DISTRIBUTED DECISION MAKING USING
A DISTRIBUTED MODEL

by

Robert R. Tenney

This report is based on the unaltered thesis of Robert R. Tenney, submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the Massachusetts Institute of Technology in June 1979. The research was conducted at the M.I.T. Laboratory for Information and Decision Systems with support provided by the Office of Naval Research under ONR Contract N00014-77-C-0532.

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139
For Dr. Zachery Smith
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.1</td>
<td>Static Abstraction</td>
<td>86</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Dynamic Abstraction</td>
<td>99</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Domular Abstraction</td>
<td>107</td>
</tr>
<tr>
<td>4.</td>
<td>Global Coordination</td>
<td>114</td>
</tr>
<tr>
<td>4.1</td>
<td>Singly Connected Systems</td>
<td>115</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Introduction</td>
<td>116</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Static Teams</td>
<td>122</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Silent Coordination</td>
<td>132</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Selfish Coordination</td>
<td>137</td>
</tr>
<tr>
<td>4.1.5</td>
<td>Short-Term Coordination</td>
<td>147</td>
</tr>
<tr>
<td>4.1.6</td>
<td>Coordination with Abstraction</td>
<td>160</td>
</tr>
<tr>
<td>4.1.7</td>
<td>Short-Term Coordination with Abstraction</td>
<td>171</td>
</tr>
<tr>
<td>4.1.8</td>
<td>Conclusion</td>
<td>175</td>
</tr>
<tr>
<td>4.2</td>
<td>More General Problems</td>
<td>187</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Static Strategies</td>
<td>193</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Silent Strategies</td>
<td>198</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Selfish Strategies</td>
<td>201</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Strategies with Abstraction</td>
<td>207</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Conclusion</td>
<td>211</td>
</tr>
<tr>
<td>5.</td>
<td>Application Example</td>
<td>215</td>
</tr>
<tr>
<td>5.1</td>
<td>Background</td>
<td>217</td>
</tr>
<tr>
<td>5.2</td>
<td>System Topology</td>
<td>221</td>
</tr>
<tr>
<td>5.3</td>
<td>Basic Elements</td>
<td>231</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Elevators</td>
<td>232</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Transportation</td>
<td>237</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4</td>
<td>Evaluation of Coordination Strategies</td>
<td>241</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Short Term Coordination</td>
<td>244</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Abstraction</td>
<td>247</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Hybrid Coordination</td>
<td>249</td>
</tr>
<tr>
<td>6.</td>
<td>Conclusion</td>
<td>251</td>
</tr>
<tr>
<td>6.1</td>
<td>Summary</td>
<td>251</td>
</tr>
<tr>
<td>6.2</td>
<td>Future Work</td>
<td>256</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Specific</td>
<td>256</td>
</tr>
<tr>
<td>6.2.2</td>
<td>General</td>
<td>259</td>
</tr>
</tbody>
</table>

References                      261
Biography                       265
<table>
<thead>
<tr>
<th>Number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Control System Design Methodologies</td>
<td>18</td>
</tr>
<tr>
<td>2.1</td>
<td>Information Flow in a Typical Organization</td>
<td>36</td>
</tr>
<tr>
<td>2.2</td>
<td>Equivalent Domular Structure</td>
<td>37</td>
</tr>
<tr>
<td>2.3</td>
<td>Decision Tree with Static Evaluators</td>
<td>40</td>
</tr>
<tr>
<td>3.1</td>
<td>Example Topology</td>
<td>51</td>
</tr>
<tr>
<td>3.2</td>
<td>Dynamics of $D_2$</td>
<td>52</td>
</tr>
<tr>
<td>3.3</td>
<td>Dynamics of $D_1$</td>
<td>54</td>
</tr>
<tr>
<td>3.7</td>
<td>Interdependence of $\sigma$ and $\rho_i$</td>
<td>66</td>
</tr>
<tr>
<td>3.9</td>
<td>Problems with Forward Search Convergence</td>
<td>75</td>
</tr>
<tr>
<td>3.10</td>
<td>Stochastic System with Reachability Differences</td>
<td>80</td>
</tr>
<tr>
<td>3.11</td>
<td>Use of Abstraction: Static Case</td>
<td>89</td>
</tr>
<tr>
<td>3.12</td>
<td>Abstraction for Prediction</td>
<td>97</td>
</tr>
<tr>
<td>3.13</td>
<td>Closed-Loop Abstraction</td>
<td>103</td>
</tr>
<tr>
<td>3.14</td>
<td>Open-Loop Abstraction</td>
<td>106</td>
</tr>
<tr>
<td>3.15</td>
<td>Abstraction of Superior</td>
<td>111</td>
</tr>
<tr>
<td>4.1</td>
<td>Topologies</td>
<td>119</td>
</tr>
<tr>
<td>4.2</td>
<td>Induction Relation</td>
<td>121</td>
</tr>
<tr>
<td>4.3</td>
<td>One-Step Decisions</td>
<td>124</td>
</tr>
<tr>
<td>4.4</td>
<td>Static Coordination: Communication</td>
<td>130</td>
</tr>
<tr>
<td>4.7</td>
<td>Timing in Limited Prediction Selfish Coordination</td>
<td>145</td>
</tr>
<tr>
<td>4.8</td>
<td>Costs for Three-Step Look-Ahead</td>
<td>154</td>
</tr>
<tr>
<td>4.9</td>
<td>Costs for Four-Step Look-Ahead</td>
<td>157</td>
</tr>
<tr>
<td>4.10</td>
<td>Using Abstraction to Generate Commands</td>
<td>169</td>
</tr>
<tr>
<td>4.11</td>
<td>Relative Performance Bounds</td>
<td>177</td>
</tr>
<tr>
<td>4.12</td>
<td>Relative Communication Needs</td>
<td>179</td>
</tr>
<tr>
<td>4.13</td>
<td>Relative Processing Requirements</td>
<td>180</td>
</tr>
<tr>
<td>Number</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.15</td>
<td>Examples of Topologies</td>
<td>190</td>
</tr>
<tr>
<td>4.16</td>
<td>Relationships of Topology Classes</td>
<td>191</td>
</tr>
<tr>
<td>4.17</td>
<td>Doubly Connected Static Coordination</td>
<td>195</td>
</tr>
<tr>
<td>4.18</td>
<td>Modification of Acyclic Topologies</td>
<td>196</td>
</tr>
<tr>
<td>4.19</td>
<td>Most General Statically Coordinable Systems</td>
<td>199</td>
</tr>
<tr>
<td>4.21</td>
<td>Inconsistency in Cyclic Selfish System</td>
<td>204</td>
</tr>
<tr>
<td>4.23</td>
<td>Separation of Possible Interaction Set</td>
<td>208</td>
</tr>
<tr>
<td>5.1</td>
<td>System Topology: Supply, Country Elevators, and Railroads</td>
<td>222</td>
</tr>
<tr>
<td>5.3</td>
<td>System Topology: Railroads, Terminals, and Ports</td>
<td>228</td>
</tr>
<tr>
<td>5.4</td>
<td>System Topology: Subterminals, Barges, and River Houses</td>
<td>229</td>
</tr>
<tr>
<td>5.5</td>
<td>System Topology: Overview</td>
<td>230</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>Local and Centralized Models</td>
<td>57</td>
</tr>
<tr>
<td>3.5</td>
<td>Equivalent Centralized Model for Example</td>
<td>59</td>
</tr>
<tr>
<td>3.6</td>
<td>Relation of Deterministic to Stochastic Systems</td>
<td>64</td>
</tr>
<tr>
<td>3.8</td>
<td>Example Computation of Forward Search</td>
<td>74</td>
</tr>
<tr>
<td>4.5</td>
<td>Independent Control Algorithm for Example Supremal</td>
<td>136</td>
</tr>
<tr>
<td>4.6</td>
<td>Selfish Control for Example</td>
<td>142</td>
</tr>
<tr>
<td>4.14</td>
<td>Comparison of Coordination Strategies</td>
<td>162</td>
</tr>
<tr>
<td>4.20</td>
<td>Suitability of Static Coordination</td>
<td>200</td>
</tr>
<tr>
<td>4.22</td>
<td>Suitability of Selfish Coordination</td>
<td>206</td>
</tr>
<tr>
<td>4.24</td>
<td>Suitability of Abstraction</td>
<td>212</td>
</tr>
<tr>
<td>4.25</td>
<td>Match of Strategies to Topologies</td>
<td>213</td>
</tr>
<tr>
<td>5.2</td>
<td>Average Transportation Costs (1973)</td>
<td>226</td>
</tr>
<tr>
<td>5.6</td>
<td>Average Grain Storage Costs</td>
<td>236</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

Throughout the period of effort that has gone into this thesis, I have incurred debts to people far too numerous to mention. All of them deserve thanks for their support, ideas, and patience, but a few are so outstanding that they must be mentioned here.

This work began with an intense interest in the concept of decentralization of control, and a feeling in my bones that something fundamental was at the heart of difficulties encountered in going from a centralized to decentralized point of view. Nils Sandell, Jr., and Michael Athans deserve boundless praise for their willingness to let me explore the fundamental issues with no guarantee of success, and constant encouragement when little was obtained over long periods. The National Science Foundation graduate fellowship program, which supported the first two years of work, also deserves thanks for its confidence in me.

Over the past year, more people have aided my efforts to delve into these issues. Alan Willsky and Marvin Minsky helped distill some thoughts, the latter with respect to ideas in artificial intelligence which have grown from seeds planted years ago by John Bell. The Office of Naval Research has funded this last year under contract N00014-77-C-0532. Alex and Elza Levis and Beth Ducot all provided extremely useful and timely help in the development of the example of Chapter Five -- their instant access to reams of data was a godsend late this spring.

Beyond technical and financial support, however, has been the personal interactions which have proven so valuable. Dennis Fromholzer deserves thanks for his understanding of my feelings during this period (perhaps due to the fact that he was ensnared in his own thesis work
ACKNOWLEDGEMENTS

also) and my warmest appreciation goes to Suetta Baker for her companionship, understanding, and help even while pursuing studies in medical school. Finally, my parents deserve thanks for their patience, and especially my mother for her effort devoted to the typing of the manuscript.
DEDICATION

To My Parents

Robert E. and Zoe S. Tenney

June, 1979
NOTATION

Figures and tables are numbered in the same sequence to aid readers in finding a particular one. Similarly, definitions, propositions, conjectures, and examples are also numbered in their own single sequence.

For all variables, a subscript denotes the agent with which it is associated. Superscripts indicate a particular element in a set. Thus \( x_1^3 \) is the third \( x \) associated with agent one.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sets</strong></td>
<td></td>
</tr>
<tr>
<td>( I )</td>
<td>Intervals of reals</td>
</tr>
<tr>
<td>( L )</td>
<td>Links of a graph</td>
</tr>
<tr>
<td>( N )</td>
<td>Nodes of a graph</td>
</tr>
<tr>
<td>( R )</td>
<td>Reals</td>
</tr>
<tr>
<td>( S )</td>
<td>Simplex</td>
</tr>
<tr>
<td>( T )</td>
<td>Types of grain</td>
</tr>
<tr>
<td>( \Sigma )</td>
<td>State set</td>
</tr>
<tr>
<td>( P )</td>
<td>Interaction set</td>
</tr>
<tr>
<td>( U )</td>
<td>Control set</td>
</tr>
<tr>
<td>( Y )</td>
<td>Output set</td>
</tr>
<tr>
<td>( 2^X )</td>
<td>Power set of ( X )</td>
</tr>
<tr>
<td>( P(X) )</td>
<td>Set of probability distribution</td>
</tr>
<tr>
<td>( R )</td>
<td>Relation</td>
</tr>
</tbody>
</table>

**Variables**

<p>| ( B )  | Number of bins in an elevator |
| ( D )  | Domular decision element |
| ( s )  | Average steady-state loss per unit time |
| ( i, j, k, l, m, n ) | Indices |</p>
<table>
<thead>
<tr>
<th>Notation</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Numbers of modules in system</td>
</tr>
<tr>
<td>t</td>
<td>Time</td>
</tr>
<tr>
<td>T</td>
<td>Time limit</td>
</tr>
<tr>
<td>u</td>
<td>Control</td>
</tr>
<tr>
<td>y</td>
<td>Output</td>
</tr>
<tr>
<td>ρ</td>
<td>Interaction</td>
</tr>
<tr>
<td>σ</td>
<td>State</td>
</tr>
<tr>
<td>ε</td>
<td>Small positive real</td>
</tr>
</tbody>
</table>

**Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>Cost function</td>
</tr>
<tr>
<td>f</td>
<td>State transition function</td>
</tr>
<tr>
<td>g</td>
<td>State to interaction function</td>
</tr>
<tr>
<td>h</td>
<td>Output function</td>
</tr>
<tr>
<td>p</td>
<td>Probability distribution</td>
</tr>
<tr>
<td>s</td>
<td>Cost-to-produce or cost-to-respond function</td>
</tr>
<tr>
<td>v</td>
<td>Cost-to-go function</td>
</tr>
<tr>
<td>w</td>
<td>Cost-to-reach function</td>
</tr>
<tr>
<td>f(α; x)</td>
<td>Family of functions f of one variable x indexed by parameter α</td>
</tr>
<tr>
<td>a</td>
<td>Abstraction; aggregation function on state space</td>
</tr>
<tr>
<td>β</td>
<td>Specialization</td>
</tr>
<tr>
<td>γ</td>
<td>Control strategy</td>
</tr>
</tbody>
</table>

**Other**

<table>
<thead>
<tr>
<th>Other</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>Graph</td>
</tr>
<tr>
<td>H</td>
<td>Induction ordering</td>
</tr>
<tr>
<td>Notation</td>
<td>Interpretation</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>≥</td>
<td>Equals by definition</td>
</tr>
<tr>
<td>⇒</td>
<td>Implies</td>
</tr>
<tr>
<td>(, , )</td>
<td>Ordered set</td>
</tr>
<tr>
<td>{ }</td>
<td>Set</td>
</tr>
<tr>
<td>x', x''</td>
<td>Alternate values of x</td>
</tr>
<tr>
<td>x⁺</td>
<td>Value of x in future</td>
</tr>
<tr>
<td>x⁻</td>
<td>Value of x in past</td>
</tr>
<tr>
<td>x</td>
<td>Upper bound on x</td>
</tr>
<tr>
<td>x</td>
<td>Lower bound on x</td>
</tr>
<tr>
<td>fx</td>
<td>Sequence of x's</td>
</tr>
<tr>
<td>x̂</td>
<td>Abstracted version of x</td>
</tr>
</tbody>
</table>
Chapter 1. Introduction and Overview

With the development of data processing equipment of ever greater power, more and more attention is being paid to the tantalizing goal of automatically managing large, complex systems. A fascinating approach to this problem involves the use of multiple processors cooperating in their efforts to control a system. That such an approach is practical is strongly suggested by the fact that human organizations display such a distributed structure; each decision maker is concerned with its own specific tasks, yet communicates with others to help coordinate his activities with those of the others.

However, progress in the development of practical means to design a multiple processor, decentralized control system have been at best frustrating. These have generally been based on the methodologies which have evolved for the design of single processor, centralized control systems. That such progress is lacking suggests that there are fundamental differences between centralized and decentralized control problems and that these differences preclude a straightforward transfer of concepts from the centralized to the decentralized case.

The goal of this thesis is to explore an alternative approach to the problem of decentralized control which is based on the assumption that no agent has a complete model of the system available to it. Rather, each agent has knowledge of the operation of a particular subsystem - it is an expert on this subsystem. However, it has no knowledge of the structure of the system outside of this domain; assessment of the impact of its decisions on the rest of the system, and of other's activities on its special subsystem, must be gained through communication.

In the course of this exploration, the fundamental assumptions of centralized optimal control theory will be examined and modified in the
light of the above assumption. A set of concepts will be discussed which allow many of the intuitive properties of human organizations to be captured in a unified framework. This will lead to the formalization of a variety of decision strategies which can be used in the design of automatic, decentralized control algorithms.

It must be emphasized that this work is done at a very conceptual level. While it does lead to approaches which seem to be suitable for application in many settings, no claim is made that it is appropriate for all settings, nor even that it is superior to other approaches in those settings in which it can be used. As the work is done in a general framework, there is great promise that the concepts developed here can be specialized to take more advantage of properties exhibited by a particular problem; however, this specialization will be beyond the scope of this work and left as a major area of future effort.

This chapter will give some of the reasoning behind this particular approach to decentralized control, a precise problem formulation, and an overview of the major results presented in subsequent chapters. Chapter Two will describe prior efforts in this area, as well as the general guidelines that define the approach taken here. Chapters Three and Four present the major technical results; Chapter Five demonstrates the applicability of the problem formulation and control strategies to a particular system. Chapter Six will conclude the thesis with an outline of the most promising areas for future work.

1.1 Introduction

While it is inappropriate to exhaustively detail the efforts which have led to this approach (see[1] for a survey of some of them), valuable perspective can be gained by looking at them in a general light and
contrasting them with the proposed alternative. As most of the relevant work has been done by control scientists, it is best to begin there.

Figure 1.1 maps the general process for designing a control system as it appears in the most successful current methodologies. A problem is identified and defined as the first step. Then a model of the process to be controlled is built, and relevant performance criteria defined. In modern control theory, the fundamental characteristic of the model building process is the definition of its state. The state is defined intuitively as all the information needed to completely predict the future behavior of the system, and formally as the set of data such that any other data is irrelevant for determining the response of the system to future inputs. The idea is that the state captures all of the information about the past relevant to the system's behavior, and the model can be used to predict the evolution of the state through the future.

Once a model is built, it can be used to derive a control strategy which satisfies the desired performance requirements. While this derivation may involve optimization [2], model simplification, statistical analysis[3], etc., the exact means is not important. The major feature of this step is the output of a structure for the control system which works well for the model.

The final step, implementation, usually involves modifying this structure due to either implementation constraints or differences between the actual system and the model.

Previous approaches to the design of distributed control structures can be classified by their point of departure from this general process. Since such centralized design procedures, for centralized control structures, have been so successful, there has been a conscious attempt to make
Figure 1.1

Control System Design Methodologies

I. Identify Problem
   \[ \rightarrow \text{Build model in distributed framework} \]

II. Model Physical System
    Define Performance Criteria
       \[ \rightarrow \text{Decompose model; design strategy} \]
       \[ \rightarrow \text{for each piece} \]
       \[ \rightarrow \text{Add constraints to force decentralization} \]

III. Find Satisfactory Control
    Strategy
       \[ \rightarrow \text{Distribute strategy; implement} \]

IV. Implement Control System
this as late in the process as possible.

These approaches which depart at point A in Figure 1.1 work well in select cases when

1. The system is small enough that Steps II and III are practical.

2. The resulting strategy has a natural structure which can be exploited to facilitate distribution.

In these cases (which cannot generally be identified a priori), the satisfactory strategy itself is decomposed and implemented in a distributed control system. An excellent example of this process is Gallagher’s packet network routing strategy [4]. Unfortunately, most applications of this vein are a result of specific properties of the problem under study. The conclusion to [1] suggests there is potential for rectification of this disadvantage.

The second class of approaches (B) accept a model of the system from Step II, but then modify either the model, or the criteria, in order to encourage the emergence of a distributed solution structure. Among those that modify the model are time scale separation ([25-35] of [1]), interaction separation ([21-24] of [1]), chaining [5, 6], aggregation ([6-13] of [1]) and other application-specific techniques. These then use the modified structure, be it simple, decomposed, or hierarchical, to derive a control strategy that matches that structure.

Those approaches that modify the criteria either impose constraints on the information flow or computational structure to be used by the control system. Constraints on information flow seem to lead to impossible problems [7, 3] due to potential for signaling, second-guessing, non-linear coding, etc. The alternative, a priori specification of control system
structure, has been more successful in certain cases. The successes have been limited to linear systems [9] in which linear control laws are assumed, and periodically updated control structures [10] in which a centralized scheme to derive control laws is applied periodically to correct errors appearing in a decentralized scheme. In general, though, there are two defects in this type of approach:

1. No way of selecting the "best" structure is available.
2. Simply specifying a structure does not guarantee that the resulting parametric optimization problem is solvable in practice.

Other approaches exist which seek to decompose the problem by splitting both the model and objectives into subproblems, solving them, and using these solutions to approximate the overall solution. This in turn can lead to a modification of the decomposition, and the process iterated. This has often been applied in static problems [11] or open loop dynamic problems [12], but seems to bog down in stochastic feedback problems because the iterations require a huge amount of data (such as predicted state trajectories) to be transmitted between decision agents at each step.

It is the position of this thesis that the development of a theory of distributed decision making must occur by making an even more radical departure from the centralized methodology than these aforementioned approaches. This can only take place at Point C in Figure 1.1 - after a problem has been identified but before a model of the process has been built. The rationale for choosing this point of departure is expanded in the next chapter, but stems mainly from the failure of the other control theoretic approaches to produce a general theory of distributed
decision making.

If the alternative is to break away from the established procedure of starting with a single model of the system, it must take the approach of multimodelling - describing the system within a framework of decision agents, where each agent is an "expert" on some part of the system (i.e. knows its structure perfectly), is generally aware of other parts (i.e. knows their general structures), and knows how the others relate to his area (i.e. how the situation of the other systems within their general structure affects the part of the system of which he has a complete model). Each decision agent in this type of structure will be termed a domule.

The idea of multimodelling is not new. It arises from the model modification techniques mentioned earlier (particularly singular perturbations [13, 14], chaining [5, 6], and multilevel control [11, 12]) as a result of decomposing the single, complete, centralized model of Step II in the outline of Figure 1.1. What is new is the idea that the system model should be constructed directly within a framework of decision agents, without recourse to a centralized model first. The implications of this idea are dramatic. Some will be discussed in Section 2.2, but one important one is that the structure of the decision system will parallel the structure of the system to be controlled. This assumes, of course, that the system to be controlled indeed has a structure amenable to such a modelling approach.

Simon [15] and others [16] have often remarked that many large scale physical systems seem to have a natural, hierarchical structure to them.

1. From Latin dominus, "master", and -ule, "little". Also from English module.
That is, interactions at one level of system operation depend only on aggregate properties of the components. The most commonly cited example of this is the operation of the human body, in which the global health of the being depends on aggregate properties of the various organs, which in turn depend on aggregate properties of their component tissues, etc. As the ambitions of large scale theorists, the knowledge about these systems, and the amount of computing power available grow, so will the potential for application of a theory which takes direct advantage of these structures.

Thus the three principle arguments supporting the direction taken in this work are:

1. Past control theoretic technical approaches to decentralization based on a single, centralized state space model have failed to produce a generally applicable theory of distributed decision making which exhibits most of the intuitive characteristics desired of such a theory,

2. The alternative to a single, centralized state space model is a distributed model in which the dynamics of one element depend only on general characteristics of other models, and

3. There exist many large scale systems of interest which exhibit the structure required for the construction of a distributed model.

1.2 Problem Formulation

This section will make precise some of the ideas of the last section in a formal setting. It begins with a definition of the network topology,
and follows with a definition of a domule. A simple example of this formal structure is discussed in Section 3.1; a more detailed description of a physical system which fits it is deferred to Chapter 5.

The topology of a network is expressed by a graph $\mathbf{G}$ consisting of a finite set of nodes $\mathcal{N}$ and a set of links.

$$\mathbf{G} = (\mathcal{N}, \mathcal{L})$$

For convenience, let the nodes be numbered $1, 2, \ldots, |\mathcal{N}| = N$ in some unique way. The links connect one node to another unidirectionally

$$\mathcal{L} \subseteq \mathcal{N} \times \mathcal{N}$$

where $(i, j) \in \mathcal{L}$ indicates a link connecting node $i$ to node $j$. This graph will describe the underlying dynamic influence of $i$'s model on $j$'s; thus, the fact that $i$'s always affects $i$'s is implicit in $i$'s possession of its own model. Hence, $\mathbf{G}$ will be restricted to being non-reflexive:

$$(i, i) \notin \mathcal{L}$$

Each link represents not just dynamical interaction, but the corresponding interface in the decision making structure. Since neither is necessarily bidirectional, no assumption about the symmetry of $\mathbf{G}$ need be made.

Links represent the relationships of the domules at each node to one another, but their relationships to the system inputs and outputs, and to objectives, must be considered.

Inputs: Each input to the system must be determined by a specific domule (the one which models its direct effects), but that one may determine several others also. Other domules may model no inputs, and hence have no inputs to select.
Outputs: Each output of the system will similarly be associated with a specific domule which models its derivation from local variables.

Objectives: Certain domules will have embedded in them specific objectives related to their individual models. Others will have none - their function will be to mediate the actions of the specified domules so that the objectives are achieved (how this is done will depend on the philosophy of the designer and the specific problem).

Thus the topology of the domule network is specified by:

\[ N : \text{set of nodes} \]
\[ L : \text{links between nodes} \]

The domule \( D_n \) represented by the \( n^{th} \) node exists as a pair: the underlying, local model \( M_n \), and the corresponding decision agent \( A_n \).

\[ D_n = (M_n, A_n) \] (1.4)

To understand the role of the agent, the structure of the model must be defined.

In automata theory, a time invariant model of a system is a quintuple

\[ M = (U, Y, \Sigma, f, h) \]

where

\[ U = \text{set of admissible inputs} \]
\[ Y = \text{set of possible outputs} \]
\[ \Sigma = \text{set of internal states} \]
\[ f = \text{next-state function} \]
\[ h = \text{output function} \]
A modular model is an extension of this to allow the distributed structure to be made explicit.

Each local model is complete in the sense that it has a sort of Markov property. There will exist a state set \( \Sigma_j \), but the next-state map takes arguments other than \( \sigma_j \in \Sigma_j \) in order to reflect the influence that a part of the system modelled by another module has on the state transitions of \( D_j \). These interaction variables are drawn from sets \( P_{ij} \) which reflect the influence of the subsystem modelled by \( D_1 \) on that modelled by \( D_j \). They are derived from the state set of \( D_1 \) by aggregation functions:

\[
\varepsilon_{ij}: \Sigma \to P_{ij}
\]

(1.5)

specifying an aggregate state \( \sigma_{ij} \) for each state \( \sigma_i \). \( \varepsilon_{ij} \) will be constrained to be noninvertible; there must be some \( \sigma_1^1 \) and \( \sigma_1^2 \) such that

\[
\varepsilon_{ij}(\sigma_1^1) = \varepsilon_{ij}(\sigma_1^2) \quad \sigma_1^1 \neq \sigma_1^2
\]

(1.6)

for each \( i,j \). (Note that when one model does not influence another, that is, \( (i,j) \notin \mathcal{L} \), then \( P_{ij} \) contains only one element. It might seem more natural to define \( P_{ij} = \phi \) in this case, but formal considerations make a singleton set more appealing.)

Thus the state spaces of interest will be

- \( \Sigma_1 \) the state of \( D_1 \)'s detailed model,
- \( P_{ij} \) the aggregate version of \( \Sigma_1 \) presented to \( D_j \), and a singleton if \( (i,j) \notin \mathcal{L} \),

for each \( i, j \in \mathbb{N} \). Control and observation spaces are defined the same way:

- \( U_1 \) the set of controls from which \( D_1 \) may choose
- \( Y_1 \) the set of measurements which may appear at \( D_1 \)
Now the model possessed by $D_i$ can be defined. It is an octuple,

where

$\Sigma_i$ local state set

$\{P_{ij}\}$ aggregate state sets

$U_i$ input set

$Y_i$ output set

$f_i$ next state function

$h_i$ next output function

$\{g_{ij}\}$ aggregation functions

$c_i$ local cost function

and where

$f_i : \Sigma_i \times P_i \times \cdots \times P_N \rightarrow \Sigma_i \tag{1.7}$

$g_{ij} : \Sigma_i \rightarrow P_{ij} \tag{1.8}$

$h_i : \Sigma_i \times P_i \times \cdots \times P_N \rightarrow Y_i \tag{1.9}$

$c_i : \Sigma_i \times \Sigma_i \times P_i \times \cdots \times P_N \times U_i \rightarrow \mathbb{R} \tag{1.10}$

(1.7) expresses the constraint that state transitions depend only on local state and neighboring interaction variables, and (1.10) the same for outputs. Extension of this model to stochastic systems is discussed in Section 3.2.2.

(1.10) defines the local cost function $-c_i(\sigma_i, \sigma_1^+, \rho_{11}, \rho_{21}, \cdots, \rho_N, \rho_1)$ is the cost of moving from state $\sigma_i$ at some time $t$ to $\sigma_i^+$ at $t+1$ when interaction variables $\rho_1, \rho_2, \cdots, \rho_N$ are present and $u_1$ is applied.

An important feature of this formulation is that the notion of a centralized state has been replaced with the notion of several local states. No local state is enough to determine future local response to local inputs; future interactions from other domules are necessary for this. Thus since centralized control strategies require as much knowledge as possible about the value of the state variables, it is to be expected that local decision
making here will be based on gathering as much information as possible about the local state and future interactions.

For the sake of simplicity, it will be assumed that the observation functions \( h_i \) are one-to-one: each agent knows the state and interaction variables at each time \( t \) without ambiguity. This avoids complications introduced by the estimation problem and allows effort to be focussed on the coordination problem.

The relationship of the local cost functions \( c_i \) to an overall objective will not be specified yet. As will be shown in Chapter 4, this formulation allows consideration of problems where several organizations interact, each agent operating to optimize the performance of its organization without regard to its effect on others. Section 4.2 evaluates the effect of objective structure (relationship of the \( c_i \) to overall goals) on the suitability of various coordination mechanisms.

This, then, is the formal problem which underlies the sequel: given the agents and their associated, specific models, develop coordination strategies which allow their activities to be coordinated to achieve some system goal which is defined from the \( c_i \)'s.

1.3. Overview of Results

This section presents a synopsis of the results obtained in detail in succeeding chapters. Naturally at this point they can be discussed only in general and somewhat loose terms; details are left to the individual chapters.

Chapter 2 delves more deeply into the issues presented in this chapter. The first half reviews efforts in several areas which affected the course taken in this work. The principle influence of optimal control theory has already been discussed, but this work bears direct
relationships with past efforts in decision theory, the study of organizational decision making, and artificial intelligence. The latter is important as it is a field which essentially studies the same problem of decision making in a complex environment as motivates this work; there is a great potential for interplay between the quite different approaches embodied there and in this work.

The second half of Chapter 2 develops the philosophies which guide the development of the strategies for coordination which are presented later. The problem formulation of the preceding section is incomplete in many ways as only the structure of the models underlying the decision system is specified, not the structure of the decision system itself. As this involves a fair degree of arbitrariness, it is postponed until Chapter 2 when it can be presented in the light of past efforts. Of the several guidelines developed, two stand out as being of principle importance: communication between two agents is done only in terms of the interaction variables between them, and uncertainty about future effects of outside agents is resolved by assuming the worst case possibility will occur. The practical result of the first is to force the structure of the control system to parallel that of the system itself; the latter leads to decoupling of the control efforts as one agent is given some freedom to operate slightly suboptimally by another's assumption that it will take the worst possible actions. It also introduces a new role to communication: reduction of uncertainty as to how outside agent's actions will affect, or be affected by, those of a specific agent.

Chapter 3 develops some preliminary concepts related to the problem formulation of Section 1.2. First, a simple example of a two-module system is presented which will serve as a common illustration of many
points to follow. Then Section 3.2 explores some properties of the domular formulations: the fact that there is an equivalent centralized model for every domular model which assures that the problem is well posed, and the ability of the formulation to be extended to include stochastic effects.

The last two sections of Chapter 3 develop two important concepts which can be used as cornerstones for the construction of coordination strategies. The first is rooted in the philosophy borrowed from decision analysis and artificial intelligence of working forward in time in comparing decision options. The fundamental idea is that if two options lead to the same resulting state, and one is less costly than another, then that one should be selected. This leads to an algorithm which allows the locally optimal control to be selected by an agent with knowledge of only a finite future of interactions - and hence provides a measure of the need for interaction prediction. The second is a very promising one: constructing simplified models of a decision process which, while incomplete, provide some way of determining what actions a decision maker may take without simulating the complete problem it faces. This technique, dubbed abstraction, can be used in coordination strategies to give one agent some way to predict the activities of others as seen through interactions without complete knowledge of their local models.

Chapter 4 presents a series of coordination strategies for domular systems. The first half is restricted to consideration of a special class of system topologies, and a system wide goal of minimizing the sum of local costs. While such problems are fairly general in applicability, they are selected as each of six coordination strategies can be used on them. These six are developed in this first half; they include:
1. Static strategy: achieves global coordination by examining all possible futures, thus serving as a benchmark.

