component of the operational network planning model. This section will summarize the model and present extensions, including performance improvements to the basic model and modifications required to support the network planner.

6.2.1. The Basic Model

The formulation of the basic model relies upon the following definition of the problem:

> "Given a set of inbound trains available for classification and the make-up [consist] and schedule for a set of outbound trains, determine the sequences for switching the inbound trains and pulling back [assembling] the aggregated blocks which optimizes an objective function relating to the tardiness of the outbound trains."

The problem definition includes certain assumptions about the operations of the yard:

Assumption 1: All cars for an outbound block will make their assigned connections.

Assumption 2: An entire receiving track will be humped before another is started (No partial humping of inbound trains).

Assumption 3: Sufficient track space exists to allow a "sort-by-block" classification strategy, with each car passing over the hump once (single-stage).

Assumption 4: There exists a single hump, where the hump is available for continuous operation.

Assumption 5: One assembly locomotive is used and is in continuous service.

Assumption 6: All processing times are deterministic.

There are several areas that demand additional discussion. The basic model requires that all inbound trains are ready to be processed. In the broad context of network planning, a planning period will include train arrivals, so not all inbound trains are available for humping at the start of the period. Modifications to handle this case are proposed in the "extensions" section of this chapter. Second, typical yards can assemble multiple outbound trains simultaneously, but the basic model is built upon a single assembly (Assumption 5). The intuitive manner to handle the relaxation of this assumption is to move from a strict two-machine model to the case where the second machine is actually multiple machines operating in parallel, a notion that was discussed earlier in the chapter (see Figure 6.2). Dealing with multiple assemblies in the yard controller requires modeling the specific movements of engines through the trim end of the yard, the assembly tracks, and the departure yard. However, it is sufficient to represent the assembly process as multiple parallel machines for the level of fidelity sought in the model that supports operational network planning.

At first glance, the two-machine approach appears to fit nicely into the mold of a two-machine flow shop problem, since the machine ordering for each job is the same. For most regular[1] measures of performance, this problem is NP-Hard (Rinnooy Kan [1974]), but good heuristics exist for finding near-optimal schedules. However, some assumptions of a flow shop are *not* satisfied in the yard sequencing problem and we must follow a different approach.

The measure of performance used in the basic model is the minimization of maximum tardiness ($T_{\max}$). Tardiness penalizes late service completion without rewarding early completion. The choice of minimizing maximum tardiness allows us to develop a schedule that can be easily verified to satisfy some threshold for tardy departures. However, the fact that $T_{\max}$ is non-zero would indicate *nothing* about the number of tardy trains. Alternative objective measures would be the minimization of average tardiness, minimization of weighted average tardiness, or minimization of the *number* of tardy departures.

The general idea behind the algorithm is the following. Outbound trains consist of blocks, which are built in the classification yard. Based on scheduled train departures and specific orderings of blocks on outbound trains, due times are generated for blocks. Block due times correspond to the times at which they must arrive in the departure yard such that their outbound train will not be tardy. These due times are used in the tardiness-related objective. In single machine scheduling problems, sequencing according to earliest due date (EDD) commonly provides optimal solutions for due date-related criteria (see Baker [1974]). This basic model uses a modified EDD calculation to determine both the hump and assembly sequence.

The algorithm used in the basic model is presented below in five steps. Each step contains a definition of notation and presentation of theorems that

[1] A *regular* performance measure is one that is nondecreasing in task completion time.

are used. The theorems proved in Ferguson a complete summary of notation is provided in Appendix D.

6.2.1.1. **Step 1:** *Determine BlockDue Times*

The notation required for this step is as follows:

Given	
N	Number of outbound trains
n_j	Number of blocks in outbound train j
(i,j)	i^{th} block on outbound train j
p_{ij}	Pull-out/Assembly duration for block (i,j)
d_j	Due time for train j
Determine	
d_{ij}	Due time for block (i,j)
Internal	
$\hat{d}$	Temporary storage of due times

Block due dates are determined by the relationship:

$$d_{ij} = d_{i'j} - p_{i'j} \tag{6.1}$$

where block (i,j) directly precedes block (i',j) in train j. Thus, due times are set to be the *latest* times at which block completion will not affect the lateness of the outbound train.

The algorithm for the calculation of due times is given by:

```
for $j = 1,\ldots,N$                /* Outbound trains */
    $\hat{d} \leftarrow d_j$
    for $i = n_j,\ldots,1$          /* Blocks in current outbound train */
        $d_{ij} \leftarrow \hat{d}$
        $\hat{d} \leftarrow d_{ij} - p_{ij}$
    end for
end for
```

6.2.1.2. **Step 2:** *Determine the Slack Lengths of Hump Job Sets*

A Hump Job Set (HJS) is the set of inbound trains that have cars connecting to a particular outbound block. There is a one-to-one correspondance between outbound blocks and HJSs.

Slack length is a key measure that relates an inbound train's hump processing to an outbound block. To determine slack length, it is necessary to find the position in the inbound train of the last car connecting to the block of interest. The slack length is the difference between the time at which that car is humped and the time at which the train's last car is humped. It is a measure of the extra processing time required for the current train to finish before another train can be humped.

Given this notion of slack length, minimizing the time it takes to build a single block is done in accordance with the following theorem:

Theorem 6.1: To minimize the lateness of a block, sequence only the inbound trains that have connections for that block. Of those trains, put the train with the largest slack length last in the sequence. The order of the preceding trains is irrelevant.

This theorem is still valid when we consider building multiple blocks, with one exception. If, in the *m* trains included in this HJS, the first *m-1* trains in the set also complete a second outbound block, then the otherwise irrelevant sequence of those trains should be chosen so as to build the second block as quickly as possible. We will call this phenomenon *secondary hump sequencing* and will discuss it further as a modification to the basic model.

The following notation is used in this step:

Given

N, n_j, (i,j)	<As defined above>
M	Number of inbound trains
m_k	Number of cars on inbound train k
HJS_{ij}	Hump Job Set corresponding to block (i,j)
μ	Hump processing rate (cars per minute)
pos_{ij}^k	Position of last block (i,j) car on inbound train k

Determine

s_{ij}	Maximum slack length in Hump Job Set (i,j), where $s_{ij} = \max_{k \in HJS_{ij}} \left(s_{ij}^k\right)$

Internal

s_{ij}^k	Slack length for block (i,j) on inbound train k
$last_{ij}$	Last train ordered in HJS_{ij}

```
for $k = 1,\ldots,M$
   scan $(i,j)$ for all $k \in HJS_{ij}$
      $s_{ij}^k \leftarrow \frac{m_k - pos_{ij}^k}{\mu}$
   end scan
end for

for $j = 1,\ldots,N$
   for $i = 1,\ldots,n_j$
      $s_{ij} \leftarrow 0$
      scan $k \in HJS_{ij}$
         if $s_{ij}^k > s_{ij}$ then
            $s_{ij} \leftarrow s_{ij}^k$
            $last_{ij} \leftarrow k$
         end if
      end scan
   end for
end for
Re-order $HJS_{ij}$ such that $last_{ij}$ is last element in set
```

6.2.1.3. **Step 3:** *Determine Hump Job Set Sequence by Modified EDD Rule*

To satisfy the due dates of the assembly jobs, the HJS sequence needs to be determined. If the hump process is isolated as a single machine minimizing maximum tardiness (or lateness), then a well-known result from sequencing theory is to process the hump job sets in the order of Earliest Due Date (EDD). However, Ferguson shows that in addition to its dependence upon the due dates, the HJS sequence is sensitive to both the processing times for the pull-out/assembly jobs and the slack lengths of the HJSs. This leads to the critical theorem:

Theorem 6.2: For a given sequence of assembly jobs, the sequence of Hump Job Sets that minimizes the maximum lateness of the assembly jobs is given by ordering the HJSs according to increasing value of $d_{ij} - p_{ij} + s_{ij}$

The additional notation is introduced as follows.

Given	
$N,\ n_j,\ (i,j),\ d_{ij},\ p_{ij},\ s_{ij}$	<As defined above>
Determine	
$\hat{d}_{ij}$	Modified due date calculation for HJS/block (i,j)
Ω	The ordered list of assembly jobs/HJSs arranged in increasing order of $\hat{d}_{ij}$

And the resulting algorithm is the following:

```
for $j = 1,\dots,N$
   for $i = 1,\dots,n_j$
      $\hat{d}_{ij} \leftarrow d_{ij} - p_{ij} + s_{ij}$
      insert $(i,j)$ into ordered list $\Omega$
   end for
end for
```

6.2.1.4. **Step 4:** *Determine Inbound Train Hump Order*

Having established the HJS sequence, this step identifies the specific hump sequence for inbound trains. The additional notation for this problem is as follows:

Given	
$n_k,\ HJS_{ij},\ s_{ij}^k,\ M,\ \Omega,\ \mu$	<As defined above>
β_0	Fixed time required between trains in hump sequence
Determine	
hs_k	Hump start time for inbound train k
pr_{ij}	Pull-out/assembly ready time for block (i,j)
Internal	
A	Ordered set of trains assigned to be humped
B	Set of trains not assigned to be humped

h_k	Hump processing duration for inbound train k
τ	Current time

The algorithm to perform the calculation is the following:

```
initialize  $A \leftarrow \varnothing$
initialize  $B \leftarrow k, \forall k=1,\ldots,M$  /*set of inbound trains*/
while $\Omega \neq \varnothing$
    remove first $HJS_{ij}$ from $\Omega$
    while $HJS_{ij} \neq \varnothing$
        remove train $k$ from $HJS_{ij}$  /* $k \in HJS_{ij}$ ordered per Step 2*/
        if $k \in B$ then
            $\tau \leftarrow \tau + \beta_0$
            $hs_k \leftarrow \tau$
            $h_k \leftarrow \frac{m_k}{\mu}$
            $\tau \leftarrow \tau + h_k$
            add $k$ to $A$
            remove $k$ from $B$
        end if
    end while
    $pr_{ij} \leftarrow \tau - s_{ij}^k$
end while
```

6.2.1.5. **Step 5:** *Determine Assembly Start, Finish and Resulting Tardiness*

The final two theorems relate to the perfomance measure of lateness, the positive component of which is tardiness. The method for calculating due times is shown in Step 1, Equation 6.1. Since the due times are the basis for the lateness/tardiness performance measure, the following theorem relates the cumulative effect of late assemblies.

Theorem 6.3: In a desired sequence of continuously scheduled assembly jobs, if any assembly job is late, then the succeeding job will be at least that late.

This leads directly to the result for the total lateness of a *train*:

Theorem 6.4: The lateness of an outbound train equals the maximum of the lateness of its blocks.

In general, for any job j, whose due time is d_j and actual completion time is c_j, the lateness and tardiness measures are given to be, respectively:

$$L_j = c_j - d_j \tag{6.2}$$
$$T_j = \max(0, L_j) \tag{6.3}$$

For this final step, the following notation is used:

Given	
pr_{ij}, p_{ij}, d_{ij}	<As described above>
Determine	
ps_{ij}	Pull-out/assembly start time for block (i,j)
c_{ij}	Completion time of (i,j), where $c_{ij} = ps_{ij} + p_{ij}$
T_j	Tardiness of outbound train j
$T_{\max}$	Maximum tardiness of trains
Internal	
$\hat{\Omega}$	Copy of Ω, the ordered list of assembly jobs/HJSs
τ	Current time

The algorithm to perform these calculations is given by:

```
initialize  $T_{\max} \leftarrow 0$
initialize  $T_j \leftarrow 0 \quad j=1,\dots,N$
initialize  $\tau \leftarrow 0$
while $\Omega' \neq \varnothing$
  remove first block $(i,j)$ from $\hat{\Omega}$
  $ps_{ij} \leftarrow \max(\tau, pr_{ij})$
  $c_{ij} \leftarrow ps_{ij} + p_{ij}$
  $\tau \leftarrow c_{ij}$
  if $d_{ij} - c_{ij} > T_j$
    $T_j \leftarrow d_{ij} - c_{ij}$
    if $T_j > T_{\max}$
      $T_{\max} \leftarrow T_j$
    end if
  end if
end while
```

To summarize the algorithm, it takes as input the outbound train schedule, the order of blocks on the outbound trains, and the car-to-block assignments for all cars on the inbound trains. The algorithm assumes that all trains are available for humping at the beginning of the planning period.

The first step is to create the due times for individual blocks. The inbound trains are then ordered within each Hump Job Set such that the last train in each HJS has the largest slack length. Next, the hump sequence of HJSs is determined, from which the hump sequence of inbound trains is derived. Given this sequence, the algorithm calculates the times at which trains begin humping and the time at which blocks finish being humped and are ready for assembly $\left(pr_{ij}\right)$. Using the HJS sequence, the corresponding blocks are assembled. The assembly start times $\left(ps_{ij}\right)$ and block completion times $\left(c_{ij}\right)$ are determined. Thus, the algorithm can determine the tardiness for any block and, by Theorem 6.4, for any outbound train.

One of the examples from Ferguson [1993] is revisited in Tables 6-1 through 6-5, which correspond to the five steps of the algorithm. The example considers a situation in which five inbound trains are ready to be humped. Predefined blocks are to be created and assembled in a predefined order into outbound trains. Two of the blocks exist in the classification yard at the beginning of the planning period and their slack lengths are set to zero.

For outbound trains (Trains 6-9), departure schedules, consists (in station order), and block due times, d_{ij}, are shown in Table 6-1. Block-to-car connections and the position of cars are known for the inbound trains. We can, therefore, determine the hump process duration for each train, based on an average hump rate of 1.5 cars per minute; the last car position in each train, as counted from the rear of the consist; and the resulting slack length, s_{ij}^{k}, for block (i,j) on inbound train k. These calculations are shown in Table 6-2, along with the maximum slack lenghts for each HJS.