2. Silent coordination: assumes no communication, so each agent chooses locally optimal control assuming worst case interactions from others.

3. Selfish strategy: similar to silent, but allows each agent to receive and compute predictions of interaction sequences, thus reducing the uncertainty faced by each agent.

4. Finite look-ahead: planning is done over a fixed future horizon, and worst case behavior assumed beyond that.

5. Abstraction: the techniques developed in Section 3.4 can be used to supply each agent with simplified descriptions of the behavior of the system (and agents) outside of it.

6. Finite look-ahead and abstraction: the last two are combined in a hybrid scheme.

Section 4.1.3 includes a summary of each strategy, an example of its use in current human organizations, and a comparison of the performance and processing/communication requirements of each.

The second half of Chapter 4 studies the problems encountered when each of these strategies is applied in a more general system structure. Each strategy has a class of topologies for which it is suited; these are summarized in Section 4.2.5. Other interesting conclusions can be drawn for objective structures which involve individual agents acting in their own interests; or groups of agents, each group acting towards a common goal, but in conflict with one another. Each group can be coordinated...
as a team of Section 4.1, but it behooves each to acquire either ab-
stract models or predictions of future interactions from other groups.

Chapter 5 turns to a specific application of the domular problem
formulation and strategies derived in Chapter 4. Although many appli-
cations are suitable, the availability of relevant information led to the
choice of an integrated grain storage, transportation, and processing
company as an illustrative example. Models can be constructed for
domules representing grain suppliers, elevators, railroads, barges,
and markets. State variables represent the amount, type, and grade of
grain in storage or in transit at each facility; interaction variables
represent the flow of grain from one to another in a given time period.
Because of the structure of such a company, the strategies of Section 4.1
are almost directly applicable, and brief descriptions of the problems
and advantages of using each in such a context are given in the second
half of Chapter 5.

Finally, Chapter 6 concludes the work with a sketch of the future
paths to be explored leading from those followed here. There are many
of these, both in terms of specific goals and examination of ramifica-
tions of more conceptual issues.
Chapter 2. Background and Approach

This chapter will survey some of the work done in related areas which serves as a backdrop to the domule concept, then synthesize some general philosophical guidelines for the development of a domular theory of distributed decision making. Its tone will be that of a general discussion of issues; more formal details will be deferred to Chapters 3 and 4.

2.1 Background

The formulation of the domular concept has its roots in many fields. The most influential, systems theory and modern control, has been covered in Section 1.1 and will also provide the setting for Section 2.2. This section is devoted to three neighboring areas which supply the intuition and insight behind many of the issues that sprout from the domular system structure. These areas are

a. Decision theory and decision analysis

b. Organization theory, particularly as seen in management science

c. Artificial intelligence and problem solving

Each of these will be the topic of a particular subsection.

2.1.1 Decision Theory

It is interesting to note that the fundamental philosophy of the scientific method of decision making, dating back to Kant [17], is centralized: collect all relevent data in one place, formulate and evaluate possible alternatives, and select the best one. This long standing, almost inherent assumption of centralization suggests that the basic principles underlying the process of decision making should be examined as one moves toward distribution of that process among several agents.
In recent years, the extensions to decision theory have been proceeding along two distinct paths which relate to decentralization: the study of multiple decision makers with conflicting objectives (games) and ways to combine values held by individual elements of a decision organization into one value reflective of the whole.

The theory of games, particularly those involving two agents in a situation where one's gain is another's loss, has been well studied. One important conclusion is that optimal decisions may not exist unless one considers probabilistic decisions, where the actual decision is selected from those available according to an optimal probability distribution. It is to be expected that this problem manifests itself in a modular system if one agent's actions affects another's, and vice versa, and they are not acting cooperatively (see Section 4.2.3).

Also of interest is the formulation of organizational goals from those of the individuals within it. This, in turn, requires a thorough examination of ways to express those goals. The standard approach has been to derive some notion of utility to a decision maker [19, 20], and then maximize the sum of utilities over all elements of an organization. This is partially the approach taken in modular systems; the local cost functions can be interpreted as "negative utility" functions. As pointed out in [21], there are many ways in which one can formulate organizational utility from individual utility; each has advantages and disadvantages relative to a simple sum. It is this potential for variety in the overall utility pattern which led to the postponement of assumptions about it until Chapter 4.

Some of the efforts in this area begin to tie into those of artificial intelligence (Section 2.1.3). [22] discusses the problems
encountered when the set of possible alternatives is too large to be handled (centrally) and thus compromises must be made. One of the intriguing suggestions made there is that social organizations surmount this problem by incrementally (and iteratively) adjusting the decision strategy until a satisfactory solution is achieved. (If one assumes that this adjustment is made by elements changing their perception of the expected activity of other agents, an algorithm similar to that of Section 4.2.4 is suggested.) This leads directly to the consideration of behavior of existing human organizations as a source of insight into the process of making decisions in a complex environment.

2.1.2 Organization Theory

The study of how existing organizations actually go about making decisions has been spurred more by the goal of improving that process than by transferring it to an automatic control setting. Nonetheless, the insight gained from those efforts is quite valuable in aiding this latter task.

Chief among the analysts of organizational decision making is Herbert Simon [23-26]. Two of the major conclusions he arrived at are

1. Organizations are not optimizing: they satisfice.

That is, they attempt to improve their utility only to a certain level, and then turn their efforts to other things.

2. Organizational structure has a significant impact on the decision process. By boxing a manager into his own area of responsibility, alternatives which he might consider do not occur to him as they are outside his scope of work. Thus this avoids the
complexity of too many alternatives.

Other work has been done on this latter point by observing humans, placed in a particular organizational form, attack specific, given tasks [27]. As mentioned in the last section, the organizational decision process can be viewed as that of a series of incremental changes in individual strategies which eventually reach an equilibrium [28].

Of prime interest here is the form in which such organizations as businesses, governments, military services, etc., take. Figure 2.1 illustrates this canonical structure. The head of the organization is supported by two branches, divided along functional lines. The staff function is responsible for environmental assessment: acquiring information, processing it to generate alternative courses of action, etc. This information is modified, combined, and evaluated as it proceeds up the organizational hierarchy. At the top, the head decides on the best course of action to follow, then instructs line management to implement it. As these instructions are passed down through line management, they are successively expanded based on each manager’s abilities.

In many organizations, though, each decision maker in it implements both staff and line functions [29]. For example, in a military command and control system, each officer is responsible for generating situation assessments for his superior, and relaying commands to his subordinates. This provides a "short circuit" for information indicated by dashed lines in Figure 2.1: each decision maker uses the information it gathered while performing the staff function to aid in the propagation of commands from its superior to its subordinates while performing the line function.

Figure 2.2 shows how this process fits into a modular framework. If it is assumed that interactions flow from subordinate to superior
Figure 2.1
Information Flow in a Typical Organization

Staff function  |  Line function
Figure 2.2

Equivalent Domular Structure
(i.e. actions taken by a subordinate affect not only his own situation, but also the more broadly scoped situation faced by the superior) then the management process consists of reports passed upward concerning available options and their associated costs (or profits generated, or utilities), and instructions passed downward as to what to do next. This is precisely the concept behind the coordination strategies of Sections 4.1.2 and 4.1.5.

Much more can be said about the relevance of organization theory to modular decision systems, but most of it is rather intuitive in nature. Instead, the next section will discuss a completely different source of insight into distributed decision systems.

2.1.3 Artificial Intelligence

While not focussing on decentralization as a central issue, the field of artificial intelligence is nonetheless relevant as it, in part, consists of efforts to design ways to solve problems in a complex environment. Much of these have been rather ad hoc, but still give insight into ways of dealing with a problem when the set of possible alternatives is large enough to prevent exhaustive enumeration. This section will discuss two ideas in the field: the first is the generic approach to problem solving which involves searching through a tree of alternatives, ignoring branches that are likely to be unsatisfactory, and the second is the knowledge organization structures, and attendant planning strategies, that comprise frame theory.

The General Problem Solving methodology of Newell and Simon (in[30]) begins to extend the approach of decision theory to more complex settings by structuring a problem as involving a set of available transformations (decisions), a starting state, and a goal state(s). A decision tree can
be constructed by considering all possible sequences of transformations, and a solution obtained by simulating the effect of those transformations on the external world and selecting one sequence which leads to the goal. However, this usually leads to a tremendous number of possible sequences, many of which are "obviously" unsuitable. Hence they augment this scheme with static evaluators - functions which assign to each possible intermediate node a value which reflects, in some heuristic manner, its "nearness" to the goal. Derivation of these functions is often extremely difficult, but given one, it can be used to guide the search for an appropriate sequence of transformations.

This works as follows: Assuming the evaluator provides a reasonable assessment of the possibility (expressed in, say, minimum worst case cost) of reaching a goal from an intermediate state, one can

1. Find the branch in the decision tree terminating with the state which has the best evaluation.
2. Examine all alternatives which can be taken when in this state by extending the tree, finding the state resulting from it, and evaluating it.
3. Repeat until a path to the goal is found.

In Figure 2.3, two intermediate states have been expanded in this way. The next to be examined will be the one resulting from $A_2$ which evaluates to 5 - the alternatives available subsequent to the originally very promising $A_1$ all lead to less desirable states.

This process, when combined with the notions of state and transition cost functionals, can lead to an algorithm for decision making which serves as an alternative to dynamic programming (Section 3.3) and in which the static evaluator can be found algorithmically assuming a worst case future (Section 4.1.3).
Figure 2.3

Decision Tree with Static Evaluators

A_{i j k} Alternative
\times State which evaluates to x
The other, more exciting, concept arising from artificial intelligence is that of a frame [31] as an organizational unit of knowledge specific to a particular part or context of the external world. It is, in essence, a model of a subunit of the world, linked to other frames which represent parts of the world which can be reached from it. Thus a frame system is very similar to a domular network: each is comprised of local models of the world, with local information on the effect of actions, and these models can be affected, and affect, "neighboring" units. While this point will not be pursued further here, it can serve as the starting point for fascinating future work which bridges the gap between optimal control and artificial intelligence through something similar to the domular structure.

2.2 **Philosophy of Approach**

Having established the point of departure from conventional approaches to decentralized decision making in Section 1.1, it is worthwhile to discuss the general reasons behind their inadequacy before formulating the alternative. These, combined with the features desired of a distributed decision making theory, lead to the guidelines which were used in the formal definition of a domular structure of Section 1.2, and will shape the development of coordination strategies in Chapter 4.

The four major concepts which play together to produce most of the interesting results of centralized control theory are:

1. The Markovian notion of state [32].
2. Baye’s law of probability theory.
3. The use of a global, scalar performance index.
4. Dynamic programming [33].

Since specification of the state of a Markov process is sufficient (and
necessary in a minimal realization) to determine the response of the system in the future to the inputs, it is only necessary for a decision maker to track the state at a particular time. Any information from observations which affects this state estimation must be used in order to achieve optimal performance. With the state in hand, dynamic programming allows the controller to compute the input strategy which optimizes the future value of the performance index. It is important to note that the dynamic programming allows the current input to be chosen in anticipation of future inputs which complement it in order to achieve the optimum.

When all of the above four concepts are kept, but multiple decision makers are used, problems develop. Each agent can still compute his conditional probabilities on the process state, but the dynamic programming requires knowledge of, or at least conditional probabilities for, inputs supplied by other agents. This, in turn, leads to the requirement that the agent possess models for the other agents (which will have memory at least in the form of conditional probabilities, and hence a state space), that the other agents possess models of the one (and models of its models of them), etc. [34]. The problem of computing the optimal decentralized control strategy in this framework becomes intractible.

The crux of this problem seems to be the centralized concept of state. If every decision agent knows the state space, and the dynamics which describe state transitions, then the dynamic programming will force it to try to identify and account for everything that affects those dynamics (and particularly other inputs). This must be prevented if practical strategies are to emerge.

One way to prevent this single minded pursuit of all knowledge affecting the state is to give each agent only a partial model of the
state space and associated dynamics. Then, even if it had additional informa-
tion, it could not use it. Hence, it would not try to second guess the other agents, nor attempt to use the system as a communication channel [35]. However, the models employed by all agents must in some way describe the entire system, and each agent must be made aware that other parts of the system exist and affect the portion of it which has been assigned to him.

This conclusion can also be reached from another line of reasoning. The desire to employ distributed decision making systems stems from several sources, and one of the major ones is that decision mechanisms of sufficient complexity to handle many problems simply are not available. For example, bureaucracies grow as one person is unable to cope with all the information gathering, planning, and doing required to handle some situation. Since this is the case, one way to approach the design of these systems is to postulate, from the start, that only limited complexity decision elements are available. While complexity may have several ways of being defined, one way to measure the complexity of a decision element is by the size of the model which it is capable of using in its de-
liberations.

Thus, by two routes, the first principle of this research is sug-
gested:

Principle 2.1: Each decision element possesses a limited model of the system under control.

Now consider two such elements (modules) such that the actions of one (A) directly affect the dynamics of the other (B). Clearly there is a need to communicate between the two. B needs to advise A of the actions he has taken so their effects can be accounted for; A needs to inform B
of his objectives so B can plan actions to help A achieve them. This leads to a second principle.

Principle 2.2: Domules must be able to communicate when the behavior of one directly affects the behavior of the other.

In order to avoid "tricky" strategies where an infinite bandwidth channel is used to communicate everything to a central site, which then computes the usual centralized strategy, communication constraints of some sort must be imposed. Experience has shown that the inclusion of communication costs or constraints is extremely difficult to do analytically in a way which permits reasonable solutions to emerge. Hence the approach taken in this work will be to justify and adhere to certain principles regarding communication which are expressed in a general form.

The fact that two domules have almost completely separate models limits the set of things which they might communicate. All that two interacting agents share is the set of interaction variables produced by one which affect the other. Thus this is the only common context which they share as a basis for communication; it leads to

Principle 2.3: Interdomular communication will take place only in terms of quantities directly related to the underlying interaction variables. This may include functions on the space of such variables, or sequences of variables, values of them, or even simplified models defined in terms of them. It is meant to exclude communication of one agent's model to another, as the model involves quantities other than the interaction variables.

A common approach to distributing the processing load between decision agents is to use iteration between them. While often effective,
this usually requires tremendous communication capacity as several iterations have to be made at each time step in order to determine a set of current control inputs. For this reason, the strategies to follow adhere to:

Principle 2.4: Iterative techniques which involve communication between agents at each step of the iteration will be avoided.

Finally, the very structure of the problem formulation ensures that agents will often be ignorant of many things going on which might eventually affect them. Since a probabilistic approach to dealing with this uncertainty inevitably leads to a requirement for centralized or iterative processing, it is preferable to resolve the uncertainty by assuming a worst case. This has the appealing advantage of granting local autonomy to decision makers: communication serves as a vehicle for generating agreements between two agents restricting what each will do. However, each can be free to choose one of several alternatives within the agreed-upon restrictions, knowing that the other will prepare for the worst case choice possible.

Principle 2.5: Uncertainty about an agent's future actions will either be resolved through communication or assumed to take on the worst-case values.

These are the five general principles guiding the approach to distributed decision making presented here. It is useful to examine some of the consequences of these principles before proceeding further. Foremost is the fact that, since each agent will use only his piece of the model for making decisions, the structure of the module network will mirror that of the system under control. This has two nice properties: the
decision making structure is easy to relate to that of the system, facilitating testing and verification of any implementation of it, and a change in one part of the system can be made, and only the domules modelling that part need be updated. This latter property is even more appealing when combined with another consequence of the above principles: there can be no global optimization. This is, there can be no way the strategies of one domule can depend on the model of another except through communicated quantities. Thus each must compute his decision based on

1. his local model
2. real time observations
3. real time communication in terms of the interface.

Thus, if the local model in one domule is replaced with another, corresponding to modification of the underlying subsystem, no other domule need be affected. The change in other agent's strategies dictated by the new model must come about as a result of the new things communicated, in real time, to the neighbors of the domule (and then their neighbors, etc.). Thus models can be altered as the corresponding system components age or are replaced; and the domule system will adapt itself to the change. Thus this approach to system control using a distributed model may provide some of the robustness which has been suggested is a natural result of decentralization. It undoubtedly comes at a cost of increased total computation, but the decomposition of the computation into many small units helps ameliorate the additional burden.

Summarizing these rather general comments about the domular approach to distributed decision making, six principles are suggested as
guides to the formulation and use of the approach along with an intuitive justification of them. They are:

1. Limited complexity (model) decision elements are postulated,

2. Direct communication must occur only when the behavior of the part of the system modelled by one influences the behaviour of that of the other,

3. That communication involves only quantities directly related to the underlying interaction variables,

4. Iteration is to be avoided.

5. Uncertainty is resolved through communication or a worst-case assumption,
Chapter 3. Preliminaries

Before delving into the various approaches to system coordination which are possible, some technical groundwork must be laid on which the coordination strategies can be built. This chapter concentrates on the development of four independent areas which will reappear in a unified framework in Chapter 4.

First, a simple example problem will be detailed. This will serve two purposes: it will illustrate the problem formulation as introduced in Section 1.2, and it will provide a common example for purposes of comparison of various aspects of the subsequent techniques and strategies. This permits this comparison to be done in a way such that various relationships can be pointed out with a minimum of extraneous detail.

Secondly, some of the basic properties of the formulation of Section 1.2 will be discussed. Principle among these are the equivalence of the distributed model to a classical centralized model, which allows technical issues such as well-posedness of the system model to be handled easily, and a way of placing stochastic problems in the same framework. This latter result is not as simple as might be expected and is a potential source of difficulty in application of this work to stochastic systems. However, the strategy discussions of Chapter 4 will include comments on extension of the deterministic approaches to the stochastic case.

In another vein, the third section deals with solution of a local decision problem given its current state and a predicted interaction sequence. This reduces to solving a time varying Markov decision problem, a standard setting for the use of dynamic programming. However, as one would like to limit the requirements for prediction of interactions to
a finite horizon, a solution to such decision problems will be given which
works forward in time - allowing for the current decision to be made with
knowledge of the model for only a finite future; a decision which is sub-
optimal with vanishingly small probability as the horizon increases.
This will naturally be used to determine local controls to be applied
to the system given a coordination strategy has provided the necessary
predictions.

Finally, a section will be devoted to the study of deriving simplified
models of a decision maker's behavior as seen by another agent affecting,
or affected by, it. This process of abstraction will be useful in allowing
an agent to determine some of the effects of a decision it makes on
the rest of the system without having a full model of that part of the
system in all its complexity.

All of the above issues are important as they support the strategies
to be discussed later. Hence this chapter serves to present these pre-
liminaries as independent concerns; the next will tie them together in
the form of a distributed decision making system.

3.1 Example Problem

This section presents a simple, two module distributed model for
use later in illustration of essential concepts. It is not meant to be
meaningful in any applications context; it is merely to serve as an
illustrative "toy" problem. As future sections develop these concepts,
some of the structure of, and reasons for choosing, this particular
example will become clear.

The example system consists of two subsystems. That modelled in
module $D_2$ generates interactions which affect that part modelled in $D_1$;
see Figure 3.1 for this simple topology. Each submodel is deterministic, has four distinct states, two distinct controls, and its own function of states describing the local costs.

The affecting module, D2, is the most simple to describe. Its dynamics are illustrated in Figure 3.2 - control \( u_2^1 \) cycles through the states, while \( u_2^2 \) either holds state 1 or 4 or alternates between 2 and 3. The local cost function, \( c_2^\sigma (\sigma_2, \sigma_2^+, u_2) \) is independent of \( \sigma_2^+ \) and \( u_2^1 \); it reduces simply to a function of the state resulting from application of \( u_2 \) when in \( \sigma_2 \). As shown in Figure 3.2, it is zero for all resulting states except 3, for which it is unity. The aggregation function \( \phi_{22}^\sigma (\sigma_2) \), maps states 1 and 2 into interaction variable \( \phi_{21}^1 \); states 3 and 4 go to \( \phi_{21}^2 \).

Three important properties of these dynamics for D2 are:

1. It can achieve a steady state, zero local cost by trapping in states 1 or 4; thus this defines a purely local optimal control strategy (see Section 4.3).

2. Not all interaction sequences can be generated when D2 must start from any given state. For example, if started in state 1 (producing interaction variable 1), there is no way to generate a 2 as the next interaction.

3. There is no way to drive D2 along a state sequence that is cyclic with period 3. (This will be used to show that D2 cannot generate interactions which are the best from D1's point of view.)
Figure 3.1

Example Topology
Figure 3.2

Dynamics of $D_2$
Of these, the second is most important - it shows how the dynamics of D₂ impose constraints on the space of possible interaction sequences. The first points out that even if a sequence is possible, it may require non-zero expense to produce.

Moving on to D₁, the state transition function $f_1(\cdot,\cdot,\cdot)$ must be defined for each state when either of the two interaction variables is present, and for application of either of the two controls. The transitions for each of these four cases is shown in Figure 3.3, along with the local cost function which is again only dependent on the reached state.

While a bit complicated, this model for D₁ has several interesting aspects:

1. The locally optimal behavior is to stay in low cost states 1 and 2. It is not possible to do this; hence the average steady state cost of the system is greater than zero.

2. The locally optimal sequence of states is 1, 2, 3, l, 2, 3, ... resulting in an average cost of $1/3$ per time step. This requires an interaction sequence 1, l, 2, l, 1, 2, ... which is cyclic with period 3. By point (3) above, D₂ cannot produce any such sequence, hence the addition of constraints on feasible interaction sequences imposed by D₂'s dynamics further limits the optimal achievable cost.

3. Recall that one of D₂'s locally optimal strategies is to trap in state 1, producing interaction sequence 1, l, 1, l, ... This sequence causes D₁ to trap
Figure 3.3

Dynamics of $D_1$

Interaction

Control
in the set of states \( \{3,4\} \) - precisely those which are most expensive. Hence a locally optimal control strategy for \( D_2 \) forces \( D_1 \) into a very expensive mode of operation.

Comments (2) and (3) above explain the need for coordination in this example - the locally optimal strategy for \( D_1 \) (\( D_2 \)) requires (produces) unacceptable behavior on the part of the other. \( D_1 \) needs information on the cost and feasibility of producing sequences of \( \rho_{Z_1} \)'s from \( D_2 \); \( D_2 \) needs to know how effectively \( D_1 \) can respond to a proposed interaction sequence. This theme of coordination is the basis of Chapter 4.

For comparison's sake, it is possible to find the optimal strategy to control this system. Using the centralized equivalent model (Section 3.2), the optimal state cycle for \( D_2 \) is

\[
(1, 2, 3, 4, 4, 4, 4, \ldots )
\]

(3.1) costing an average of 0, locally, and producing interactions

\[
(1, 1, 2, 2, 2, 2, 2, \ldots )
\]

(3.2) \( D_1 \) can respond with state sequence

\[
(1, 2, 3, 1, 3, 1, 3, 1, \ldots )
\]

(3.3) leading to an average steady state cost of 1/2. Note the period of the cycles is 2, as \( D_2 \) and \( D_1 \) combine for a total, average steady state cost of 1/2 per unit time.

To summarize, this section presents an example of the model structure proposed in Section 1.2 which will serve as a common base for illustration. The known optimal performance will serve as a lower bound to judge the effectiveness of coordination strategies proposed in Chapter 4.
3.2 Properties of the Problem

This section will present two properties of the formulation of Section 1.2 of interest. The first shows that it is equivalent to a centralized model with state, control (and output) sets being the Cartesian produce of individual state, control (and output) sets. The centralized state transition function is a straightforward construction of local $f_i$'s and aggregation functions $g_{ij}$. The second shows the relationship between the deterministic setting and its stochastic extension; a caveat on the idea of predicting interaction sequences will be pointed out. The purpose of this section is to develop some of the properties of the domular structure before delving into the actual decision making mechanisms.

3.2.1 Centralized Equivalence

This section shows how to construct a centralized version of the distributed model used in domular systems. With this, it is easy to see that there are no inherent inconsistencies or ill-posed aspects of the formulation. However, the sizes of the various sets involved are much larger, in general, than the sets used to define the local models. Thus, while the centralized equivalent will be discussed here for the sake of completeness, all subsequent sections will be strictly concerned with the distributed form of the system model.

Table 3.4 summarizes the analogies between the local models and the centralized equivalent. To show that the equivalent is indeed a well defined model, the quantity

$$ (s_i, \ldots, s_n) \triangleq s $$

must be shown to possess the properties of a state. These are basically that
Table 3.4
Local and Centralized Models

<table>
<thead>
<tr>
<th></th>
<th>Local to D₁</th>
<th>Centralized</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Set</td>
<td>$\Sigma_1$</td>
<td>$\Sigma_1 \times \ldots \times \Sigma_N$</td>
</tr>
<tr>
<td>Control Set</td>
<td>$U_1$</td>
<td>$U_1 \times \ldots \times U_N$</td>
</tr>
<tr>
<td>Observation Set</td>
<td>$Y_1$</td>
<td>$Y_1 \times \ldots \times Y_N$</td>
</tr>
<tr>
<td>Interactions:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incoming</td>
<td>$P_{j1}$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>Outgoing</td>
<td>$P_{i1}$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>State Transition</td>
<td>$f_1$</td>
<td>$(f_1 \circ \tilde{\varepsilon}<em>{11}, \ldots, \tilde{\varepsilon}</em>{1N})$, \ldots, $f_N \circ \tilde{\varepsilon}_{NN})$</td>
</tr>
<tr>
<td>Observation Function</td>
<td>$h_1$</td>
<td>$(h_1, \ldots, h_N)$</td>
</tr>
<tr>
<td>Aggregation Function</td>
<td>$g_{jj}$</td>
<td>$\ldots$</td>
</tr>
</tbody>
</table>
1. $\sigma(t)$ can be uniquely determined given $\sigma(t)$ and $u(t)$

2. $\gamma(t)$ can be uniquely determined given $\sigma(t)$

Consider the $i^{th}$ component of $\sigma(t)$, $\sigma_i(t)$. This is the local state of Di; it is determined by

$$\sigma_i(t) = f_i(\sigma(t), \rho_{\mu_i}(t), \ldots, \rho_{\mu_{i+1}}(t), u_i(t))$$

(3.5)

or equivalently

$$\sigma_i(t) = f_i(\sigma(t), g_{i1}(\sigma(1)), \ldots, g_{iN}(\sigma(N)), u_i(t))$$

(3.6)

Since $\sigma(t), \sigma_i(t), \ldots, \sigma_N(t)$ are components of $\sigma(t)$, and $u_i(t)$ is the $i^{th}$ component of $u(t)$, $\sigma_i(t)$ can be found uniquely from $\sigma(t)$ and $u(t)$. Since this is true for all $i$, all of $\sigma(t)$ can be found thusly.

Similarly, the $i^{th}$ component of the output $\gamma_i(t)$ is given by

$$\gamma_i(t) = h_i(\sigma(t), \rho_{\mu_i}(t), \ldots, \rho_{\mu_{a+N}}(t))$$

(3.7)

or

$$\gamma_i(t) = h_i(\sigma(t), g_{i1}(\sigma(1)), \ldots, g_{iN}(\sigma(N)))$$

(3.8)

Thus each component of $\gamma(t)$ can be determined from $\sigma(t)$ using compositions of $h_i(t)$ and the $g_{ij}(\cdot)$. This demonstrated

**Proposition 3.1**

For each domular model, there exists a corresponding centralized model with identical behavior, and the correspondence is given in Table 3.4.

Note this is true regardless of the detailed structure of the sets $\Sigma$, $\Omega_i$, $\Omega_i$, and $R_j$, and for all system topologies.

**Example 3.2**

The centralized version of the standard example of Section 3.1 is shown in Table 3.5. The state transition function is given as the control
Table 3.5
Equivalent Centralized Model
For Example

<table>
<thead>
<tr>
<th>State</th>
<th>To</th>
<th>Occupancy cost (additive)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>(1,2)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(1,3)</td>
<td>4 3 2 1</td>
<td>0</td>
</tr>
<tr>
<td>(1,4)</td>
<td>3 4 1 2</td>
<td>0</td>
</tr>
<tr>
<td>(2,1)</td>
<td>2 1 4 3</td>
<td>0</td>
</tr>
<tr>
<td>(2,2)</td>
<td>1 3</td>
<td>0</td>
</tr>
<tr>
<td>(2,3)</td>
<td>4 3 2 1</td>
<td>1</td>
</tr>
<tr>
<td>(2,4)</td>
<td>3 4 1 2</td>
<td>0</td>
</tr>
<tr>
<td>(3,1)</td>
<td>2 1</td>
<td>1</td>
</tr>
<tr>
<td>(3,2)</td>
<td>4 3</td>
<td>1</td>
</tr>
<tr>
<td>(3,3)</td>
<td>2 1</td>
<td>2</td>
</tr>
<tr>
<td>(3,4)</td>
<td>3 4</td>
<td>1</td>
</tr>
<tr>
<td>(4,1)</td>
<td>4 3 2 1</td>
<td>1</td>
</tr>
<tr>
<td>(4,2)</td>
<td>2 1 4 3</td>
<td>2</td>
</tr>
<tr>
<td>(4,3)</td>
<td>1 2 3 4</td>
<td>1</td>
</tr>
<tr>
<td>(4,4)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A. Dynamics

<table>
<thead>
<tr>
<th>Centralized Control</th>
<th>Equivalent Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1,1)</td>
</tr>
<tr>
<td>2</td>
<td>(1,2)</td>
</tr>
<tr>
<td>3</td>
<td>(2,1)</td>
</tr>
<tr>
<td>4</td>
<td>(2,2)</td>
</tr>
</tbody>
</table>

B. Controls
to be used to transfer from \((\sigma(t), \sigma_2(t-1))\) to \((\sigma(t), \sigma_2(t))\); no entry indicates the transfer is not possible in one step. Multiple controls effecting the same transfer are omitted; only the lowest one's index is shown. Part B shows the equivalence between centralized control indices and pairs of local controls. The observation function is trivial due to the assumption that each local model observes its state perfectly; observation of the interaction variable by \(D_1\) provides redundant information (on \(\sigma_2\)) and thus is not a factor.

Finally, if the total system cost is the sum of the local costs, state occupancy costs as shown can be computed using

\[
C(\sigma(t)) = C_1(\sigma_1(t)) + C_2(\sigma_2(t))
\]

(3.9)

The principle point to be made about the centralized version of a domular system, aside from the fact that it exists, is that the size of the sets involved in it is the product of the sizes of the corresponding local sets. As seen in the example, even a small domular system with small local models transforms to a centralized model much more complex than any component. It is not hard to imagine systems composed of a few dozen domules, each with a few dozen internal states, that transform to completely intractable centralized equivalents. The advantage of the domular formulation is that it exploits the system structure to avoid the complexity that the centralized approach requires; but this assumes that the structure exists. Also, it is not as clear how to use the distributed model as it is how to use the centralized one — but that is the subject of Chapter 4.

3.2.2. Stochastic Equivalence

While the precise formulation of Section 1.2 is cast in a deterministic framework, it is of course extremely desirable to include probabilistic
effects in domular models. Many stochastic control problems can be reduced to deterministic equivalents by selecting appropriate state spaces (such as those of probability functions on the underlying state space), and indeed a similar approach can be taken with domules (although care must be taken). This section will discuss the relationship of stochastic domular systems to the formulation of Section 1.2 in some depth, and other comments about extending results for deterministic systems to stochastics will be interspersed where appropriate, but the main emphasis elsewhere is on the deterministic case. This is for two reasons: first, stochastic problems are similar in structure to deterministic ones, and hence a stochastic framework can obscure some points with needless complexity. Secondly, one of the major tools to be used to handle limited communication problems will be that of uncertainty as to other agent's future actions, and this introduces into the deterministic framework many of the issues (such as information) normally associated only with stochastic situations.

As in the original formulation, concern here is limited to the case of perfect state (and interaction) information. Hence stochastics will enter only in state transition maps, not measurements. Extension of the formulation to this situation is easy, but the usual approach to using the resulting models for prediction, by propagating state probability densities, must be handled with care.