Knowing the values of d_{ij}, s_{ij}, and p_{ij} allow the modified due dates $\left(\ddot{d}_{ij}\right)$ to be computed. The sequence that results is given by {D, E, F, G, C, A, B} and is applied to both the HJSs and the corresponding assembly jobs. Knowing the trains that comprise each HJS, we can establish the hump sequence for the inbound trains to be {4, 3, 2, 1, 5}. [The actual sequence used by the yard master in the case study was {1, 2, 3, 4, 5}.] Given the hump sequence, the hump pro-

Table 6-1: Outbound Train Schedule and Due Time Calculations (Step 1)

Train	D_j	Block	d_{ij}	p_{ij}
6	270			
		D	210	60
		F	270	60
7	300			
		E	300	120
8	375			
		G	315	60
		A	375	60
9	420			
		C	360	60
		B	420	60

Table 6-2: Inbound Train Schedule and Slack Length Calculations (Step 2)

Train	# Cars	h_k	Block	Last Car	s_{ij}^k
1	75	50	-	-	-
2	12	8	F	1	0
3	58	39	F	26	17
			E	18	12
			A	4	2
			C	1	0
4	80	54	D	54	36
			A	15	10
			F	9	6
			E	1	0
5	90	60	-	-	-

	Block: A	B	C	D	E	F	G
$s_{ij} = \max\limits_k \left(s_{ij}^k\right)$	10	0	0	36	12	17	0

cessing durations (h_k), the slack lengths $\left(s_{ij}^k\right)$, and a fixed intertrain hump processing duration (β_0) of twenty minutes, the hump completion times for each block are calculated and shown in Table 6-4. Note that the hump completion times are equivalent to the pull-out/assembly ready times $\left(pr_{ij}\right)$.

Table 6-3: Modified EDD Calculations (Step 3)

	D	F	E	G	A	C	B
d_{ij} (Step 1)	210	270	300	315	375	360	420
p_{ij} (given)	60	60	120	60	60	60	60
s_{ij} (Step 2)	36	17	12	0	10	0	0
$d_{ij} - p_{ij} + s_{ij}$	186	227	192	255	325	300	360

Table 6-4: Hump Completion Times for Each Block (Step 4)

Train Sequence	D	F	E	G	A	C	B
43215 (basic model)	38	161	121	-	131	133	-
12345 (actual)	195	225	231	-	224	157	-

Table 6-5: Block Completion Times (Step 5) and Tardiness Calculations

Sequence Determined By		D	F	E	G	A	C	B
	d_{ij}:	210	270	300	315	375	360	420
Basic Model	Order:	1	3	2	4	6	5	7
	c_{ij}:	98	301	241	361	481	421	541
	ℓ_{ij}:	-112	31	-59	46	106	61	121

Train i:	6	7	8	9
L_i:	31	-59	106	121

		D	F	E	G	A	C	B
Actual	Order:	1	2	3	4	5	6	7
	c_{ij}:	255	315	375	495	555	615	675
	ℓ_{ij}:	45	45	75	180	180	255	255

Train i:	6	7	8	9
L_i:	45	75	180	255

Table 6-6: Comparison of Related Objective Values (in minutes)

Performance Measure	12345 (actual)	43215 (basic model)
Maximum Lateness (Tardiness)	255	121
Average Lateness	138.75	49.75
Average Tardiness	138.75	64.5
Number of Tardy Departures	4	3

Finally, given the block ready times and the assembly sequence, we can determine the assembly completion time for each block based upon the pull-out/assembly times, p_{ij}. The assembly sequence, as noted above, is performed in the same order as the HJS sequence. [Recall the one-to-one correspondence between an HJS and a block.] Table 6-5 shows the resulting completion time for each block. The lateness for each outbound train, from Theorem 6.4, is also shown in the table. The maximum lateness, per the sequences used by the yard master, was 255 minutes, while the basic model reduced that measure to 121 minutes. This demonstrates the basic model's effectiveness in reducing maximum lateness.

Sequencing to reduce one performance measure often results in reductions in other measures. In Table 6-6, for the actual and basic model sequences, we show the values for some related measures, including average lateness, average tardiness (no reward for early departures), and number of late departures. In this example, the basic model sequence outperforms the actual sequence used by the yard master relative to any measure.

6.2.2. Extensions to the Basic Model

This basic model has several shortcomings that need to be addressed. We will refer to the basic model plus the modifications discussed in this section as the Disassembly/Assembly Sequencing Model, or DASM. A summary of required modifications includes:

- Secondary hump sequencing (as described in 6.2.1)
- Revisiting the slack length calculations

- Improving the assembly sequence, in light of early ready times
- Maintaining precedence for blocks in each outbound train
- Rolling arrivals (versus considering only trains already in yard)
- Multiple, simultaneous assemblies
- Incorporating train values, or *weights*, in sequencing decisions

These modifications can be grouped into two areas. First, there are opportunities for improving the performance of the algorithm for its intended role (see problem statement in Section 6.2.1). These include the first four items in the list above. The remaining items are modifications to create a model that can be used as the yard component in the network planner. This section will describe each and offer conjectures for extensions to the algorithm that handle the modifications.

Secondary Hump Sequencing

This phenomenon was discussed earlier. Consider the case where a Hump Job Sequence (HJS) is being processed (i.e., the *primary* HJS). Processing this train set may result in another HJS (i.e., the *secondary* HJS) being completed. In other words, the set of trains comprising the secondary HJS is wholly contained in the primary HJS. When determining the optimum sequence of trains within the primary HJS, the basic model states that the ordering of all but the last train in the set is irrelevant. However, when a secondary HJS is wholly contained in the primary HJS, the "irrelevant" members of the primary HJS should be arranged so as to also minimize the hump completion time of the secondary HJS. This translates to putting all trains for the secondary HJS first in the processing sequence, with its maximum slack length train at the end of this secondary sequence. The only time when the secondary sequence need not be considered is when the last train of the primary HJS is also contained in the secondary HJS. In this case, the processing order of all but the last train becomes irrelevant.

Slack Length Calculation Revisited

In the basic model, slack length is determined independently of the Hump Job Set sequence. Slack length is dependent only upon the trains contained in the HJS. In Step 3, the HJS sequence is based partially upon the slack lengths. The resulting problem is illustrated in the following example.

Outbound Block A (connects to outbound train 4)		
	Hump Job Set:	Inbound Train 1 ($s^1_{A,4} = 40$ min.)
		Inbound Train 2 ($s^2_{A,4} = 0$ min.)
		Inbound Train 3 ($s^3_{A,4} = 20$ min.)
Outbound Block B (connects to outbound train 5)		
	Hump Job Set:	Inbound Train 1 ($s^1_{B,5} = 0$ min.)
		Inbound Train 2 ($s^2_{B,5} = 40$ min.)

We have two HJSs, one corresponding to Block A and the second corresponding to Block B. For each HJS we calculate the maximum slack length to determine the last train humped within each HJS. Thus, Train 1 would be the final train humped in the Block A HJS and Train 2 would be the final train humped in Block B HJS. The HJS slack lengths, $\max\limits_{k \in HJS_{ij}} \left(s^k_{ij}\right)$, are given by

$$s_{A,4} = 40 \tag{6.4}$$

$$s_{B,5} = 40 \tag{6.5}$$

Assume that the HJS sequence is determined (in Step 3) such that the B set is humped prior to the A set. The B set includes inbound Trains 1 and 2, which are also members of the A set. Upon the completion of the B set, the only option for the last hump position within the A set is Train 3, with a slack of 20 minutes. Thus, the slack length for the A set becomes 20 minutes. The HJS sequence depends upon the slack lengths but we have shown the slack lengths to be dependent upon the HJS sequence. This interdependence must be resolved.

Changing the algorithm to accommodate this condition would involve a modification to Step 3. After the initial calculation of modified due dates $\left(\hat{d}_{ij}\right)$ and the subsequent ordering of jobs, the slack lengths for each Hump Job Set should be updated and the HJS sequence re-calculated. Since cycling could occur, an heuristic will be developed to reconcile this inconsistency.

Improving the Sequence of Assembly Jobs

The basic model, through the modified earliest due date rule, creates a sequence that is used for *both* the block assemblies and the corresponding Hump Job Sets. This section provides an improvement heuristic that takes advantage of situations in which the pull-out/assembly ready times, pr_{ij}, are not ordered consistently with the due dates, d_{ij}. We will revisit the case study example from Ferguson [1993] and show the benefits of this improvement heuristic.

The basic model expected assembly to occur in the same sequence as the HJSs. This makes sense when the pull-out/assembly ready times are consistent with the ordering according to the $\hat{d}_{ij}$ values. However, two things prevent this consistency from always occurring. First, a full block may exist in the classification yard at the beginning of the planning period, but its value for $\hat{d}_{ij}$ (where $s_{ij} = 0$) is such that the assembly/pull-out of this job is not scheduled for later in the sequence. Second, a case similar to the secondary hump sequencing problem described earlier, is the completion of a secondary HJS while processing a primary HJS. Thus, the ready time for this secondary block is earlier than planned, yet the assembly of blocks is still based upon the values of $\hat{d}_{ij}$. The end result is that the secondary HJS retains its designated slot in the sequence, even though we may be able to squeeze it into an earlier slot with an improvement in the performance measure.

The improvement heuristic is as follows. Using a schedule developed by the basic model, create a subset of assembly jobs whose ready times are inconsistent with the ordering defined by the modified EDD rule. This set is ordered by pull-out/assembly ready time, pr_{ij}, and is scanned from the first ordered element. For the current element, we attempt to insert it earlier in the assembly job sequence where there is sufficient idle time to process it and where it remains *viable*. The viability of a job is maintained if the sequence being considered does not change the block (or station) order of its outbound train. After the element is either entered into an earlier position in the sequence or rejected from consideration for early processing, it is removed from the set. The set is then scanned again for the next ordered element. Any time an element is deemed nonviable, it is removed. When the set is empty, the procedure is stopped.

We will illustrate this modification with the example presented in Tables 6-1 through 6-5. Recall that two of the blocks (G and B) connecting to outbound trains exist in the classification yard at the beginning. Thus, the original assembly sequence {D, E, F, G, C, A, B} can be modified by the procedure just described. Our list of blocks whose pull-out/assembly *ready* times (pr_{ij}) are *inconsistent* with the pull-out/assembly *due* times (d_{ij}) includes G and B. We begin with G, which is *viable* for earlier insertion because it is the first block in outbound Train 3. Due to the idle time at the beginning of the schedule (as shown in Table 6-5) we can insert G in the first slot for pull-out/assembly, just ahead of Block D. Since C is scheduled for assembly immediately before A, B can only be considered for switching slots with A. This does not improve the perfomance measure so B does not move forward in the sequence. As shown in Figure 6-7, the resulting sequence {G, D, E, F, C, A, B} lowers $T_{\max}$ from 121 minutes to 106 minutes, a significant improvement. Table 6-8 summarizes the other performance measures presented earlier. Keep in mind that the basic model offered a dramatic improvement of the yard master's sequence and the heuristic presented in this section improves the performance one step further.

Table 6-7: Modified Block Assembly Sequence and Tardiness Calculations

Sequence Determined By		D	F	E	G	A	C	B
	d_{ij}:	210	270	300	315	375	360	420
Basic Model	Order:	2	4	3	1	6	5	7
w/ Assembly	c_{ij}:	120	301	241	60	421	361	481
Heuristic	ℓ_{ij}:	-90	31	-59	-315	106	1	61
	Train i:		**6**	**7**	**8**	**9**		
	L_i:		**31**	**-59**	**106**	**61**		

Table 6-8: Comparison of Related Objective Values (in minutes)

Performance Measure	**12345 (actual)**	**43215 (basic model)**	**43215 (w/ assembly heuristic)**
Maximum Lateness (Tardiness)	255	121	106
Average Lateness	138.75	49.75	34.75
Average Tardiness	138.75	64.5	49.5
Number of Tardy Departures	4	3	3

Maintaining Precedence Ordering for Assembly Jobs

In Steps 4 and 5 of the basic model, the values of $\hat{d}_{ij}$ determine the sequence in which the HJSs are humped and the corresponding assembly jobs pulled. However, these values may result in a block being humped and pulled earlier than a job that it is supposed to precede. Thus, in Step 4, an additional check should be added to handle these (possibly rare) cases. An example of this phenomenon, which will occur when unusually large slack lengths exist for one of the trains, is shown below.

Outbound Block A (connects to outbound train 5)

Due Time: 100 minutes Processing Duration: 30 min.

Hump Job Set: Inbound Train 1 ($s^1_{A,4}$ = 10 min.)

Inbound Train 2 ($s^2_{A,4}$ = 15 min.)

Outbound Block B (connects to outbound train 5)

Due Time: 70 minutes Processing Duration: 25 min.

Hump Job Set: Inbound Train 3 ($s^1_{B,5}$ = 10 min.)

Inbound Train 4 ($s^2_{B,5}$ = 60 min.)

Looking at the due times, which are calculated using Step 1, we see that B must be pulled ahead of A, if the appropriate block order for Train 5, i.e., {B,A}, is to be maintained. The values for $\hat{d}_{ij}$ are 85 minutes and 105 minutes for blocks A and B, respectively. Thus, the HJS sequence and the assembly sequence would place A ahead of B. This sequence is not viable. A check must be added to the algorithm to prevent blocks from connecting to the same outbound train in the wrong station order. When blocks connect to different trains or they belong to a train without a fixed block order, then this enforcement is not necessary.

Rolling Arrivals

The modifications, to this point, have dealt with changes to the basic model, improving its performance within the original problem statement. Starting with this section, we will address changes that must be made in order to use the model as the yard component of the network planner.

Consider the problem statement as defined for the basic model. It assumes that all inbound trains to be humped have *arrived* and are ready for processing. When we plan an entire shift or entire day (which was not the stated purpose of the basic model), this assumption is not valid as trains will arrive intermittently during the planning period. We will consider two methods with which to modify the algorithm to handle rolling arrivals.

The first approach is to break the planning period into smaller planning sub-periods, defined by the interarrival times of inbound trains. More specifically, these planning sub-periods are the "hump-ready" interarrival times, or the interval between the time one train has been received and inspected by the yard and the time the next scheduled train has been received and inspected. The general approach toward solving the problem is to run the algorithm independently for each planning sub-period in the order of the sub-periods. Then the output of a planning period (i.e., current location of all trains in the receiving yard, blocks in the classification yard, and partial trains in the departure yard) will serve as the initial conditions for the following sub-period. When two train arrivals are *sufficiently close*[2], special attention must be paid to the boundary conditions between the two sub-periods. Though this approach runs the risk of losing benefits gained by considering the entire planning period, its implementation is straightforward.

The second approach is to plan the *entire* period as though all trains are, in fact, hump-ready at the beginning of the planning period. This gives a sequence for HJSs and for block assemblies. A simple heuristic is to take the ordered set of train arrivals (in increasing order of arrival times) and scan it

[2] Given that the first train is chosen to be humped immediately upon its arrival, if the second train arrives during the humping of the first train, they are *sufficiently close*.

from the last element to the first element (i.e., from last arrival to first arrival), calling the current train T*. Within an ordered set of HJSs, find the first HJS containing T*, and call this HJS*. If HJS* is scheduled for humping *after* the scheduled arrival of the train, then the sequence of HJSs will not change and we scan the train arrival set for the next latest arrival.