**DEFINITION 3.3:** A stochastic domular system if one which can be modelled in a distributed fashion:

a) Globally: in exactly the same way as a deterministic system

b) Locally: each local model $M_i$ consists of:
State space $\Sigma_i$
Control set $U_i$
Observation set $Y_i$
State transition probabilities
$$p(\sigma^+ | \sigma, \rho_{ii}, ..., \rho_{ni}, u_i)$$
Observation probabilities
$$p(y_i | \sigma, \rho_{ii}, ..., \rho_{ni})$$
Aggregation probabilities
$$p(\rho; j | \sigma)$$
State transition costs
$$c_i(\sigma, \sigma^+, \rho_{ii}, ..., \rho_{ni}, u_i)$$

This is just the extension of the deterministic form, with $f_i$, $g_j$, and $h_i$ replaced by appropriate conditional densities. Note that the state transition costs can be reduced to costs of using $u_i$ given state $\sigma_i$ and interactions $\rho_{ij}$ as
$$\sum_{\sigma_i^+} p(\sigma^+ | \sigma, \rho_{ii}, ..., \rho_{ni}, u_i) c_i(\sigma, \sigma^+, \rho_{ii}, ..., u_i)$$
(3.10)

As in the deterministic case, the sequel will assume that all sets are discrete and finite. Further, to reduce conceptual complexity, it will be assumed that

1. Observations of states and interactions are perfect:
$$p(y_i | \sigma, \rho_{ii}, ..., \rho_{ni}, u_i) \in \{0, 1\}$$
(3.11)

2. The aggregation probabilities are pure:
$$p(\rho; j | \sigma) \in \{0, 1\}$$
(3.12)

Both assumptions allow the structure of the stochastic problem to be preserved without adding complications arising from having to condition all computations on all past observations (the condition on aggregation
probabilities is required as interactions are essentially observations of external states).

It is worth noting that the approach of the previous section can be applied in this case to yield a stochastic centralized model equivalent to the distributed one.

In systems theory, one can often place a stochastic problem in a higher dimensional, deterministic framework. This allows many general results proved for deterministic systems to be extended directly to the stochastic case. Table 3.6 shows the correspondences used; generally a set $S$ is lifted to the set of probability measures on $S$, $P(S)$. For stochastic systems, $p(\sigma(t))$ is a state because it allows $p(\sigma(t+\tau))$ and $p(y(t+\tau))$ to be uniquely determined for all $\tau \geq 0$ using

$$p(\sigma(t)) = \sum_{\sigma(t)} p(\sigma(t)|\sigma(t-1); u(t-1)) p(\sigma(t-1))$$

(3.13)

$$p(y(t)) = \sum_{\sigma(t)} p(y(t)|\sigma(t)) p(\sigma(t))$$

(3.14)

It is tempting to take the same approach to stochastic domular systems:

**CONJECTURE 3.4.** The model of definition

3.3 can be regarded as a deterministic system with local state sets $P(\Sigma)$, observation sets $P(\chi)$, control sets $U$; and interaction sets $P(\rho_i)$. That is, given $p(\sigma(0)), u(\tau)$, and $p(\rho_i(\tau))$ for all $i = 1, \ldots, N$ and $\tau \geq 0$, one can determine $p(\tau(t))$ and $p(y(t))$ uniquely.

This conjecture is false.
Table 3.6

Relation of Deterministic to Stochastic Systems

<table>
<thead>
<tr>
<th>Centralized</th>
<th>Deterministic</th>
<th>Stochastic</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Space</td>
<td>Σ</td>
<td>P(Σ')</td>
</tr>
<tr>
<td>Control Space</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>Outputs</td>
<td>Y</td>
<td>P(Y)</td>
</tr>
<tr>
<td>State Transition</td>
<td>f(σ,u)</td>
<td>p(σ'</td>
</tr>
<tr>
<td>Observation</td>
<td>h(σ)</td>
<td>p(y</td>
</tr>
</tbody>
</table>
PROOF: For any $t \geq 0$, $t^+ = t + 1$,

$$p(\sigma_i(t^+)) = \sum_{\sigma_i(t)} p(\sigma_i(t^+)|\sigma_i(t), \rho_{i1}(t), ..., \rho_{iN}(t))$$

$$= \sum_{\sigma_i(t)} p(\sigma_i(t^+)|\sigma_i(t), \rho_{i1}(t), ..., \rho_{iN}(t))$$

only if $\sigma_i(t)$ and the $\rho_{ji}(t)$ are independent.

However, $\rho_{ji}(t)$ is in general dependent on $\rho_{ji}(t-1)$ through the dynamics of $D_j$, and $\sigma_i(t)$ is dependent on $\rho_{ji}(t-1)$ through

$$p(\sigma_i(t)|\sigma_i(t-1), \rho_{i1}(t-1), ..., \rho_{iN}(t-1))$$

so $\sigma_i(t)$ and $\rho_{ji}(t)$ are not independent.

Hence the conjecture is false.

EXAMPLE 3.5. This exemplifies the interdependence of $\sigma_i(t)$ and $\rho_{ji}(t)$, providing a counterexample to the proposition that they are independent. Figure 3.7 shows a simple two domain system, with each element having only one control and two states. $D_2$ oscillates between its two states; each state generates an interaction driving $D_1$ to a corresponding state. Thus $D_1$ lags $D_2$ by one time step; hence either

1. $D_1$ is in 1 seeing 2 from $D_2$
2. $D_1$ is in 2 seeing 1 from $D_2$

If $D_2$ starts with $p(\sigma_i(t)) = \frac{1}{2}$, then
Figure 3.7

Interdependence of $\sigma_i$ and $\rho_{2i}$.
\[
p(\sigma_i, \rho_{ji}) = \begin{cases} 
\gamma_i \sigma_i, \rho_j^t \lor \sigma_i, \rho_j^t \\
0 & \text{else}
\end{cases}
\]

(3.18)

which does not decompose into independent distributions on $\Sigma_i$ and $\tilde{P}_{j1}$.

Thus the difficulty in using $p(\pi)$ as a state for stochastic $D_1$
is that it becomes dependent with the interaction "replacements" $p(\rho_{ji})$.

By conditioning on the $\tilde{\rho}_{ji}$ through time $t$, this dependency can be made explicit:

\[
p(\sigma(t+1)|\tilde{\rho}_{j1}(t+1), \ldots) = p(\sigma_{j1}(t+1)|\tilde{\rho}_{ji}(t), \ldots)
\]

(3.19)

since $\rho_{ji}(t+1)$ does not affect the evolution of $\sigma_{ji}(t+1)$,

\[
= \sum_{\tilde{\sigma}_{ji}(t)} p(\sigma_{j1}(t+1)|\tilde{\sigma}_{ji}(t), \tilde{\rho}_{ji}(t), \ldots) p(\sigma_{i1}(t)|\tilde{\rho}_{ji}(t), \ldots)
\]

(3.20)

\[
= \sum_{\tilde{\sigma}_{ji}(t)} p(\sigma_{j1}(t+1)|\tilde{\sigma}_{ji}(t), \tilde{\rho}_{ji}(t), \ldots) p(\sigma_{i1}(t)|\tilde{\rho}_{ji}(t), \ldots)
\]

(3.21)

giving a form whereby $p(\sigma_{i1}(t)|\tilde{\rho}_{ji}(t), \ldots, \tilde{\rho}_{ji}(t))$ can be found recursively for all $t > 0$. Then

\[
p(\sigma_i(t)) = \sum_{\tilde{\rho}_{ji}(t)} p(\sigma_{j1}(t)|\tilde{\rho}_{ji}(t), \ldots, \tilde{\rho}_{ji}(t)) p(\tilde{\rho}_{ji}(t), \ldots, \tilde{\rho}_{ji}(t))
\]

(3.22)

gives local state occupancy probabilities for $D_1$ at time $t$.

Thus in order to find $p(\sigma_i(t))$, (3.22) requires $p(\tilde{\rho}_{ji}(t), \ldots, \tilde{\rho}_{ji}(t))$

This is the reason conjecture 3.4 is false - one needs the probabilities of sequences of $\rho_{ji}$'s in order to compute $p(\sigma_i(t))$, not sequences of probabilities. If $\tilde{\rho}_{ji}(t)$ is independent of $\tilde{\rho}_{kj}(t)$, $k \neq j$, then

\[
p(\tilde{\rho}_{ji}(t), \ldots, \tilde{\rho}_{ji}(t)) = \prod_j p(\tilde{\rho}_{ji}(t))
\]

(3.23)

holds and $D_1$ needs the $p(\tilde{\rho}_{ji}(t))$ from each neighbor in order to find $p(\sigma_i(t))$

(This is true for a large class of topologies detailed in Section 4.1.)

This result, that $D_1$ requires probabilities of interaction sequences (i.e. densities on $\tilde{P}_{j1}$) instead of the proposed probabilities (i.e.
t densities on $P_i$) to predict local state occupancy probabilities, and hence performance, is important in constructing stochastic domular coordination algorithms (see Section 4.1.5, for example). Aside from this, stochastic domules are as suitable as models as deterministic ones.

This section presents some of the properties of the domule concept; the following will discuss some other concepts, all of which combine in Chapter 4 to allow coordination to occur.

3.3 Time-varying Markov Decision Problems

This section is motivated by consideration of the decision problem faced by an agent in a domule system given it has access to predictions of the interactions from others that will affect it. As mentioned in 2.2, the approach of the coordination strategies is to select interactions, then use them to solve the local problems. With the interaction sequences known, the local problem reduces to a time-varying (due to the $p_{ij}$ changing) Markov decision problem (in order to minimize the local component of the overall cost).

One of the mainstays of solution techniques for this class of problem has been the concept of dynamic programming. It allows current decision options to be evaluated by the results of each of them at future stages of the process. As such, it requires knowledge of how the decisions can affect the future course of the state evolution at each point in time; this requires a complete model of the system for all future times.

When decisions are made by a domular agent, however, an alternative to dynamic programming would be useful. If interactions are not predicted for all future times, then the time-varying system model mentioned above is not available for all future times. It would be valuable to have an
approach to this type of decision problem which does not require a system model for the entire infinite future; this section explores such an approach which projects the effects of each current decision option forward in time only far enough to make a good decision (e.g. optimal, or optimal with probability greater than 1-\(\epsilon\)). Such an approach will be called forward search, for reasons that are clear, and will be discussed first in a deterministic setting, then in the more difficult stochastic setting.

3.3.1. Deterministic Forward Search

Consider a system with state space \(\Sigma\), input set \(\mathcal{U}\), state transition function \(f\), and state transfer cost function \(c\):

\[
\sigma(t+1) = f(\sigma(t), u(t), t)
\]

with goal to:

\[
\min_{\mathcal{A}(\tau)} \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} c(\sigma(t), \sigma(t+1), u(t), t) \quad (3.24)
\]

This will be the general setting of this section.

The above formulation captures the local domular decision problem as well as many others. For predicted sequences \(\mathcal{D}_i\) to \(D_1\):

\[
f(\sigma(t), u(t), t) = f_i(\sigma(t), \rho_i(t), \ldots, \rho_{n_i}(t), u(t)) \quad (3.26)
\]

\[
c(\sigma(t), \sigma(t+1), u(t), t) = c_i(\sigma_i(t), \sigma_i(t+1), \rho_i(t), \ldots, u(t+1)) \quad (3.27)
\]

where

\[
c_{ij}(\rho_{ij}(t)) = \begin{cases} 0 & \text{if } \rho_{ij}(t) \text{ is as predicted by } D_1 \\ \infty & \text{else} \end{cases}
\]

(These latter terms penalize \(D_1\) if it fails to produce interactions which are scheduled to affect \(D_i\), thereby assuring consistency of assumptions between \(D_i\)'s expectations and \(D_1\)'s actions. It will be the
responsibility of the coordination process which selects predicted $\tilde{y}_j$'s to assure a finite (feasible) solution to (3.25) exists."

Forward search is a variant of forward dynamic programming which can be applied to the above general setting. It starts with the system in state $\sigma$ at some time $t$, say $t = 0$, and works forward from there. However, rather than deriving the optimal control for every possible cost-to-go function at sequential times, it directly answers the specific question "which $u$ should be applied in this situation?"

Consider some other time $t > 0$, and each state the system can occupy then. The minimum cost to reach $\sigma(t)$ from $\sigma(0)$, subject to a specific choice of $u(0)$, is $\omega(\sigma(0), \sigma(t), u(0), t)$:

$$\omega: \Sigma \times \Sigma \times \mathcal{U} \times \{0, 1, \ldots\} \rightarrow \mathbb{R}_+$$

(3.29)

(Note $\omega(\sigma(0), \sigma(t), u(0), t) = \infty$ is interpreted as meaning $\sigma(t)$ is not reachable from $\sigma(0)$ using any sequence of controls commencing with $u(0)$.)

It can be computed recursively as:

$$\omega(\sigma, \sigma', u, 1) = c(\sigma, \sigma', u, u)$$

(3.30)

$$\omega(\sigma, \sigma', u, t+1) = \min_{\sigma''} \omega(\sigma, \sigma'', u, t) + c(\sigma', \sigma'', u, t)$$

(3.31)

(3.30) gives the one-step cost incurred using $u$ to get from $\sigma$ to $\sigma'$;

(3.31) gives the cost of going from $\sigma$ to $\sigma'$ at $t+1$ as the cheapest combination of ways to get there through some $\sigma'$ at time $t$.

Now, suppose that for all $\sigma'$,

$$\omega(\sigma, \sigma', u, t) \leq \omega(\sigma, \sigma', u', t)$$

(3.32)

so that it is less expensive to reach any state at time $t$ using some control sequence starting with $u$ than for the cheapest sequence starting with $u'$. Then it can be concluded that, since the optimal state sequence must
pass through some \( \sigma^+ \), \( u \) is always preferable to \( u' \) as the control to be applied at time 0. This gives

**PROPOSITION 3.6.** If (3.32) is satisfied for all \( u' \), then the optimal control to apply at time zero is \( u \).

**PROOF:** Let \( v(\sigma^+) \) be the optimal cost-to-go function for time \( t \) as it would be computed by dynamic programming. Then the total cost incurred along the optimal trajectory is

\[
\min_{\sigma^+} \omega(\sigma, \sigma^+, u, t) + v(\sigma^+) \quad (3.33)
\]

with the optimal control being

\[
u^* = \arg \min_{\sigma^+} \min_{u} \omega(\sigma, \sigma^+, u, t) + v(\sigma^+) \quad (3.34)
\]

or,

\[
\min_{\sigma^+} \omega(\sigma, \sigma^+, u^*, t) \leq \min_{\sigma^+} \omega(\sigma, \sigma^+, u, t) + v(\sigma^+) \quad (3.35)
\]

(3.32) implies

\[
\omega(\sigma, \sigma^+, u, t) + v(\sigma^+) \leq \omega(\sigma, \sigma^+, u, t) + v(\sigma^+) \quad (3.36)
\]

which gives (3.35) by taking the min of both sides and letting \( u = u^* \).

Thus (3.32) can serve as an optimality condition to select the optimal control at time 0, and it requires computation of the cost-to-reach function \( \omega \) only, not the entire cost-to-go function \( v \). This gives the

**ALGORITHM 3.7. Deterministic Forward Search**

For each time step \( t \) for which a decision
must be made:

Let $\sigma = \sigma(t)$; set $S = \mathcal{U}$ as the set of remaining possibly optimal controls

For each $\tau \in \{0, 1, \ldots, T\}$ until $|S| = 1$:

For each $\sigma^* \in \Sigma$ and $u \in S$ compute

$$
\begin{align*}
\omega(\sigma, \sigma^*, u, \tau) &= \begin{cases} 
    c(\sigma, \sigma^*, u, \tau) & \tau = 1 \\
    \min_{\sigma^* \in \Sigma} \omega(\sigma, \sigma^*, u, \tau-1) + c(\sigma^*, \sigma^*, u, \tau-1) 
\end{cases}
\quad (3.37)
\end{align*}
$$

For each $u'$ such that there exists a $u$

$$
\omega(\sigma, \sigma^*, u, \tau) \leq \omega(\sigma, \sigma^*, u', \tau) \quad \forall \sigma^* 
\quad (3.38)
$$

Set $S = S - \{u'\}$

Apply $u(t)$ as the remaining element of $S$.

As an aside, note that the computation of $\omega(\sigma, \sigma^*, u, \tau)$ can be written as

$$
\omega(\sigma, \sigma^*, u, \tau+1) = \min_{\sigma^*} \omega(\sigma, \sigma^*, u, \tau) + \min_{\sigma^*} c(\sigma^*, \sigma^*, u, \tau) 
\quad (3.39)
$$

Defining $N \times N$ matrices ($N = |\Sigma|$)

$$
\begin{align*}
W(u, t) &\triangleq [\omega(\sigma, \sigma^*, u, t)] \\
C(t) &\triangleq [\min_{u'} c(\sigma, \sigma^*, u', t)] 
\end{align*}
\quad (3.40) (3.41)
$$

(3.39) can be written as

$$
W(u, t+1) = W(u, t) (\min_{u'}) C(t) 
\quad (3.42)
$$

where $\min_{u'}$ is the scalar product defined as the minimum of the componentwise sums.

**Example 3.8.** Consider the case of the superior
domule in the example of 3.1 when in state 1,
receiving interaction 2, and subsequent interactions
predicted as 2, 1, 2, 2. Which control should it apply? First, find

\[
C(t) = \begin{cases} 
\begin{bmatrix} 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 
\end{bmatrix} & \text{if } \rho_{11}(t) = \rho_{11}^1 \\
\begin{bmatrix} 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 
\end{bmatrix} & \text{if } \rho_{11}(t) = \rho_{11}^2 
\end{cases}
\] (3.43)

Since \( \sigma(\phi) = \sigma_1^{-1} \), only the first row of need be computed:

\[
W(\omega_{11}^1, 1) = \begin{bmatrix} \phi_1 & \phi_2 & \phi_3 & \phi_4 \end{bmatrix}
\] (3.44)

\[
W(\omega_{11}^2, 1) = \begin{bmatrix} \phi_1 & \phi_2 & \phi_3 & \phi_4 \end{bmatrix}
\] (3.45)

since \( \rho_{11}(0) = \rho_{11}^2 \).

Table 3.8 shows succeeding computations of \( W(\omega_{11}, t) \). Comparability is achieved at \( t = 5 \), so \( \omega_{11}^1 \) is the optimal control to apply at time 0. This will drive the superior to state \( 3 \), so forward search for the control at \( t = 1 \) will start in state \( 3 \) with an interaction sequence commencing with \( (2, 1, 2, 2) \). It can be shown that, for this example, one needs at most five steps for forward search to converge in all cases.

The issue of convergence of the finite search algorithm is a thorny one. It does not converge in all cases; Figure 3.9 shows two cases where it will never converge. The first exemplifies a reachability problem; hence for state 1
Table 3.3

Example Computation of Forward Search

<table>
<thead>
<tr>
<th>t</th>
<th>( W(u_1^1, t) )</th>
<th>( W(u_1^2, t) )</th>
<th>( c(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>([- - - 1])</td>
<td>([- - 1 -1])</td>
<td>(c_2)</td>
</tr>
<tr>
<td>2</td>
<td>([- 1 2 -1])</td>
<td>([1 - - 2])</td>
<td>(c_1)</td>
</tr>
<tr>
<td>3</td>
<td>([- - 2 2])</td>
<td>([- 1 3 3])</td>
<td>(c_2)</td>
</tr>
<tr>
<td>4</td>
<td>([2 2 3 3])</td>
<td>([3 3 2 2])</td>
<td>(c_2)</td>
</tr>
<tr>
<td>5</td>
<td>([3 3 3 3])</td>
<td>(\geq)</td>
<td>([2 2 3 3])</td>
</tr>
</tbody>
</table>
Figure 3.9
Problems with Forward Search Convergence

A: Reachability

B: Periodicity
\begin{align}
\omega (\sigma^+, \sigma^-, \omega^+, t) &= \left[ \omega_1(t) \right] \\
\omega (\sigma^+, \sigma^-, \omega^+, t) &= \left[ \omega_2(t) \right]
\end{align} \tag{3.46}

where both \( \omega_1(t) \) are finite— and hence the two cases will never satisfy (3.32). The second possesses a periodic cycle which is entered at different points, causing the \( \omega (\sigma^+, \sigma^-, \omega^+, t) \) to oscillate between the two forms of (3.46) out of phase with one another—so again comparability is never achieved.

An aid to proving convergence in special cases is

**PROPOSITION 3.8.** Forward search converges in finite time \( T \) if the backwards dynamic programming solution is independent of the terminal cost after \( T \) steps.

**PROOF:** This is a direct result of the optimality of forward search. If dynamic programming converges to \( u^* \) independently of the terminal cost \( v(\sigma^+, T) \), then

\[ \omega (\sigma^+, \sigma^-, \omega^+, T) + v(\sigma^+, T) - \omega (\sigma^+, \sigma^-, \omega^+, T) \leq v(\sigma^+, T) \tag{3.47} \]

for all \( \omega^+ \) and \( v(\sigma^+, T) \). Setting \( v(\sigma^+, T) = 0 \), this gives (3.32), the forward search optimality condition.

This is useful in cases where dynamic programming itself converges in finite time (as is the case with a large class of finite state problems) but in many cases it converges only asymptotically to a solution (such as for linear quadratic control problems). In such cases, a reasonable modification to the convergence criterion (3.32) is the \( \varepsilon \)-optimality condition:
choose \( u \) if

\[
\omega(\sigma, \sigma^{-}, u, t) \leq \omega(\sigma, \sigma^{-}, u', t) + \varepsilon
\]

for all \( u' \) and some small positive \( \varepsilon \). This assures that if \( u \) is chosen, performance resulting from that decision will be at most worse than had the actual optimal decision been made. This will lead to finite convergence for many problems with continuous state spaces.

Thus there exists an algorithm which allows the optimal local control to be determined by working only a finite time into the future. This bodes well for the coordination approach adopted in this work; it means that the search for optimal interaction sequence predictions need only provide such which are accurate for a finite time. While this is not an easy task, Chapter 4 will outline some approaches to it. Most of these will make use of the local cost-to-reach function

\[
\omega(\sigma, \sigma^{-}, t) \triangleq \min_u \omega(\sigma, \sigma^{-}, u, t)
\]

which will be used to evaluate bounds on the optimal local cost in response to a predicted sequence.

However, the scope of application of this technique is not limited to modular problems. It applies to any time-varying process, as do the results of the next section on stochastics.

3.3.2. Stochastic Forward Search

Extension of the ideas of forward search to stochastic settings is not as simple as might be expected. The usual approach of embedding a stochastic problem in an equivalent, higher dimensional space runs into difficulty as the space is not reachable in finite time from any starting state. Hence difficulties arise in comparing alternatives when each
allows a state (probability distribution on $\Sigma$) to be reached which the
other does not. Two approaches to avoid this problem can be taken:
the first derives a bound which limits the comparison to less than the
entire space; the second assumes a terminal criterion which can be in-
voked after an arbitrary number of steps of forward search. This latter
is the method that will be used in Chapter 4, where such a criterion
arises naturally.

As in Section 3.1.2, let $\mathcal{P}(\Sigma)$ be the set of distributions on $\Sigma$,
and $p(\sigma^+|\sigma, u, t)$ the probability of transferring from $\sigma$ at time $t$ to
$\sigma^+$ at time $t+1$ when $u(t) = u$. The distribution $p(\sigma, t)$ is a state in
the deterministic system

$$p(\sigma^+, t+1) = \sum_{\sigma} p(\sigma^+|\sigma, u, t) p(u|\sigma, t) p(\sigma, t) \quad (3.50)$$

where $p(u|\sigma, t)$ is the (randomized) decision strategy used at $t$. The cost
for a single time step,

$$c: \mathcal{P}(\Sigma) \times \mathcal{P}(\Sigma) \times \mathcal{P}(\mathcal{U} \mid \Sigma) \times \{0, 1, \ldots\} \to \mathbb{R}_+ \quad (3.51)$$
is

$$c(\rho(\sigma, t), \rho(\sigma^+, t+1), p(u|\sigma, t), t) =
\begin{cases}
\sum_{\sigma} c(\sigma, \sigma^+, u, t) p(\sigma^+|\sigma, t+1) p(\sigma, t) p(u|\sigma, t) \\
\infty \quad \text{if} \ (3.50) \ \text{not satisfied}
\end{cases} \quad (3.52)$$
or equivalently

$$c(\rho(\sigma, t), p(u|\sigma, t), t) =
\sum_{\sigma} c(\sigma, \sigma^+, u, t) p(\sigma^+|\sigma, u, t) p(u|\sigma, t) p(\sigma, t) \quad (3.53)$$

Equation (3.52) hints at the reachability problem - the cost of
reaching a $p(\sigma^+, t+1)$ from a $p(\sigma, t)$ is infinite if an appropriate
$p(u|\sigma, t)$ cannot be found. That such cases exist is easily illustrated;
in fact, only under very special conditions is the state space completely
reachable in a finite number of steps.

EXAMPLE 3.9. Consider the (time invariant) system of Figure 3.10. Let \( \sigma (t) = \sigma ^{-1} \).

Then the application of \( \omega \) yields state \([1/2, 1/2]\), where \( \omega \) leads to \([1/3, 1/3]\). For forward search to be useful, one would like the set of states reachable from \([1/2, 1/2]\) after \( t \) steps

\[
\mathcal{X}_1(t) = \left\{ \left[ \lambda, \mathcal{N}_1 \right] \mid \frac{1}{2} \left( \frac{1}{3} \right)^{t-1} \leq \lambda \leq \frac{1}{2} \left( \frac{1}{2} \right)^{t-1} \right\}
\] (3.54)

(assuming randomized strategies if necessary) to equal the set of states reachable from \([1/3, 1/3]\):

\[
\mathcal{X}_2(t) = \left\{ \left[ \lambda, \mathcal{N}_2 \right] \mid \frac{1}{3} \left( \frac{1}{3} \right)^{t-1} \leq \lambda \leq \frac{1}{3} \left( \frac{1}{2} \right)^{t-1} \right\}
\] (3.55)

However, the state \( \left[ \left( \frac{1}{3} \right)^{t} \mathcal{N}_3 \right] \) is always in \( \mathcal{X}_1(t) \) but never in \( \mathcal{X}_1(t) \); \( \left[ \left( \frac{1}{2} \right)^{t} \mathcal{N}_4 \right] \) is always in \( \mathcal{X}_2(t) \) but never \( \mathcal{X}_2(t) \). Hence not only is the complete space of distributions not reachable, but the subsets reachable from each of two initial ones may (and usually) never match.

Additional information on the structure of the system can help alleviate this difficulty while preserving the forward search concept. If the future state transition probabilities and costs are known to be elements of a particular set, then the two approaches mentioned earlier can be employed. Let \( S \) be the set of all possible pairs \( \left\{ \left( \sigma^*, \mathcal{N}_1, \mathcal{N}_2 \right) \right\} \) describing state transition probabilities and costs. While the actual probability-cost pairs relevant to a particular time may change from \( t \)
Figure 3.10

Stochastic System with Reachability Differences

\[ p(\sigma+|\sigma, \omega_i):  \]

\[ 1 \quad \text{\(1/2\)} \quad 2 \quad 1 \]

\[ 1/2 \]

\[ p(\sigma+|\sigma, \omega_3):  \]

\[ 1 \quad \text{\(2/3\)} \quad 2 \quad 1 \]

\[ 1/3 \]
to \( t+1 \) etc., assume \( S \) is time invariant - it completely describes such pairs.

**DEFINITION 3.10**: \( S \) is **completely ergodic** if there exists some positive integer \( T \) such that the \( T \)-period state transition probability

\[
\rho (\sigma(t+1) | \sigma(t), \varpi(t)) = \begin{cases} 
\sum_{\sigma(t)} \rho (\sigma(t+1) | \sigma(t), \varpi(t)) \rho (\sigma(t) | \sigma(0), \varpi(t-1)) \\
\rho (\sigma(t) | \sigma(0), \varpi(0)) 
\end{cases} 
\]  
(3.56)

satisfies

\[
\rho (\sigma(t) | \sigma(0), \varpi(t-1)) \geq \varepsilon 
\]  
(3.57)

for all sequences \( \varpi(t) = (\varpi(0), \ldots, \varpi(t)) \), all sequences of one-step transitions \( \rho (\sigma(t) | \sigma(0), \varpi(0)), \ldots, \rho (\sigma(t) | \sigma(t-1), \varpi(t-1)) \) and \( \varepsilon > 0 \).

If \( S \) is completely ergodic, then no matter which sequence of elements of \( S \) determine the dynamics of the system for the next \( T \) periods, and no matter which controls are applied, there is always an \( \varepsilon \) probability of reaching any state \( \sigma(t) \) from any \( \sigma(0) \). This leads to a bound on the cost-to-go function \( \nu(\sigma; t) \) which, in turn, helps relieve the forward search convergency problem.

The bound of interest is on the quantity

\[
\nu(t) \overset{\Delta}{=} \max_{\sigma(t)} \nu(\sigma, t) - \min_{\sigma(t)} \nu(\sigma, t) 
\]  
(3.58)

Note that

\[
\nu(\sigma^+, t+T) \leq (1-\varepsilon) \max_{\sigma(t)} \nu(\sigma, t) + \varepsilon \min_{\sigma(t)} \nu(\sigma, t) + \max_{\sigma(t), \varpi(t), \varpi(t+1)} c(\sigma, \sigma^+, \varpi(t), \ldots, \varpi(t+T-1)) 
\]  
(3.59)

(where \( c(\sigma, \sigma^+, \varpi(t), \ldots, \varpi(t+T-1)) \) is the expected cost of transferring
from $\sigma$ at $t$ to $\sigma^+$ at $t+T$ using the controls listed) if $S$ is completely ergodic. Also

$$
\nu(\sigma^+, t+T) \geq (1-\epsilon) \min_{\sigma} \nu(\sigma, t) + \epsilon \max_{\sigma} \nu(\sigma, t)
$$

$$
+ \min_{\sigma, \lambda(t), \ldots, \lambda(t+T-1)} c(\sigma, \sigma^+, \lambda(t), \ldots, \lambda(t+T-1))
$$

so

$$
\left( \max_{\sigma^+} \nu(\sigma^+, t+T) - \min_{\sigma} \nu(\sigma, t+T) \right) \leq
$$

$$
(1-2\epsilon) \left( \max_{\sigma} \nu(\sigma, t) - \min_{\sigma} \nu(\sigma, t) \right)
$$

where

$$
c = \max_{\sigma^+} c(\sigma, \sigma^+, \lambda(t), \ldots, \lambda(t+T-1)) - \min_{\sigma, \lambda(t), \ldots, \lambda(t+T-1)} c(\sigma, \sigma^+, \lambda(t), \ldots)
$$

Equation (3.61) gives a way to solve for a bound on $\nu(t)$ recursively; since $\epsilon > 0$ it does converge to $c/2\epsilon$.

Recall that the original forward search optimality criterion was derived from the fact that $\lambda'(t)$ is preferable to $\lambda^t(\sigma(t))$ if

$$
\min_{\lambda'(\sigma(t))} \omega(\rho(\sigma(t)), \rho'(\sigma(t)), \lambda'(t)) + \nu(\rho'(\sigma(t))) 
\leq
\min_{\rho^t(\sigma(t))} \omega(\rho(\sigma(t)), \rho^t(\sigma(t)), \lambda^t(\sigma(t))) + \nu(\rho^t(\sigma(t)))
$$

Since $\nu(\sigma^t(t)) - \nu(\sigma^t(t)) \leq \nu(t)$, and the knowledge of $\sigma(t)$ at time $t$

gives

$$
\nu(\rho^t(\sigma(t))) = \sum_{\sigma(t)} \rho^t(\sigma(t)) \nu(\sigma(t))
$$

(3.63) leads to $\lambda'(t)$ preferable if

$$
\min_{\lambda'(\sigma(t))} \omega(\rho(\sigma(t)), \rho'(\sigma(t)), \lambda'(t)) \leq \min_{\rho^t(\sigma(t))} \omega(\rho(\sigma(t)), \rho^t(\sigma(t)), \lambda^t(\sigma(t))) - \nu(t)
$$

This proves

**Proposition 3.11.** An alternative optimality criterion for stochastic forward search is that

$\lambda'(t)$ is preferable to $\lambda^t(\sigma(t))$ if

$$
\omega(\rho(\sigma(t)), \rho'(\sigma(t)), \lambda'(t)) \leq \omega(\rho(\sigma(t)), \rho^t(\sigma(t)), \lambda^t(\sigma(t))) - \frac{\epsilon}{2\epsilon}
$$

(3.66)
for some $\rho^1(\sigma(t))$ which holds for all $\rho^2(\sigma(t))$.

The advantage of this form is that even if $\rho^1(\sigma(t))$ is not reachable from $\rho(\sigma(o))$ using $\omega^2(o)$, the cost to reach it may be low enough that it can be compared with $\rho^2(\sigma(t))$. This comparison is possible because $\xi/2\varepsilon$ bounds the difference on the cost-to-go's resulting from any $\rho(\sigma(t))$ pair. (This bound can be tightened considerably, but (3.66) illustrates the point most simply.) Thus a priori knowledge of some of the future system structure, gained through $S$, can be used to relax the requirement that every state distribution must be less expensive to reach using $\omega^1(o)$ than $\omega^2(o)$.

While this approach does relieve the reachability problem, convergence of the algorithm might still take prohibitively long, particularly if $\xi/2\varepsilon$ is quite large. Since the forward search must be repeated at each time step, this could lead to an extremely expensive processing requirement for the controller. If this is a problem, then one can adopt the approach, mentioned in 2.1.3, which is used in artificial intelligence: static terminal evaluation. That is, one posits a $\nu_\tau(\sigma)$, chooses a depth of forward search $T$, and uses the criterion

$$\nu^* = \arg\min_{\nu(0)\geq0} \min_{\rho(\sigma(o))} \omega(\rho(\sigma(o)), \rho(\sigma(t)), \omega(o) + \sum_{\sigma(t)} \rho(\sigma(t)) \nu_\tau(\sigma(t)))$$  \hspace{1cm} (3.67)

The selection of $\nu_\tau(\sigma)$ is the crucial issue here. If $\nu_\tau(\sigma)\geq0$ , then the forward search finds the optimal myopic strategy for the fixed horizon $T$. If $\nu_\tau(\sigma)$ dominates $\omega(\rho(\sigma(o)), \rho(\sigma(t)), \omega)$, then it effectively forces the system to find the decision most likely to drive it to the state that minimizes $\nu_\tau(\sigma)$. It is possible to consider adaptive
schemes, where the controller estimates a $\nu_r(\sigma)$ based on past results - but these are fraught with stability problems and ill suited to the time-varying structure of the problem.

However, the $\nu_r(\sigma)$ most consistent with the philosophies of the domular problem is that which is derived assuming a worst-case (costwise) model structure beyond time $T$. This is, given that a control is applied, the system model will create dynamics which force the results of that control to be as expensive as possible. This essentially sets up a game - the controller acts against an antagonistic model, and the resulting steady state value of the game is used as $\nu_r(\sigma)$. Section 4.1.3 gives the details of this approach, as exactly the same situation arises in a domule when it has no way to predict the interactions which will affect it.

Thus forward search is more difficult in the stochastic case, although ways can be found to circumvent the reachability problem. The more practical approach, though, is to work forward only a finite distance in time, then terminate with a cost function that in some manner reflects the future structure of the problem. By working forward in time, it is possible to defer use of $\nu_r(\sigma)$ if it dominates $\omega(\cdot, \sigma, \cdot, \cdot, \cdot, \cdot, \cdot, r)$ and hence limit the sensitivity of the decision strategy to that $\nu_r(\sigma)$.

In all, the forward search approach provides an alternative to dynamic programming as it focuses attention on the short-term cost-to-reach function rather than the long-term cost-to-go function. However, much remains to be done in terms of applying the concept to problems of special structure which lead to reduction of the computation required.

3.4 Abstraction

Abstraction is a very powerful technique used in human decision making to guide the selection of one particular course of action out of
a bewildering array of possibilities. This is generally done by simplifying the problem—stripping it of confusing details, solving the resulting problem, then modifying the solution to include adjustments necessitated by the details originally ignored. Thus, at least conceptually, this seems a relevant technique to explore in the context of automatic decision making in complex systems.

The thrust of this section is to formalize some aspects of the idea of abstracting a decision problem, and identify some of the issues involved. The concept of abstraction, while powerful, is vague; this section will only begin to touch on the use of generalization, simplification, etc. as techniques applicable to decision making. The two themes of major import which unite it are the general decision making approach using abstraction mentioned in the opening paragraph, and the idea of using abstraction to produce a simplified model of a decision maker and associated decision problem as they might appear to an outside observer. While the first is of general interest, and may lead to a change in the way one can approach a general, complex decision problem, the second is of specific interest in the case of domular systems. The concepts developed, which allow partial prediction of the activities of a decision maker/problem pair without complete knowledge of either, lead directly to the domular coordination scheme discussed in Section 4.1.6.

This section will approach the concept of abstraction as a decision tool in a sequence of three steps. At first, it will be limited in scope to the area of static, or one-step decision problems. Ideas formulated in this context will be extended to a more general, dynamic setting in the next section, then focussed on the domular structure in the final
subsection. Throughout, all problems will be assumed deterministic; stochasticity add obscuring detail with no corresponding increase in
generality - the approach of the previous sections on stochasticity, em-
bedding in a higher dimensional space, seems to be perfectly applicable
to the framework erected here.

3.4.1. **Static Abstraction**

The best place to begin an exploration of the concept of abstraction
is at the basic structure of a static decision problem. The uses of ab-
straction in this simple context can lead to strategies which can be em-
ployed in the more complex dynamic and domilar settings.

**DEFINITION 3.12:** A static decision problem is a

quintuple

\[(\Sigma, \mathcal{U}, \mathcal{Y}, f, c)\]

with \(\Sigma\) = set of states of nature

\(\mathcal{U}\) = set of controls

\(\mathcal{Y}\) = set of outcomes

\(f: \Sigma \times \mathcal{U} \rightarrow \mathcal{Y}\) outcome generation function

\(c: \Sigma \times \mathcal{U} \times \mathcal{Y} \rightarrow \mathbb{R}_+\) objective function

and goal to

\[\text{minimize } c(\sigma, u, y) \text{ subject to } f(\sigma, u) = y \quad (3.68)\]

by choosing an appropriate \(u\) for the state of

nature \(\sigma\) which is relevant.

Such a decision problem can either be solved "off-line", in which the

1. Note that if \(u\) is prohibited when \(\sigma\) occurs, one can set \(c(\sigma, u, y) = \infty\)

for all \(y\).
decision function

\[ \gamma : \Sigma \rightarrow \mathcal{U} \]  
\[ \gamma(\sigma) = \arg \min_{\mathcal{U}} c(\sigma, \omega, f(\sigma, \omega)) \]  

is found and stored for all possible \( \sigma \), or "on-line", when (3.70) is evaluated only when \( \sigma \) is known. The choice is usually influenced by the relative cost of storage for \( \gamma \) and on-line processing for (3.70), which in turn often depends on special structure exhibited by \( f \) or \( c \).

The use of abstraction can occur in either on-line or off-line solutions, but will be discussed only in the on-line setting here. Thus the decision maker is faced with selecting \( \omega \) given a particular \( \sigma \). The most general solution is to compute

\[ c'(\sigma, \omega) \triangleq c(\sigma, \omega, f(\sigma, \omega)) \]  

for each \( \omega \), and select the one resulting in the lowest such value. This requires \( |\mathcal{U}| \) evaluations of both \( f \) and \( c \), as well as minimization over a set of size \( |\mathcal{U}| \). It also results in the optimal decision in each case.

This seems simple enough. What is the motivation for trying any other approach? Since one cannot improve the objective performance of a decision maker by any schemes other than this "brute force" approach, why is it so rarely (if ever) used in practice? The answer must lie, as stated in Section 2.2, in the need to reduce the computational load required for implementation. This is clear when \( \mathcal{U} \) is infinite; it also applies when \( \mathcal{U} \) is finite but "large" in some sense.

It is difficult to approach the issue of computational complexity in a general sense without reference to the primitive operations out of which a decision algorithm may be structured. Clearly the primitives used in
automata theory are at too low a level to be of interest in a general control problem; arbitrary function evaluations may be too high. Thus the central dilemma facing the application of abstraction is discovered: implementation issues force the use of abstraction in practice, yet simultaneously inhibit the construction of a general theory due to the specificity required.

In light of these considerations, this section will attempt to capture the intuitive notion of what abstraction is, and how it can be used, but avoid delving into specific applications or claiming to construct a general theory.

Mentioned earlier, one use of abstraction in human decision making is to find a "simpler" decision problem, \((\Sigma, \hat{\mathcal{U}}, \hat{\mathcal{Y}}, \hat{\mathcal{E}}, \hat{\mathcal{C}})\), when faced with a "hard" decision, solve the abstracted version, and use that solution to aid in solving the original. This process is illustrated in Figure 3.11; find an abstraction

\[
\alpha((\Sigma, \mathcal{U}, \mathcal{Y}, \mathcal{E}, \mathcal{C})) = (\hat{\Sigma}, \hat{\mathcal{U}}, \hat{\mathcal{Y}}, \hat{\mathcal{E}}, \hat{\mathcal{C}})
\]

which takes the original decision problem \(D\) into an abstracted version \(\hat{D}\); solve \(\hat{D}\) for the \(\hat{\mathcal{U}}^*\) best for it, then use some specialization function

\[
\beta((\Sigma, \mathcal{U}, \mathcal{Y}, \mathcal{E}, \mathcal{C}), \hat{\mathcal{U}}^*) = (\Sigma', \mathcal{U}', \mathcal{Y}', \mathcal{E}', \mathcal{C}')
\]

which reduces \(D\) to some simpler form \(D'\). \(D'\) can then be solved to yield the overall choice \(\mathcal{U}'^*\).

Note:

1. The process can be recursed; use \(\hat{\alpha}\) to abstract \(\hat{D}\) to some \(\hat{\hat{D}}\); use \(\hat{\mathcal{U}}^*\) to form a \(\hat{D}'\) which yields the \(\hat{\mathcal{U}}^*\) for (3.72), etc.

2. The price paid in practice for using this approach
Figure 3.11
Use of Abstraction: Static Case
is that $u^*$ may not be optimal for $D$ as the abstraction serves as an approximation, and therefore errors are made. It is desirable to structure $\alpha$ and $\beta$ so that tight bounds on this error can be found.

3. In the vein of the above, it may be necessary to redefine optimality with respect to $\hat{D}$ and $D'$. If suboptimality of $u^*$ is possible ($u^* \neq u^*$), then some flexibility can be introduced in the solution of $\hat{D}$ or $D'$.

EXAMPLE 3.13: Consider the search for the minimum of a highly complex function. Let

$$D = \begin{cases} \Sigma = \{\sigma\} & \mathcal{U} = \mathcal{Y} = \mathbb{R} \\ f(\sigma, u) = f_0(u) \\ c(\sigma, u, y) = y \end{cases} \tag{3.74}$$

Thus $\sigma$ plays no role, $f_0$ is the function to be minimized, and the search is over the set of possible minima, $\mathbb{R}$.

Exact solution requires testing of $f$ at all points. A solution using abstraction is to divide $\mathbb{R}$ into intervals of length $\ell > 0$, take a single sample of $f$ in each interval, and find the interval containing the smallest sample. Then this interval can be the subject of a more detailed search. In this case
\[\hat{\Sigma} = \{ \hat{\alpha} \} \]
\[\hat{\mathcal{U}} = \{ (n\epsilon, (n+1)\epsilon), n = 0, 1, \ldots \} \]
\[\hat{\gamma} = \mathcal{R} \]
\[\hat{f}(\hat{\alpha}, \hat{\omega}) = f(x) \quad x = \inf(\hat{\omega}) \]
\[\hat{z}(\hat{\alpha}, \hat{\omega}, \hat{\gamma}) = \hat{\gamma} \]

is the abstraction resulting from \( D \). Its solution is the \( \hat{\omega} \) containing the minimum sample: \([n^*\epsilon, (n^*+1)\epsilon)\). This reduces \( D \) to

\[\Sigma' = \{ \sigma' \} \]
\[\mathcal{U}' = [n^*\epsilon, (n^*+1)\epsilon) \]
\[\gamma' = \mathcal{R} \]
\[\hat{f}'(\sigma', \omega') = c'(\sigma', \omega', \gamma') = f_0(\omega') \]

which is simpler than \( D \) in the sense that the region of search is smaller. The result of the entire process is some \( \sigma^* \) which, while not necessarily the point at which \( f_0(\cdot) \) is minimized, is hopefully near it and certainly better than many others.

Naturally not every \( \alpha \) and \( \beta \) are useful in such a scheme. It is worthwhile developing properties which \( \alpha \) and \( \beta \) must have in order to serve as a useful abstraction. First, in order for \( \hat{D} \) and \( D' \) to be more simple than \( D \), one would like

\[|\hat{\mathcal{U}}| \leq |\mathcal{U}| \quad |\mathcal{U}'| \leq |\mathcal{U}| \]

in order to reduce the scope of the search.
Secondly, if the state of nature $\sigma$ is known in $D$, then the abstracted state should also be known. Thus specification of $\sigma$ uniquely defines $\widehat{\sigma}$:

$$\alpha(\sigma) = \widehat{\sigma}$$  \hspace{1cm} (3.78)

That is, the $\widehat{\sigma}$ corresponding to $\sigma$ is $\alpha(\sigma)$, and the set of all such $\widehat{\sigma}$ is identical to $\widehat{\Sigma}$ (no extraneous $\widehat{\sigma}$'s).

As in the Example 3.13, selection of a $\widehat{U}^*$ should reduce the scope of the search over $U$ for a solution to $D$; this suggests letting each $\widehat{U}$ correspond to a subset of $U$. The union of these subsets need be neither

1. Mutually exclusive - the same $u$ may be represented by several $\widehat{U}$'s
2. Collectively exhaustive - a $u$ need not be represented
   (particularly if it is never a solution) in order to gain simplicity.

Thus there is a relation $R_u$ between elements of $U$ and $\widehat{U}$

$$u R_u \widehat{U} \text{ if } \widehat{U} \text{ represents a subset of } U \text{ containing } u$$  \hspace{1cm} (3.78)

Similarly, a $\widehat{\gamma}$ can be interpreted as representing a subset of $\gamma$, yielding

$$\gamma R_\gamma \widehat{\gamma} \text{ if } \widehat{\gamma} \text{ represents a subset of } \gamma \text{ containing } \gamma$$  \hspace{1cm} (3.79)

This allows for the following interpretation and structure on $\widehat{f}$: given $\widehat{\sigma}$, and a decision to apply $\widehat{U}$, one would like the $\widehat{f}$ for $\widehat{D}$ to give a $\widehat{\gamma}$ such that

$$\widehat{f}(\widehat{\sigma}, \widehat{U}) = \widehat{\gamma} \supset \{ \gamma = f(\sigma, u) \mid \alpha(\sigma) = \widehat{\sigma}, u R_u \widehat{U} \}$$  \hspace{1cm} (3.80)
Thus the \( \hat{\gamma} \) modelled as resulting from \( \hat{\sigma} \) and \( \hat{\omega} \) by \( \hat{z} \) represents a subset of \( \gamma \) containing all possible \( \gamma \) resulting from \( \sigma \) represented by \( \hat{\sigma} \) and \( \omega \) represented by \( \hat{\omega} \). This will prove valuable in allowing \( D \) to be used to predict the outcome of a decision maker's attempts to solve \( D \), as seen by an outside party.

Finally, \( \tilde{\varepsilon}(\hat{\sigma}, \hat{\omega}, \hat{\gamma}) \) should serve the same purpose: generate a subset of \( \mathbb{R} \) (the range space of \( \zeta(\cdot, \cdot, \cdot) \)) which contains

\[
\left\{ \zeta(\sigma, \omega, \gamma) \mid \alpha(\sigma) = \hat{\sigma}, \omega \in \mathbb{R} \setminus \hat{\omega}, \gamma \in \mathbb{R} \setminus \hat{\gamma} \right\}
\]

(3.81)

the set of possible results of applying \( \omega \)'s when \( \sigma \)'s occur, and generating \( \gamma \)'s, in \( D \) for each such triple represented by the corresponding abstracted variables in \( D \). This set may be complicated in structure; only its smallest and largest elements are of interest. Thus it is natural to define \( \tilde{\varepsilon} \) as

\[
\tilde{\varepsilon} : \hat{\Sigma} \times \hat{\mathbb{U}} \times \hat{\gamma} \to \mathbb{E}
\]

(3.82)

where \( \tilde{\varepsilon}(\hat{\sigma}, \hat{\omega}, \hat{\gamma}) = [\tilde{\eta}, \tilde{\eta}'] \in \mathbb{E} \) implies

\[
\tilde{\gamma} \leq \zeta(\sigma, \omega, \gamma) \leq \tilde{\eta}' \quad \text{for all} \quad \omega, \gamma \text{ represented by } \sigma, \omega
\]

(3.83)

and \( \mathbb{E} \) is the set of intervals on \( \mathbb{R} \).

This discussion motivates

**DEFINITION 3.14**: An abstraction \( \hat{D} \) of a decision problem \( D \) satisfies:

\[
|\hat{\Sigma}| \leq |\Sigma| \quad \alpha : \Sigma \to \hat{\Sigma}
\]

(3.84)

\[
|\hat{\mathbb{U}}| \leq |\mathbb{U}| \quad R_\omega \subseteq \mathbb{U} \times \hat{\mathbb{U}}
\]

(3.85)

\[
R_\gamma \subseteq \gamma \times \hat{\gamma}
\]

(3.86)
\{ y \mid y \in \mathcal{X} \} \supseteq \{ y = f(\sigma, u) \mid \mathcal{X}(\sigma) = \mathcal{X}, u \in \mathcal{U} \} \quad (3.87) \\
\text{if } \hat{F}(\hat{X}, \hat{Z}) = \hat{Y}

[\eta, \eta] \supseteq \{ \alpha(\sigma, u, \gamma) \mid \mathcal{X}(\sigma) = \mathcal{X}, u \in \mathcal{U}, \gamma \in \mathcal{Y} \} \quad (3.88) \\
\text{if } \hat{S}(\hat{X}, \hat{Z}, \hat{Y}) = [\eta, \eta]

EXAMPLE 3.15: The identity abstraction of \( D \) is \( D \); (3.84)-(3.88) are satisfied with \( \alpha \), \( \mathcal{R}_u \), and \( \mathcal{R}_Y \) the identity functions and relations, \( \hat{F} = f \), \( \hat{S} = \mathcal{S} \), and noting that \( \mathcal{R} \) (the range of \( \mathcal{C} \)) is isomorphic to a subset of \( \mathbb{I} \) (the range of \( \mathcal{E} \)) of elements of the form \( [\eta, \eta] \).

The trivial abstraction (or null abstraction) of \( D \) is a decision problem with

\[ |\hat{S}| = |\hat{A}| = |\hat{Y}| = 1 \quad (3.89) \]

and \( \alpha, \mathcal{R}_u \), and \( \mathcal{R}_Y \) universal relations.

If \( \hat{D} \) is an abstraction of \( D \), then one can make the decision described by \( D \) using

ALGORITHM 3.16: Static decision making with abstraction. Given a decision problem \( D \) and an abstraction \( \hat{D} \),

1. Determine the state of nature in \( D \), \( \sigma \)

2. Map \( \sigma \), using \( \alpha \), into the corresponding state \( \hat{\sigma} \) in \( \hat{D} \)
3. Solve \( \hat{D} : \hat{u}^* = \arg \min_{\hat{u}} \mathbb{E}(\hat{\sigma}, \hat{\omega}, \hat{f}(\hat{x}, \hat{\omega})) \)

4. Construct \( D' = (\Sigma, \{\omega | \omega \in R_{\omega} \hat{u}^*\}, \gamma, f, c) \)

5. Solve \( D' : u'^* = \arg \min_{u' \in \Omega} \mathbb{E}(\sigma, \omega, f(\sigma, \omega)) \)

Note the minimization in Step 3 selects the minimum of a set of intervals. This requires an order relation on intervals; three candidates are

a) \([r_1, r_2] \leq [r_3, r_4] \) if \( r_1 \leq r_3 \)
b) \([r_1, r_2] \leq [r_3, r_4] \) if \( r_2 \leq r_3 \)
c) \([r_1, r_2] \leq [r_3, r_4] \) if \( r_2 \leq r_3 \)

That this algorithm arrives at a decision is clear; as suggested by the ambiguity in the definitions of \( \leq \) on intervals, it need not be the optimum. (a) and (c) are total orders on \( \Pi \); (b) is a partial order and therefore Step 3 may not yield a unique solution if it is used. In this case, the set of least elements may replace \( \hat{u}^* \).

Use of (a) or (c) provides a bound on the suboptimality of the ultimate decision \( u'^* \). Using (a), the interval \([r_1, r_2] \) is selected which has the least lower bound. By (3.88), the actual cost achievable by any \( u \) in response to \( \sigma \), where \( u \in R_{\omega} \hat{u} \) and \( u(\sigma) = \hat{\omega} \), lies in \([r_1, r_2] \). Hence the maximum suboptimality suffered by restricting \( D' \) to consider choices related to \( \hat{u}^* \) is \( r_2 - r_1 \).

Using (c), which from the point of view of \( D \) is the min max (best worst-case) ordering, yields the bound \( c \leq r_2^* - \min \{r_1\} \).
where \( e \) is the difference between actual cost and the optimal cost, \( \Gamma^* \) is the upper bound of the interval \( \ominus(\mathcal{A}, \mathcal{A}^\circ(\mathcal{A}, \mathcal{A}^*)) \), and \( \{ \gamma \} \) is the set of lower bounds on intervals \( \ominus(\mathcal{A}, \mathcal{A}, \mathcal{A}^\circ(\mathcal{A}, \mathcal{A})) \) for each \( \mathcal{A} \in \mathcal{A} \).

As pointed out at the start of the section, this notion of abstraction is an exploratory one - its goal is to capture the concept of abstraction formally and raise questions about the issues involved. Thus, at this point, it is unclear whether \( a, b, \) or \( c \) should be used, although \( c \) is most consistent with the philosophy of selecting the conservative (best worst case) option as well as providing a measure of potential suboptimality.

What is clear is that this procedure can be a useful one in aiding practical decision making in a complex environment. The approach of suppressing detail by aggregation is powerful in reducing computation required, but leads to a possible loss of optimality. If optimality is viewed in the larger scene as a tool to aid in the derivation of "reasonable" decisions, then perhaps this loss of optimality in the abstracted case is not a hindrance.

Thus it seems that abstraction may be a valuable tool in making practical decisions. What about its other use - providing a simple view of a decision maker/problem to an outside agent? This agent might be interested in predicting the outcome of the decision, without wanting (or being capable of) delving into the details of \( D \) and solving it completely in order to predict the optimal outcome.

As characterized by Figure 3.12, this process is a way of bypassing the search over \( \mathcal{A} \) to find some limits on \( \gamma \) or \( \Gamma \). Again, if the state \( \mathcal{A} \)
Figure 3.12

Abstraction for Prediction
is known in D, then a model of D, $M_\delta$, can be assumed to be in an initial state which is known. Thus any $\hat{\mathcal{F}}$ represents an aggregate of $\sigma$'s, and again

$$\alpha : \Sigma \rightarrow \hat{\Sigma} \quad |\hat{\Sigma}| \leq |\Sigma|$$

(3.90)

Given $\hat{\mathcal{F}}$, the model would like to predict the outcome, $\gamma$, which will be generated by D assuming it is solved optimally. Since $\alpha$ is an aggregation, $\hat{\gamma}$ will represent a subset of $\gamma$ containing at least the set

$$\hat{\gamma}(\hat{\mathcal{F}}) = \hat{\gamma} \supseteq \{ \gamma = f(\sigma, u^*) \mid \alpha(\sigma) = \hat{\sigma}, u^* = \min_u c(\sigma, u, f(\sigma, u)) \}$$

(3.91)

which is the set of optimal outcomes for each initial $\sigma$ which is mapped to $\delta$ by $\alpha$. By letting this be a proper containment, an $\hat{\gamma}$ may be constructed of special form which is easy to compute or store.

A similar process can be used to predict the cost incurred by a solution to D. Letting

$$\hat{\varepsilon} : \hat{\Sigma} \rightarrow \Pi$$

(3.92)

then

$$\hat{\varepsilon}(\hat{\mathcal{F}}) = [\varepsilon, \bar{\varepsilon}] \supseteq \{ \min_u c(\sigma, u, f(\sigma, u)) \mid \alpha(\sigma) = \hat{\sigma} \}$$

(3.93)

Thus if $\hat{\varepsilon}$ is determined, $M_\varepsilon$ supplies bounds $\varepsilon, \bar{\varepsilon}$ on $\sigma$, the cost incurred by the optimal decision. Again, by letting this interval be larger than absolutely necessary, a $\hat{\varepsilon}$ may be constructable which is easier to compute than $c$.

In summary, the techniques of abstraction can be used to generate $M_\varepsilon$ which can serve as either:

**DEFINITION 3.16:** An outcome predictor of a decision problem D is a quintuple $(\hat{\Sigma}, \hat{\gamma}, \hat{\mathcal{F}}, \alpha, \mathcal{R}_\gamma)$
satisfying (3.90), (3.91) and
\[ \mathcal{R}_\gamma \subseteq \gamma \times \hat{\gamma} \]  

or

**DEFINITION 3.17.** A cost predictor of a
decision problem \( D \) is a triple \((\hat{\mathcal{X}}, \hat{\mathcal{Y}}, \alpha)\)
satisfying (3.90), (3.92), and (3.93)

The principle conclusion of this subsection is that abstraction can be
captured in a formal structure in two ways. The first aids in the solu-
tion of complex problems, but pursuing this line of development would
stray beyond the bounds of the domular problem. The second, constructing
abstracted models to predict outcomes or costs resulting from a decision
maker solving a decision problem, is of great interest in the practical
solution of domular problems. In short, the latter uses aggregated state
information to restrict the set of possible outcomes (or costs) to those
which might result from the optimal decision. Extension of this idea to
a domular framework is the goal of the following two subsections.

3.4.2 **Dynamic Abstraction**

The use of abstraction in a dynamic setting can be approached in two
ways, both of which allow an observer outside of the decision agent to
predict, in some measure, the actions of the agent as reflected in the
system outputs. The first involves solving for the agent's optimal
strategy, then constructing a reduced form model of the optimal closed
loop system. While effective, this necessitates solving the original
decision problem and thus negates the advantages of abstraction as a
tool to ease its solution. The second approach involves constructing
a non-Markov model which allows some prediction of possible outputs (and costs) which may appear when a particular "state" is occupied. By drawing conclusions as to which output sequences may be optimal, or near-optimal, based on this simpler model, the original decision can be made by searching only through the decision (sequences) which can produce one of these candidate outputs (sequences). A principle requirement of such a non-Markov model is that the observer be able to track the state occupied given past system outputs, even if future outputs and costs cannot be predicted perfectly.

DEFINITION 3.18: A dynamic decision problem consists of:

- state space: \( \Sigma \)
- control set: \( \mathcal{U} \)
- output set: \( \gamma \)
- state transition function \( f: \Sigma \times \mathcal{U} \to \Sigma \)
- output function \( h: \Sigma \to \gamma \)
- incremental cost function \( c: \Sigma \times \mathcal{U} \to \mathbb{R} \)

with the objective

\[
\min_{\alpha} \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} c(\sigma(t), u(t)) \tag{3.95}
\]

subject to

\[
\sigma(t+1) = f(\sigma(t), u(t)) \tag{3.96}
\]

The solution to such a problem is a sequence of controls \( u^*(t) \).

With these, the system obeys

\[
\sigma^*(t+1) = f(\sigma^*(t), u^*(t)) \tag{3.97}
\]

\[
y^*(t) = h(\sigma^*(t)) \tag{3.98}
\]
or, in closed loop form, using the control law
\[ y^* (x^* (t)) = u^* (t) \]  \hspace{1cm} (3.99)

(3.97) becomes
\[ x^* (t+1) = f (x^* (t), y^* (x^* (t))) \]  \hspace{1cm} (3.100)

The cost incurred at each stage is
\[ c (x^* (t), u^* (t)) = c (x^* (t), y^* (x^* (t))) \]  \hspace{1cm} (3.101)

Equations (3.98)-(3.101) define an input-less system, which produces outputs \( y^* (t) \) and incurs costs. An abstraction of this system, which no longer involves decision making, can be found using

**Definition 3.19**: A closed loop dynamical abstraction \( D \) is a system with
- state space \( \hat{\Sigma} \)
- output space \( \hat{\gamma} \subseteq 2^\gamma \)
- abstraction function \( \alpha : \Sigma \rightarrow \hat{\Sigma} \)
- output function \( \hat{h} : \hat{\Sigma} \rightarrow \hat{\gamma} \)
- state transition function \( \hat{f} : \hat{\Sigma} \times \gamma \rightarrow \hat{\Sigma} \)
- cost function \( \hat{c} : \hat{\Sigma} \times \gamma \rightarrow \mathbb{II} \)

satisfying
\[ \alpha (\sigma) = \hat{\sigma} \Rightarrow h (\sigma) \in \hat{h} (\hat{\sigma}) \]  \hspace{1cm} (3.102)

\[ \alpha (\sigma) = \hat{\sigma} \text{ and } h (\sigma) = y \text{ implies } \alpha (f (\sigma, y^* (\sigma))) = \hat{f} (\hat{\sigma}, \hat{y}) \]  \hspace{1cm} (3.103)

\[ \alpha (\sigma) = \hat{\sigma} \text{ and } h (\sigma) = y \text{ implies } c (\sigma, y^* (\sigma)) \in \hat{c} (\hat{\sigma}, \hat{y}) \]  \hspace{1cm} (3.104)

Equation (3.102) requires that if \( \sigma \) is abstracted to \( \hat{\sigma} \) in the aggregate state space \( \hat{\Sigma} \), then the output \( y \) resulting from \( \sigma \) is an element of the set of outputs resulting from \( \hat{\sigma} \) through \( \hat{h} (\hat{\sigma}) \). Given
σ and its associated output γ, (3.103) and (3.104) require that the state following σ be aggregated into a ˘σ given by ˘ in applied to ˘σ and γ, and with cost in the interval ˘e(˘σ, γ). The use of the abstraction in prediction is thus:

**ALGORITHM 3.20**: Prediction from a closed loop dynamical abstraction:
1. Given some ˘σ(t), the possible outputs at t are in the set ˘H(˘σ(t)).
2. Given one of those outputs γ, the optimal cost incurred at t lies in the interval ˘e(˘σ, γ), and the next σ(t+1) will be abstracted to ˘σ(t+1) = ˘f(˘σ, γ).

**EXAMPLE 3.21**: Consider the inferior domule of the canonical example. The optimal decision strategy there results in the closed loop system illustrated in Figure 3.13A. Forming an abstraction by aggregating states 2 and 3 results in 3.13B; knowledge that the system is in state [2, 3] is insufficient to predict whether the output will be 1 or 2. However, if it is 2, a cost in the interval [1, 1] will be incurred and the system moves to a state in {4}. Here future outputs of 2 and costs in [0, 0] can be predicted.
Figure 3.13

Closed Loop Abstraction

A. Optimal Closed Loop System

B. Abstraction of A.
Thus a closed loop abstraction allows the observer to use outputs generated by the system to help determine the state to which the abstracted model moves, then uses this state to determine a set of possible future outputs and costs. An interesting philosophical point is that the observations provide information to the observer even though the system is completely deterministic; a simplified model requires such information to supplement the knowledge built into it. This illustrates that modelled knowledge and communicated information are interrelated (and interchangeable).

However, to avoid the requirement that $D$ be solved for $\gamma^*$ before $\hat{D}$ can be constructed, another approach must be used. $\hat{D}$ may be constructed directly from $D$, by taking into account all of the structure of $D$. The idea is to construct $\hat{D}$ to predict not just the optimal outputs, state transitions, and costs resulting from $D$'s actions, but any possible such quantities.

**DEFINITION 3.22:** An open loop dynamical abstraction is similar to a closed loop abstraction but with \((3.103)\)-(\(3.104\)) replaced by:

\[
\alpha(\sigma) = \hat{c} \text{ and } \eta(\sigma) \in \hat{\gamma} \Rightarrow \alpha(f(\sigma, u)) = \hat{c}(\hat{c}, \hat{\gamma}) \tag{3.105}
\]

\[
\alpha(\sigma) = \hat{c} \text{ and } \eta(\sigma) \in \hat{\gamma} \Rightarrow c(\sigma, u) \in \hat{c}(\hat{c}, \hat{\gamma}) \tag{3.106}
\]

for all $\sigma, u$.

Naturally, for any given state space aggregation $\alpha$, the open loop abstraction will exhibit less usefulness in prediction because of the additional state transitions, and associated costs, which must be
considered. However, the lack of a need to first find $\mathcal{Y}$ can be very useful in eventually solving $D$ as well as predicting its future.

EXAMPLE 3.23: Again consider the example inferior domain, and the abstraction illustrated in Figure 3.14. With this extremely simple model, next-output prediction is impossible; therefore so is next-state prediction. However, it can be used to prove that the optimal strategy traps the system in class $\{1, 4\}$. Referring back to the original dynamical model, this actually specifies $\mathcal{Y}(\sigma^i)$ and $\mathcal{Y}(\sigma^y)$.