Otherwise, if the arrival of T* occurs after it is to be humped as part of HJS*, scan the HJS sequence beginning with the HJS that occurs just after HJS*. If the current HJS does not contain T*, the HJS may be moved earlier in the sequence, just ahead of HJS*, if this will improve the objective. We continue scanning the HJS sequence and moving HJSs forward in the sequence if they do not contain T*. This proceeds until HJS* is scheduled consistently with the arrival of T* or until the list of remaining successor HJSs is empty. HJS* is then marked so that it can not be considered for early processing when other HJSs are scanned. This procedure is repeated for all train arrivals (again, in decreasing order of arrival times).

Multiple Assemblies

The basic model assumes a single assembly mechanism. However, most yards contain more than a single assembly mechanism, i.e., parallel train assemblies can be performed. Thus, the second machine in this "two-machine" model should be replaced with a set of parallel machines. The interaction between the parallel machines will depend upon the physical layout of the assembly/pull-out tracks, the departure yard, and the trim end of the classification yard.

Scheduling parallel machines to minimize maximum lateness is a problem that, except for a few cases, is NP-Hard (see Blazewicz, *et al.* [1994]). Thus, in an effort to arrive at a good solution, we must rely upon heuristic algorithms. The approach is to schedule assembly jobs according to the modified EDD rule in Step 3 of the original algorithm. Assuming all assembly mechanisms can support assembly onto *any* track in the departure yard, we can simply assign each block to the next available assembly mechanism. Alternatively, if we wish to have all blocks in a given outbound train processed by the same assembly mechanism, the assignment will be determined by the assembly mechanism availability at the time the first block in that outbound train is to be processed. In this case, we risk having unnecessary, inserted idle time due to potentially large separations between the ready times of the individual blocks on the same outbound train.

Incorporating Train Values

One feature of this problem that has not been captured is to consider the *value* of the freight that is connecting between trains. In this case, time is not the single driving factor for sequencing decisions. Rather, the freight *value* also provides a discriminator for sequencing decisions. The basic model does not address this and treats all connections and outbound trains as equally important. Some well-known results for weighted performance measures exist and may provide good solutions to the weighted version of this problem (e.g., Conway, *et al.* [1967], Baker [1974], French [1982], and Blazewicz, *et al.* [1993]).

An additional issue is determining how to represent a train's value. The values used in the yard problems should be consistent with those used to make sequencing decisions on the line. This poses some interesting challenges that must be addressed in future research.

6.2.3. High-Level Network Planning Revisited: Modifications to the Heuristic Coordinating Algorithm

Recalling the heuristic from Section 5.2.2, we wish to now consider the implications of incorporating DASM as the yard component of the network planner, in lieu of the descriptive yard model originally used. Revisiting the notional design of the operational network planner (see Figure 5.1), the substitution of yard models results in the arrangement shown in Figure 6.3.

In terms of data flow between the coordinating algorithm and the DASM, additional input is required. When the descriptive model was used in Chapter 5, we did not require scheduled departures to be passed as input to the yard component. Since the objective function in the DASM is dependent upon due times, this information is essential. Thus, if we are to utilize the DASM with the heuristic network coordinating algorithm, the heuristic must generate departure schedules for each yard at each iteration.

If the network planner is operating from a fixed schedule, the coordinating algorithm would simply pass down those fixed times to the yard and line components at each iteration. The more difficult case is when the coordinat-

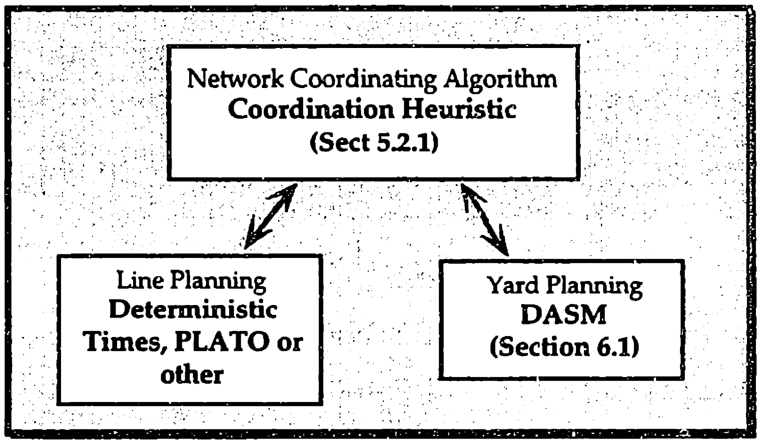


Figure 6.3: Network Planner using DASM

ing algorithm is only given schedule *guidance* and has the flexibility to adjust arrival and departure times. In the examples presented in Chapter 5, there are intermediate points in the algorithm at which freight is scheduled to depart the line or yard *earlier* than it is scheduled to arrive. In order to avoid situations such as these, the heuristic coordinating algorithm must adjust departure schedules, not just the arrival schedules.

An alternative to passing departure schedules is to change the objective function in DASM so that it is not based upon criteria involving due dates. Such performance measures include minimizing mean flow time[3], minimizing mean weighted flow time, or minimizing makespan[4] . Modifying the DASM to incorporate these objectives would involve theoretical work for the derivation of the algorithm. Once this work is done, incorporating it into the basic model can be done quickly.

The final set of data that was not used in the descriptive yard model but could be used in DASM is freight value. The object model described in both Chapters 4 and 5 includes freight values, so the structure already exists to support their use. Since the piece of freight will maintain its value through-

[3] Flow time, for a job j, is the sum of the job's waiting time and processing time. This is equivalently $F_j = c_j - r_j$, where c_j is completion time (or the time the job leaves the system) and r_j is the ready time (or the time the job enters the system).

[4] Makespan is the total length of the schedule, or $\max_j(c_j)$, where c_j is completion time of job j

out its trip through the network, the heuristic coordinating algorithm will not have to adjust the freight's value at any iteration, but it will need to pass this information to the line and yard components.

6.3. Extensions for Yard Control Decision Support Tools

In Chapter 3, we presented a hierarchical approach toward planning a railroad. This notion of hierarchical planning applies not only to the planning that occurs *among* the strategic, tactical, and operations planning levels, but *within* each of these planning levels. At the operational planning level, our hierarchy comprises an operational network planner and line control and yard control decision support tools. Consistency among the decisions that are made at each level of the planning hierarchy is paramount.

To enforce this consistency at the operational planning level, we will use the DASM not only as the yard component of the network planner, but also as the foundation for the yard controller. DASM provides the sequencing for the primary effects (i.e., humping and assembling), but the yard controller requires another level of fidelity to model the assignment of freight to all appropriate resources in the yard, such as inspection crews, engine crews, and tracks. This section presents a notional model for unifying these decisions and offers suggestions for models to solve these additional resource assignment problems.

The implementation involves three phases (see Figure 6.4). Phase I deals with the primary effects of hump and assembly sequencing. Within this phase, we assume that sufficient track space exists in the receiving, classification, and departure yards to handle the scheduled flow of trains. We also assume that the time from train arrival until the time the train is available for humping is *train dependent* but *sequence independent*. The times determined in Phase I planning include the hump completion times for each train; the time at which an outbound block begins to occupy a classification track; the time at which a block is pulled from its classification track; the time at which an outbound train begins to occupy a departure yard track; and, finally, the time at which the outbound trains depart the yard. These times,

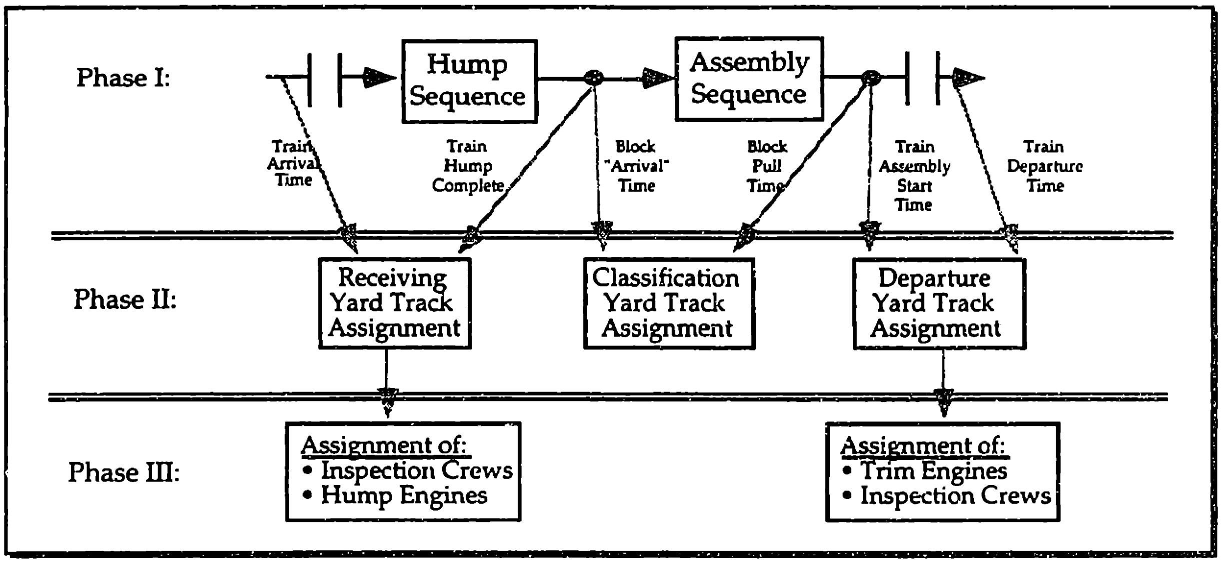

Phase I:
Hump Sequence
Assembly Sequence
Train Arrival Time
Train Hump Complete
Block "Arrival" Time
Block Pull Time
Train Assembly Start Time
Train Departure Time
Phase II:
Receiving Yard Track Assignment
Classification Yard Track Assignment
Departure Yard Track Assignment
Phase III:
Assignment of: • Inspection Crews • Hump Engines
Assignment of: • Trim Engines • Inspection Crews

Figure 6.4: Yard Decision Support Model using DASM

along with the scheduled arrival times for inbound trains, are passed as boundary conditions for the assignments made in Phases II and III.

In Phase II, we have three independent track assignment problems. The temporal element puts a wrinkle into what would otherwise be a straightforward assignment problem. Assigning inbound trains to receiving tracks, given the times generated in Phase I, is an easy problem. Since the only stipulation for a train to be assigned to a receiving track is that it fits, we can use a greedy heuristic with the following rule: assign an inbound train upon its arrival to the shortest possible receiving track. Thus, if we have an list of receiving tracks (T = number of tracks), ordered from shortest to longest, and a list of inbound trains (N = number of trains), ordered from earliest to latest, this rule results in a worst-case complexity of $O(NT)$ comparison operations. This simple rule is also applicable to the departure yard track assignments.

Difficulty arises in making the classification track assignments. Depending on the operating policy of the terminal, each block may have a bowl track to which it is assigned (or "home track"). The only circumstances that would prevent a block from being assigned to its home track is that the block is too long to fit on its assigned track or that another block is currently occupying that track. An alternative is to assign the block to a track that minimizes impact upon the classification and assembly operations. First, the preferred reassignment would result in no other blocks being displaced from their home track. Second, the assignment should minimize the impact upon assembly time, which is dependent upon the physical layout of the trim end of the classification yard, the assembly tracks, and the departure yard.

Phase III involves the assignment of the inspection and engine crews to perform work. These decisions are constrained by the time windows determined in Phase I and the location of trains and blocks determined in Phase II. The assignment of crews involves the highest fidelity model of the three-phase hierarchy. This is evident in the case of assigning trim engines to assembly jobs. Proper assignment of the engines will depend upon the detailed layout of the trim end of the classification yard, the configuration of the assembly tracks, the layout of the departure yard, and the presence of crossover tracks between the tracks in each of these three areas.

To summarize the notional model of the yard controller, we utilize the DASM in Phase I to determine the hump and assembly sequences, the primary effects in this planning problem. Given these sequences, we can determine the amount of time each piece of freight needs to spend in each area of the yard. These times are fed to the Phase II planner, which assigns trains and blocks to specific tracks. Then, given the time windows and locations of the freight, the assignment of specific crews to process the freight are made in Phase III. Thus, the yard controller will arrive at detailed yard operating plans that are consistent with the schedule objectives determined by and passed down from the network planner because of the DASM's role in both the yard controller and the yard component of the network planner.

6.4. Chapter Summary

This chapter demonstrated the use of scheduling and sequencing theory for determining hump and assembly sequences in the yard. A previously developed model served as the starting point for this chapter. The intent of the basic model was to provide yard masters with a method to make these sequencing decisions, given a set of inbound trains ready to be humped, a set of blocks to be built, and a set of outbound trains to be assembled from those blocks. The method is fast, simple, and easy to implement as a computer program.

The primary goal of this chapter was to present the basic model and the modifications required for its operation as the yard component of the operational network planner. In addition, suggestions were offered for improving its performance on the problems it was originally intended to solve. Improvement of the revised model was demonstrated on an example provided in the reference (Ferguson [1993]) of the original model. We referred to the new model as the Disassembly/Assembly Sequencing Model (DASM).

Due to its complexity (or lack thereof), the DASM is ideal for use as the yard component of the network planner. In addition, the same model can serve as the foundation for a decision support tool to perform yard control. Such a tool would require a high level of modeling fidelity. This chapter proposed a three-phase decision process for making these more detailed decisions. The first phase is simply the DASM. The two additional phases handle the higher fidelity decisions, including the assignment of specific crews and tracks to trains and blocks. Using the DASM both at the network level and as the primary driver in the yard control decision support tool, we can ensure consistency between the schedule objectives passed down from the network planner and the decision support tools' ability to meet them.

The next step in researching this area would be to incorporate the yard sequencing model within the network planning framework described in Chapter 5. In that framework, the yard sequencing model should be linked with deterministic representations of line running times and the heuristic network coordinating algorithm (Section 5.2.2) to investigate the convergence properties of the combination of the three modeling components. A following step would be to add a line sequencing model, such as PLATO, into the framework as the line component of the network planner, and explore the results.

7. Summary and Future Research

The major contribution of this thesis is the introduction of a plan for hierarchical railroad planning and the design and formulation of a yard planning model that can be used to support network planning. This represents initial work in the area of real-time network planning and control. Extensions to this research are numerous. This chapter summarizes the work presented in the thesis and it describes areas for future research.

7.1. Summary of Thesis

The fundamental problem addressed in this thesis is **the development of a yard model to support operational network planning**. The first three chapters provide background information on the industry, on yard operations, and on recent modeling advances in yard and network planning. The third chapter provides a major transition in the thesis by expanding a historical taxonomy of rail models and proposing a hierarchical railroad planning system. Within that context, the final three chapters present yard planning models in addition to methods for coordinating decisions made across a network. This section will reiterate the key points of the thesis as presented at the end of each chapter.

Chapter 1 *Introduction*

The purpose of this chapter was to present general trends in the railroad industry, to motivate an understanding of the need for reliable service, and to demonstrate the industry's inability to meet that need. Unless greater reliability is provided, customers will seek alternate modes of freight transportation.

Events in the railroad industry in the last two decades have been highlighted by two major influences: the massive cost-cutting measures as the result of deregulation and the growth of intermodal service. Competition among rail carriers forced cost reductions. Railroads slashed personnel, track ownership, and equipment ownership. Externally, the rail industry faced stiff competition from motor carriers, whose industry had also been deregulated. Market share, in terms of total revenue, began to skew heavily toward the motor carriers. However, as the trucking industry began facing high driver turnover and increased fuel costs, they began to use rail carriers to move some of their long haul freight. Intermodal revenues for the rail carriers are now second only to coal revenues. In order to maintain this growth, service reliability must be improved.

Chapter 2 *Tutorial on Terminal Processing*

This tutorial provided a top-level description of freight processing at terminals. Three fundamental areas are used to process the freight: the receiving yard, the classification yard, and the departure yard. Variations exist in any yard, and the yard masters must consider many factors when planning a shift's work. Through a specific example of a major hump yard, this chapter provided some insight into terminal operations and the information available to the yard masters for making freight sequencing decisions and resource assignments.

Chapter 3 *A Paradigm for Railroad Modeling*

This chapter began with presentation of a railroad modeling taxonomy attributed to Assad [1980]. Citing recent modeling advances found in academic literature and in practice at major North American rail carriers, we added to that list of models. As it pertains to the operational planning and control of trains, the literature revealed that line planning research (i.e., real-time determination of meets and passes) is more advanced than yard sequencing research. There has been no work reported on real-time network planning and control, which combines line and yard planning.

In addition to describing the taxonomy and the literature review, this chapter presented a notional model for a railroad hierarchical planning system. This transcends the placement of models into a taxonomy and implies a well-defined framework that links all planning models—from strategic train routing decisions down to real-time line and yard control decisions. Beyond the notional architecture presented for such as system, this chapter explored the details of hierarchical planning at the operational level. Finally, we pinpointed the area at which this thesis is directed: the development of a yard model to support real-time railroad network planning.

Chapter 4 *Descriptive Modeling: An Object Oriented Framework*

This chapter offered a general design for developing yard models for use within a real-time network planning and control system. Such a design begins with an understanding of the data required to support the decision making. Using Object Oriented Modeling (OOM), we presented a static representation of yards and trains that will support a variety of modeling methods, such as heuristic rules, heuristic planning algorithms, and optimization models. Upon these static structures, we developed a dynamic yard model using a deterministic simulation approach. Finally, possible extensions were presented and include the development of a full simulation, yard planning algorithm testbed, or real-time yard control decision support model.

Chapter 5 *Modeling The Network*

This chapter offered a network representation that is consistent with the object oriented yard model presented in Chapter 4. A testbed has been created based upon this object oriented network design. It allows algorithms to be easily inserted and tested with other algorithms as part of the operational network planner. For the purpose of demonstration, the deterministic simulation created in Chapter 4 was incorporated as the yard planning component. Furthermore, a coordination heuristic was derived to serve as the network coordinating algorithm. For the final component, fixed line-haul times were used. The combination of the three components was illustrated on two examples.

Chapter 6 *A Yard Sequencing Model*

This chapter demonstrated the use of scheduling and sequencing theory for determining hump and assembly sequences in the yard. A previously developed model served as the starting point for this chapter. The intent of the original model was to provide yard masters with a method to make sequencing decisions, given a set of inbound trains ready to be humped, a set of blocks to be built, and a set of outbound trains to be assembled from those blocks. The method is fast, simple, and easy to implement.

The primary goal of this chapter was to present that basic model and modifications required for its operation as the yard component of the operational network planner. In addition, suggestions were offered for improving its performance on the problems it was originally intended to solve. Improvement of the revised model was demonstrated on an example provided in the reference (Ferguson [1993]). We refer to the new model as the Disassembly/Assembly Sequencing Model (DASM).

Due to its complexity (or lack thereof), the DASM is ideal for use as the yard component of the network planner. In addition, the same model can serve as the foundation for a decision support tool to perform yard control. Such a tool requires a high level of modeling fidelity. This chapter proposed a three-phase decision process for making these more detailed decisions. Using the DASM as the basis for the yard controller and at the network level, we can ensure consistency between the schedule objectives passed down from the network planner and the plans developed by the decision support tools.

7.2. Future Research

The areas of future research that need to be addressed fall into four categories: the hierarchical network planning system, the network planner, the deterministic yard simulation, and the yard sequencing model. Recommendations for specific research topics are included as bullet points under each heading.

Hierarchical Network Planning System

- The sequence of tactical planning models needs to be refined. This includes models for train schedule creation, train routing, freight blocking, and train make-up decisions. The results of using such a planning sequence should be contrasted with the performance attained under current operations. Furthermore, the benefits and drawbacks of solving these problems separately versus simultaneously should be identified.
- Recent advances in the optimization of tactical airline problems, such as fleet assignment and crew scheduling, should be explored for use in tactical level rail planning.
- The relationship between the tactical planning and the operational planning levels need further definition. In the hierarchical planning structure, models at the higher levels should represent the system consistently with the planning tools used at the lowest levels, albeit with a lower level of fidelity. This consistency must be enforced. The issue of aggregating operational decisions to describe yards and lines at the tactical and strategic planning levels needs to be investigated.

The Network Planner

- Recall the two requirements placed upon the network coordinating algorithm: first, it must ensure consistency between the planning subproblems; and second, it should enforce network objectives among the subproblems. The coordination heuristic currently only meets the first criterion. It should be modified (heuristically, of course) to meet the second requirement.
- A mechanism needs to be developed to create initial network schedules against which the network planner can be tested. A major burden in testing algorithms is developing network schedules by hand. The convergence properties of the coordination heuristic could then be explored under a wide range of traffic schedules and densities.
- Sequencing models, versus the deterministic simulation, should be incorporated into the network planner as the yard planning component. The convergence properties of the heuristic may be different under these circumstances and should be explored.

- The operational network planner may have an additional role in creating schedules at the tactical level. Using expected seasonal demands (that also account for weekly demand patterns), the same network planner *may* be able to create schedule objectives, or timetables, at the tactical level.
- Interfaces with a car scheduling system must be developed so that real-time planning decisions are not limited to schedule adjustments. Such an interface would allow adjustments also to be made to train consists. This implies interfaces with a locomotive distribution system to adjust power assignments to trains whose weights change.

Deterministic Yard Simulation/Object Oriented Modeling Structure

- The level of fidelity of the object model should be enhanced to allow more detailed representations of track topologies, specifically at the trim end of the yard (i.e., classification tracks, pull-out leads, and departure yard tracks).
- The object model should be generalized to allow tailoring for flat switching yards and intermodal yards.
- The deterministic simulation needs to be validated as a predictive mechanism for yard put-through times, if it will be used in that manner. This requires using large data sets from multiple yards.
- A testbed environment, as defined in the end of Chapter 4, should be developed from the deterministic simulation and used to analyze yard planning algorithms.

Yard Sequencing Model

- The Disassembly/Assembly Sequencing Model (DASM) should be implemented within the object oriented yard modeling structure. Analysis of the model as a "stand-alone" planner should be performed using real data. Performance comparisons should be made between the DASM and the actual decisions made in the yard. This will validate the model for use within the network planner and the yard control decision support tool.
- DASM should be considered for use as the basis for a yard controller. This includes formally defining the Phase II and Phase III planning problems and formulating assignment algorithms to solve these problems.
- Consideration should be provided for a similar approach to solving the sequencing problem for flat switching and intermodal yards.

- The sequencing model does not currently consider the value of freight. Weighted performance measures should be incorporated into the algorithm. Prior to that, a considerable amount of study must be devoted to determining these weights. They should be consistent with freight values as represented by the line component.

Appendix A - Selkirk Yard Layout

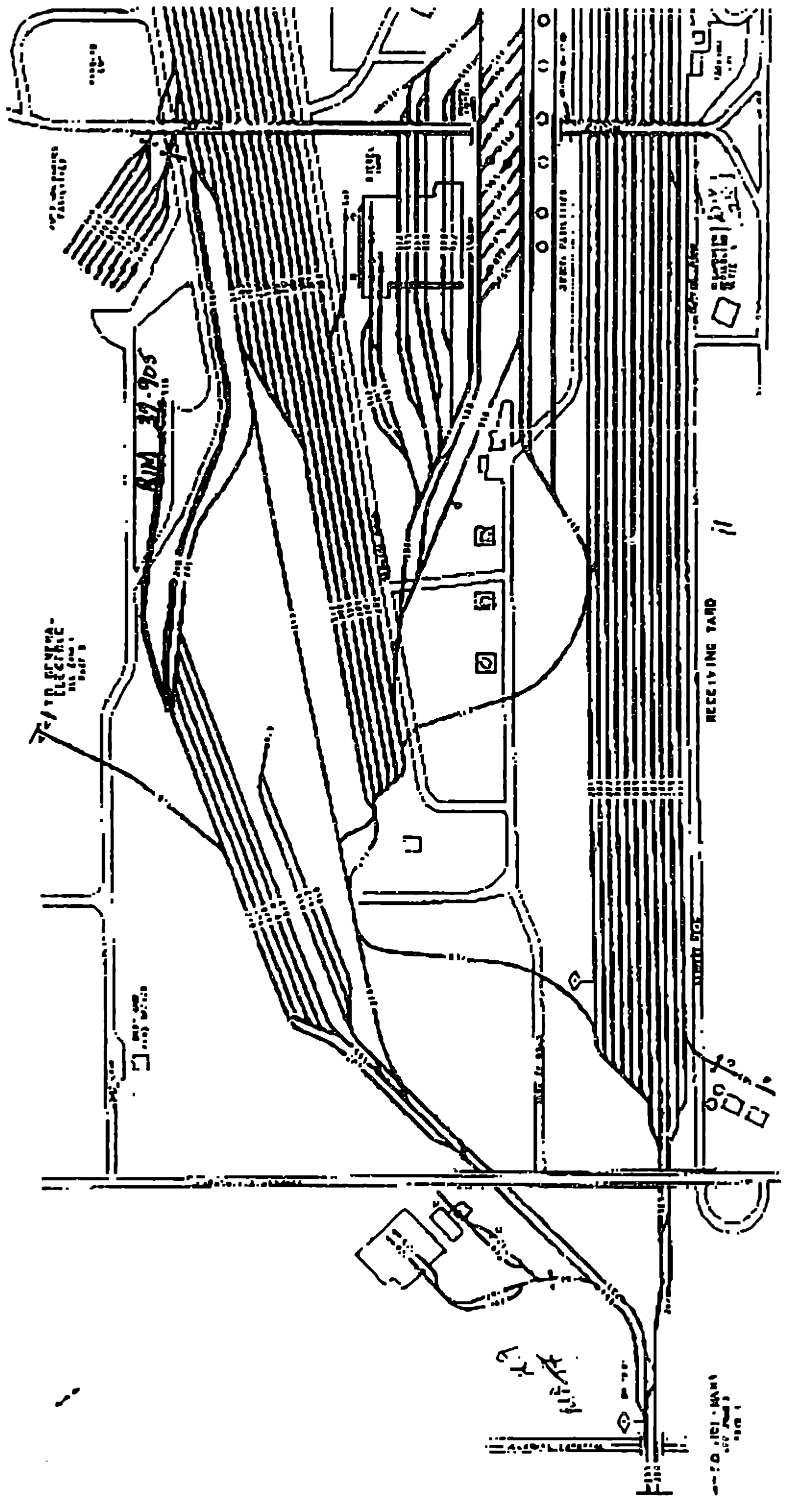
The purpose of this appendix is to provide drawings of the terminal described in Chapter 2, Tutorial on Terminal Processing. The intent is to present the physical size of a typical hump yard.

The original drawing has been divided into three sections. The first section is the receiving (i.e., west) end of the yard. Eastbound traffic enters the west end of the yard on the main tracks. Westbound trains enter from the opposite end of the and are routed to the receiving yard through the middle of the yard on one of two *fast freight* tracks. The eleven receiving tracks are shown in the lower part of the drawing. The other areas include tracks used to move locomotives to the diesel shop, tracks used to move cars to the local yard, and tracks at the west end of the north departure yard.