Thus the abstraction can be used to reduce the set of possible optimal $\mathcal{Y}$'s, although prediction is not feasible. This is a simple example of an obviously potentially powerful technique.

Between the poles of open- and closed-loop discussed above lies a spectrum of abstraction techniques. If local analysis of $D$ can show, for example, that $\mathcal{U}^i$ is definitely not the optimal control to be applied in state $\sigma^j$, then the conditions (3.105)-(3.106) can be relaxed and not required to hold for $\sigma^j$ and $\mathcal{U}^i$. That is, since

$$\mathcal{Y}^i(\sigma^j) \neq \mathcal{U}^i$$

one need not consider the consequences of applying $\mathcal{U}^i$ when in $\sigma^j$ when forming the abstracted problem.

This is a particularly interesting observation in light of the discussion of forward search in Section 3.3.1. While the search may have to be carried far into the future to determine the sole optimal control,
Figure 3.14
Open Loop Abstraction

\[ \frac{1}{[0,0]} \quad \frac{2}{[\infty, \infty]} \quad \frac{1}{[0,0]} \quad \frac{2}{[1,1]} \]

\[ \frac{2}{[0,0]} \quad \frac{1}{[\infty, \infty]} \quad \frac{2}{[1,1]} \]
many candidates may be eliminated with a search of much shorter horizon. Thus, while the long-term search may be prohibitively expensive to perform, a short horizon search may be used to weed out some possible options for the control, then an abstraction constructed, which need not consider the eliminated options. Thus a sort of local (in time) examination of the original problem, followed by an abstraction to capture the longer horizon but with a smaller state space, can be combined in an approach to deriving an optimal, or near optimal, input.

The concept of abstraction in a dynamic setting, and other ideas which spring from it, are just beginning to be explored. This section has shown how dynamical abstractions can be constructed, and illustrated how they can be used both in prediction and as a step in determining a control. Based on this, the next section will sketch how abstraction fits into the domular framework.

3.4.3 Domular Abstraction

The extension of the previous ideas to domular systems is guided by the fact that the observer external to the decision maker postulated above will here be the agent of another domule. As shown in Chapter 4, that agent will be particularly interested in either

a) the interactions which might be generated by the first, and the cost associated with their production, which affect it, or

b) the cost which the first would have to incur in order to respond optimally to the interactions generated by the latter.

This section will apply the general concepts of dynamical abstraction developed above to satisfying these two interests.
Point (a) above arises when a superior domule needs to predict the interactions which may be generated by an inferior one in the future. There are two distinct questions which can be posed:

1. What interactions might be generated?

2. What interactions can be generated?

The answer to the second is a subset of the answer to the first, which gives the superior foresight into future interactions when the inferior cannot communicate them directly. The latter forms the basis for the superior deriving orders to the inferior as to what interactions are desired - and certainly the requested interaction must be feasible. Both will be explained in Section 4.1.6 into detailed coordination mechanisms.

As may be suspected from the use of the inferior canonical domule as an example of dynamic abstraction in the last section, the generation of abstractions of an inferior for a superior is exactly the same as general dynamical abstraction. Because \( \hat{\mathcal{G}}(x, \gamma) \) is an interval, indicating the range of costs which may be incurred in producing an output in \( \hat{\gamma} \) when in \( \hat{\mathcal{G}} \), it answers both questions (1) and (2). The minimum of the interval indicates the lowest possible such cost; thus a finite minimum indicates some \( \hat{\gamma} \) is possible. The maximum is the worst-case cost; if it is infinite, then there is some \( \gamma \) in \( \hat{\gamma} \) which may not be producible when \( D \) occupies a state in \( \hat{x} \).

EXAMPLE 3.24. Refer to the abstraction of Figure 3.14. If the inferior occupies a state in \( \{1,4\} \), and is producing output 2, then the abstraction shows that the next state will lie in \( \{1,4\} \) also (as the transition to \( \{2,3\} \) is
infinitely expensive). Then, either interaction 1 or 2 may be produced; 1 with cost no greater than 0, 2 with cost either in \([0, 0]\) or \([\infty, \infty]\).

Thus 1 or 2 might be produced; only 1 can be guaranteed as feasible with finite cost.

Thus the abstraction of the inferior can be used to supply the superior with knowledge of which interactions the inferior may generate and the bounds on the costs incurred given a particular interaction is generated. Each will be used in 4.1.6 to help coordinate, although through two different mechanisms. Each mechanism must also be able to use a model of the superior supplied to the inferior - and this brings us back to point (b) posed at the beginning of this section.

As the superior generates no interactions of interest to the inferior, the model of the superior would be expected to give only cost information, as did the cost predictor in the static case. The interactions from the inferior act as control inputs to the superior; the abstraction should provide bounds on the cost incurred given the superior is in some abstract state \(\hat{\Sigma}\) and the inferior generates some interaction \(\gamma\).

**DEFINITION 3.25:** An abstraction of a superior domain consists of

- state space \(\hat{\Sigma}\)
- input space \(\mathcal{P}_j\)
- state transition function \(\hat{f}\): \(\hat{\Sigma} \times \mathcal{P}_j \rightarrow \hat{\Sigma}\)
- cost function \(\hat{c}\): \(\hat{\Sigma} \times \mathcal{P}_j \rightarrow \mathbb{I}\)
- abstraction relation \(\hat{\mathcal{R}}_k \subseteq \Sigma \times \hat{\Sigma}\)

satisfying
for each $\sigma \mathcal{R}_x \hat{\sigma}$, for each $u$, for each $\rho_j:

\hat{f}(\sigma, \rho_j, u) \mathcal{R}_x (\sigma, \rho_j)

(3.107)

c(\sigma, \rho_j, u) \mathcal{C}(\hat{\sigma}, \rho_j)

(3.108)

This definition uses an abstraction relation $\mathcal{R}_x$ as there may not exist a nontrivial abstraction function $\alpha$ satisfying (3.107)-(3.108). Each $\hat{\sigma}$ represents a set of $\sigma$'s, and these sets are not mutually exclusive. (3.107) requires that all $\sigma$'s which may result from interactions $\rho_j$: received while in states in $\hat{\sigma}$ are indeed in $\hat{f}(\sigma, \rho_j)$. (3.108) plays the same role for costs; ideally, $c(\sigma, \rho_j)$ should be the smallest interval satisfying (3.108).

EXAMPLE 3.26: Consider the superior of the canonical example of 3.1. Note that repeated interactions of 1 drive the state to either 3 or 4. This suggests the abstraction of Figure 3.15. Application of interaction 2 can lead to any state from states 3 or 4, so it acts as a reset to class $\{1, 2, 3, 4\}$. Sequential productions of interaction 1 lead successively to classes $\{2, 3, 4\}$ and $\{3, 4\}$, trapping in the latter and incurring constant cost of 1 per step. All other transitions cost something in $[0, 1]$, depending on the exact state and control.

An interesting point illustrated by this is that a nontrivial abstraction exists only if the interactions can drive the superior into a
Figure 3.15

Abstraction of Superior
proper subset of the state space regardless of controls applied by it.
If the superior's controls are sufficiently powerful to allow it to reach
any state from any other for all interaction sequences, then the only
possible abstraction of it is the one with
\[
\hat{\Sigma} = \{ \Sigma \}
\]  
(3.109)

This severely limits the applicability of abstractions of superiors; unless
the superior is not completely reachable, no abstraction can capture a
partial model of its behavior. Thus the coordination algorithms which
use abstraction will be most effective when only those of inferiors are
used.

This concludes the discussion of abstraction in static, dynamic, and
domular settings. It is a discussion of preliminary concepts - those of

1. Solving a decision problem and building a simplified model
   of the resulting closed loop system.
2. The more powerful approach of simplifying a problem,
   solving it, and using the solution to guide the search
   for a solution to the original problem.
3. Predicting the outputs of an optimally controlled system
   without completely duplicating the optimization process.
4. Predicting (reducing uncertainty about) interactions
   produced by an inferior domule.
5. Computing bounds on the cost to produce interaction
   sequences by an inferior.
6. Computing bounds on the cost to respond to interactions
   by a superior, provided it is not completely reachable.

Much more work needs to be done in terms of solidifying this concept:
finding special classes of problems for which it is particularly appro-
riate, examining stochastic applications to mix the probabilistic and
uncertain aspects of modelling, and tightening up the formal framework
in which abstraction can be embedded.
Chapter 4. Global Coordination

This chapter will blend the separate concepts of domular structure, forward search, and abstraction into a series of unified strategies for coordinating the control of a complex system. By allowing various types of communication between interacting agents, several points on the trade-off curve of agent complexity versus optimality can be examined.

The global problem explored here is that of a team, where the objective is to minimize the sum (over time and domules) of the local cost functions. The first section will discuss the team problem for a set of system structures particularly well suited to coordination - those which are only singly connected. The second section will discuss the extension of the procedures presented in this context to more general settings, both in terms of objectives and system topology.

Throughout this chapter run three unifying ideas.

1. Coordination is done by selecting predictions of interactions which yield good performance, then solving the resulting independent local problems to derive local control inputs,

2. Working forward in time only as far as is necessary or feasible, and

3. Using uncertainty to gain local autonomy for agents when complexity precludes perfect coordination.

These reflect the design considerations discussed in Section 2.2.

Finally, while most discussion in this chapter will be in the context of deterministic problems, the considerations of Section 3.2.2, suggest straightforward ways to generalize the results to stochastic situations.
Comments relevant to this generalization will be made at the end of each subsection.

4.1. Singly Connected Systems

By restricting attention to a subset of possible system topologies, many issues can be addressed in a fairly simple context. Singly connected systems have a nice property in that the effects of one agent's decisions can affect those of another through only one series of agents. This allows the coordination process to proceed in an orderly manner, with no fear of deadlocks or endless iteration.

This section will begin by defining the coordination problem, then proceed to strategies which make use of successively more communications capability. The first of these is that of static, or one-step, problems, where the future does not affect present decisions. Introducing dynamics to this greatly increases the difficulty of the problem, so a decision strategy is first developed for the case of no interagent communication. This is first extended by allowing agents to predict interactions generated by it to others, then by allowing full coordination but for only a finite future. Then, by excluding online communication but allowing a priori negotiation, abstraction can be brought to bear to give each agent some information on what the rest of the system is doing in order to improve its own decisions. Finally, by combining full coordination over a finite horizon with abstraction to reduce long-term uncertainty, a hybrid coordination mechanism can be developed. The concluding section discusses some implementation considerations of this mechanism, particularly in terms of systems with dynamics that slowly change over time.
4.1.1. Introduction

Before describing coordination strategies, the problem formulation of Section 1.2 will be specialized to the case at hand. This involves:

1. Specifying the relation of the local costs to the global objective.

2. Specifying the class of topologies of interest.

3. Introducing an ordering on agents to serve as the skeleton of some of the strategies.

The classic team objective is to

\[
\min_{u_i(t)} \lim_{T \to \infty} \frac{1}{T} \sum_{i=1}^{N} \sum_{t=0}^{T-1} c_i(t, c_i(t+1), \ldots, u_i(t)) \quad (4.1)
\]

where \( c(\cdot, \cdot, \ldots, \cdot, \cdot) \) is the cost incurred within the \( i \)th model at time \( t \) when the various other arguments occur. The key points of this objective are that it is additively separable in time and space, optimizes long term performance, and requires that decisions be made to achieve a single goal. The dynamics of the models relate the costs at one time to those at another.

In the domain setting, this has one drawback. As the individual agents must make decisions based on incomplete knowledge of the system structure, each will often face uncertainty as to some effects of its decisions. Since the principle of communication in terms of interaction variables prohibits assigning one agent to evaluate (4.1) as time proceeds, this global objective must be modified to allow for this uncertainty. Whenever an agent faces a decision where complexity considerations prevent it from evaluating the effects of its decisions completely, it will be assumed that it will make a decision using
\[
\min_{u_i(t)} \max_{T} \lim_{T \to \infty} \sum_{i=1}^{N} \sum_{t=0}^{T-1} c_i(t) \quad \text{(4.2)}
\]

where \( T \) is a set of uncertainties reflecting the agent's lack of knowledge of the complete system structure. The exact set \( T \) will depend on what is being communicated to the agent; see Section 4.1.3 for a simple example.

Using uncertainty to allow incomplete knowledge of the system model can be expected to lead to a conservative decision strategy as each agent plans for the worst case, which may never happen. The cost predicted by a strategy using uncertainty will be an upper bound on the cost actually incurred, which in turn will be greater than the global optimum due to incomplete coordination.

In the case of stochastic problems note that the uncertainty will be expressed in terms of uncertain probability measures on various sets. An agent will be able, in general, to relate its actions to system cost via an uncertain probability function. Thus stochastic coordination will take a form using both probabilistic and uncertain modelling.

In short, the objective faced by each agent is to minimize the worst case total cost of system operation, using its own local model and whatever outside information it receives.

The class of topologies to be considered is that where each agent is "connected" to any other in exactly one way. Recall that the system structure is described by a graph \( G \) with node set \( \mathbb{N} \) and interaction paths \( \mathbb{L} \).

**DEFINITION 4.1:** \( G \) is **singly connected** if,

for any \( n_1, n_k \) distinct nodes in \( \mathbb{N} \), then
there is exactly one ordered subset of
$\mathbb{N} \times (n_1, \ldots, n_{k-1}, n_k)$ such that
1. Either $(n_i, n_{i+1})$ or $(n_{i+1}, n_i)$
is in $L$, and
2. $n_i \neq n_j$ for any $i, j$.

This requires that there is exactly one path from each $n_i$ to another $n_k$,
where the path can follow interaction effects in either direction. Figure 4.1 illustrated such a graph, as well as several non-singly connected
topologies.

Given a particular singly connected system, the domules comprising
it can be partitioned into three sets.

DEFINITION 4.2: A domule $n$ is a
superior domule if there is no other
$n'$ such that $(n, n')$ is an edge of $G$.

DEFINITION 4.3: A domule $n$ is an inferior
domule if there is no other $n'$ such that
$(n', n)$ is in $G$.

DEFINITION 4.4: A domule $n$ is a middle
domule if it is neither superior nor
inferior.

For example, superior, inferior, and middle domules are labelled S, I,
and M respectively in the system of Figure 4.1A. These classes are of
interest as the algorithm an agent uses will depend on its relation to
the rest of the system as captured by the above definitions.
Figure 4.1

Topologies

A. Singly Connected

B. Not Singly Connected
A final introductory concept is that of constructing a number of orderings on any singly-connected graph which serve as a pattern for induction on that graph.

**DEFINITION 4.5:** An induction relation \( H \)

on the nodes of a singly connected graph is one satisfying:

1. If \((n_i, n_j)\) is in \( G \), then either \((n_i, n_j)\) or \((n_j, n_i)\) is in \( H \), but never both.
2. For any \( n_i \), there is at most one \( n_j \) such that \((n_i, n_j)\) is in \( H \).

\( H \) provides a way of working through \( G \) in such a way that each link is traversed once, and the process terminates at some node. It is an immediate consequence of property 2 that there is exactly one \( n_i \) such that no \( n_j \) exists for which \((n_i, n_j)\) is in \( H \).

**DEFINITION 4.6:** A node \( n_i \) precedes \( n_j \) if \((n_i, n_j)\) is in \( H \); it succeeds \( n_j \) if \((n_j, n_i)\) in in \( H \). A node with no predecessor is an initial node; the one with no successor is the terminal node.

Figure 4.2 illustrates these definitions.

As will be seen in the next section, \( H \) determines the "flow" of the coordination process. As there are \( N \) possible \( H \)'s (one for each terminal node), this will imply that there are \( N \) possible "flows". Section 4.2 will mention how this freedom can be used to allow more general global objectives.
Figure 4.2

Induction Relation

\[ \text{Initial} \rightarrow \text{G} \rightarrow \text{H} \]

\[ \text{Terminal} \]
With the form of the global objective, and the above terminology, established, the first and most simple coordination mechanism can be investigated.

4.1.2. Static Teams

The one-step problem about to be posed and solved is a modification of the type of problem solved by spatial dynamic programming [36].

It is included here as an illustration of the use of $\mathcal{H}$ to guide the coordination process, so that agents are inductively included in it until the entire system is coordinated.

The salient points of this problem are:

1. The formulation permits interaction effects.

2. Arbitrary communication is allowed between interacting agents.

3. $\mathcal{H}$ structures the coordination process.

The formulation will not assume each agent is activated at time zero and must choose $u_i(o)$ to

$$\min_{u_i(o)} \sum_i c_i(\sigma_i(o), \sigma_i(o), \rho_{i1}(o), \ldots, \rho_{in}(o), u_i(o))$$  \hspace{1cm} (4.3)

as this breaks the problem into $N$ independent, local decision problems.

$$\min_{u_i(o)} c_i(\sigma_i(o), \sigma_i(o), \rho_{i1}(o), \ldots, \rho_{in}(o), u_i(o))$$  \hspace{1cm} (4.4)

since the interactions $\rho_{i}(o)$ are fixed. Rather, each agent will make a decision at time $t_i$, where

$$1. \text{ If } (\pi_i, \pi_j) \text{ is in } \Theta, \text{ then } t_j = t_i + 1$$

$$2. \text{ For some } \pi_i, t_i = 0$$  \hspace{1cm} (4.5)

Thus if $\pi_i$ produces interactions with $\pi_j$, $\pi_i$ will apply its control
first so that the resulting interactions will affect $\eta_j$. This necessitates cooperation between agents $i$ and $j$ to trade off local costs.

(The existence of a set of $t_i$ satisfying (4.5) results from the singly connected property. The agent $i$ where $t_i = 0$ is the first agent to make a decision; its decision time is set to 0 without loss of generality. Such an agent must be in an interior domule of $G$.)

EXAMPLE 4.7: Consider the topology of Figure 4.3A. $D_6$ is the most inferior domule; hence $t_6 = 0$. At this time, $D_6$ must apply $u_6(0)$ which will drive it to $q_6(i)$, generating $\rho_{6,4}(i)$ to influence $D_4$. $D_4$, in turn must apply $u_4(i)$, etc. The process is diagrammed in Figure 4.3B.

It will be assumed that all agents know the initial state of their local model at time zero. The coordination will thus occur just before the first control is applied to the system.

Formally, the one-step decision problem is to

$$\min_{u_i(t_i)} \sum_{i=1}^{N} c_i(s_i(t_i), s_i(t_i+1), \rho_i(t_i), \ldots, u_i(t_i))$$

subject to

$$\sigma_i(t_i+1) = f_i(s_i(t_i), \rho_i(t_i), \ldots, u_i(t_i))$$

$$\rho_{ij}(t_j) = g_{ij}(s_i(t_i))$$

where (4.8) interconnects the domules so that coordination is necessary.

The coordination strategy is one of iterating through the system structure, in a way determined by the induction relation $\mathcal{H}$, successively
Figure 4.3

One-Step Decisions

A. Topology

B. Decision structure
computing a function on the \( R_j \) which reflects the minimum cost incurred by a branch of the system in either producing or responding to a particular interaction \( \rho_j(t_j) \). The role of an agent depends on whether it is in an initial module, where it must initiate this computation, a terminal module, where it terminates, or neither.

**Algorithm 4.8: Static Team Coordination**

(Single Pass)

**Case I: Initial Modules**

**Subcase A: Superior Modules**

Each such agent \( i \) can compute

\[
q_j^* (\rho_j) \equiv \min_{u_i} c_i(\sigma_i(t_i), \sigma_i(t_i+1), \rho_j, \ldots, u_i(t_i)) \quad (4.9)
\]

which is the value of the minimum cost response to \( \rho_j \) when \( D_i \) is in \( \sigma_i(t_i) \).

This function is then passed to \( D_j \), the module which affects it (since \( D_i \) is initial and superior, there is only one such \( D_j \)).

At some later point, \( D_j \) will pass an optimal \( \rho_j^* \) to \( D_i \), which then chooses

\[
u_i^* (t_i) = \arg\min_{u_i} c_i(\sigma_i(t_i), \sigma_i(t_i+1), \rho_j^*, u_i) \quad (4.10)\]

**Subcase B: Inferior Module**

The dual of a superior module, an inferior initial agent \( i \) can find

\[
q_j^* (\rho_j) \equiv \min_{u_i} \left[ c_i(\sigma_i(t_i), \ldots, u_i) \right]_{q_j^* (\sigma_i(t_i+1)) = \rho_j} \quad (4.11)
\]

which is the value of the minimum cost.
incurred when in initial state $\pi_i(t_i)$ to
produce interaction $\rho; j$ to $D_j$, the successor
to $D_i$ in $\pi_i$. After passing this function
to $D_j$, $D_i$ will receive a $\rho; j$ to be produced
and apply

$$u_i^*(t_i) = \arg\min_{u_i} \{ c_i(\pi_i(t_i),...,u_i)|g_j(\pi_j(t_j),u_j)) = (4.12) \rho; j \}
$$

Since initial domules are always either in-
erior or superior, this concludes Case I.

Case II: Non-initial and Nonterminal Domules

Subcase A: The successor to $D_i$, say $D_j$,
affects $D_i$ ($\pi_i, \pi_j$) is in $\pi_i$, while $\pi_j, \pi_i$ is in $\pi_i$. As with superior initial domules,
$D_i$ here will compute the minimum cost to
respond to a $\rho; j$ from the successor. After
it receives $s_i(u; \rho; u)$ from all domules
which it affects, and $s_i(u; \rho; u)$ from those
other than $D_j$ which affect it, it can find
the cost incurred in the part of the system
preceeding it (in $\pi_i$). For a particular
choice of $\rho; k$ and $\rho; u$, this cost is

$$d_i(\rho; 1,...,\rho; i, \rho; k,...,\rho; u, u_i) = \sum_k s_i(u; \rho; u)$$

$$+ \sum_j s_i(\rho; j) + c_i(\pi_i(t_i),...,u_i)$$

provided

$$g_i(u; (\pi_i(t_i), u)) = \rho; k \quad k \text{ preceding } i$$

(4.14)
The first term in (4.13) is the cost of responding to the $\rho_{ik}$ by predecessors of $D_i$ affected by it; the second, the cost of producing the $\rho_{ik}$ that affect it, and the third the local cost incurred by $D_i$ itself. The minimum cost incurred by $D_i$ and all its predecessors in responding to $\rho_{ij}$ is then

$$S_{ij}(\rho_{ij}) = \min_{\rho_{ik}} \min_{\rho_{il}} \min_{d_i} d_i(\rho_{il}, \ldots, \rho_{in}, u_i) \quad \text{subject to (4.14)}$$

(4.15)

This can be sent to $D_j$.

$D_j$ will respond with some $\rho_{jk}$ as a prediction of the optimal interaction to $D$ which it will produce. $D_i$ can find

$$u_i^*(k) = \arg\min_{u_i} \min_{\rho_{ik}} \min_{d_i} d_i(\rho_{ik}, \ldots, \rho_{in}) \quad \text{subject to (4.14)}$$

(4.16)

as well as the $\rho_{ik}, l\neq j$ and $\rho_{in}$ which also minimize (4.16). These latter predictions of optimal desired and produced interactions can be sent to the predecessors of $D_i$.

Subcase B. The successor to $D_i$ is affected by $d_i$ ($n_i, n_j$ is in both $\mathcal{G}$ and $\mathcal{H}$).

This case is similar to the above except that $S_{ij}(\rho_{ij})$ is transmitted to $D_i$'s successor $D_j$. 
as the minimum cost to produce $\rho_{ij}$.

Given all interaction functions $s_{ki}(\rho_{ki})$ and $s_{l;j}(\rho_{l;j})$, the cost to produce a $\rho_{ij}$ is

$$s_{ij}(\rho_{ij}) = \min_{\rho_{ki}} \min_{\rho_{l;j}} \sum_{k=1}^{N} s_{ki}(\rho_{ki}) + \sum_{l \neq j}^{N} s_{l;j}(\rho_{l;j}) + c_i(\sigma(t_i), \ldots, u_i)$$  \hspace{1cm} (4.17)

subject to

$$\sigma_{l;i}(\sigma(t_i + 1)) = \rho_{l;j} \hspace{1cm} l \neq j$$  \hspace{1cm} (4.18)

This can be sent to $D_j$.

When $D$ returns $\rho_{ij}^*$, the optimal control $u_i^*(t_i)$ and interactions desired and predicted, $\rho_{ki}^*$ and $\rho_{l;j}^*$, can be found as the minimizing arguments to (4.16) with $\rho_{ij}^*$ inserted for the value of $\rho_{ij}$.

Case III. Terminal Domule

The terminal domule $i$ will receive $s_{ij}(\rho_{ij})$ from all domules it affects, indicating their minimum cost to respond to $\rho_{ij}$, and $s_{ji}(\rho_{ji})$ from all that affect it, conveying their costs of producing $\rho_{ji}$.

It can find

$$d_i(\rho_{i;j}, \ldots, \rho_{N;i}, \rho_{i;\ldots}, \rho_{in;i}, u_i) = \sum_{j=1}^{N} s_{ij}(\rho_{ij})$$

$$+ \sum_{j=1}^{N} s_{ji}(\rho_{ji}) + c_i(\sigma(t_i), \ldots, u_i)$$ \hspace{1cm} (4.19)
The optimal total system cost is then,

$$\min_{\mu} \min_{\beta;\rho} \min_{\rho_{\mu}} \mathcal{J}_i(\rho_{\mu}, \ldots, \rho_{\mu}, \mu, \beta;\rho)$$  \hspace{1cm} (4.20)

subject to

$$\mathcal{G}_i(\mu, \beta;\rho) = \rho_{\mu}$$

with $\mu, \rho_{\mu},$ and $\rho_{\mu}$ the arguments

that minimize (4.19). The $\rho_{\mu}$ and $\rho_{\mu}$ can be passed to the appropriate agents.

This is a distributed algorithm for achieving optimal coordination of the agent's activities. Figure 4.4 shows the communicated quantities for the example of Figure 4.3. Note that the coordination process occurs in two phases - first, the optimal cost to produce (or respond to) an interaction is inductively computed through the ordering of $\mathcal{A}$. The terminal module computes the optimal total system cost, and starts the second phase of finding optimal interaction predictions. For this, the inductive order is reversed as each agent uses the functions from the first pass to determine its own control and other interaction predictions.

**PROPOSITION 4.9:** The controls computed by algorithm 4.3 are indeed optimal.

**Proof:** The proof of optimality lies in showing that the overall objective (4.6) is equivalent to the decomposed objectives (4.9), (4.11), (4.15), (4.17), and (4.20).

This is purely algebraic, done by inserting each of the decomposed objectives, and system equations (4.17),
Static Coordination: Communication

Figure 4.4

\[ u^*_1 \downarrow s_{21}(\rho_{21}) \quad s_{22}(\rho_{22}) \quad s_{23}(\rho_{23}) \quad s_{24}(\rho_{24}) \]

\[ u^*_2 \downarrow s_{31}(\rho_{31}) \quad s_{32}(\rho_{32}) \quad s_{33}(\rho_{33}) \]

\[ u^*_3 \downarrow \]

\[ u^*_4 \downarrow \]

\[ u^*_5 \downarrow \]

\[ u^*_6 \downarrow \]

\[ \rightarrow \text{Phase I} \]

\[ \rightarrow \rightarrow \rightarrow \text{Phase II} \]
iteratively into (4.20) and removing appropriate terms from the scope of
minimizations.

In stochastic one-step team problems, the same algorithm applies
with modifications like those mentioned in 3.2.2. Rather than defining
the cost-to-produce or cost-to-respond functions on the set of interaction
variables, they can be defined on the set of probability densities on those
sets. Control selection can be replaced by control law selection (choose
$Y^*$ mapping observed interactions into controls) and the stochastic,
perfect state information case can be handled as above.

Also, the Algorithm 4.8 can be extended to handle open-loop dynamic
problems by reformulating the one-step problem slightly. By replacing
consideration of states, controls, and interactions with state sequences,
control sequences, and interaction sequences over a time horizon $[0,T]$,
and solving for open-loop optimal decisions, the problem structure remains
the same. The $s_i(\cdot)$ must be defined on $P_{ij}^T$, rather than the $P_{ij}$, and
each of the minimizations modified to be summed over time and to include
local model dynamics (use of interaction sequences obviates the need for
other model's dynamics). This produces what is essentially the spatial
dynamic programming algorithm [36], which suffers from the problem that
long time horizons lead to high communication requirements (exponential
in $T$). In order to handle long time horizons, closed-loop feedback, or
stochastics, other techniques are needed. These will be introduced in
Sections 4.3 and 4.6; meanwhile, a modified version of this approach to
handling dynamics will appear in Section 4.5.
Static team problems can be solved in an optimal, distributed fashion. The structure of the solution algorithm rests on the induction order $|H|$, which provides a sequence for passing information on costs of producing and responding to interactions so that coordination can occur. The next section will introduce a way of coordinating dynamical systems using uncertainty to overcome a lack of communication capability.

4.1.3. Silent Coordination

As suggested by the title, this section addresses the question of making decisions in an environment where agents cannot communicate at all. Each must seek to minimize its local contribution to total system cost, with no way of anticipating other agent's decisions or gauge its own effect on others. Because of the uncertainty as to interactions affecting it, each agent must adopt the min-max philosophy of finding the best worst-case control.

By prohibiting all communication, there is no way for two agents to set up a protocol for signaling through system dynamics. Derivation of such a signaling strategy requires knowledge of the sender's dynamics at the receiver. While such strategies can improve overall system performance (and in fact are forced when decentralized control is viewed as a constrained global optimization problem), they are difficult to derive and often exhibit undesirable properties (such as high sensitivity to system parameter variations or modelling error). Replacing the centralized derivation of decentralized strategies with a distributed derivation using uncertainty serves to decouple the agents and give them local autonomy. In this extreme case of total decoupling, performance will necessarily be no better than with the optimal signalling strategy; later sections will introduce communication to limit uncertainty and improve the
effectiveness of the coordination.

Reflecting these considerations, the silent coordination problem
is: find \( u_i(t) \) given \( \sigma_i(t) \) and all \( \rho_ji(t) \) (from the perfect information assumption) to

\[
\min_{u_i} \max_{\rho_ji} \min_{\sigma_i} \lim_{t \to \infty} \sum_{\tau=t}^{t+T-1} c_i(\sigma_i(\tau), \sigma_i(\tau+1), \rho_ji(\tau), \ldots, u_i(\tau)) \tag{4.21}
\]

subject to local model dynamics. Note that the max min over \( \rho_ji \)'s and \( \sigma_i \) must be interlaced properly to reflect the fact that each choice of \( u_i(t) \) is made with knowledge of the current and past values of all \( \rho_ji \)'s available.

The solution to this problem is a modified version of dynamic programming as used in dynamic game theory. Forward search is not needed as the problem is time invariant; the inclusion of the maximization over \( \rho_ji \) completely decouples each local problem. Let \( \overline{V}_i(\sigma_i, t) \) be the best guaranteed cost-to-go when the local system is in state \( \sigma_i \) at time \( t \) just before knowledge of the \( \rho_ji(t) \) is obtained. Then, at time \( t-1 \), knowing \( \rho_ji(t-1) \), \( D_i \) will seek to

\[
\min_{u_i} \left\{ c_i(\sigma_i(t-1), \sigma_i(t), \rho_ji(t-1), \ldots, u_i) + \overline{V}_i(\sigma_i(t), t) \right\} \tag{4.22}
\]

for the current state \( \sigma_i(t-1) \). Assuming all neighbors act antagonistically the \( \rho_ji(t-1) \) which actually appear are those which

\[
\max_{\rho_ji(t-1)} \min_{u_i} \left\{ c_i(\sigma_i(t-1), \ldots, u_i) + \overline{V}_i(\sigma_i(t), t) \right\} \tag{4.23}
\]

giving

\[
\overline{V}_i(\sigma_i, t-1) = \max_{\rho_ji} \min_{u_i} \left\{ c_i(\sigma_i(t-1), \ldots, u_i) + \overline{V}_i(\sigma_i(t), t) \right\} \tag{4.24}
\]
This is the backwards dynamic programming algorithm for solving a dynamic game; it approaches a form

$$\bar{V}_i(\sigma_i, t) = \bar{V}_i(\sigma_i) + \bar{g}_i(t)$$  \hspace{1cm} (4.25)

if the state space is reachable. In the deterministic case, this form is found by averaging $\bar{V}_i(\sigma_i, t)$ over the period of its cyclical behavior; $\bar{g}_i$ is the average cost over that period. In the stochastic case, asymptotic convergence is assured when the system is ergodic for all possible control strategies.