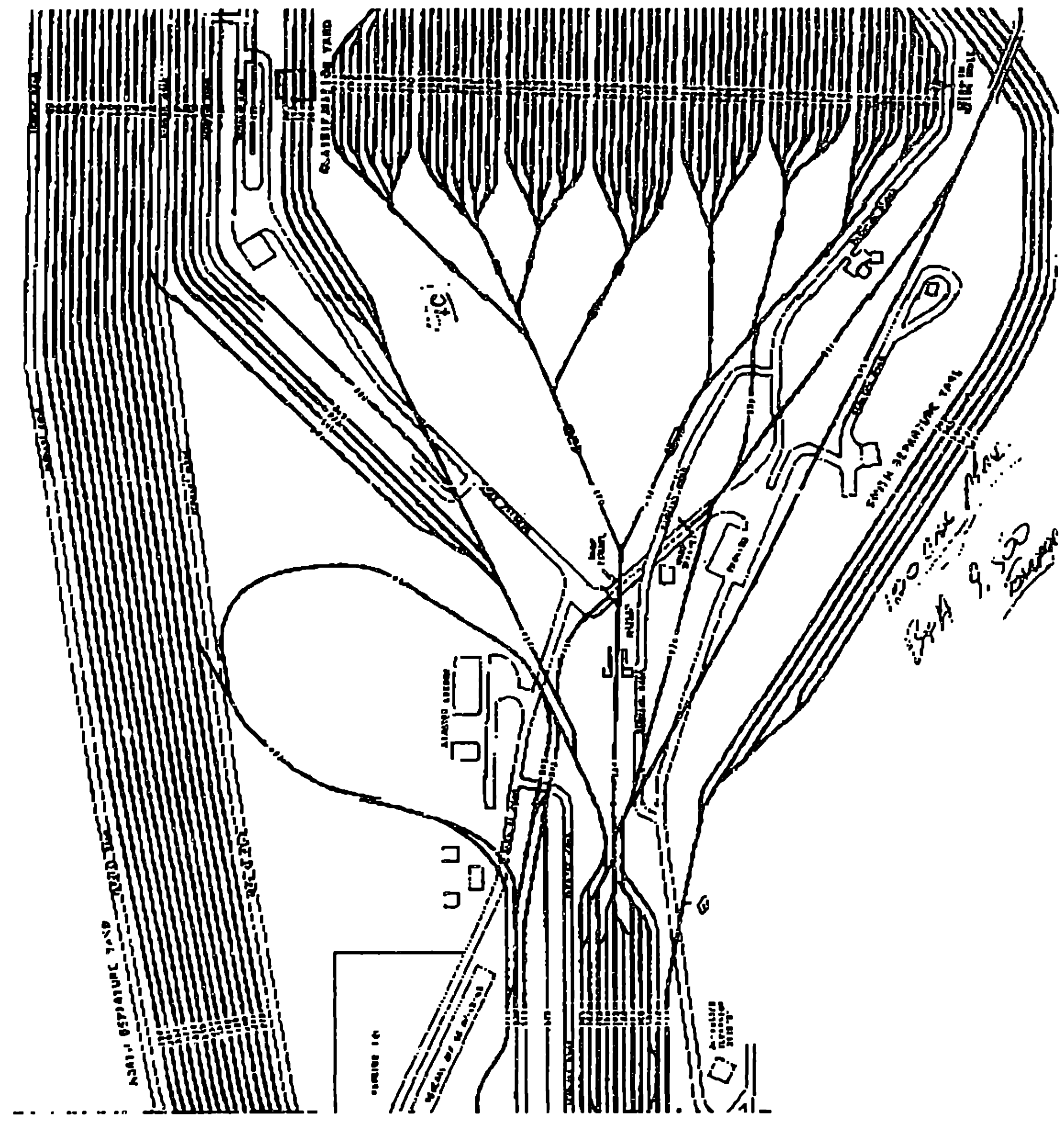
The middle section of the drawing highlights the hump, which is the single track running between the receiving yard to the classification yard. This diagram illustrates the complex set of switches that must be set for routing freight cars onto the correct track in the bowl. Here, we see the seven groups of ten tracks that comprise the bowl.

The final section shows the complicated web at the assembly end of the yard. Of note in this drawing are the three drill tracks, which can be found just south of the two main tracks that depart to the east. Blocks are pulled onto the drill tracks and the switches are aligned so that the trim engines can then push the freight onto the assigned track in either the north or south departure yard.

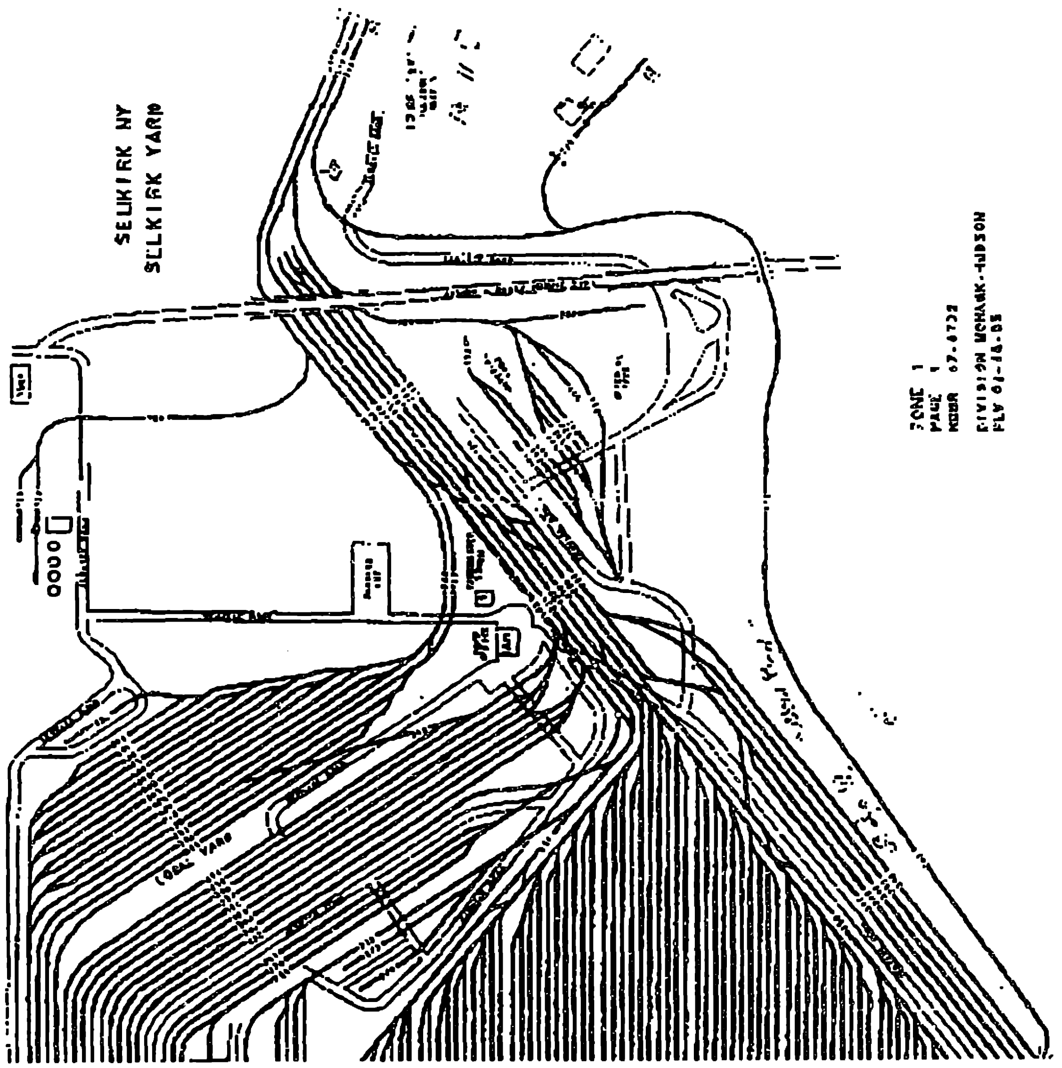
Receiving End of Yard



Hump, Bowl, and Departure Yards



Classification Yard, Drill Tracks, Departure Yards



Appendix B - Glossary of Terms for Object Oriented Modeling (OOM)

This section presents fundamental OOM terms and concepts. It is intended to be a simple reference so that the reader unfamiliar with OOM will be able to fully understand the design of the concepts and notation presented in Chapter 4, Descriptive Yard Modeling: An Object Oriented Framework.

The philosophy behind OOM is to develop a model using the real-world system as the basis for the design. Understanding and abstracting the key components of the real system and their relationship to each other is the key to successfully modeling the system using OOM. We will begin our discussion of OOM by first presenting the four fundamental concepts of object-orientation:

1) *Data Encapsulation.* Also known as identity, this characteristic means that all data and functions can be enveloped into discrete *objects*, which have *attributes* and *behavior*.

2) *Data Abstraction.* Also known as classification, abstraction means that all objects with the same attributes and behavior can be grouped as a class (denoted in **boldface**). Note that for different objects within a class , the common attributes may have different values for each object. For instance, a class **Circle** may have the attribute *radius*, yet for each instance, or object, in that class, the *radius* may have a different value.

3) *Polymorphism.* The same operation may induce different behavior on different classes. As an example, consider an **Ellipse** object and a **Rectangle** object, which are derived from the base class **Shape**. A common behavior may be to display the shape on the monitor. Therefore, *display()*, may be a function in the base class, but the actual behavior (i.e., the drawing routine) will be different, depending on whether the shape is a rectangle or an ellipse. Polymorphism is often implemented using the notion of *virtual functions*.

4) *Inheritance.* Classes may derive attributes or behavior from one or more different class. Inheritance implies a hierarchical structure. Classes that inherit are called *derived classes*, while those from which derived classes inherit are called *base classes*. For the most part, inheritance is used in order to mimic the real world existence of the objects being modeled. However, in some instances, inheritance is used as a convenient computational method for grouping classes that require the same functions or data members.

To represent these four characteristics, OOM uses three standard pictorial models to describe the static, dynamic, and functional aspects of the system model:

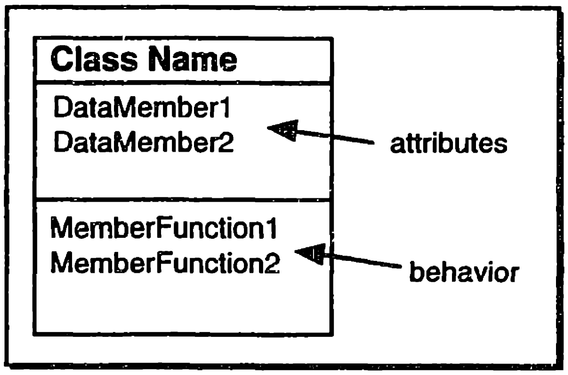
1) *Object Model.* This model defines the classes and how they relate to each other. All *data members* (attributes) and *member functions* (behavior) are defined at this level. An object model captures the static behavior of the system and is considered the most important of the three types of model, since the focus of OOM is to represent a systems as objects. The dynamic behavior will follow easily if the static relationships and attributes are defined carefully and accurately.

2) *Dynamic Model.* This model describes the system as its objects and their relationship change over time. The major concepts include *events*, or external stimuli, and *states*, which represent the values of objects.

3) *Functional Model.* This model describes the computations within a system. This model does not take into account the ordering of events. In relation to the other two, this model describes what happens, the dynamic model describes when it happens, and the object model describes "what it happens to."

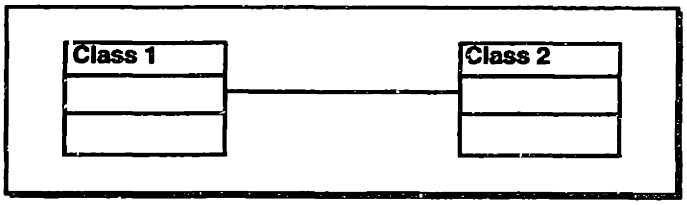
In the object, or static, model, a standard set of notation is used to show the relationships between the different classes. Here, we provide a concise summary of commonly used symbols:

Class Description:



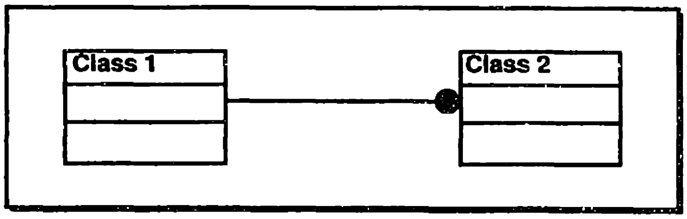
This symbol refers to generic class. Specific instances of a class are *objects.* In the description of the yard model, all classes will be represented in the general form. In most models, one is interested in describing the relationship between two or more objects and how those relationships change when some external force is applied to the system. Described below are some of the key associations that are used in Chapter 4.

One-To-One Association:



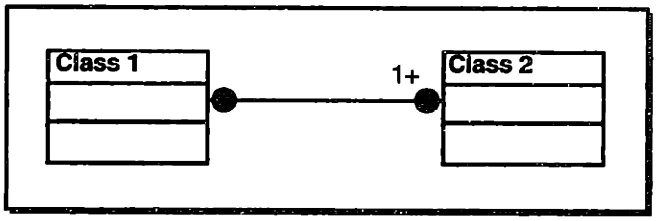
Here, two classes have a one-to-one relationship. A specific example would be if **Class 1** represents a State in the US and **Class 2** represents the capital city of that State. In terms of a implementing such a relationship, each object would have a single pointer to the other object.

One-To-Many Associations:



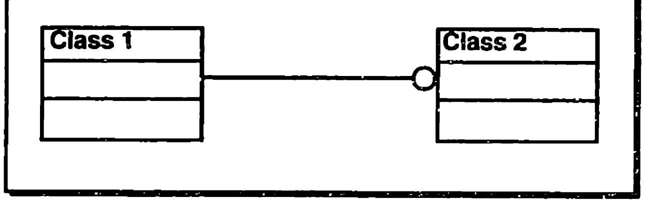
One class may be related to many objects of another class, yet each instance of a **Class 2** object has a relation with one instance of **Class 1**. The "-to-many" relationship is symbolized by the black *multiplicity ball*. An example would be **Class 1** representing the United States and **Class 2** representing the states. For implementation, **Class 1** would contain a linked list of pointers to the multiple objects of **Class 2** and those objects of **Class 2** would have a single pointer back to the **Class 1** object.

Many-To-Many Associations:



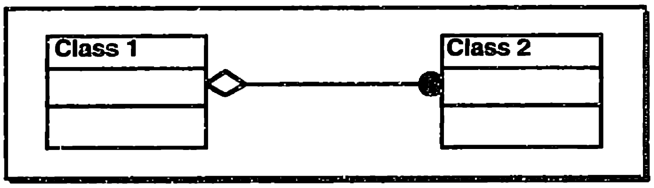
This relationship is similar to the above, except that the **Class 2** objects may relate back to more than a single **Class 1** object. The 1+ notation on the multiplicity ball places the requirement that the **Class 1** object must be related to one or more **Class 2** objects. One example could be courses at a university (**Class 1**) and students (**Class 2**). A course will exist only if one or more students enroll, and students may be enrolled in zero or more courses. The case of zero courses could represent students on Co-op or those enrolled for thesis/dissertation work. For the implementation, **Class 1** would have a list (possibly empty) of pointers to **Class 2** objects, while **Class 2** would have its own non-empty list of pointers to **Class 1** objects.

The Zero-One Relationship



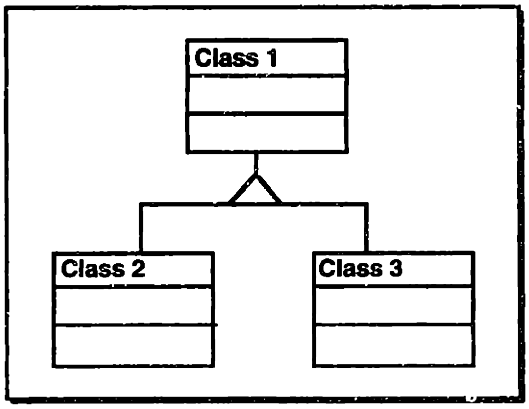
A hollow multiplicity ball indicates that **Class 1** is related to either "zero or one" **Class 2** objects. An example would be (in a monogamous society) **Class 1** is a man (or woman) and **Class 2** as a woman (or man) and the relationship between them is marriage. Implementing this relationship involves a **Class 1** object having a single pointer to a **Class 2** object or to NULL. All **Class 2** objects would have a single pointer to a **Class 1** object. You will notice that every **Class 2** object points to a **Class 1** object, but this is not the case for the reverse relationship.

Aggregation:



This relationship is stronger than an association. In this case, a **Class 1** object is an aggregate object which is made up of zero or more **Class 2** objects. In classifying a relationship as an aggregation, one should be able to say that the real-world relationship could be described as one class being *part of* the other. Although the diagram shows that a **Class 1** object *consists-of* zero or more **Class 2** objects, we can define the multiplicity as any kind of relationship. An example is the aggregation of **Door** objects to **Automobile** objects. The multiplicity ball, instead of having a "1+" notation, would have a "2, 4" notation. Such an aggregation would mean that an automobile consists of either 2 or 4 doors. The aggregation does not have to be a complete description of all the components of the aggregate system. It only needs to include those component in which you are interested in modeling.

Inheritance:



Inheritance is indicated by the open triangle beneath a class (the *base* class). One or more classes may inherit either data members or member functions from the base class. These are known as subclasses, or *derived* classes. An example would be computer monitors. The general characteristics and behavior would be included in the base class **Monitor** and these may include size, refresh rate, and resolution. The derived classes would then include **Color monitors** and **Black and white monitors**. Both would inherit the same members from the base class, but the *color* may include the number of colors support, while the black and white may include the level of grey scales supported. The differentiation between the two could go even deeper. But the key point to inheritance is that classes with common attributes or behavior can be derived from a base class that contains those common items.

This section has been provided to give the reader an understanding of the basic concepts and syntax of object-oriented modeling. The definitions and some of the examples are derived from **Object-Oriented Modeling and Design,** by Rumbaugh, *et al.* [1991]. For a more information about OOM, this book provides more detailed definitions and examples.

Appendix C - Sample Input and Output Files for Yard Model

This appendix provides sample input and output files for the deterministic simulation described in Chapter 4. The general formats of these files are presented in that chapter. The example presented here is from a nineteen train schedule running over the set of tracks as depicted in the screen dumps (Figure 4.12 and 4.13).

BLOCK AND CAR INPUT

Contains block number, origin, destination, number of cars in block, and listing of all cars in the block, including car number, length, weight, and value. In this case, cars are identical in all respects. The complete list of blocks is shown below in five columns.

```
101  8    4   5             40   50  1000  1       79   50  1000  1  112  4    5   9             156  50  1000  1
     1    50  1000  1       41   50  1000  1       80   50  1000  1       119  50  1000  1       157  50  1000  1
     2    50  1000  1       42   50  1000  1       81   50  1000  1       120  50  1000  1       158  50  1000  1
     3    50  1000  1       43   50  1000  1       82   50  1000  1       121  50  1000  1       159  50  1000  1
     4    50  1000  1  105  15   1   9             83   50  1000  1       122  50  1000  1       160  50  1000  1
     5    50  1000  1       44   50  1000  1       84   50  1000  1  113  9    6   4             161  50  1000  1
     6    50  1000  1       45   50  1000  1       85   50  1000  1       123  50  1000  1       162  50  1000  1
     7    50  1000  1       46   50  1000  1       86   50  1000  1       124  50  1000  1       163  50  1000  1
     8    50  1000  1       47   50  1000  1       87   50  1000  1       125  50  1000  1       164  50  1000  1
102  10   4   5             48   50  1000  1       88   50  1000  1       126  50  1000  1  118  11   7   5
     9    50  1000  1       49   50  1000  1  109  8    3   9             127  50  1000  1       165  50  1000  1
     10   50  1000  1       50   50  1000  1       89   50  1000  1       128  50  1000  1       166  50  1000  1
     11   50  1000  1       51   50  1000  1       90   50  1000  1       129  50  1000  1       167  50  1000  1
     12   50  1000  1       52   50  1000  1       91   50  1000  1       130  50  1000  1       168  50  1000  1
     13   50  1000  1       53   50  1000  1       92   50  1000  1       131  50  1000  1       169  50  1000  1
     14   50  1000  1       54   50  1000  1       93   50  1000  1  114  8    6   3             170  50  1000  1
     15   50  1000  1       55   50  1000  1       94   50  1000  1       132  50  1000  1       171  50  1000  1
     16   50  1000  1       56   50  1000  1       95   50  1000  1       133  50  1000  1       172  50  1000  1
     17   50  1000  1       57   50  1000  1       96   50  1000  1       134  50  1000  1       173  50  1000  1
     18   50  1000  1       58   50  1000  1  110  11   5   9             135  50  1000  1       174  50  1000  1
103  12   4   10       106  6    1   5             97   50  1000  1       136  50  1000  1       175  50  1000  1
     19   50  1000  1       59   50  1000  1       98   50  1000  1       137  50  1000  1  119  6    7   10
     20   50  1000  1       60   50  1000  1       99   50  1000  1       138  50  1000  1       176  50  1000  1
     21   50  1000  1       61   50  1000  1       100  50  1000  1       139  50  1000  1       177  50  1000  1
     22   50  1000  1       62   50  1000  1       101  50  1000  1  115  8    6   8             178  50  1000  1
     23   50  1000  1       63   50  1000  1       102  50  1000  1       140  50  1000  1       179  50  1000  1
     24   50  1000  1       64   50  1000  1       103  50  1000  1       141  50  1000  1       180  50  1000  1
     25   50  1000  1  107  14   3   6             104  50  1000  1       142  50  1000  1       181  50  1000  1
     26   50  1000  1       65   50  1000  1       105  50  1000  1       143  50  1000  1  120  10   1   5
     27   50  1000  1       66   50  1000  1       106  50  1000  1       144  50  1000  1       182  50  1000  1
     28   50  1000  1       67   50  1000  1       107  50  1000  1       145  50  1000  1       183  50  1000  1
     29   50  1000  1       68   50  1000  1  111  11   5   3             146  50  1000  1       184  50  1000  1
     30   50  1000  1       69   50  1000  1       108  50  1000  1       147  50  1000  1       185  50  1000  1
104  13   1   4             70   50  1000  1       109  50  1000  1  116  7    6   4             186  50  1000  1
     31   50  1000  1       71   50  1000  1       110  50  1000  1       148  50  1000  1       187  50  1000  1
     32   50  1000  1       72   50  1000  1       111  50  1000  1       149  50  1000  1       188  50  1000  1
     33   50  1000  1       73   50  1000  1       112  50  1000  1       150  50  1000  1       189  50  1000  1
     34   50  1000  1       74   50  1000  1       113  50  1000  1       151  50  1000  1       190  50  1000  1
     35   50  1000  1       75   50  1000  1       114  50  1000  1       152  50  1000  1       191  50  1000  1
     36   50  1000  1       76   50  1000  1       115  50  1000  1       153  50  1000  1  121  13   1   7
     37   50  1000  1       77   50  1000  1       116  50  1000  1       154  50  1000  1       192  50  1000  1
     38   50  1000  1       78   50  1000  1       117  50  1000  1  117  10   7   5             193  50  1000  1
     39   50  1000  1  108  10   3   10            118  50  1000  1       155  50  1000  1       194  50  1000  1
```

```
    195 50 1000 1      251 50 1000 1  133 12  7  6            362 50 1000 1      417 50 1000 1
    196 50 1000 1      252 50 1000 1      306 50 1000 1       363 50 1000 1      418 50 1000 1
    197 50 1000 1  127 14  5  3           307 50 1000 1       364 50 1000 1      419 50 1000 1
    198 50 1000 1      253 50 1000 1      308 50 1000 1       365 50 1000 1      420 50 1000 1
    199 50 1000 1      254 50 1000 1      309 50 1000 1       366 50 1000 1  144 12  1  5
    200 50 1000 1      255 50 1000 1      310 50 1000 1  138 5   4  1            421 50 1000 1
    201 50 1000 1      256 50 1000 1      311 50 1000 1       367 50 1000 1      422 50 1000 1
    202 50 1000 1      257 50 1000 1      312 50 1000 1       368 50 1000 1      423 50 1000 1
    203 50 1000 1      258 50 1000 1      313 50 1000 1       369 50 1000 1      424 50 1000 1
    204 50 1000 1      259 50 1000 1      314 50 1000 1       370 50 1000 1      425 50 1000 1
122 12  3  8           260 50 1000 1      315 50 1000 1       371 50 1000 1      426 50 1000 1
    205 50 1000 1      261 50 1000 1      316 50 1000 1  139 9   5  6            427 50 1000 1
    206 50 1000 1      262 50 1000 1      317 50 1000 1       372 50 1000 1      428 50 1000 1
    207 50 1000 1      263 50 1000 1  134 14  7  3            373 50 1000 1      429 50 1000 1
    208 50 1000 1      264 50 1000 1      318 50 1000 1       374 50 1000 1      430 50 1000 1
    209 50 1000 1      265 50 1000 1      319 50 1000 1       375 50 1000 1      431 50 1000 1
    210 50 1000 1      266 50 1000 1      320 50 1000 1       376 50 1000 1      432 50 1000 1
    211 50 1000 1  128 10  5  10          321 50 1000 1       377 50 1000 1  145 2   1  5
    212 50 1000 1      267 50 1000 1      322 50 1000 1       378 50 1000 1      433 50 1000 1
    213 50 1000 1      268 50 1000 1      323 50 1000 1       379 50 1000 1      434 50 1000 1
    214 50 1000 1      269 50 1000 1      324 50 1000 1       380 50 1000 1  146 8   3  1
    215 50 1000 1      270 50 1000 1      325 50 1000 1  140 9   5  3            435 50 1000 1
    216 50 1000 1      271 50 1000 1      326 50 1000 1       381 50 1000 1      436 50 1000 1
123 9   3  5           272 50 1000 1      327 50 1000 1       382 50 1000 1      437 50 1000 1
    217 50 1000 1      273 50 1000 1      328 50 1000 1       383 50 1000 1      438 50 1000 1
    218 50 1000 1      274 50 1000 1      329 50 1000 1       384 50 1000 1      439 50 1000 1
    219 50 1000 1      275 50 1000 1      330 50 1000 1       385 50 1000 1      440 50 1000 1
    220 50 1000 1      276 50 1000 1      331 50 1000 1       386 50 1000 1      441 50 1000 1
    221 50 1000 1  129 2   5  10      135 6   7  5            387 50 1000 1      442 50 1000 1
    222 50 1000 1      277 50 1000 1      332 50 1000 1       388 50 1000 1  147 12  3  4
    223 50 1000 1      278 50 1000 1      333 50 1000 1       389 50 1000 1      443 50 1000 1
    224 50 1000 1  130 12  6  5           334 50 1000 1  141 14  5  3            444 50 1000 1
    225 50 1000 1      279 50 1000 1      335 50 1000 1       390 50 1000 1      445 50 1000 1
124 13  3  6           280 50 1000 1      336 50 1000 1       391 50 1000 1      446 50 1000 1
    226 50 1000 1      281 50 1000 1      337 50 1000 1       392 50 1000 1      447 50 1000 1
    227 50 1000 1      282 50 1000 1  136 14  4  5            393 50 1000 1      448 50 1000 1
    228 50 1000 1      283 50 1000 1      338 50 1000 1       394 50 1000 1      449 50 1000 1
    229 50 1000 1      284 50 1000 1      339 50 1000 1       395 50 1000 1      450 50 1000 1
    230 50 1000 1      285 50 1000 1      340 50 1000 1       396 50 1000 1      451 50 1000 1
    231 50 1000 1      286 50 1000 1      341 50 1000 1       397 50 1000 1      452 50 1000 1
    232 50 1000 1      287 50 1000 1      342 50 1000 1       398 50 1000 1      453 50 1000 1
    233 50 1000 1      288 50 1000 1      343 50 1000 1       399 50 1000 1      454 50 1000 1
    234 50 1000 1      289 50 1000 1      344 50 1000 1       400 50 1000 1  148 13  3  1
    235 50 1000 1      290 50 1000 1      345 50 1000 1       401 50 1000 1      455 50 1000 1
    236 50 1000 1  131 12  6  3           346 50 1000 1       402 50 1000 1      456 50 1000 1
    237 50 1000 1      291 50 1000 1      347 50 1000 1       403 50 1000 1      457 50 1000 1
    238 50 1000 1      292 50 1000 1      348 50 1000 1  142 7   5  9            458 50 1000 1
125 3   3  9           293 50 1000 1      349 50 1000 1       404 50 1000 1      459 50 1000 1
    239 50 1000 1      294 50 1000 1      350 50 1000 1       405 50 1000 1      460 50 1000 1
    240 50 1000 1      295 50 1000 1      351 50 1000 1       406 50 1000 1      461 50 1000 1
    241 50 1000 1      296 50 1000 1  137 15  4  8            407 50 1000 1      462 50 1000 1
126 11  5  10          297 50 1000 1      352 50 1000 1       408 50 1000 1      463 50 1000 1
    242 50 1000 1      298 50 1000 1      353 50 1000 1       409 50 1000 1      464 50 1000 1
    243 50 1000 1      299 50 1000 1      354 50 1000 1       410 50 1000 1      465 50 1000 1
    244 50 1000 1      300 50 1000 1      355 50 1000 1  143 10  1  9            466 50 1000 1
    245 50 1000 1      301 50 1000 1      356 50 1000 1       411 50 1000 1      467 50 1000 1
    246 50 1000 1      302 50 1000 1      357 50 1000 1       412 50 1000 1  149 4   3  8
    247 50 1000 1  132 3   6  1           358 50 1000 1       413 50 1000 1      468 50 1000 1
    248 50 1000 1      303 50 1000 1      359 50 1000 1       414 50 1000 1      469 50 1000 1
    249 50 1000 1      304 50 1000 1      360 50 1000 1       415 50 1000 1      470 50 1000 1
    250 50 1000 1      305 50 1000 1      361 50 1000 1       416 50 1000 1      471 50 1000 1
```

TRAIN SCHEDULE INPUT

Having created the block objects from the previous input file, the inbound and outbound train objects are created based on the following assignments of blocks to trains. Both types of trains include a scheduled consist and an actual consist. When this file is read, the blocks are added to the scheduled consist. When the dynamic portion of the model (i.e., the deterministic simulation) starts, the scheduled consists for the arriving trains are copied over to the actual consists. The actual consists for the outbound trains are not built until the blocks are actually assembled in the model.