The antagonistic assumption gives $\bar{V}_i(\sigma_i, t)$ as an upper bound to the actual cost-to-go when in $\sigma_i$ at time $t$. A lower bound can be found assuming that neighbors are perfectly cooperative; that they will produce just those $\rho_j$ needed to drive $D_1$ optimally.

$$\underline{V}_i(\sigma_i, t) = \min_{\rho_j} \min_{u_i} \{ c_i(\sigma_i(t-1), \ldots, u_i) + \bar{V}_i(\sigma_i(t), t) \} \quad (4.26)$$

computes this lower bound; it converges to the form

$$\underline{V}_i(\sigma_i, t) = \bar{V}_i(\sigma_i) + \bar{g}_i(t) \quad (4.27)$$

The value of equations (4.25) and (4.27) are twofold: they provide bounds on the actual average local cost incurred

$$\bar{g}_i \geq \lim_{T \to \infty} \frac{1}{T-1} \sum_{t=0}^{T-1} c_i(\sigma_i(t), \ldots, u_i(t)) \geq \bar{g}_i \quad (4.28)$$

and a decision strategy

$$u_i^*(t) = \arg \min_{u_i} \{ c_i(\sigma_i(t), \ldots, u_i(t)) + \bar{V}_i(\sigma_i(t+1)) \} \quad (4.29)$$

The tightness of the bounds, $\bar{g}_i - \bar{g}_i$, gives an indication of the need for communication to improve the local performance of $D_1$; the decision strategy
can be implemented either by constructing the decision function mapping state and interaction variables into controls or by precomputing $\mathcal{V}_i(\sigma_i)$. The latter is more amenable to inclusion in more complex strategies; see Section 4.1.5.

This discussion leads to

**ALGORITHM 4.10: Independent Control**

For each domain:

- Compute $\mathcal{V}_i(\sigma_i,t)$ using (4.24) and (4.25)
- Choose $u_i(t)$ using (4.29)
- To evaluate the local effectiveness of this strategy, compute $\overline{\gamma}_i - \overline{\gamma}_i$ using (4.24)-(4.27)

**EXAMPLE 4.11:** Consider the supremal domain in the example of Section 3.1. The worst case interactions are those where $\rho_i \neq \rho'_i$, forcing the supremal to trap in expensive states 3 and 4. The best case ones are those such as 1, 1, 2, 1, 1, 2 ..., giving a state sequence 1, 2, 3, 1, 2, 3, ... The computation of $\mathcal{V}_i(\sigma_i,t)$ and $\gamma_i(\sigma_i,t)$ are summarized in Table 4.5. The former has period 1; the latter, period 3. Since each one increases in cost by one unit in each period, the actual average cost lies in $[\mathcal{L}_i, \mathcal{L}_i]$. The silent decision strategy for $D_1$ is found
Table 4.5

Independent Control Algorithm for Example Supremal

<table>
<thead>
<tr>
<th>$v_i(\sigma_i, t)$</th>
<th>(t)</th>
<th>T-5</th>
<th>T-4</th>
<th>T-3</th>
<th>T-2</th>
<th>T-1</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma_1)</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\(\sigma_1\) one period

<table>
<thead>
<tr>
<th>$v_i(\sigma_i, t)$</th>
<th>(t)</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma_1)</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Stationary bounds on cost-to-go:

<table>
<thead>
<tr>
<th>(\bar{v}_1(\sigma_1))</th>
<th>(v_1(\sigma_1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/3</td>
</tr>
<tr>
<td>2</td>
<td>2/3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1/3</td>
</tr>
</tbody>
</table>

Bounds on average cost:

\[\bar{c}_1 = 1\]

\[\underline{c}_1 = 1/3\]
as indicated by (4.29). If $D_1$ is in 3, say, and sees interaction 2, then it chooses

$$\arg\min_{u_i} \{u_i: 1+i, u_i: 0+i\} \quad (4.30)$$

which is $u_i$. This drives the system to state 1, taking some advantage of the fact that 2 is not the antagonistic input when $D_1$ is in 3.

As indicated above, this silent strategy works equally well when $D_1$ contains a stochastic model. The stochastics model known dynamical effects within $D_1$; the max min uncertainty, the effects of the parts of the system outside of and unknown to $D_1$.

The silent strategy applies to systems where each agent knows nothing about the system outside of its local model, either a priori or in real time. This precludes the use of a stochastic model for predicting interactions as the statistics of interaction arrivals reflect knowledge of the outside. This requires the use of uncertainty and min max decision making; this, in turn, leads to suboptimality in the team sense. The next sections present ways to improve overall performance by reducing the uncertainty for which each agent must plan.

4.1.4 Selfish Coordination

The first step to improving performance might be to allow communication along links of $G$ in which each agent predicts future interactions. As another agent receives these predictions, it can find a minimum cost response to them, and thereby predict the interactions which it will generate for other agents. The bound on local cost incurred by each
will be reduced as uncertainty is reduced; however, the overall coordination is still suboptimal since no agent considers making a locally suboptimal decision in order to generate an interaction sequence which greatly lowers the costs incurred in parts of the system affected by it. For this reason, this communication strategy is labelled "selfish".

One difficulty with this approach is that the algorithm for silent control, dynamic programming, cannot be used to determine locally optimal controls. Because the interactions are generally time varying, dynamic programming would require them to be specified for all future times. The forward search algorithm of Section 3.3 can avoid this problem if it converges in finite time; a sequence of predictions need be made only as far into the future as is required for it to converge.

To be fully effective, each agent needs to know one piece of data on the rest of the system: the length of prediction required. Let $T_i$ be the longest time to converge for forward search in $D_i$ for sequences $\tilde{\alpha}_i$. If $D_i$ is a superior domule, it requires $T_i$ length predictions from those affecting it. Each of these latter, $D_j$, must supply the maximum of the $T_i$ required by superiors; in order to supply the $T_i^{th}$ prediction, $D_j$ itself requires $T_j$ predictions beyond $T_i$. Thus each domule affecting $D_j$ must supply $T_i + T_j - 1$ predictions to $D_j$. By continuing this way, the length of the predictions required of any agent can be found.

ALGORITHM 4.12: Selfish control with known prediction lengths

A Priori:

Superior Domules:

Set $N_1 = T_i$  \hfill (4.31)
Transmit $N_i$ to all affecting it.

Middle Domules: Receive $N_j$ from all $D_j$ it affects.

Set $N_i = \max \{ N_j \} + T_i - 1$ \hspace{1cm} (4.32)

Send $N_i$ to all affecting it

Inferior Domules: Receive $N_j$ from all $D_j$ it affects.

Set $N_i = \max \{ N_j \} + T_i - 1$ \hspace{1cm} (4.33)

Real Time:

Inferior Domules: Solve for locally optimal controls and resulting state sequence over the period $N_i-T_i+1$. (This will involve forward search looking through $N_i$ steps.) Map this state sequence onto interaction variables for all other domules $D_j$ it affects using $\rho_{ij}(s) = \rho_{ij}$. Send these sequences to appropriate places.

Middle Domules: Receive interaction sequences from all that affect it. Use forward search to find controls, state, and interaction sequences. Note that instead of

$$\omega_i(s, s', t) = \min_{s_{i, t}} \omega_i(s, s, t) + \min_{u_i} c_i(s, s', t, u, t)$$ \hspace{1cm} (4.34)

the interactions set up

$$\omega_i(s, s', t) = \min_{s_{i, t}} \omega_i(s, s, t) + \min_{u_i} c_i(s, s', \rho_{i, n}, \ldots, \rho_{m}, u, t)$$ \hspace{1cm} (4.35)
as a result of the fact that it is they, not $D_{1}$'s dynamics, that are time varying. Send the resulting predicted interactions to appropriate places.

*Superior Domules: Receive interaction sequences. Use forward search to find current control to apply.*

The above form of the algorithm

1. Requires prior communication to determine $N_{1}$
2. Provides predictions as far ahead as necessary
3. Contains much redundancy

The latter arises from the fact that, as it stands above, the algorithm requires all non superior domules to compute controls from the present to $N_{1}-T_{1}+1$ steps in the future. However, the first $N_{1}-T_{1}$ of these were computed one step in the past. Likewise, it requires transmission of interaction predictions that are redundant over $N_{1}-T_{1}$ of their length. While stochastic problems require this redundancy, since predicted probabilities of interaction sequences change due to updated state information, deterministic problems require that the sequences of interaction predictions be extended only one additional step each time. Thus the algorithm can be modified so that only the decision $N_{1}-T_{1}$ is found, yielding state and interaction predictions for time $N_{1}-T_{1}+1$. This reduces both processing and communications requirements at a cost of storage to retain the parts of the predictions which are not retransmitted.
EXAMPLE 4.13. Again consider the example system of 3.1. Forward search in the supral converges in at most five steps; the infimal requires four steps. Thus $N_1 = 5$, $N_2 = 8$, and the infimal must send interaction predictions five steps long to the supral.

Assume each is in its state one. Forward search in the infimal gives the locally optimal infimal control, state and interaction sequences given in Table 4.6. Note that derivation of $u_i(4)$ will indeed require searching to time $8 = N_2$, four steps beyond 4. The infimal will choose to remain in the costless state 1, producing interaction sequences of $\alpha_i^1$. The supral uses this prediction of five consecutive $\alpha_i^1$'s to decide to apply $u_i(4) = u_i^1$. The subsequent interactions will force it into expensive states 3 and 4, so the long term average supral cost is 1. Thus the selfish behavior: the infimal chooses to stay in an inexpensive state, but this forces the supral into costly states.

If the storage for the long predictions required for inferior domules, or the capability for the a priori communication phase of
Table 4.7

Selfish Control for Example

<table>
<thead>
<tr>
<th>t</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\rho$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\mathcal{U}$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Infimal

Supremal

$\sigma$ | 1
$\mathcal{U}$ | 1
Algorithm 4.12 is lacking, an alternative selfish algorithm is possible. Let each agent be capable of generating predictions at most \( T \) steps into the future. Inferior domules can achieve this length of prediction; others will be more limited. If a domule receives predictions over \( T \) steps, it can generate predictions over \( T - T_1 + 1 \) steps only. (It can make at least a one step prediction as it computes the current control to apply, and predicts its effect.) If this is not enough for a neighbor affected by it, i.e. \( T - T_1 + 1 \leq T_j \), that neighbor \( D_j \) must find an alternative way to choose its current control when its forward search algorithm fails to converge.

This can be found by considering that, beyond the horizon for which predictions are received, \( D_j \) is completely uncertain as to what the interactions will be. This suggests using forward search to compute the optimal cost-to-reach function over the period for which predictions exist, and then using the relative cost functions \( \bar{\nu}_i(\sigma) \) to provide an evaluation of the best state to be in at the start of the period of uncertainty. Let predictions have been received for \( t_1 \) steps; forward search gives

\[
\omega_i(\sigma; \sigma^*_i, u_i, t_i) = \min_{\sigma_i} \omega_i(\sigma; \sigma^*_i, u_i, t_i-1) + \min_{\sigma_i} c_i(\sigma, \sigma^*_i, \ldots, u_i) \tag{4.36}
\]

as the optimal cost to reach \( \sigma^* \) at \( t_1 \) from the current state \( \sigma \) using \( u_i \) as the present control. The relative worst case value of being in \( \sigma^* \) at \( t_1 \) is \( \bar{\nu}_i(\sigma^*_i) \); thus the total relative cost when in \( \sigma^* \) and applying \( u_i \) is

\[
\min_{\sigma^*} \omega_i(\sigma; \sigma^*_i, u_i, t_i) + \bar{\nu}_i(\sigma^*) \tag{4.37}
\]
The decision to be made at the current time is thus

$$u_i^* = \arg\min_{u_i} \min_{\sigma^*} \sum w_i(\sigma, \sigma^*, u_i, t_i) + \nabla_i(\sigma^*)$$

which minimizes the sum of the cost to reach some \(\sigma^*\) at \(t_i\) and the relative cost of being in that \(\sigma^*\).

Figure 4.7 illustrates the timing: \(D_j\) receives \(T\) predictions but can generate only \(T - T_i\) controls and hence \(T - T_i + 1\) state and interaction predictions. If \(D_i\) requires \(T_i > T - T_i + 1\) steps, then the worst case relative costs terminate it. This gives the control to be applied currently; it can be used to predict the next state and hence the next interaction.

Note this uses \(\nabla_i(\sigma^*)\) as a static position evaluator as described in 2.1.3. It summarizes future behavior past the point at which detailed information (the interaction predictions) is available. It will be used in this capacity in the next section also.

Summarizing, when Algorithm 4.12 cannot be used, but communications exist to transmit interaction predictions along edges of \(G\), the following alternative can be used:

**ALGORITHM 4.14: Limited Prediction Selfish Control**

**Inferior:** Apply the inferior Algorithm 4.12, but only for some limited horizon \(T\).

**Middle:** Receive interaction prediction sequences from all affecting it. If:

a) Forward search converges: use it to determine as many controls, state, and interactions as possible.
Figure 4.7

Timing in Limited Prediction Selfish Control

Resulting control

Resulting interactions to $D_1$

Resulting states

Derived controls

Predictions to $D_j$

$T - T_j$
b. Forward search does not converge over period for which all interactions have been predicted: Use (4.38) to determine current control. Predict next state, interaction.

Send interaction predictions to those affected.

Superior: Use middle algorithm, but only to determine current control. No predictions are needed.

Generalizing either Algorithm 4.12 or 4.14 to the stochastic case is not difficult. Rather than seeking to predict interactions, each agent can produce probabilities that particular sequences will occur, at least for a finite horizon. The probabilities will be conditioned on the current state of the agent's part of the system, and will be a function of the predicted probabilities on interactions affecting the agent. Stochastic forward search must be slightly modified from that presented in 3.3.2 in order to account for the probabilistic nature of the time variations in the model; this involves working forward in time along each interaction sequence and using the predicted probabilities to weigh the responses against one another. Note that non-termination problems with stochastic forward search can be resolved by applying (4.37) at some time.

This selfish coordination method thus produces an improvement over the silent problem as the predictions of interactions reduce the uncertainty of each local controller and thus make its efforts at reducing
the local cost more effective.*

A key point of Algorithm 4.14 is its two phase character - using detailed information to derive control strategies over the short term, and using uncertainty to allow for effects beyond the short time horizon. This dichotomy - detailed short term and less detailed long term information, - will be seen in future algorithms.

4.1.5 Short Term Coordination

In this section, a more effective form of coordination is discussed where communication is bidirectional and done in terms of minimum costs to produce or respond to possible sequences of interactions over a finite period. The strategy is to use the static algorithm to select the best overall interactions for some fixed horizon $T$, and let uncertainty model interactions after $T$. This requires a way to estimate the cost to produce or respond to each interaction sequence by each part of the system (as ordered by $\mathcal{H}$). The process of constructing this estimate will provide bounds on total system cost which are at least as tight as those given for selfish coordination.

* (Note that this communication improves the bounds on total cost - not necessarily the actual cost. It is conceivable that, for some problems, the silent strategy generates controls that just happen to be globally optimal. Providing information on upcoming interactions may cause the control decisions to be modified - away from the global optimum. Thus partial information is more harmful than no information in terms of actual cost; but the guarantees on performance are clearly better in the selfish case.)
One interesting characteristic of this method of coordination is that, even in deterministic problems, some issues usually associated with stochastic problems appear. For example, consider an agent at time 0 which, in planning its response to interactions from one affecting it, has to choose between applying $\omega_k$ and $\omega_i$ at time 1. If $\omega_k$ leads to a situation such that appearance of some $\rho_{j;^k}$ at time $T+1$ forces a terribly expensive cost to be incurred relative to the worst possible given $\omega_i$ is chosen, then the agent will plan to choose $\omega_i$, to minimize the worst case cost. Thus the cost to produce some $\rho_{j;^k}$ will reflect this. However, if at time 1 agent 1 finds that the one affecting it can avoid producing $\rho_{j;^k}$ at $T+1$, it may plan to use $\omega_i$ at time 1, resulting in a new cost to produce $\rho_{j;^k}$. This illustrates the fact that an agent may be able to improve its near term plans as the future comes more sharply into focus.

The effect of this is much the same as that of observational information on the system state in stochastic settings: one makes current plans based on the best estimate of the future but modifies them at each step as more information appears.

With these understandings, that

a) Each agent computes cost-to-produce or -respond functions based only on locally available information,

b) Missing information is assumed to take its worst case value,

c) Each agent can update its plans each time step to include new information,

the finite horizon coordination algorithm is a straightforward extension of the static Algorithm 4.8.
ALGORITHM 4.15: Finite Horizon Coordination

Case I: Initial Domules

Subcase A: Superior Domules

Each such agent $i$ can compute a function

$$s_{ij}(\tilde{\rho}_{ji}); P_{ij}^T \rightarrow \mathbb{R}$$

(4.39)

to convey the worst-case cost of responding to a sequence $\tilde{\rho}_{ji}$.

This is

$$s_{ij}(\tilde{\rho}_{ji}) = \min_{\sigma^+} \omega_i(\tilde{\rho}_{ji}; \sigma^+, \sigma^+, \tau) + \bar{\nu}_i(\sigma^+)$$

(4.40)

where $\omega_i(\tilde{\rho}_{ji}; \sigma^+, \tau)$ is the minimum cost of going from current state $\sigma$ to state $\sigma^+$ at $T$ steps from now, $\bar{\nu}_i(\sigma^+)$ is the "surcharge" for terminating in $\sigma^+$ at $T$, and $\tilde{\rho}_{ji}$ is the driving interaction sequence. This function can be passed to $D_{ij}$, which will eventually return $\tilde{\rho}_{ji}^*$. Forward search can then determine $\omega_i^*$ to be applied.

Subcase B: Inferior Domules

The function to be found is the cost-to-produce function

$$s_{ij}(\tilde{\rho}_{ij}) = \min_{\sigma(t+\tau)} \min_{\sigma(t+\tau)} \sum_{t=t}^{t=T-1} \epsilon_i(\sigma(t), \sigma(t+\tau), \ldots) + \bar{\nu}_i(\sigma(t+\tau))$$

(4.41)

where

$$\sigma(t+\tau) = f_i(\sigma(t), \omega_i(t))$$

(4.42)
and subject to

\[ \rho_{ij}(t) = g_{ij}(\sigma_{i}(t)) \]  \hspace{1cm} (4.43)

This is the lowest cost way to produce \( \hat{\rho}_{ij} \) subject to the dynamics of \( D_{i} \).

Note that since \( D_{i} \) is inferior, \( \bar{v}_{i}(\sigma) \) is the optimal relative cost to go as no interactions affect \( D_{i} \). \( s_{ij}(\hat{\rho}_{ij}) \) can be sent to \( D_{j} \).

When \( D_{j} \) returns \( \hat{\rho}_{ij} \), \( D_{i} \) can select the appropriate \( u_{i}(t) \) to minimize (4.41).

Case II. Non-initial and Nonterminal Donules

Subcase A: The successor to \( D_{i} \) affects \( D_{i} \), so a cost-to-respond function is needed. This function is computed by minimizing the sum of local costs, costs to produce other interactions that affect \( D_{i} \), and costs to respond to interactions generated by \( D_{i} \).

\[ s_{ij}(\hat{\rho}_{ij}) = \min_{\sigma(t+1)} \min_{\rho_{ki}} \min_{\Hat{u}_{i}^{t+1}} \sum_{t} \sum_{k} s_{ui}(\hat{\rho}_{ki}) \]

\[ + \sum_{I} s_{ik}(\hat{\rho}_{iI}) + c_{i}(\sigma_{i}(t), ..., u_{i}(t)) \]  \hspace{1cm} (4.44)

where

\[ \sigma_{i}(t+1) = f_{i}(\sigma_{i}(t), \rho_{i}(t), ..., \rho_{wi}(t), u_{i}(t)) \]  \hspace{1cm} (4.45)
and $\rho_{m,n}(\tau)$ is the $\tau-t$th element of $\rho_{m}^{n} \in \mathcal{P}_{m}^{n}$. Note the elements of $\tilde{\rho}_{j}$ appear in terms for the local cost and dynamics. This $s_{ij}(\tilde{\rho}_{j})$ can be passed to $D_j$ which will respond with some $\tilde{\rho}_{j}$; $D_i$ finds the $\tilde{\rho}_{k}^{n}, \tilde{\rho}_{j}^{n},$ and $\omega_{j}(t)$ which are the arguments that minimize (4.44), and send them to the appropriate places.

Subcase B: $D_i$ affects the successor of $D_j$, so a cost-to-produce function is required. This is found in a similar way:

$$s_{ij}(\tilde{\rho}_{ij}) = \min_{s_{i}(t+\tau)} \min_{\rho_{j}} \min_{\tilde{\rho}_{ki}} \min_{\omega_{j}(t)} \sum_{t+\tau}^{t+\tau-1} \sum_{l \neq k} s_{ik}(\tilde{\rho}_{ik})$$

$$+ \sum_{k} s_{ki}(\tilde{\rho}_{ki}) + \omega_{i}(s_{i}(t+\tau)) + \epsilon_{i}(\ldots) \quad (4.46)$$

subject to (4.43) and (4.45). This is passed to the successor, $D_j$, which responds with a $\tilde{\rho}_{ij}^{n}$. $D_j$ finds the $\tilde{\rho}_{j}^{n}, \tilde{\rho}_{ki}^{n},$ and $\omega_{j}(t)$ which minimize (4.46) and sends them to the appropriate places.

Case III. Terminal Domule

This agent receives costs to produce and respond from all branches of the system. It sums them with the local costs, and selects as a plan those sequences which minimize this
total:
\[
\min_{\tau_i(t+1)} \min_{\tilde{\rho}^i} \min_{\tilde{u}_i} \sum_{\tau=t}^{t+T-1} \sum_{k} s_{ki}(\tilde{\rho}^i) + \sum_{i \not= j} z_{ij}(\tilde{\rho}^i) + \tilde{v}_i(s_{i(t+1)} + \zeta_i(s_{i(t)}, \ldots, u_i(t)))
\]
\[(4.47)\]
It applies $u^m_i(t)$, and sends the $\tilde{\rho}^m_i$ and $\tilde{\rho}_i^m$ to appropriate neighbors.

Since this algorithm allows the effects of one's decisions on one's neighbors to be included, albeit for a finite horizon, it is to be expected that it leads to performance bounds better than those when the selfish strategy is used. That this is the case can be seen through the argument that the trajectory selected by a selfish strategy is always available for selection by the finite horizon strategy, and another is selected only if it is known to be better, in a global sense, than that selfish prediction. The use of the min max static evaluator to terminate the evaluation of the sequence cost guards against choosing decisions which later force a domule into a highly undesirable state.

**Example 4.16.** Consider the use of a three step look-ahead with the canonical example of Section 3.1. Let $|H|=G$ and both models be in their respective state 1 at time 0.

Algorithm 4.15 can be applied as follows:

1. D is an initial inferior domule, so it must compute the cost to produce each sequence $(\rho_i(1), \rho_i(2), \rho_i(3))$ based on its knowledge of its model and that $\sigma_i(0) = \sigma_1$. The cost of producing
\((1, 1, 1)\) is

\[
\ell_n((1, 1, 1)) = \min_{\sigma_1(0)} \min_{\sigma_2(0)} \min_{\sigma_3(0)} \sum_{\tau=0}^{2} c_\ell(\sigma(\tau), ..., u_\ell(\tau))
\]

subject to

\[
q_{\ell_n}(\sigma(\tau)) = \rho_{\ell_n}(\tau)
\]

which is clearly 0 as \(D_{\ell}\) can stay in state 1. Figure 4.8 shows the cost to produce each other sequence of length 3 (note: the static evaluator \(\overline{\nu}_\ell(\sigma)\) is zero except for \(\sigma^3\), where it is 1 as \(D_{\ell}\) must pass through \(\sigma^3\), costing 1.) Since \(\sigma(0) = \sigma^3\), it is impossible to produce \(\rho_\ell(\tau) = \rho_{\ell_n}(\tau)\), so the entire right subtree is infeasible.

2. \(D_{\ell}\) passes the cost function \(\ell_n(\rho_{\ell_n})\) to \(D_1\).

3. \(D_1\), as a terminal domain, uses (4.47) to determine (upper bounds on) the total cost incurred. (The static evaluator \(\overline{\nu}_1(\sigma)\) is as computed in Example 4.11: 1 for all states.) In this case, the advantage to \(D_{\ell}\) of any sequence is ex- actly offset by the cost of responding to it in \(D_1\). Thus \(D_1\) can select any of the four sequences with \(\rho_\ell(\sigma) = \rho_{\ell_n}(\sigma)\).
Figure 4.8

Costs for Three-Step Look-Ahead

Time:

\[ \rho_i \]

\[ \rho_{21} \]

\[ \rho_{11} \]

Cost to produce:

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 1 & \infty & \infty & \infty \\
\end{array}
\]

Cost in \( D_1 \):

\[
\begin{array}{ccccccc}
4 & 3 & 2 & 3 & 4 & 3 & 2 & 3 \\
\end{array}
\]

Total:

\[
\begin{array}{ccccccc}
4 & 4 & 4 & 4 & \infty & \infty & \infty \\
\end{array}
\]
say \((1, 1, 1, 1)\) as the one to use.

4. \(D_1\) passes \((1, 1, 1, 1) = \tilde{\mathbf{\alpha}}_\star\) to \(D_2\)
    and applies \(u_i(0) = u_i\) as its control.

5. \(D_2\) receives \(\tilde{\mathbf{\alpha}}_\star\), and applies \(u_i(0) = u_i\)
    to achieve it.

6. The above takes \(D_1\) into \(2\) and leaves \(D_2\) in \(1\); the process can be repeated
    at time \(1\), etc.

The above example shows two difficulties which can be encountered
with the finite look-ahead strategy. They are

1. Communication load grows exponentially with the length
   of the horizon (unless the system has special structure)

2. Horizons which are relatively short often do not give
   much improvement in the performance, as the cost estimates
   are more strongly affected by the static evaluator than
   the dynamical structure. (This suggests an improved
   evaluator will help the short horizon case greatly.

   See the following two sections.)

The first point above focusses practical use of this strategy on
engineering the tradeoff between communication capability and coordinated
system performance. As the horizon \(T\) increases, so does communication,

PROPOSITION 4.17. As \(T\) increases, the
performance of a system using finite horizon
coordination approaches the optimum.

PROOF: As \(T\) increases, the space over
which interdomular coordination is performed, $P_{ij}^T$, approaches the space which would be used if the static coordination algorithm were used, $P_{ij}^\infty$. Since the latter generates the optimal interaction sequence, the sequence chosen using $P_{ij}^T$ will approach this optimum.

EXAMPLE 4.18. Consider coordination of the example using four time steps. The tree of costs is shown in Figure 4.9 (the right subtree is infinite as it is entirely infeasible). Ambiguity remains between sequences $(1,2,2,1)$ and $(1,2,2,2)$, but these are selected over all others - and so the initial sequence $(1,2,2)$ is selected. This is the optimal interaction sequence from these initial conditions; however, there is still insufficient communication to guarantee that the system performs optimally.

If ambiguities are resolved in the worst way possible, this strategy achieves an average steady state cost of $2/3$; better than the comparable 1 with three step look ahead but worse than the optimal 1/2.
Figure 4.9

Costs for Four Step Look Ahead

Time:
1
2
3
4

Cost to produce:
0 1 2 1 ∞ 2 1 1

Cost to respond:
5 3 4 3 3 3 3

Total:
5 5 5 5 ∞ 5 4 4
From these examples, one might attempt to show

**CONJECTURE 4.19.** The optimal interaction
sequence, from any set of initial conditions,
will be proposed by Algorithm 4.15 as having
cost no greater than any other such sequence.

However, this is false; a counter example can be constructed similar to
the one used to show that selfish behavior is not necessarily superior
to silent behavior. If a particularly expensive trapping state exists
in the superior which can be forced by the inferior, and access to it is
achieved by following the optimal interaction path over the horizon $T$,
then the static evaluator for the superior will be high at the terminus
of that path. The coordination strategy will select another sequence to
avoid the possibility of the inferior's actions eventually forcing such
a trap.

Finally, the extension of Algorithm 4.15 to stochastic problems is
straightforward. As suggested in Section 3.2.2, the sets of interaction
sequences can be replaced by sets of probability functions on the under-
lying set of sequences. Thus, while

$$s_{ij}: \mathcal{P}_{ij}^T \rightarrow \mathbb{R}_e$$  \hspace{1cm} (4.50)

the corresponding function in stochastic settings takes

$$s_{ij}: \mathcal{S} \mathcal{P}_{ij}^T \rightarrow \mathbb{R}_e$$  \hspace{1cm} (4.51)

where $\mathcal{S}^n$ is the unit simplex embedded in $\mathbb{R}^n$. Thus $s_{ij}(\cdot)$ gives the ex-
pected cost to produce (or respond to) a given distribution on $\mathcal{P}_{ij}^T$. 
Note that for minimization to be meaningful in the computation of the \( s_{ij} (\cdot) \)'s one must consider randomized strategies (e.g., if one could produce one distribution on \( P_{ij}^T \) using one control sequence for expected cost \( \nu_j \), or achieve it as the result of a randomized choice between two other strategies for expected cost

\[
\lambda \nu_k + (1 - \lambda) \nu_j < \nu_i \quad 0 \leq \lambda \leq 1 \quad (4.52)
\]

then the randomized choice is superior). This gives \( s_{ij}(\cdot) \) the property of being piecewise linear and convex \( \cup \) over \( P_{ij}^T \), with each region of uniform gradient convex, as well as having the region where \( s_{ij}(\cdot) < \infty \) convex. All these can be used to reduce the communication required to send \( s_{ij}(\cdot) \) from \( D_i \) to \( D_j \) (or \( D_j \) to \( D_i \)) by sending descriptions of the linear regions in terms of the hyperplanes containing them, rather than each the value at point of \( P_{ij}^T \).

In summary, the short term coordination strategy improves performance by coordinating decisions to avoid behavior that is expensive over the next \( T \) time steps, in terms of total system cost. Locally expensive states beyond \( T \) are avoided as the static evaluator used to terminate the interaction valuation computations considers them. The overall picture of this type of suboptimal, dynamical coordination is one where each agent acquires enough information to model total system cost for a finite horizon, and reverts to complete uncertainty to model interactions beyond that horizon. The next sub-section will discuss a way to reduce uncertainty using shared, a priori reduced models of the system; the subsequent one will use these reduced models to limit uncertainty beyond the horizon of this finite look-ahead algorithm.
4.1.6 Coordination with Abstraction

Turning from strategies which use increasing amounts of on-line communication to achieve improved performance, this strategy will draw on the concepts of Section 3.4.3 to provide each agent with a simplified model of the system external to it which it can use to reduce the uncertainty it sees as to the interactions which may affect it, or the way its decisions may lead to increased or decreased costs on the part of sub-systems which it affects. To avoid stability problems which can arise in techniques which try and identify such models on-line, the approach here will be to provide them in a single computational sweep through the system prior to actual decision making. This limits its use to time invariant local models; however, slowly changing models, or agents controlling sub-systems which might be replaced with new versions, can be handled by repeating the model production computations either periodically or when the replacement occurs.

The procedure embodied in this sweep through the structure will be to use the induction relation $M$ to construct abstracted models of successively larger sections of the system until the terminal domain has a single, simplified model of the entire system. Each agent receives abstracted models from its predecessors, constructs a hybrid model by adding in its own state dynamics, then passes on abstraction of the hybrid to its successor. The abstracted models can be used in two ways in decision making:

1. In a variant of the silent strategy, the abstractions reduce the uncertainty as to which interactions will be produced by inferiors, thus improving performance bounds, or
2. Each module receives "orders" from its successor indicating desired future behavior, then uses the abstractions of its predecessors to construct orders for them which provide the best bounds on performance (and thus must be feasible).

As pointed out in Section 3.4.3, abstraction in modules is more useful when a model of an inferior is constructed and passed to its superior than vice versa. Thus this strategy is most applicable in cases where \( \mathcal{M} \) parallels \( \mathcal{G} \) as much as possible, and is thus best suited to tree-structured systems (hierarchies). However, this section will cover the general case of singly connected systems.

The approach will be to present the strategy in three steps. First, the process involved in constructing the abstracted models will be given. Then, the two decision-making processes will be discussed separately; naturally, combinations of them can also be imagined.

**Algorithm 4.20: Construction of System**

**Abstractions:**

**Initial Modules:** Construct abstracted models, \( \hat{D}_i \), as discussed in Section 3.4.3 and pass to successors.

**Intermediate Modules:** Receive abstractions from predecessors. Construct intermediate model \( D'_i \) with

state space:

\[
\Sigma'_i = \Sigma_i \times \prod_{j} \hat{\Sigma}_j
\]

state transition function:

\[
f_i: \Sigma'_i \times \prod_{j} P_j \times \cup i \rightarrow \Sigma'_i
\]
cost function:
\[ c'_i : \Sigma'_i \times \mathcal{P}_j \times \mathcal{U}_i \rightarrow \mathbb{R} \]

aggregation function:
\[ g'_{ij} : \Sigma'_i \rightarrow \mathcal{P}_j \]
satisfying
\[
\begin{align*}
c'_i ((\sigma_i, \alpha_i, \cdots, \alpha_m), \rho_{i1}, \cdots, \rho_{in}, u_i) &= \\
(\hat{\ell}_i(\sigma_i, \rho_{i1}, \cdots, \rho_{in}, u_i), \hat{\mathcal{E}}_i(\sigma_i, \rho_{i1}, \cdots)) \quad & (4.53) \\
c'_i ((\sigma_i, \alpha_i, \cdots, \alpha_m), \rho_{i1}, \cdots, \rho_{in}, u_i) &\geq \\
\hat{c}_i (\sigma_i, \rho_{i1}, \cdots, \rho_{in}, u_i) + \sum_{j=1}^{n} \hat{c}_j (\sigma_j, \rho_{j1}) \quad & (4.54) \\
g'_{ij} ((\sigma_i, \alpha_i, \cdots, \alpha_m)) &= g_{ij} (\sigma_i) \quad & (4.55)
\end{align*}
\]

This is a dynamical decision problem with states \( \hat{\Sigma}_i \), controls \( \mathcal{U}_i \), and outputs \( \mathcal{P}_{ij} \). An abstraction of it can be generated and passed to the successor of \( D_i \) as a \( \hat{D}_i \).

Terminal Domain: Operates the same as an intermediate; but the resulting abstraction \( \hat{D}_i \) is kept as a simplified model of the entire system.

Several points must be noted about this algorithm:

1. The construction of the intermediate abstraction is for the case where all predecessors of \( D_i \) in \( \mathcal{H} \) affect it. Hence the abstractions include a state transition and cost function \( \hat{F}_j \) and \( \hat{C}_j \) which include \( r_j^i \), the output of \( D_j \), in the determination of future states and cost bounds. For a successor which is affected by \( D_i \), the
abstraction model includes $\rho_j$ which acts as an external input to $D_j$. Thus (4.53)-(4.55) must be modified appropriately to include $\rho_j$ when $D_j$ is affected by $D_1$, and remain $\rho_j$ when $D_j$ affects $D_1$.

2. Similarly, the algorithm assumes the successor to $D_1$ is affected by it; if the opposite is true, there is no need for the "output" function $\psi_{ij}(\sigma_i)$.

3. The index $j$ in the definitions of $\Sigma_i$, $f_i$, etc. ranges over sets determined by context. For example, the Cartesian product in the definition of $\Sigma_i$ ranges over all $j$ such that $D_j$ has supplied an abstraction to $D_1$.

4. Equation (4.53) requires that the state transitions act according to module dynamics - independently except as linked by $\rho_j$.

5. Equation (4.54) forces the combined cost bounds to be included in the resultant cost bound $c'(\sigma, \rho_{i_1}, \ldots \rho_{i_n}, u)$. Note that addition of intervals is done in the usual set addition sense:

$$i_1 + i_2 = [\min i_1 + \min i_2, \max i_1 + \max i_2]$$ (4.56)

6. As mentioned in Section 3.4.2, there is a variety of ways to generate abstractions, from open-loop to closed-loop, depending on the degree to which the local control law can be determined a priori. If forward search shows a control option to be superfluous based on local information, it can be dropped from inclusion in (4.53)-(4.55).
7. Finally, although this section assumes that the abstracted models can be communicated among agents, it seems that realizations of this approach in human organizational decision systems involves the transfer of agents from one position to another. For instance, an apprentice learns (develops) a general idea of the issues and problems of his trade by assisting a master. When sufficiently experienced (and thus having a good feel for the capabilities of an apprentice) he is promoted to master himself, and the abstraction of the decisions faced by an apprentice accompany him. Then, as a master, he can generate orders for his own apprentices without excessive input from them.

Algorithm 4.20 is the basis for constructing abstractions throughout the system, both for the strategies for using them which follow, and the hybrid strategies of the next section. It involves communicating the abstract models \((\hat{\mathbf{Z}}_c, \hat{\mathbf{r}}_d, \hat{\mathbf{\kappa}}_d)\) from \(D_i\) to its successor. While this may appear to violate the principle of communication of quantities expressed in terms of interaction variables only, it should be noted that the most useful abstractions may be those which have abstract states corresponding to the last few interaction variables produced or received, and all abstractions deal with the \(\rho_\gamma\) as either input or output variables with costs bounded by the \(\hat{\mathbf{\kappa}}_d\).

The use of these abstractions in generating actual decisions can proceed along two lines. The first is essentially the silent coordination strategy with abstractions used to eliminate some infeasible incoming
interactions from inclusion in the derivation of the min max control law. The second is a conceptual dual of the selfish strategy: based on its knowledge of which interactions are always feasible, a superior selects an interaction sequence which is best from its local point of view and orders the inferior to produce it. This requires real-time communications capability in opposition to $G$: from domules to those which affect it.

The former is the most simple strategy to describe. Each agent uses the model created by combining its local model with the abstractions received to derive its local decision: selecting the current input which minimizes the worst possible future costs incurred by the system. Since the abstractions will show that some interactions are infeasible, the set of possible worst cases is reduced, and hence performance bounds improve.

**Algorithm 4.21: Silent Coordination with Abstraction**

Each domule uses its combined model $D_i$ to solve

$$\overline{\gamma} (\sigma^i) = \min \left\{ \max \left\{ \sum \gamma_i (\sigma^i + c_i (\sigma^i, \rho, \ldots, u_i)) + \gamma \right\} \right\}$$

$$\sigma^* = f_i (\sigma^i, \rho, \ldots, \rho, u_i)$$

where $\rho_i$ is restricted to the set of those feasible (non-infinite costs of production) when $D_j$ occupies $\gamma_j$ which is the $j$th component of $\sigma^i$. The second max in (4.57) is over the interval $c_i (\ldots)$. The solution
\( \nabla_i' (\sigma_i') \) is a static evaluator for the expanded state set \( \Sigma' \); it and \( \bar{g}_i' \) play roles analogous to \( \nabla_i(\sigma) \) and \( \bar{g} \) in silent coordination: \( \nabla_i'(\sigma_i') \) is the relative cost of being in \( \sigma_i' \), \( \bar{g}_i' \) is the average worst-case cost per unit time incurred in \( D_i \).

The decision rule is then:

\[
 u_i(t) = \arg \min_{u_i} \max_c c_i'(\sigma_i', \rho_{i1}, \ldots, \rho_{iN}, u_i) + \bar{g}_i'(\sigma_i')
\]

(4.59)

where

\[
\sigma_i^+ = \hat{f}_i'(\sigma_i', \rho_{i1}, \ldots, \rho_{iN}, u_i)
\]

(4.60)

The abstractions thus allow \( D_i \) to operate in a slightly larger state space, and it uses \( \Sigma' \) just as \( \Sigma \) was used in silent coordination. The resulting average cost \( \bar{g}'_i \) and static evaluator \( \nabla' (\sigma_i') \) are used in the decision rule just as their counterparts in silent coordination are. Clearly \( \bar{g}'_i < \bar{g}_i \) as the added structure of the dynamics of \( \Sigma' \) remove some possibilities of interaction sequences to be considered; hence the control law will not be made more conservative. This \( \nabla' (\sigma_i') \) will also be used in the next section to terminate forward search when the finite look ahead algorithm is used.

Note that this strategy makes the control decision based on \( \sigma_i' \), not \( \sigma_i \), as well as the \( \rho_{ij} \). Thus each agent must use the observed \( \rho_{ij} \) to constantly update the states \( \hat{\sigma}_j \) occupied in each of the abstractions. Thus the processing required at each agent includes that required to generate the abstractions, to track the occupied state of the abstractions, and
to implement the decision rule.

As mentioned earlier, the abstractions can be used another way. Algorithm 4.21 used them to eliminate impossible interactions; they can also give information on interactions which are certain to be possible. If each domule is given authority to select the interactions which it desires to receive, as well as its controls, then it must be certain that the interactions it selects are indeed possible. Combining these two observations yields an algorithm which requires communication from affected to affecting domules, in which the latter are given instructions as to what to produce for the former. This has considerable intuitive appeal.

ALGORITHM 4.22. Coordination with Abstraction and Commands

Assume that the system topology is a hierarchy, with the terminal domule, identical with the superior domule, at the root of the tree. Then each domule \( D_i \)

1. Receives an order to produce a \( \tilde{\rho}_{ij} \) for some period \( T \) ahead

2. Uses forward search (terminated with \( \tilde{v}_i'(\sigma_i') \) above) to find a \( \sigma_i' \) and \( \tilde{\rho}_{ki} \)

for all \( D_k \) affecting \( D_i \), which

\[
\min \min \max \sum_{t=0}^{T-1} c_i'(\sigma_i', \rho_{ki}, \ldots, \rho_{mk}, \lambda_i, \omega_i) + \tilde{v}_i'(\sigma_i'(T))
\]

subject to

\[
\sigma_i'(t+1) = f_i'(\sigma_i(t), \rho_{ki}(t), \ldots, \omega_i(t))
\]
where each \( \rho_{\mu;}(t) \) is feasible given the 
\( \hat{\sigma}'(t) \) specified by \( \sigma'(t) \).

While this algorithm is specified for tree structures only, it is easy to extend to arbitrary singly connected systems by combining it with the selfish strategy. Commands passed from \( D_i \) to \( D_j \) only make sense when \( D_j \) affects \( D_i \); hence \( \mathcal{H} \) must contain \((j,i)\) as well as \( \mathcal{G} \).

When \( \mathcal{H} \) contains \((i,j)\) and \( \mathcal{G} \) contains \((j,i)\), \( D_j \) cannot pass a command to \( D_i \) as \( D_j \) affects \( D_i \). Hence \( D_j \) must pass its prediction of interactions to \( D_i \), which can then pass predictions or commands it generates locally to relevant agents. (Figure 4.10 illustrates this.) The difference between this and the selfish strategy, though, is that \( D_j \) will, through the abstraction \( \hat{D}_i \), have some information (bounds) on the cost to be incurred by the subsystem preceding, and including, \( D_i \). Thus the predictions of \( \hat{\delta}_j \) will be selected by \( D_j \) in a way which, to some extent, takes into account the expense incurred by that subsystem responding to them.

This algorithm captures the essence of the situation described in comment (7) after Algorithm 4.20 concerning the promotion of an apprentice to the position of master overseeing other apprentices. Because the master understands the minimal capabilities of the apprentices, he can give them assignments without prior communication with them concerning their individual situations. While not the most effective form of coordination (due to the presence of the conservative, worst-case consideration imposed by the max over the interval sum of \((4.61)\)) it does represent an improvement over the process of Algorithm 4.21. This latter has the master trying to predict what actions the apprentice
Figure 4.10

Using Abstraction to Generate Commands

A. Topology

B. Communicated quantities
would take given no guidance from the master— and the inability of the apprentice to see the ramifications of his actions on the master's problems is likely to lead to significant suboptimality.

As was stressed at the end of Section 3.4.3, this work on the formalization of the concept of abstraction, and its use in coordination strategies, is very preliminary. It clearly has great potential in aiding the decision making process in a modular system without imposing the communication load of the finite look-ahead strategy, and captures many of the intuitive aspects of human organizational decision making. However, the formal framework in which it is expressed, and many of the theoretical conclusions which can be drawn regarding its use, require much more work.

With the above caveat, the promising results of this section can be summarized as follows:

1. If the system is singly connected, an abstracted model of increasingly large subsystems can be propagated until at least one agent has a simplified model of the entire system.

2. These subsystem abstractions can be used to:
   a) limit the set of possible interaction sequences producible by an inferior
   b) bound the cost incurred by a subsystem affected by an agent
   c) determine interaction sequences which can always be produced by an inferior, and bound the cost involved in that production.

3. Coordination can be achieved by using (a) and (b) above to
select a local control which ignores impossible sequences of incoming interactions, and includes the cost-to-respond bounds on superiors.

4. Coordination can also be achieved using (b) and (c) as a superior selects which interaction sequence an inferior should produce.

5. Abstraction seems necessary to capture this command coordination scheme, which is seen in many human organizations, as the previous ones could not.

In short, abstraction allows communication of some (but not complete) knowledge of system structure to supplant real-time communications. The next section discusses a hybrid of the strategies of this and the previous section which provides still better coordination with still greater processing and communication requirements.

4.1.7 Short Term Coordination with Abstraction

Most of the groundwork for this strategy has already been laid. By augmenting the finite horizon coordination scheme with the abstracted models, tracking the state of the latter as time evolves, and using the static evaluator $\nabla_i^{'}(\tau_i^{'})$ rather than $\nabla_i^{'}(\tau_i)$ to terminate the forward search, improvements in system performance can be achieved. These result from the reduction in uncertainty as to the possible interactions affecting a particular module beyond the finite, detailed coordination horizon. This is just as abstraction alone reduced the uncertainty, and thus improved performance bounds, for an agent using the silent strategy.

As with the methods presented in the last section, this approach operates in two parts. First, Algorithm 4.20 can be used to establish
the abstracted models wherever necessary; and all comments made there apply equally well here. The second stage, where actual control decisions are made, is described as

**ALGORITHM 4.23.** Finite Horizon Coordination with Abstraction

**Case I: Initial Domules**

Neither superior nor inferior initial domules have abstractions of other domules available, hence they both operate exactly as in 4.20. They produce \( \tilde{\sigma}_{ij} (\tilde{\rho}_{ij}) \) and transmit it as the cost-to-respond and cost-to-produce function respectively. Upon receiving a \( \tilde{\rho}_{ij}^T \), each applies the control given by local forward search.

**Case II: Non-initial and Non-terminal Domules**

Both cases of \( D_i \) affecting its successor \( D_j \), and \( D_i \) being affected by \( D_j \), are handled as in equations (4.44)-(4.46) with \( \tilde{\nu}_i (\sigma_i) \) being replaced by \( \tilde{\nu}_i (\sigma_i') \). However, additional processing is required to propagate

\[
\sigma_i' (t+\tau) = f_i' (\sigma_i' (t), \rho_i' (t), \ldots, u_i(t)) \tag{4.63}
\]

in order that \( \tilde{\nu}_i (\sigma_i' (t+\tau)) \) can be evaluated (\( T \) is the horizon length).
Note that, in evaluating costs over the period \( T \) where interactions are considered exactly, \( \sigma_i \) and \( \bar{c}_i(\cdot) \) are used; the combination state of \( \sigma_i \) is used only at the termination of the forward search minimization. This is because \( \sigma_i \) is necessary only to change the knowledge of the effect of events beyond the horizon \( T \) where exact knowledge does not exist.

When \( D_i \) receives \( \bar{\sigma}_i \) from \( D_j \), it finds the corresponding sequences to request of its predecessors as in the finite horizon case (by minimizing the modified (4.44) or (4.46)).

Case III: Terminal Domain

As in Algorithm (4.20), this agent receives costs from its successors, then uses equation (4.47) (again with \( \bar{\sigma}_i \) replacing \( \bar{\sigma}_i \)) to select its control \( \bar{u}_i(\cdot) \) and \( \bar{\rho}_j(\cdot) \). These latter are then communicated to the \( D_j \) to initialize the second, backwards sweep of the coordination process.

This hybrid algorithm thus uses the finite horizon strategy to consider near-term (within \( T \) steps) interactions exactly and long-term (beyond \( T \) steps) interactions in a worst case sense, with the latter
limited by the additional knowledge of system structure supplied by the abstract models. (Note that this is not explicit; it is done implicitly through the modified static evaluation mechanism.) Thus it is easy to show

PROPOSITION 4.24: The bound on worst case total system cost is at least as low for the hybrid algorithm than for finite horizon look-ahead.

Proof: The set of possible interactions to be considered by each agent as possible worst cases beyond the horizon $T$ is at least as small for the hybrid as for the other; hence the worst-case bound is at least as low.

Also,

PROPOSITION 4.25: The worst-case bound for the hybrid is at least as low as that for (silent) abstraction coordination.

Proof: Again, the hybrid has less uncertainty as the first $T$ steps of interaction production (or response) costs are modelled exactly, where in the abstract algorithm they can only be bounded by computations based on $\hat{c}_j$.

As has been pointed out previously, it is important to note that the above propositions deal only with performance bounds; actual costs may be higher.
for the hybrid as the added information causes a naive, but coincidentally optimal, decision to be modified.

The process captures by this strategy is even more similar to the characteristics of human organizational decision making than that of abstraction. While abstraction serves as a general model to rule out unreasonable things that another agent may do over the long term, the finite-horizon part models a negotiation process as exact plans for the near future are agreed upon by examining all possible options. This similarity with processes which have evolved in practice suggests that this hybrid algorithm is also a prime candidate for further research into expansion and clarification of the preliminary concept described here.

4.1.8 Conclusion

This section will briefly compare the salient properties of the principle coordination algorithms proposed for singly connected domular topologies. Rather than just summarizing the underlying processes involved in each, it will compare and contrast their properties in terms of performance bounds, processing requirements at each agent, and inter-agent communication needs.

Although some sections presented several variants of an algorithm, only one of each class will be considered here. For simplicity, only deterministic cases will be considered. All strategies involving a finite horizon will be compared for the same value of T. While precise measurement of processing and communication is difficult, some qualitative conclusions can be drawn given the assumption of no further structure than that of domular decomposition; such structure could clearly exhibit properties which drastically alter the requirements (and indeed, the search
for such structures is quite important). These quantities will be discussed in terms of both a priori, or initialization, needs and those required for a typical, single step of the on-line coordination process.

The specific algorithms included here are:

A. Static team coordination (Algorithm 4.8) based on an infinite future.

B. Silent coordination (Algorithm 4.10) in which the static evaluator $\nabla_{o}(\sigma)$ is computed a priori.

C. Selfish coordination with known prediction lengths (Algorithm 4.12) including the initialization pass to determine the minimum lengths of predictions required.

D. Finite horizon coordination (Algorithm 4.15).

E. Silent coordination with abstraction (Algorithm 4.20 to construct abstract models, followed by 4.21 as the coordination). The modified evaluator $\nabla_{e}'$ will be assumed to be calculated along with the abstract model $D_i$.

F. Finite horizon coordination with abstraction (hybrid) (again, Algorithm 4.20 to construct models, but followed by Algorithm 4.23 as the on-line strategy).

Figure 4.11 illustrates the partial order on performance bounds of the above. Any coordination is better than the silent strategy; the best is, of course, the static, infinite planning horizon algorithm which effectively models all system dynamics as they affect each agent. All others fit between these poles, with the hybrid algorithm superior to each of its components as discussed in Propositions 4.24 and 4.25.
Figure 4.11

Relative Performance Bounds

Static

Hybrid

Abstraction  Finite Horizon

Selfish

Silent
If the horizon $T$ of the finite horizon approach is greater than any of the $T_i$ used in the communication of predictions in the selfish case, then the finite horizon scheme is superior as indicated by the dashed line. This results from the fact that the sequence that would be predicted by the selfish strategy is eliminated by the finite one only if the latter finds another sequence with better worst-case global performance.

Not unexpectedly, each improvement in performance is achieved at the expense of more communication. Figure 4.12 illustrates the relative communication requirements of these strategies for both the initialization phase and a typical single time unit pass. In terms of the latter, static requires most, as the $s_{ij}(\bar{s}_j)$ must be communicated as functions on an infinite dimensional space $\mathbb{R}^\infty_{ij}$. Abstraction and silent require no real-time communication, but the abstraction technique buys its performance advantage through a large initialization load as the abstract models are passed from one agent to the next. Finite horizon and selfish have identical initialization requirements involved in passing along information on the time horizon to be used; finite horizon and the hybrid communicate the same on-line quantities - costs on $\mathbb{R}^T_{ij}$. Combining these two orderings shows that overall relative communications load is indeed quite similar to the relative performance.

Finally, Figure 4.13 shows the processing requirements for a typical (i.e. non-initial and non-terminal) agent, again for both initialization and on-line phases. Silent and finite horizon require processing in the former to construct $\bar{v}_i$, the worst-case static evaluator, while the two using abstraction must compute $\bar{v}_i'$, which is defined on the larger space $\Sigma_i'$, as well as the abstract models themselves. The selfish strategy
Figure 4.12

Relative Communication Needs

Hybrid

↑

Abstraction

↑

Finite Horizon,
Selfish

↑

Silent (Static)

A. Initialization

Static

↑

Hybrid, Finite Horizon

↑

Selfish

Silent, Abstraction

B. Real time
Figure 4.13

Relative Processing Requirements

Selfish ----> Abstraction, Hybrid

Silent, Finite Horizon

Static

A. Initialization

Static ----> Hybrid

Finite, Horizon

Selfish ----> Abstraction

Silent

B. Real time
must find a $T_i$ such that forward search converges for all interaction sequences in that time; this involves a search over all such sequences.

The real time requirements are more in parallel with performance. The static requires most, as at each step the cost functions on $\mathcal{F}_{ij}$ must be constructed. Silent requires least, as $\mathcal{F}$ is used to find the current decision. Abstraction is similar to silent, with the fact that $\mathcal{F}'$ is defined on the larger space as the only reason it needs more. Selfish requires forward search to find the current control given the received predictions; finite horizon requires forward search along each possible input interaction sequence in order to construct the $\alpha_{ij}(\tilde{\mathcal{F}}_{ij})$. Finally, hybrid requires the processing capability of both the finite horizon and abstraction techniques. Thus again it is seen that each algorithm that yields better performance bounds than another also requires more on-line processing power.

The above qualities are summarized in Table 4.14. Naturally the entries reflect broad, qualitative judgements, based on a general system. It must be re-emphasized that a major goal of future work along these lines should be to find system structures which are exceptions to the general trends displayed here.

Other qualities of each strategy are worthy of note in a summary such as this. They are:

A. **Static Coordination:** Each agent finds the minimal costs to produce/respond to each possible interaction, then passes this to another. The latter matches this with its options, selects the overall best resulting interaction, and both implement it.
Table 4.14
Comparison of Coordination Strategies

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Performance</th>
<th>Communication</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Initial</td>
<td>Real time</td>
</tr>
<tr>
<td>Static</td>
<td>Best</td>
<td>Least</td>
<td>Most</td>
</tr>
<tr>
<td>Silent</td>
<td>Worst</td>
<td>Least</td>
<td>None</td>
</tr>
<tr>
<td>Selfish</td>
<td>Poor</td>
<td>Some</td>
<td>Some</td>
</tr>
<tr>
<td>Finite Horizon</td>
<td>Fair</td>
<td>Some</td>
<td>Much</td>
</tr>
<tr>
<td>Abstraction</td>
<td>Fair</td>
<td>Much</td>
<td>None</td>
</tr>
<tr>
<td>Hybrid</td>
<td>Good</td>
<td>Most</td>
<td>Much</td>
</tr>
</tbody>
</table>
Variations: The one-step and infinite horizon formulations are equivalent; the former emphasizes the static nature of the algorithm, the latter its relation to a dynamical setting.

Example: It is difficult to find an example of practical application of this algorithm due to its extreme communication/processing requirements (in the dynamical setting). For an example of the one-step strategy, consider the decisions made by the two people searching for a mutually pleasurable point to meet for lunch. Typically, one will suggest a list of alternatives, rating them according to some scale. The other will then adjust the ratings in accordance with his own tastes, and suggest a solution. This is a coordinated decision, made with no consideration of the future, and (hopefully) maximizes total pleasure.

B. Silent Coordination: Each agent makes its decision to minimize local costs assuming the worst possible interactions will be generated by other parts of the system in the future.

Variations: None.

Example: Consider the plight of tenants of a large apartment building with inadequate plumbing. Each desires to take hot showers, but must settle for something slightly cooler in order to avoid the possibility of being scalded when another turns a shower off.
There is no coordination; each makes his own decisions independently and must guard against the worst case in an uncertain future.

C. **Selfish Coordination**: Each agent receives predictions of interactions which affect it, plans its own activities, and passes on predictions of how those will affect yet others. 

**Variations**: The two variations deal with how the various agents determine the horizon over which they will generate predictions to those they affect. Either an initial pass can be used to allow each to determine the horizon it requires and pass this on to its affectors, or inferior agents can arbitrarily select a horizon and all others generate whatever predictions are possible.

**Example**: Consider the case of the public works department's road maintenance crews, and commuters. Interactions between the former and latter are the times, locations, and extent of repair work. Typically, the department will decide what maintenance work needs to be done, schedule it, and publish the schedule. Each commuter must then plan his own activities around this set schedule.

D. **Finite Horizon Coordination**: Each agent receives from those around it a list of all possible interaction sequences for the next $T$ time steps and associated costs to produce or respond. Based on this, and local
optimization, it generates such quantities for others. One agent receives these estimates from all around it, selects the best overall set, and passes this selection back. Each local optimization is done assuming a worst-case future beyond T steps.

Variations: None

Example: Imagine a homeowner with a broken appliance trying to schedule a visit by a competent maintenance man. Neither knows the situation faced by the other. The man suggests a set of times at which he can show up (the interaction) over the next T days and costs of each (due to odd hours, etc.). The homeowner takes these options, adds in factors associated with his plans, and selects the best. Note that he must assume the worst case: that the man will be completely booked up after the T day period, so he cannot decide to postpone the visit beyond T days.

E. Coordination with Abstractions: Prior to any decision making activity, the agents construct simplified models of other parts of the system. These can then be used to bound the cost-to-produce or respond functions without real-time communication, and local decisions made on this restricted worst-case basis. Abstraction is most effective when used for the cost-to-produce function, and thus in hierarchical systems.

Variations: The abstracted models can be used to guide
the local decision processes to find controls that are more beneficial in a system wide sense, or to generate commands as to what interactions affecting agents should produce.

Example: An assembly line illustrates the use of abstraction well. Each worker possesses a model of the entire plant expressed in local terms: either the line is moving and tasks must be performed at some regular rate, or it is broken and nothing need be done. This model serves to help schedule his activities based on local variables (e.g. parts supply) in a more coordinated manner than would be possible without it (e.g. if tasks arrived at unpredictable intervals).

F. Finite Horizon Coordination with Abstraction: This strategy uses the abstract models of (E) to reduce the uncertainty of external system behavior beyond the T step horizon of (D). It thus requires the initial construction of models, followed by real-time communication of the details of the next T steps.

Variations: None.

Example: Reconsider the homeowner/maintenance man scheduling problem of (D). If, in their conversation, the man starts out by saying he is always free for emergencies on Thursdays, this establishes a simple model of his availability to the homeowner. Thus the latter need not assume that the man will be booked completely after the T days of consideration,
and this may allow him to postpone the repair work
due to pressing previous engagements.

This concludes the discussion of coordination of systems which
exhibit the singly connected topology and a team objective structure.
A variety of methods to achieve such coordination exist, each using only
local model knowledge and communication in terms of interaction variables.

4.2. More General Problems

Section 4.1 dealt with the problem of coordinating the activities
of a domular system with a singly connected topology, where the objective
was the team goal: minimize the sum of the local costs incurred. This
section will examine the six strategies proposed in that context for app-
licability to wider classes of topology or objective.

The structure of this section will be to parallel the development
of 4.1, starting with a classification of the topological and objective
structures to be considered. Section 4.2.1 will discuss the effect of
alternate choices of each on the static coordination scheme and also,
because the considerations are similar, for the finite horizon look-
ahead approach. Succeeding sections will deal with the silent, selfish,
and abstracted coordination strategies, omitting the hybrid combination
of abstraction and finite horizon planning as its properties can be de-
duced straightforwardly from those of its components.

This involves topologies and objectives more complex than those
of the example of Section 3.1, so no attempt will be made to use it as the
sole illustrator of points made subsequently. Also, discussion will be
confined to the deterministic case for reasons of clarity. The goal of
each section will focus on detailing which combination of structures each
approach is suited to, and thus will only explain why a pair is compatible, 
or illustrating why it is not.

The topological structures of interest are

A. Hierarchical trees: a subset of singly connected systems
   with a single superior node.
B. Singly connected systems.
C. Doubly connected systems: similar to singly connected, but
   interactions can flow both ways between successive agents
   in the network.
D. General acyclic systems: systems with no cycles such
   that an agent's actions eventually propagate through to
   affect itself.
E. General systems: arbitrary topologies.

Singly connected systems are described in Definition 4.1; others are for-
mally defined by:

**DEFINITION 4.26**: A **hierarchical tree** system is
one in which no agent affects more than one other:
for each i, there is at most one j such that

\((i,j) \in I\)

Thus \(G\) is singly connected.

**DEFINITION 4.27**: A system is **doubly connected** if
it contains a singly connected system, and can
be constructed from that system by adding only
links \((i,j)\) if \((j,i)\) is in that system.

If

\[ I_1 \supseteq I_2 \]  \hspace{1cm} (4.64)
and $\mathbf{L}_2$ describes a singly connected system, then

$$\mathbf{L}_2 \leq \{ (i,j) | (i,i) \in \mathbf{L}_2 \text{ or } (j,i) \in \mathbf{L}_2 \}$$ (4.65)

A doubly connected system thus allows bidirectional flows of interactions between agents in what otherwise would be a singly connected system. It thus is excellent for modelling systems structured around the exchange of interactions between agents (e.g. markets).

**DEFINITION 4.28**: An acyclic topology is one such that given $(i,j)$ is a link in it, there is no sequence of links starting at $j$ which leads back to $i$. Equivalently, there is no pair of nodes $i$ and $j$ such that both $(i,j)$ and $(j,i)$ are elements of the transitive closure of $\mathbf{L}$.

Finally, a generally structured system is one with no restriction on $\mathbf{G}$. Figure 4.15 illustrates each of the above types.

Figure 4.16 shows the relations between these classes of topologies. Any hierarchy is singly connected, which is doubly connected, which is general. Figure 4.15 C shows a doubly connected system which is not acyclic; 4.15 D illustrates the opposite. Each strategy will be applicably to a particular level of generality in this relation.

Three types of group objective can be considered. The team objective, defined previously, grouped all agents together in order to achieve a common goal. For simplicity, this goal was assumed to be to minimize the sum of local costs. The antithesis of this is the individual
Figure 4.15

Examples of Topologies

A. Hierarchy

B. Singly connected

C. Doubly connected

D. Acyclic

E. General
Figure 4.16

Relationships of Topology Classes

Hierarchy

Singly connected

Acyclic

General

Doubly connected
objective structure, where each agent is interested only in minimizing local costs. Between these two lies a range of group objectives, in which various groups of agents work together to achieve a common goal, but each group has a different goal. This captures the notion of several organizations competing with one another.

**DEFINITION 4.29:** The individual objective has each agent $D_i$ striving to

$$\min \max \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} c_i(\sigma_i, \sigma_i^+, \rho_i, \ldots, \rho_i^+, \omega_i)$$

(4.66)

where the maximum operation is over all uncertainty not resolved by communication.

**DEFINITION 4.30:** The group objective involves defining a partition $F^m$ on the set of domines $IN$ with

$$|F^m| = m$$

(4.67)

$$\bigcup_{i \in F^m} m_i = IN \quad m_i \cap m_j = \emptyset \text{ iff } i \neq j$$

(4.68)

Then, for each agent $j$ in group $m_i$, the objective is to

$$\min \max \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{k \in m_i} c_k(\sigma_k, \sigma_k^+, \rho_k, \ldots, \omega_k)$$

(4.69)

As in the team objective, the functional in (4.69) is expressed as a sum of local costs within the group. Other combinations of costs can be handled in general by communicating sequences of costs of interactions, rather than just the total cost involved in producing or responding to
an interaction sequence.)

The above definitions establish a framework of fifteen topology-objective structures for study with each algorithm. The next four subsections examine the way each strategy relates to each such structure.

4.2.1. Static Strategies

The static strategy can be extended with little difficulty to doubly connected systems, and to acyclic topologies if the communication principles are relaxed somewhat. It will not work with general topologies as cycles preclude the use of an induction order. Since it provides perfect coordination between many agents, it is not useful for the individual objective structure, but does work well within a group if some other strategy is used to provide intergroup coordination. (The results of this section can be applied equally well to finite horizon coordination by restricting attention to the T steps of the future and using uncertainty to model the remaining time.)