The format of the file shows, for each train, the train number, number of blocks in train, train origin, train destination, departure time from origin, arrival time at destination, and list of block numbers. The yard for which this model is being run is Yard 2, so each inbound train has 2 as its destination and each outbound train has 2 as its origin. The boldface type is not included in the file.

```
INBOUND TRAINS
1111  3     4     2     0000  0000
      101   102   103
1112  3     1     2     0000  0028
      104   105   106
1113  3     3     2     0018  0118
      107   108   109
1114  2     5     2     0028  0128
      110   112
1115  3     6     2     0116  0216
      113   115   116
1116  3     7     2     0133  0233
      117   118   119
1117  2     1     2     0220  0320
      120   121
1118  4     3     2     0636  0736
      122   123   124   125
1119  3     5     2     0748  0848
      126         128   129
1120  3     6     2     0844  0944
      130   131   132
1121  3     7     2     1026  . 1126
      133   134   135
1122  3     4     2     1103  1203
      136   137   138
1123  4     5     2     1120  1220
      139   140   141   142
1124  3     1     2     1449  1549
      143   144   145
1125  4     3     2     1518  1618
      146   147   148   149
OUTBOUND TRAINS
2222  4     2     1     2618  2718
      132   138   146   148
2223  3     2     3     1848  1948
      111   114   127
2224  2     2     3     2126  2226
      131   134
2225  2     2     3     2220  2320
      140   141
2226  2     2     4     1216  1316
      104   113
2227  2     2     4     2618  2718
      116   147
2228  3     2     5     1028  1128
      101   102   106
2229  2     2     5     1233  1333
      117   118
2230  3     2     5     1944  2044
      120   123   130
2231  2     2     5     2203  2303
      135   136
2232  2     2     5     2549  2649
      144   145
2233  2     2     6     1736  1836
      107   124
2234  2     2     6     2220  2320
      133   139
2235  1     2     7     1320  1420
      121
2236  2     2     8     1736  1836
      115   122
2237  2     2     8     2618  2718
      137   149
2238  2     2     9     1118  1218
      105   109
2239  4     2     9     2220  2320
      110   112   125   142
2240  1     2     9     2549  2649
      143
2241  2     2     10    1118  1218
      103   108
2242  3     2     10    1848  1948
      119   126   128
2243  1     2     10    1848  1948
      129
```

YARD TOPOLOGY

Having created the freight, the next step is to create the physical structure of the yard. The names in boldface do not appear in the file. We list the number of tracks in each area, followed by the length of each track. Tracks with length 9999 (i.e., the inbound main tracks) are sufficiently long to handle any traffic of concern. These inputs correspond to those visualized in the screen dumps shown in Figure 4.12 and 4.13.

MAIN:	2	9999	9999								
RECEIVING:	8	3000	3400	3800	4200	4200	3800	3400	3000		
CLASSIFICATION:	25	600	700	800	900	1000	1100	1200	1300	1400	1500
	1600	1700	1700	1800	1900	1800	1600	1500	1400	1300	1200
	1000	800	600	500							
DEPARTURE:	6	4000	4200	4400	4200	4100	4000				

YARD PROCESSING RATES

The next step is to specify the rates at which the activities occur in the yard. In the early design of the model, the hump and assembly processes were treated as **Activities** versus **Areas** (see Chapter 4 for definitions), so they were input with the activities. There is some information regarding track lengths for the hump track and the assembly track.

The length of the hump track is shown to be 9999 because there is never a limit on the size of the train that can pass over the hump. Likewise, in the current configuration of the model, the single assembly/pull-out track is configured as shown in Figure 4.1. In this case, blocks are pulled directly from the classification tracks to the departure yard (i.e., no push-pull activity). Thus, like the hump, the assembly track *in this case* has no train size limit.

Assembly Cut-off time:	0			
Assembly Mode:	2			
Inbound Inspection:	2	20	.4	
	0	2400	1200	1200
	0	2400	1200	1200
Hump Engines:	3	10	10	
	0	2400	1200	1200
	0	2400	1200	1200
	0	2400	1200	1200
Hump:	1	20	.5	9999
Trim Engines:	3	10	10	
	0	2400	1200	1200
	0	2400	1200	1200
	0	2400	1200	1200
Assembly:	1	20	.5	9999
Outbound Inspection:	2	20	.4	
	0	2400	1200	1200
	0	2400	1200	1200

The cut-off mode specifies whether we wait for all cars to be in place in the classification yard before building the outbound train (i.e., = 0) or if we start building the train some time, *c*, before its schedule departure time (i.e., = *c*). The assembly mode, as described in Chapter 4, specifies the manner in which engines are assigned to pull the blocks. Mode 1 is assembly by assigning a single engine to build an entire outbound train. Mode 2 is assembly by assigning all available engines to the next train to be built.

For each of the activities, except for **Hump** and **Assembly**, the inputs are: number or crews; β_0, the fixed time (in minutes) associated with an activity; β_1, the variable time (in minutes per unit length) associated with an activity; and, for each crew, the times for shift start, shift end, break start, and break end. For hump engines and trim engines, we assume only a fixed time component, so the entry for β_1 serves merely as a place holder. Finally for the **Hump** and **Assembly** activities, we show the number of tracks; the fixed processing rate, β_0; the variable processing rate, β_1; and the track length.

INVENTORY INPUT FILE

The purpose of this file is to initialize the state of the system by inputting information regarding trains and blocks that are already in the system at the start of the current planning period. From this data, the model will create the initial associations between the trains and blocks and their physical location in the yard, as well as the resources they are utilizing, such as tracks and crews.

In the example used here, there is a single train, Train 2223, in the yard at the beginning of the period. The data for the train are: train number, time of event completion (10), current processing activity (P = train departure), track number (departure track 1), crew number (none), lead track number (none), number of blocks, and block ID numbers.

```
Trains in Inventory:   1
                       2223   10   P   1   0   0   3   111   114   127
Blocks in Inventory:   0
```

SCHEDULE OUTPUT FILE

After the initialization of the data structures and prior to the running of the dynamic part of the model, the scheduled arrival and departures are output to file.

Following the run of the deterministic simulation, the actual arrivals and departure are output. The actual arrivals are the same as the scheduled arrivals, whereas the departures times change.

Scheduled Arrivals and Departures:

```
** ARRIVAL LISTING: **
ID    Origin  DepTime   Destin  ArrTime   # Blks  Length  Weight  Value
=================================================================

1111  4       0000      2       0000      3       1500    30000   30
1112  1       0000      2       0028      3       1700    34000   34
1113  3       0018      2       0118      3       1600    32000   32
1114  5       0028      2       0128      2       750     15000   15
1115  6       0116      2       0216      3       1200    24000   24
1116  7       0133      2       0233      3       1350    27000   27
1117  1       0220      2       0320      2       1150    23000   23
1118  3       0636      2       0736      4       1850    37000   37
1119  5       0748      2       0848      3       1150    23000   23
1120  6       0844      2       0944      3       1350    27000   27
1121  7       1026      2       1126      3       1600    32000   32
1122  4       1103      2       1203      3       1700    34000   34
1123  5       1120      2       1220      4       1950    39000   39
1124  1       1449      2       1549      3       1200    24000   24
1125  3       1518      2       1618      4       1850    37000   37

** DEPARTURE LISTING: **
ID    Origin  DepTime   Destin  ArrTime   # Blks  Length  Weight  Value
=================================================================

2222  2       2618      1       2718      4       1450    29000   29
2223  2       1848      3       1948      3       1650    33000   33
2224  2       2126      3       2226      2       1300    26000   26
2225  2       2220      3       2320      2       1150    23000   23
2226  2       1216      4       1316      2       1100    22000   22
2227  2       2618      4       2718      2       950     19000   19
2228  2       1028      5       1128      3       1200    24000   24
2229  2       1233      5       1333      2       1050    21000   21
2230  2       1944      5       2044      3       1550    31000   31
2231  2       2203      5       2303      2       1000    20000   20
2232  2       2549      5       2649      2       700     14000   14
2233  2       1736      6       1836      2       1350    27000   27
2234  2       2220      6       2320      2       1050    21000   21
2235  2       1320      7       1420      1       650     13000   13
2236  2       1736      8       1836      2       1000    20000   20
2237  2       2618      8       2718      2       950     19000   19
2238  2       1118      9       1218      2       1150    23000   23
2239  2       2220      9       2320      4       1250    25000   25
2240  2       2549      9       2649      1       500     10000   10
2241  2       1118      10      1218      2       1100    22000   22
2242  2       1848      10      1948      3       1350    27000   27
2243  2       1848      10      1948      1       100     2000    2
```

Actual Arrivals and Departures

```
** ARRIVAL LISTING: **
ID      Origin  DepTime     Destin  ArrTime     # Blks   Length  Weight  Value
===========================================================================

1111    4       0000        2       0000        3        1500    30000   30
1112    1       0000        2       0028        3        1700    34000   34
1113    3       0018        2       0118        3        1600    32000   32
1114    5       0028        2       0128        2        750     15000   15
1115    6       0116        2       0216        3        1200    24000   24
1116    7       0133        2       0233        3        1350    27000   27
1117    1       0220        2       0320        2        1150    23000   23
1118    3       0636        2       0736        4        1850    37000   37
1119    5       0748        2       0848        3        1150    23000   23
1120    6       0844        2       0944        3        1350    27000   27
1121    7       1026        2       1126        3        1600    32000   32
1122    4       1103        2       1203        3        1700    34000   34
1123    5       1120        2       1220        4        1950    39000   39
1124    1       1449        2       1549        3        1200    24000   24
1125    3       1518        2       1618        4        1850    37000   37

** DEPARTURE LISTING: **
ID      Origin  DepTime     Destin  ArrTime     # Blks   Length  Weight  Value
===========================================================================

2222    2       2851        1       2951        4        1450    29000   29
2223    2       0010        3       0110        3        1650    33000   33
2224    2       1746        3       1846        2        1300    26000   26
2225    2       2116        3       2216        2        1150    23000   23
2226    2       0826        4       0926        2        1100    22000   22
2227    2       2631        4       2731        2        950     19000   19
2228    2       0456        5       0556        3        1200    24000   24
2229    2       0936        5       1036        2        1050    21000   21
2230    2       1636        5       1736        3        1550    31000   31
2231    2       1856        5       1956        2        1000    20000   20
2232    2       2521        5       2621        2        700     14000   14
2233    2       1231        6       1331        2        1350    27000   27
2234    2       2006        6       2106        2        1050    21000   21
2235    2       1011        7       1111        1        650     13000   13
2236    2       1121        8       1221        2        1000    20000   20
2237    2       3001        8       3101        2        950     19000   19
2238    2       0716        9       0816        2        1150    23000   23
2239    2       2336        9       2436        4        1250    25000   25
2240    2       2411        9       2511        1        500     10000   10
2241    2       0606        10      0706        2        1100    22000   22
2242    2       1416        10      1516        3        1350    27000   27
2243    2       1451        10      1551        1        100     2000    2
```

CAR CONNECTION SUMMARY OUTPUT

In this model, we can summarize yard performance with a measure of the percentage of missed connections or with a measure of the average yard time. Recall the processing information input file, where the first entry was the cut-off time for assembly. That variable was set to zero, indicating that the outbound trains should not impose a cut-off and that all scheduled connections should be made. Thus, in the file shown below, which notes the connection times by train and by block, all of the connections have status of "Made." If cut-offs were imposed, blocks not making connections would be marked as "Missed." At the end of the file, the average yard time (in minutes, for made connections) and the "made connection" percentage are shown. These calculations could be easily modified to show the average lateness of outbound departures. The format of the file is, for each trains, the file lists the block number, the schedule yard time (in minutes) for the block, the status of the blocks, and the actual yard time (in minutes).

```
TRAIN #2222

132   994   Made   1147
138   855   Made   1008
146   600   Made   753
148   600   Made   753

TRAIN #2223

111   1129  Made   11
114   1129  Made   11
127   1129  Made   11

TRAIN #2224

131   702   Made   482
134   600   Made   380

TRAIN #2225

140   600   Made   536
141   600   Made   536

TRAIN #2226

104   708   Made   478
113   600   Made   370

TRAIN #2227

116   1442  Made   1455
147   600   Made   613

TRAIN #2228

101   628   Made   296
102   628   Made   296
106   600   Made   268

TRAIN #2229

117   600   Made   423
118   600   Made   423

TRAIN #2230

120   984   Made   796
123   728   Made   540
130   600   Made   412

TRAIN #2231

135   637   Made   450
136   600   Made   413

TRAIN #2232

144   600   Made   572
145   600   Made   572

TRAIN #2233

107   978   Made   673
124   600   Made   295

TRAIN #2234

133   654   Made   520
139   600   Made   466

TRAIN #2235

121   600   Made   411

TRAIN #2236

115   920   Made   545
122   600   Made   225

TRAIN #2237

137   855   Made   1078
149   600   Made   823

TRAIN #2238

105   650   Made   408
109   600   Made   358

TRAIN #2239

110   1252  Made   1328
112   1252  Made   1328
125   884   Made   960
142   600   Made   676

TRAIN #2240

143   600   Made   502

TRAIN #2241

103   678   Made   366
108   600   Made   288

TRAIN #2242

119   975   Made   703
126   600   Made   328
128   600   Made   328

TRAIN #2243

129   600   Made   363

PERCENTAGE OF CONNECTIONS MADE: 100

AVERAGE YARD TIME FOR CARS: 512.618
```

Appendix D - Summary of Notation

This appendix provides a summary of the notation presented for the algorithms in Chapters 5 and 6. The notation is grouped by chapter and alphabetized within each grouping.

Chapter 5 - Heuristic Coordinating Algorithm

a_{ij}^t	Arrival time of train i at node j for iteration t
$\mathbf{a}_j^t$	Vector of train arrival times at node j for iteration t
d_{ij}^t	Departure time of train i at node j for iteration t
$\mathbf{d}_j^t$	Vector of train departure times at node j for iteration t
δ	Flag variable for convergence
i	Index for trains, $i = 1,\ldots,T$
j	Index for planning nodes, $j = 1,\ldots,N$
j_i^{next}	Next planning node visited by train i after node j
N	Number of planning nodes
$NPA_j(\ )$	Node planning algorithm for node j
T	Number of trains
T_j	Set of trains at planning node j
t	Iteration index for algorithm

Chapter 6 - The Yard Sequencing Model

Symbol	Description	Defined in Step
A	Ordered set of trains assigned to be humped	4
B	Set of trains not assigned to be humped	4
β_0	Fixed time required between trains in hump sequence	4
c_{ij}	Completion time of (i,j), where $c_{ij} = ps_{ij} + p_{ij}$	5
d_{ij}	Due time for block (i,j)	1
d_j	Due time for train j	1

$\hat{d}$	Temporary storage of due times	1
$\hat{d}_{ij}$	Modified due date calculation for HJS/block (i,j)	4
HJS_{ij}	Hump Job Set corresponding to block (i,j)	2
h_k	Hump processing duration for inbound train k	4
hs_k	Hump start time for inbound train k	4
(i,j)	i^{th} block on outbound train j	1
$last_{ij}$	Last train ordered in HJS_{ij}	3
M	Number of inbound trains	2
m_k	Number of cars on inbound train k	2
μ	Hump processing rate (cars per minute)	2
N	Number of outbound trains	1
n_j	Number of blocks in outbound train j	1
p_{ij}	Pull-out/Assembly duration for block (i,j)	1
pos_{ij}^k	Position of last block (i,j) car on inbound train k	2
pr_{ij}	Pull-out/assembly ready time for block (i,j)	4
ps_{ij}	Pull-out/assembly start time for block (i,j)	5
s_{ij}	Maximum slack length in Hump Job Set (i,j),	3
s_{ij}^k	Slack length for block (i,j) on inbound train k	3
T_j	Tardiness of outbound train j	5
$T_{\max}$	Maximum tardiness of trains	5
τ	Current time	4
Ω	The ordered list of assembly jobs/HJSs arranged in increasing order of $\hat{d}_{ij}$	4
$\hat{\Omega}$	Copy of Ω, the ordered list of assembly jobs/HJSs	5

REFERENCES

Railroad and Transportation

Armstrong, J. H., *The Railroad: What It Is, What It Does: The Introduction to Railroading*, Simmons-Boardman, Omaha NE, 1982.

Assad, A. A., Models for Rail Transportation, *Transportation Research A*, Vol. 14A, pp. 205-220, 1980(a).

Assad, A. A., Modeling of Rail Networks: Toward a Routing/Makeup Model, *Transportation Research B*, Vol. 14B, pp. 101-114, 1980(b).

Assad, A. A., Analytical Models in Rail Transportation: An Annotated Bibliography, *INFOR*, Vol. 19, No. 1, pp. 59-80, 1981.

Barnhart, C., E. L. Johnson, R. Anbil, L. Hatay, A Column-Generation Technique for the Long-Haul Crew-Assignment Problem, *Optimization in Industry 2*, Wiley, 1994.

Bodin, L.D., B. L. Golden, A. D. Schuster, W. Romig, A Model for the Blocking of Trains, *Transportation Research B*, Vol. 14B, pp. 115-120, 1980.

Chatlosh, B. R., The Simulation of the Railroad Classification Yard, *M.S. Thesis*, Michigan Technological Institute, 1991.

Christodouleas, J., A Brief Discussion of a New Formulation for the High-Level Rail Planning Problem, *Internal memorandum*, 20 January 1994.

Crainic, T. G., J. A. Ferland, J.-M. Rousseau, A Tactical Planning Model for Rail Freight Transportation, *Transportation Science, Vol.* 18 No. 2, pp. 165-184, 1984.

Crainic, T. G., J.-M. Rousseau, Multicommodity, Multimode Freight Transportation: A General Modeling and Algorithmic Framework for the Service Network Design Problem, *Tranportation Research B*, Vol. 20B, No. 3 pp. 225-242, 1986.

Daganzo, C. F., Dowling, R. G., Hall, R. W. Railroad Classification Yard Throughput: The Case of Multistage Triangular Sorting, *Transportation Research A*, Vol. 17A No. 2, pp. 95-106, 1983.

Daganzo, C. F., Static Blocking at Railyards: Sorting Implications and Track Requirements, *Transportation Science*, Vol. 20 No. 3, pp. 189-199, 1986.

Daganzo, C. F., Dynamic Blocking for Railyards: Part I. Homogeneous Traffic, *Transportation Research B*, Vol. 21B No. 1, pp. 1-27,, 1987(a).

Daganzo, C. F., Dynamic Blocking for Railyards: Part II. Heterogeneous Traffic, *Transportation Research B*, Vol. 21B No. 1, pp. 29-40, 1987(b).

Draper Laboratory Internal Publication, *PLATO Interface Document*, *C.S. Draper Laboratory*, 1994.

Deloitte Haskins & Sells, *Rail Terminal Sequencing System: General Design Manual*, Prepared for Freight Car Demonstration Program, Association of American Railroads, 1978.

Duffy, M. A. , *Statistical Process Control Applied to Rail Freight Terminal Performance*, S.M. Thesis, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, 1994.

Ferguson, III, Robert T. *Mathematical Modeling of the Railroad Switching Process*. PhD Dissertation, School of Systems Engineering, University of Virginia, 1993.

Forbes, M. A., J. N. Holt, A. M. Watts, Exact Solution of Locomotive Scheduling Problems. *Journal of the Operational Research Society*, Vol. 42 No. 10, 1991.

Gopalan, R., Operations Research at US Air, *Presentation to the MIT Operations Research Center*,, 1995.

Guignard, Monique, Kraft, Edwin R., A Mixed-Integer Optimization Model to Improve Freight Car Classification in Railroad Yards, *Working Paper 93-06-06*, Department of Operations and Information Management, The Wharton School, University of Pennsylvania, November 1993.

Haghani, A. E., Formulation and Solution of a Combined Train Routing and Makeup, and Empty Car Distribution Model, *Transportation Research B*, Vol. 23B, No. 6, pp. 433-452, 1989.

Hall, R. W., Vehicle Scheduling at a Transportation Terminal with Random Delay en Route. *Transportation Science*, Vol. 19 No. 3, August, 1985.

Hane, C. A., C. Barnhart, E. L. Johnson, R. E. Marsten, G. L. Nemhauser, G. Sigismondi, The Fleet Assignment Problem: Solving a Large-Scale Integer Program, 1994. (to appear in *Mathematical Programming*)

Hocker, G. A., *Airport Demand and Capacity Modeling for Flow Management Analysis*. SM Thesis, Operations Research Center, Massachusetts Institute of Technology, January, 1994.

Jovanovic, D. and Harker, P. T., Tactical Scheduling of Rail Operations: The SCAN I System, *Transportation Science*, Vol. 25, No. 1, February 1991.

Keaton, M. H., Designing Optimal Railroad Operating Plans: Lagrangian Relaxation and Heuristic Approaches, *Transportation Research B*, Vol. 23B, No. 6, pp. 415-431, 1989.

Keaton, M. H., Service-Cost Tradeoffs for Carload Freight Traffic in the U.S. Rail Industry, *Transportation Research A*, Vol. 25A, pp. 363-374, 1991.

Keaton, M. H., Designing Railroad Operating Plans: A Dual Adjustment Method for Implementing Lagrangian Relaxation, *Transportation Science*, Vol. 26, No. 4, 1992.

Klincewicz, J. G. Solving a Freight Transport Problem Using Facility Location Techniques, *Operations Research*, Vol. 38 No. 1, 1990.

McCarren, J. R., C. D. Martland, The MIT Service Planning Model, *MIT Studies in Railroad Operations and Economics*, Vol. 36, 1980.

Martland, C. D., PMAKE Analysis: Predicting Rail Yard Time Distributions Using Probabilistic Train Connecting Standards, *Transportation Science*, Vol. 16 No. 4, November, 1982.

Martland, Carl D., P. Little, M. Duffy, Y. Dong, Benchmarks for Rail Hump Yard Performance, *AAR Affiliated Laboratory Working Paper 94-3*, June 1994.

Martland, C.D., Little, Patrick, Duffy, Michael, Dong, Yan, Development of Concepts for Linking Railroad Terminal Control Systems To Advance Line Control Systems, *Final Report to the Charles Stark Draper Laboratory*, June 1993.

Peterson, E.R., Railyard Modeling: Part I. Prediction of Put-Through Time, *Transportation Science*, Vol. 11 No. 1, February 1977.

Peterson, E.R., Railyard Modeling: Part II. The Effect of Yard Facilities on Congestion, *Transportation Science*, Vol. 11 No. 1, February 1977.

Phillips, R. L., D. W. Boyd, T. A. Grossman, Jr., An Algorithm for Calculating Consistent Itinerary Flows. *Transportation Science*, Vol. 25 No. 3, August 1991.

Robinson, J. D., *A Simulation Testbed for Flow Management Analysis in Air Traffic Control*, SM Thesis, Operations Research Center, Massachusetts Institute of Technology, Cambridge MA, 1992.

Siddiqee, M. W., Investigation of Sorting and Train Formation Schemes for a Railroad Hump Yard, *Traffic Flow and Transportation*, Ed. G.F. Newell, pp. 377-388. Elsevier, NY, 1972.

Tsitsiklis, J. N., Discussion of Some Issues Relate to Decomposition Methods for the Line Planning Problem, Internal memorandum, November 1993.

Turnquist, M. A., M. S. Daskin, Queuing Models of Classification and Connection Delays in Raliyards, *Transportation Science*, Vol. 16 No. 2, May 1982.

Van Dyke, C. D., The Automated Blocking Model: A Practical Approach to Freight Railroad Blocking Plan Development, *Proceedings, Twenty-Seventh Annual Meeting of the Transportation Research Forum*, Vol. 27, pp. 116-121, 1986.

Voigtlaender, C. H., Intermodal Freight Transportation - An Integrated Analysis of Strategy and Operations, SM Thesis, Operations Research Center, Massachusetts Institute of Technology, June 1994.

Wong, P. J., B. Conrad, M. J. Johnson, N. Lay, Tactical Planning for Coordinating Railroad Operations, Presentation from the *Fifty Ninth Annual Meeting of the Transportation Research Board*, January 21-25, 1980.

Yagar, S., F. F. Saccomanno, Q. Shi, An Efficient Sequencing Model for Humping in a Rail Yard, *Transportation Research A*, Vol. 17A No. 4, 1983.

Sequencing Theory

Ashour, S., *Sequencing Theory*, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, 1972.

Baker, K. R., *Introduction to Scheduling and Sequencing*, Wiley, New York, 1974.

Baker, K. R., Scudder, G. D. Sequencing with Earliness and Tardiness Penalties: A Review, *Operations Research*, Vol. 38 No. 1, pp. 22-36, January-February 1990.

Bellman, R., *Mathematical Aspects of Scheduling and Applications*, Pergamon Press, Oxford, 1982.

Blazewicz, W. Cellary, R. Slowinski, J. Weglarz, *Scheduling Under Resource Constraints: Deterministic Models*, J. C. Baltzer, Basel, 1986.

Blazewicz, W., K. Ecker, G. Schmidt, J. Weglarz, *Scheduling in Computer and Manufacturing Systems*, Springer-Verlag, Berlin, 1993.

Coffman, E. G., *Computer and Job-Shop Scheduling Theory*, Wiley, New York, 1976.

Conway, R. W., W. L. Maxwell, L. W. Miller, *Theory on Scheduling*, Addison-Wesley, Reading, MA, 1967.

Davis, J. S., J. J. Kanet, Single-Machine Scheduling with Early and Tardy Completion Costs, *Naval Research Logistics*, Vol. 40, pp. 85-101, 1993.

Elmaghraby, S. E., *Activity Networks: Project Planning and Control by Network Models*, Wiley, 1977.

Fadlalla, A., J. R. Evans, M. S. Levy, A Greedy Heuristic for the Mean Tardiness Sequencing Problem, *Computers and Operations Research*, Vol. 21 No. 3, pp. 329-336, 1994.

Feo, T. A., K. Venkatraman, Bard, J. F., A GRASP^{TM} For a Difficult Single Machine Scheduling Problem, *Computers and Operations Research*, Vol. 18 No. 8, 1991.

Fisher, M. L. , A Dual Algorithm for the One-Machine Scheduling Problem, *Mathematical Programming*, Vol. 11, pp. 229-251 1976.

French, S., *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Ellis Horwood, Chichester, 1982.

Holsenback, J. E., Russell, R. M. A Heuristic Algorithm for Sequencing on One Machine to Minimize Total Tardiness. *Journal of the Operational Research Society*, Vol. 43 No. 1, 1992.

Lee, I.-S., A Worst-case Performance of the Shortest-processing-time Heuristic for Single Machine Scheduling, *Journal of the Operational Research Society*, Vol. 42, No. 10, 1991.

Liu, J., B. L. MacCarthy, Effective Heuristics for the Single Machine Sequencing Problem with Ready Times, *International Journal of Production Research*, Vol. 28 No. 8, 1991.

Panwalker, S. S., M. L. Smith, C. P. Koulamas, A Heuristic for the Single Machine Tardiness Problem, *European Journal of Operational Research*, Vol. 70, pp. 304-310, 1993.

Rinnooy Kan, A. H. G., *Machine Scheduling Problems: Classification, Complexity, and Computations*, Martinus Nijhoff, The Hague, 1976.

Sen, T., B. Borah, On the Single-machine Scheduling Problem with Tardiness Penalties, *Journal of the Operational Research Society*, Vol. 42, No. 8, 1991.

Sprecher, A. *Resource-Constrained Project Scheduling: Exact Methods for the Multi-Mode Case*, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, 1993.

Swarc, W., J. J. Liu, Weighted Tardiness Single Machine Scheduling with Proportional Weights, *Management Science*, Vol. 39, No. 5, May 1993.

Yano, C. A., Y.-D. Kim, Algorithms for a class of single-machine weighted tardiness and earliness problems, *European Journal of Operational Research,* Vol. 52, 1991.

Zheng, W.-X., H. Nagasawa, N. Nishiyama, Single-Machine Scheduling for Minimizing Total Cost with Identical, Asymmetrical Earliness and Tardiness Penalties, *International Journal of Production Research,* Vol. 31, No. 7, pp. 1611-1620, 1993.

General

Anderson, A. E., W. J. Heinze, *C++ Programming and Fundamental Concepts,* Prentice Hall, Edgewood Cliffs, NJ, 1992.

Burke, P. J., Delays in Single Server Queues with Batch Input, *Operations Research,* Vol. 23, 1975.

Hall, W. D., *Resource-Coordinated Hierarchical Planning for Real-Time Autonomous Systems, S.M. Thesis*, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1992.

Kelley, A., I. Pohl, *A Book on C: Programming in C,* Benjamin/Cummings, Redwood City, CA, 1994.

Lasdon, L. S. *Optimization Theory for Large Systems,* The MacMillan Co., New York, NY, 1970.

Rumbaugh, J., M. Blaha, W. Premerlani, W. Lorenson, *Object-Oriented Modeling and Design,* Prentice Hall, EdgewoodCliffs, NJ, 1991.