The extension to doubly connected systems is made by substituting a more complicated expression for the cost-to-produce (cr -respond) functions \( s_i(\rho_j) \) (or \( s_i(\rho_{ij}) \)) by conditioning them on a particular response (or production). Because of its structure, an induction ordering \( \mathcal{H} \) can always be found exactly as in the singly connected case. This is used to guide the propagation of the new forms of the cost functions through the system.

The modification of the cost-to-produce function is as follows. Recall \( s_i(\rho_{ij}) \) is the minimum cost to produce interaction \( \rho_{ij} \) by \( D_i \) and its predecessors. If \( D_j \) produces interactions which affect \( D_i \), counter-directionally to \( \rho_{ij} \), then a new function \( s_i(\rho_{ij}; \rho_{ij}) \) is defined as the minimum cost to produce \( \rho_{ij} \) conditioned on the production of \( \rho_{ij} \) by \( D_j \).
Similarly, the cost to respond to an interaction \( \rho_{kj} \) by \( D_k \) and its predecessors can be conditioned on their producing a particular \( \rho_{ki} \), resulting in a function \( s_{kj} (\rho_{ki}; \rho_{ij}) \). (See Figure 4.17.) (Recall that the notation \( f(\omega; x) \) is intended to convey a family of functions of the variable \( x \) indexed by the argument \( \omega \).)

\( D_j \) can compute its cost to produce/respond by minimizing

\[
\zeta_j (T_j, T^+, \rho_{ij}, \rho_{ki}, u_j) + s_{ij} (\rho_{ij}, \rho_{ij}) + s_{kj} (\rho_{kj}, \rho_{jk})
\]

(4.70)

over \( u_j, \rho_{ij}, \rho_{ij}, \rho_{ki}, \rho_{jk} \) subject to

\[
\xi_{ji} (S^+) = \rho_{ij} \quad \xi_{jk} (S^+) = \rho_{jk}
\]

(4.71)

This process of computing \( s_{ij} \) as a function of \( \rho_{ij} \) and \( \rho_{ji} \) can be propagated through the system as in the usual static algorithm.

Thus doubly connected teams can be coordinated using a static strategy which replaces the costs associated with producing/responding to interaction sequences with cost of exchanging interactions.

Acyclic topologies pose a more difficult problem. Consider the topology of Figure 4.18 A. \( D_3 \) can compute a function

\[
s: \mathcal{P}_{31} \times \mathcal{P}_{32} \rightarrow \mathbb{R}
\]

(4.72)

which is the cost to produce interaction pairs \( (\rho_{31}, \rho_{32}) \). Unfortunately, \( s \) will not in general decompose to the form

\[
s(\rho_{31}, \rho_{32}) = s_{31} (\rho_{31}) + s_{32} (\rho_{32})
\]

(4.73)

so \( D_3 \) cannot transmit separate cost-to-produce functions to \( D_2 \) and \( D_1 \). Each other agent is in a similar position: its costs depend in a nonseparable
Figure 4.17

Doubly Connected Static Coordination
Figure 4.18

Modification of Acyclic Topologies

A. Acyclic Topology

B. Modified
way on the interactions with each of its neighbors. This fact, plus the observation that one cannot define an induction order on the topology, leads to the conclusion that a typical acyclic topology cannot be coordinated using the static scheme.

However, if the principle prohibiting interagent communication in terms other than those involving the underlying interaction variables is relaxed, coordination can occur. D3 can transmit \( \cdot \cdot \cdot \) to D2, which can then derive a cost-to-produce function for \( (\rho_3, \rho_4) \) pairs. Passing this to D1 allows coordination to occur.

The effect of this procedure is essentially to restructure the system into that of Figure 4.18 B. The \( \rho_{21} \) interactions are replaced by \( (\rho_{21}, \rho_{22}) \) pairs; \( \rho_{21} \) by \( (\rho_{21}, \rho_{21}) \) pairs; and the dynamics of D2 relating \( \rho_{21} \)'s to \( \rho_{21} \)'s by the trivial extension which preserves the first component of the new pairs. (There is a slight technical difficulty involved in preserving time relationships in this approach; it can be overcome by reformulating the models slightly.) This effectively produces a singly connected system which can be handled in the usual way.

Thus the static scheme can be applied to doubly connected systems, but the lack of separability in cost functions precludes its use in acyclic systems, at least with the team objective. With the individual objective structure, each agent receiving cost-to-produce functions from affectors must assume that the least costly interaction sequence will be produced. Received cost-to-respond functions are ignored: the agent is interested only in its local cost minimization. Since the selection of the least costly to produce interaction can be done just as well by the affecting agent as by the affected, full cost-to-produce functions need not be found and communicated either. Thus, in the individual objective
case, only predictions of the locally least costly to produce interactions are sent from affector to affected - and this is just the selfish strategy. As will be shown in Section 4.2.3, the selfish strategy for individual objectives, and hence the static strategy, works for arbitrary acyclic topologies, but not doubly connected systems - exactly the opposite of that for the static scheme and team objectives.

Between team and individualistic objectives lie those involving groups. By the same reasoning as above, intergroup communication will reduce to that of predictions of groupwise least costly interactions. This is again selfish coordination, which requires that groups be related to one another in an acyclic topology. Within groups, however, the full static coordination scheme can be used: predictions from affectors are taken as given, coordination of the group as a whole done, and predictions of the resulting interactions affecting the rest of the system can be sent to appropriate places. Thus the group objective permits the use of the static strategy in a special subset of general topologies: intragroup structure must be doubly connected, and intergroup structure must be acyclic. An example of this type of topology is shown in Figure 4.19.

It is to be expected that this most general case specializes to the results for team and individual objectives when the number of groups is one, or when each group contains exactly one agent. Table 4.20 summarizes each of these results - and the specialization of the topological requirements for static group coordination to the more restricted sets of topologies.

4.2.2. Silent Strategies

By virtue of the fact that the silent strategy decomposes the coordination scheme into the search for the best worst-case local controls,
Figure 4.19

Most General Statically Coordinable System
Table 4.20

Suitability of Static Coordination

| Topology         | Team         | Objective-
<table>
<thead>
<tr>
<th></th>
<th>Group</th>
<th>Individual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Singly connected</td>
<td>Good</td>
<td>Good</td>
</tr>
</tbody>
</table>
| Doubly connected | Good         | Provided inter-
|                 |              | group topology    | None              |
| Acyclic          | Can be done with modifications | " | Reduces to selfish |
| General          | None         | Provided intra-
|                 |              | group topology is doubly connected, and intergroup is acyclic |
and thus decisions are made by each agent completely independently of all others, it is applicable to all topologies and all objective structures. Topology is not a factor as the origin of interactions impinging upon a module is of no concern to the agent; it merely assumes the worst possible case. Objective structure is immaterial as teams, and thus groups, are reduced to acting on an individualistic basis when shorn of all capacity to communicate. However, there are two relevant points which can be made before moving on to selfish strategies.

First, in the discussion of the static strategy for group and individual objective structures, it was seen that intergroup or interagent communication was reduced to that of the selfish strategy, as each could send predictions to those it affected. However, generating these predictions, and communicating them, usually involves some costs in practice. Unlike the team case where the objective of overall performance optimization motivates the use of such predictions, there is no reason for an agent to incur these costs if it is acting only in the interest of itself or its group. Unless the affected party is willing to reimburse the costs of the generation of such predictions, the interagent or intergroup strategy is likely to be the silent one, instead of the selfish one mentioned above.

Also, while the silent strategy is generally poor from a performance point of view, it is the only one of those considered here suitable for use in a general system topology. Thus in a complex system with a general, cyclic topology, the only candidate for coordination which satisfies all of the principles set forth in Section 2.2 is the silent strategy.

4.2.3. **Selfish Strategies**

While the selfish approach to coordination displays much the same behavior with respect to objective structure as the silent strategy, the
effect of topology on its usefulness is quite interesting. As touched on in Section 4.2.1, it is suitable for acyclic topologies, but not for structures involving any sort of cycle due to the fact that mutually conflicting objectives can exist which lead to a gaming situation between agents.

That the selfish strategy works with any acyclic system under the team objective is easy to see. Each agent which is affected by no other agent (inferior domules) can solve its local control problem, compute the interactions generated by applying the solution, and passing these predictions to the agents it affects. A second set of agents which are affected only by domules which have transmitted predictions of interactions can use them to construct and send their own prediction. Repeating this process through the entire system eventually results in predictions arriving at all superior domules from others affecting them, and the coordination is complete.

The reason that the selfish scheme does not run into the separability problem of the static strategy is that each agent forms its predictions as an element of the Cartesian product of outgoing interaction sets, not as a function on that product set. The element is separable as each component can be sent to the agent it affects; the function is not separable.

Where the selfish strategy fails while the static approach succeeds is in doubly connected topologies. The predictions made by one agent may depend on those made by its neighbor, which in turn depend on those made by the one. As there is no guarantee that a mutually consistent pair of such predictions exists, the selfish strategy cannot in general be used in cyclic topologies.
EXAMPLE 4.31. Consider two agents in a cyclic topology. Each has almost the same model consisting of two states and two controls, each of which acts as a reset to the corresponding state (Figure 4.21). The only difference is in the mapping from states to interactions—one is the reverse of the other.

The local objective of each is to match the state of the local system at time \( t+1 \) with the interaction received from the other agent. Thus the local cost function is

\[
c_t(s_i, s_j, \rho_i, \omega_i) = \begin{cases} 
0 & \text{if } s_i = \rho_j; \\
1 & \text{else}
\end{cases}
\]  

(4.73)

This example cannot be coordinated using a selfish strategy.

To see this, suppose \( D_1 \) assumes \( D_2 \) will produce \( \rho_i \) at time \( t+1 \). \( D_1 \) will then apply \( \omega_i \), and predict that

\[
\rho_{12}(t+1) = \rho_i
\]  

(4.74)

\( D_2 \) will receive this prediction, decide to apply \( \omega_i \), and thus send a prediction that

\[
\rho_{21}(t+1) = \rho_i
\]  

(4.75)

contradicting \( D_1 \)'s original assumption. A similar inconsistency occurs if \( D_1 \) assumes
Inconsistency in Cyclic Selfish System

A. Topology

B. Model of $D_1$

C. Model of $D_2$
D2 will produce $\rho_{ij}^2$.

The difficulty here is that encountered in game theory. Because the two agents have objectives that directly conflict, no solution exists. By allowing randomized control laws, a saddle point can be found: each predicts

$$\rho(\rho_{ij}^\star) = \rho(\rho_{ij}^\star) = \frac{1}{2}$$  \hspace{1cm} (4.76)

which leads to controls being applied with

$$\rho(u_i^\star) = \rho(u_i^\star) = \frac{1}{2}$$  \hspace{1cm} (4.77)

However, further examination of the use of randomized control laws, while important, is beyond the scope of this work.

Thus selfish strategies are useful in acyclic topologies, but cycles lead to inconsistencies which can only be resolved by introducing noisy control laws. In terms of objective structure, the selfish strategy, like the silent one, forscusses on local costs even in the team case. Thus objective structure makes little difference to the applicability of selfish strategies, although the comment of the last section concerning the use of silent coordination between groups or individuals (as a result of the cost of forming and sending predictions) applies equally well here. Table 4.22 summarizes these points.
Table 4.22

Suitability of Selfish Coordination

-Objective-

<table>
<thead>
<tr>
<th>Topology</th>
<th>Team</th>
<th>Group</th>
<th>Individual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy</td>
<td>Good</td>
<td>Good*</td>
<td>Good*</td>
</tr>
<tr>
<td>Singly connected</td>
<td>Good</td>
<td>Good*</td>
<td>Good*</td>
</tr>
<tr>
<td>Doubly connected</td>
<td>None</td>
<td>None</td>
<td>None (unless randomized)</td>
</tr>
<tr>
<td>Acyclic</td>
<td>Good</td>
<td>Good*</td>
<td>Good*</td>
</tr>
<tr>
<td>General</td>
<td>None</td>
<td>None</td>
<td>None (unless randomized)</td>
</tr>
</tbody>
</table>

* May be mixed with silent for intergroup coordination
4.2.4. Strategies with Abstraction

The construction and propagation of abstract models throughout a
domular control system poses unique problems. Flexibility is offered as
such models are separable and can be constructed in a cyclic topology;
difficulties arise as interaction production cost models are much easier
to construct than interaction response cost models, and iteration is re-
quired to improve models generated in a cyclic setting.

Because response cost models are nontrivial only under special
conditions (namely that the responder does not have a completely reach-
able state space) this section will be restricted to the consideration of
the use of cost-to-produce models described in Section 3.4.3.

In a team setting, such models are best suited for use in a hierarchi-
cal structure. The models can be propagated to the root of the structure as
each agent uses received models to limit the uncertainty about incoming
interactions which it must consider, and produces a model based on the
responses it might generate in optimal response to those interactions.
This model can then be passed to the agent it affects. Because this agent
is unique, separability is not an issue.

Section 4.1.6 described a way to coordinate singly connected teams
using both types of abstract model. Such systems can also be coordinated
using only production-cost models if it is realized that such can be re-
placed with somewhat less informative models which are separable. This
process is illustrated in Figure 4.23 for the case of a model constructed
by $D_3$, which affects both $D_1$ and $D_2$. Using the approach described in
Section 3.4.3, $D_3$ can construct an abstract model of itself which bounds
the set of possible $(\rho_{31}, \rho_{32})$ pairs producible (and the associated costs)
Figure 4.23

Separation of Possible Interaction Set
given \( D_3 \) occupies some abstracted state \( \hat{\gamma} \). This is the best possible model in the sense that the set of feasible \((\rho_{31}, \rho_{32})\) pairs is as small as possible.

A somewhat less precise model can be constructed by finding the smallest set of feasible \((\rho_{31}, \rho_{32})\) which is separable (and the smallest intervals which bound the cost of producing a pair which decompose as

\[
\hat{e}(\hat{\gamma}, \rho_{31}, \rho_{32}) = \hat{e}_{31}(\hat{\gamma}, \rho_{31}) + \hat{e}_{32}(\hat{\gamma}, \rho_{32}) \tag{4.78}
\]

where intervals \( \hat{e}_{31}(\cdot, \cdot) \) and \( \hat{e}_{32}(\cdot, \cdot) \) are added in the usual set addition fashion). This can be separated into the two sets which are the original one's projections onto \( P_{31} \) and \( P_{32} \). This produces the tightest separable bound on the producible set and its costs; the separated bounds then comprise independent abstract models which can be sent to \( D_1 \) and \( D_2 \).

For singly connected systems, (and acyclic systems in general) these models can be constructed in a single pass through the system. Inferior domules construct, separate, and communicate models to those they affect. Others repeat the process when they have received models from all that affect them. Finally, the superior domules receive theirs, and the online coordination can take place either silently or (if inner bounding models of possible \((\rho_{31}, \rho_{32})\) pairs are constructed also) with commands propagated from superiors to inferiors.

Thus abstraction is suited to acyclic topologies if the additional uncertainty introduced in the transition to separable models is tolerable. Cyclic topologies pose different, but not insurmountable, problems. Any agent can construct an abstract model of itself assuming all incoming interaction sequences are possible; this can be separated and passed to
those it affects. These can repeat the process, taking into account the constraints on possible incoming interactions which are implied through the received abstract model. If a cycle exists, eventually models will arrive at the original agent which limit the set of possible incoming sequences to something smaller than that it used in its original abstract model construction. Thus it can build a new such model which will, in general, be tighter (more precise) than the original. If this is then passed along to those it affects, an iterative process results in which models are continuously being refined.

Convergence of the resulting algorithm, at least asymptotically, is not difficult to show as each iteration shrinks the sets of possible interactions and costs, so the contraction mapping theorem applies. However, iteration violates the philosophy laid down in Section 2.2. For abstraction to be used in a cyclic topology, then, either the exclusion of iterative algorithms must be relaxed, or the process outlined above restricted to a single pass through the system. If the latter route is taken, questions emerge concerning the best ordering for the abstraction to take, as there will generally be many alternatives.

Thus abstraction is suitable for use in all acyclic topologies, and can be used in cyclic topologies if compromises are made. Turning to the effect of objective structure on its usefulness, it can be seen that the construction of abstract models is similar to the generation of predictions used in the selfish approach. If it is reasonable for an agent, individual or in a group, to supply a model to another agent, individual or member of another group, then the model construction process can work exactly as with the team objective. If it is not, then the agents in a group affected by outside domules must construct their models assuming
all incoming interactions are possible.

It is worth noting that, while the on line silent coordination using these models is unaffected by objective structure due to the separation of agents, the idea of commands generated and sent from affected to affector is limited to use within a group. There is no incentive for an agent acting as part of a group, or as an individual, to comply with outside requests to generate particular interactions. Further, the use of commands is problematical in a non-hierarchical system as there is no agent possessing a unified abstract model of the entire system which can initiate the command generation process.

Table 4.24 summarizes these conclusions. Abstraction is eminently suitable for use with hierarchies, easily adapted for use with acyclic topologies, and can, in fact, be considered for use in the most general settings.

4.2.5. Conclusion

Section 4.2 has presented some of the major results of this work. While details of various coordination schemes are strongly dependent on the system for which they are being used, it is clear that several general conclusions can be drawn regarding the suitability of the various strategies developed earlier to systems of given topology and objective structure.

Table 4.25 summarizes these conclusions. Singly connected systems are suited to the greatest variety of strategies, although hierarchies have an advantage when abstraction is used. Strategies are useful with either acyclic or doubly connected topologies, but only the silent one works well with both (with special exceptions for the static case, and if modifications are performed to the abstraction and static ones).
### Table 4.24

**Suitability of Abstraction**

<table>
<thead>
<tr>
<th>Topology</th>
<th>Team</th>
<th>Group</th>
<th>Individual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy</td>
<td>Good</td>
<td>Good</td>
<td>Good, silent only</td>
</tr>
<tr>
<td>Singly connected</td>
<td>Good, commands need care</td>
<td>Good, commands need care</td>
<td>Good, silent only</td>
</tr>
<tr>
<td>Doubly connected</td>
<td>Fair, iteration suggested, silent only</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Acyclic</td>
<td>Good, commands need care</td>
<td>Good, commands need care</td>
<td>Good, silent only</td>
</tr>
<tr>
<td>General</td>
<td>Fair, iteration suggested, silent only</td>
<td>Fair</td>
<td>Fair</td>
</tr>
</tbody>
</table>
Table 4.25

Match of Strategies to Topologies and Objectives

<table>
<thead>
<tr>
<th>Topology</th>
<th>Team</th>
<th>Group</th>
<th>Individual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy</td>
<td>Static</td>
<td>Static/Selfish</td>
<td>Silent</td>
</tr>
<tr>
<td></td>
<td>Silent</td>
<td>Silent/Silent</td>
<td>Silent</td>
</tr>
<tr>
<td></td>
<td>Selfish</td>
<td>Selfish/Selfish</td>
<td>Selfish</td>
</tr>
<tr>
<td>Singly connected</td>
<td>Static</td>
<td>Static/Selfish</td>
<td>Silent</td>
</tr>
<tr>
<td></td>
<td>Silent</td>
<td>Silent/Silent</td>
<td>Silent</td>
</tr>
<tr>
<td></td>
<td>Selfish</td>
<td>Selfish/Selfish</td>
<td>Selfish</td>
</tr>
<tr>
<td>Doubly connected</td>
<td>Static</td>
<td>Static/Selfish*</td>
<td>Silent</td>
</tr>
<tr>
<td></td>
<td>Silent</td>
<td>Silent/Silent</td>
<td>Silent</td>
</tr>
<tr>
<td></td>
<td>Abstr.*</td>
<td>Abstr./Silent*</td>
<td>Abstr.*</td>
</tr>
<tr>
<td>Acyclic</td>
<td>Static*</td>
<td>Static/Selfish*</td>
<td>Silent</td>
</tr>
<tr>
<td></td>
<td>Silent</td>
<td>Silent/Silent</td>
<td>Silent</td>
</tr>
<tr>
<td></td>
<td>Selfish</td>
<td>Selfish/Selfish</td>
<td>Selfish</td>
</tr>
<tr>
<td>General</td>
<td>Silent</td>
<td>Static/Selfish*</td>
<td>Silent</td>
</tr>
<tr>
<td></td>
<td>Abstr.*</td>
<td>Abstr./silent*</td>
<td>Abstr.*</td>
</tr>
</tbody>
</table>

*With restrictions
+With modifications
Objective structure affects strategy selection greatly: there is no need for any method but abstraction, selfish, or silent coordination in individualistic or intergroup coordination (for the latter, entries in Table 4.25 indicate intragroup/intergroup pairs.

A few notes must be appended to this table.

1. Any strategy which uses selfish or abstraction coordination between groups can, of course, also use silent coordination there.

2. Options which involve the use of randomized controls, such as cyclic selfish, have been omitted.

3. Abstraction indicates the permissible use of abstract models and silent on-line coordination; command coordination is suitable for use in hierarchical teams or hierarchical groups within other topologies.

4. Finite horizon coordination is applicable wherever the static scheme is; the hybrid finite horizon with abstraction is useful wherever both components are.

The main conclusion to be drawn is that the effort presented here has provided a new perspective on distributed decision making as it is done in current organizations, indicates many new directions to move in the search for ways to coordinate the behavior of large, complex systems, and shows that large areas remain unexplored.
Chapter 5. Application Example

Domules may be an appropriate way to formulate a control problem for any large system displaying the modular structure which the formulation of Section 1.2 assumes. The key attributes of this structure are:

1. Many identifiable subsystems.
2. Interactions between subsystems which involve fewer variables than those within a subsystem.
3. Suitability for real time, decentralized control (in terms of supporting processing and communication facilities).
4. A topology which accepts strategies as summarized in Section 4.2.5.

The original motivation behind the choice of the formulation of the domule problem was a desire to understand and improve the functioning of military command and control systems. It is easy to see how the hierarchical structure of these systems fit that of domules in general, and the finite look ahead with abstraction strategy comes close to describing some of the most important features of their operation.

Unfortunately, it is not easy at this point to construct a practical example of this application in order to demonstrate the suitability of the formulation and practical ramifications of the candidate coordination strategies. This difficulty stems from two sources:

1. There is a tremendous variety of subsystems, each with its own unique factors (such as the individual areas of responsibility of the various tactical commanders - where small geographical factors can be of extreme tactical importance and thus must be modelled).
2. It is not at all clear how to model the situations
seen by commanders other than those at the battlefront.
While some model must exist, the variables of interest
(e.g. morale, vulnerability, threat, power) are diffi-
cult to define, measure, and propagate through time.
The example of modular decision making presented here will capture
many of the features of a dynamic resource management system (such as a
command and control system) while avoiding those difficulties listed above
as they alone require tremendous effort to resolve. It will involve a
large, geographically disbursed system composed of many elements, which
interact with one another, but can be classified together into a set of
generic structures with easily defined state variables and models.
The system of this example is that of an integrated terminal company
which receives, stores, transports, and processes grain from its origin
at farms across the Midwest and Plains states to its end users at terminal
markets or export ports at the mouth of the Mississippi. As described
in [37]:
"The integrated terminal company operates terminal elevators
in more than one marketing center. The company may own line
country elevators, river houses, and sub-terminals; it may
be engaged in exporting; it may lease rail equipment; and
it may own barges and boats. While overall policy, as well
as financing of the integrated terminal company, may be pro-
vided by the home office; day-to-day merchandising activities
are carried on by the individual management at each terminal
location."
This chapter will describe how such a process can be modelled in terms of domules and the effects of using the various strategies to coordinate it.

Each element of the system (elevators, railroads, barges, etc.) can be modelled as a domule. Interaction variables describe the flow of grain from one to another in terms of type, class, quality, and quantity transferred. Coordination will be done to manage the day-to-day activities required to meet demands at various points in the system while minimizing overall operating cost. This coordination activity can be viewed as being implemented in a set of mini computers, one per system element, communicating nightly in order to arrange the activities of the next twenty-four hour period.

The remainder of this chapter is divided into four sections. The first provides general background on the sources and uses of wheat, the method of grading it, and various processes that can be performed as it is moved from farm to miller. The second section discusses the global structure of a typical integrated terminal company. Then each domular element can be described, at least in generic terms, along with its relationship to the rest of the company. The last section evaluates the suitability of each of the coordination schemes to this example. However, the major goal of this section is to show that such a system can be modelled naturally and fruitfully in a domular network, while being far too complex for the classical optimal control techniques to be applied.

5.1 Background

The basic problem faced by a grain handling company is that grain production is extremely variable due to seasonal and geographical factors,
while demand is relatively constant. This is compounded by the fact that production is widely disbursed while consumption is mostly located at a few major processing areas. This requires the company to operate storage facilities and a transportation system to coordinate supply with demand through the time and space variations.

The largest cash crop produced in the U.S., both in terms of quantity and value, is corn. However, 65% of this is used as livestock feed [37, p. 14], usually near the area it is produced. The next largest is wheat, with a 1975 production of 2,135,000,000 bushels values at approximately $14,382,000,000, and most of which is used in human food products due to its relative expense. Other produce such as soybeans, hay, sorghums, potatoes, rice, oats, and sugar beets are produced to successively less extents than wheat. This section discusses wheat, but most of the features discussed here can be extended to the distribution and storage of corn, oats, soybeans, etc.

Wheat is produced in almost all parts of the U.S. in some quantity, but by far the major production area includes the Midwest and Great Plains states. Planting occurs either in the fall (so-called winter wheat) or spring preceding the year of harvest. The yearly harvest begins in May in southern states (Texas), and proceeds northward until late September (North Dakota, Minnesota). Major factors influencing the quality and yield of the wheat crop are

1. Weather - rainfall, frost, planting time
2. Insects and disease
3. Soil conditions, including fertilization
When harvested, wheat is either trucked directly to the nearest country elevator, or stored on the farm for a period before being delivered. The choice is often made by customary procedure; for example, in Kansas, most wheat is held in commercial storage, while North Dakota farmers prefer to hold their own crop. In either case, the wheat enters the facilities of the terminal company when it reaches country elevators owned by it.

The demand for wheat stems from two major markets. Over half of the wheat produced in the U.S. is exported, with major export points along the Great Lakes, Northwest Coast (for Oregon and southern Washington produce) and particularly along the Gulf Coast and mouth of the Mississippi. Some wheat is reserved for seed, some for animal feed, but most domestic consumption is in the form of flour. This requires that the wheat be delivered to major milling centers such as Kansas City, Minneapolis, or Chicago. Of the flour produced, almost half is used by the baking industry, another 45% directly by families, and the rest by restaurants and other miscellaneous users. A major by-product of the milling process is bran and middlings which can be used for feed. The terminal company illustrated here will assume responsibility for the grain only to the point where it is delivered to a miller or loaded onto a boat for export; real-life parallels to it often include milling, baking, and packaging facilities as well.

An important aspect of dealing with grain is the method of classifying it. Each type and quality of grain is best for certain uses, and deviation from specific standards can be damaging. The seven types of wheat are:

1. Hard red winter
2. Hard red spring
3. Soft red winter
4. Duram
5. Red Duram
6. White

7. Mixed (less than 90% of one of the above)

and each tends to be grown in an area of favorable climate - making production patterns disbursed in terms of type as well as harvest time and quantity. The grade of the wheat is determined by measuring such factors as

1. Moisture content
2. Fraction of damaged kernels
3. Presence of foreign material (dirt, etc.)
4. Disease
5. Weight per unit volume

The value of each attribute is compared with a standard scale and rated, usually on a 1 to 5 scale. The wheat is then graded as the worst of those values.

Samples are taken at every loading and unloading point, of every car, truck, or bargeful of wheat in order to ensure that it is of the type and quality specified. If a delivery does not measure up to the quality specified by a customer, standard discounts (determined either from market discounts or written into the contract) apply towards the purchase.

The above factors which determine quality suggest the types of processing which can occur at points within an integrated terminal system other than the principle ones of storage and transportation. Grain can
be dried to lower its moisture content, screened to remove damaged kernels or foreign material, or mixed to improve overall quality. As an example of the latter, consider two equal sized lots of grain - the first grading 2 on all factors except damaged kernels, which grades 3, and the second grading 2 except for a grade of 1 on damaged kernels. The first is of grade 3; the second grade 2; mixing them in the right proportion can produce a quantity of wheat grading 2 on all factors, and thus 2 overall, which is more valuable than the original first lot. When all other factors vary also, there is a wide range of mixtures which can be produced - and the problem arises of having to decide whether the cost of doing so is returned in additional profits.

This section has very briefly touched on some essential background for the system description to follow. For an excellent overview in greater depth of the issues involved in handling and processing grain, see [37].

5.2. System Topology

The system structure is basically that of a singly connected graph, with the exception of supply and demand models. Interactions between elements are in terms of the quantity, type, and grade (or qualities) of grain transferred between two elements in a given time period (one day). The objective structure is that of a team seeking to minimize total systemwide operating cost. ('The system will be assumed to be of fixed topology; changes in its structure due to reorganization, purchase or sale of facilities, etc., is managed by corporate headquarters and involves investment decisions made (a) based on many factors outside of those affecting operations, and (b) over a much longer planning horizon. Techniques such as those developed in [38, 39] can assist this process.)
The basic entity which structures the definition of the topology by breaking it into subunits is the bill of lading, which describes the merchandise transferred between two agents of the company. It contains records of the date of the transaction, and a complete description of the grain transferred in terms of amount and quality of each type present. The bill of lading is thus the basis for the definition of the interaction variables between domules representing the subsystems of the company.

The information which describes such a transfer can be represented several ways. For concreteness, assume that the interactions are between a country line elevator and a railroad - extension to other interfaces is quite straightforward. Inspection techniques are based on the assumption that all grain in a car is of the same type, quality, etc. - several samples are taken from diverse points in it and compared. If similar they are mixed together and sent to a laboratory (often at the elevator) for detailed measuring and grading; if dissimilar, more samples are taken until a more accurate representation of the whole is obtained.

The inspection would produce a car-by-car description of the contents - type, weight, and quality listed for each (although the latter would be in terms of its single grade, completeness in the domule model case requires a more thorough description including the individual quality parameters). Railroad practice dictates maximum length trains (typically 100 cars) so each interaction can be described as

\[ \rho_{ij}(t) \in \mathbb{T} \times \mathbb{RR} \times \mathbb{RR}^5 \]  

(5.1)

where

\[ \mathbb{T} = \text{set of types (listed above)} \]

\[ \mathbb{RR} = \text{possible weights} \]
$\mathbb{R}^5$ possible quality ratings on the five scales listed above.

and $\sigma_{ij}(t)$ is a description of the grain transferred from agent $i$ (the elevator) to agent $j$ (the railroad) in day $t$.

With this definition of interactions in hand, the structure of the system represented by the operational arm of an integrated terminal company is as follows. The flow of produce starts at local farms, with grain being trucked to the country elevators either at harvest or after being stored some length of time on farms. A separate model can be defined describing these supply patterns for each elevator which reflects the variables associated with local grain production. (Such models must be linked by common, consistent assumptions about weather patterns, but since the agent associated with each exerts no control over any variables, and thus can only make predictions as to the single course of the future, these assumptions can be incorporated in each supply model rather than requiring a central meteorological model. This will preserve the single-connectedness property of the system without sacrificing generality.)

The interactions between supply models and the next stage, country line elevators, are expressed in terms of type, weight, and quality of the grain delivered by each truck to the elevator in a day. The elevators pass the grain on to railroads, along which they are usually located. The agent associated with the domule modelling each elevator can be thought of as the elevator's operations manager.

A single model is required for the railroad network serving a region which picks up grain from the line elevators and transports it to terminals, sub-terminals, or river houses. This allows consideration of the routing and scheduling problems faced by such an operator.
Figure 5.1 illustrates the relations between the three types of
domule involved in the intital collection of grain by the company. This
structure is repeated many times throughout the region served by it,
creating the model of a large network of facilities.

The railroads unload the grain they carry at either
a. Terminals - serving specific markets or users
b. Sub-terminals - intermediate, large capacity storage
   and processing facilities
c. River houses - for storage and shipment by barge
d. Port facilities - for export.

Since shipment by rail is more expensive than shipment by water, (when
company owned equipment is used, costs are as shown in Table 5.2 [40])
it is reasonable to assume that water transport is used wherever possible.
(This is often not the case when commercial carriers are used as railroads
commonly reduce rates dramatically over routes which parallel commercially
navigable waterways to remain competitive.) River houses are the inter-
face through which grain flows from rail to water carriers, and thus are
a subset of the sub-terminals. These are usually located far from market,
at major transportation crossroads.

Whichever of the four serves as recipient of the grain, the inter-
action variables again describe the type, amount, and qualities of the
delivered grain. Terminals or ports serving domestic or foreign markets
constitute the final step in the integrated terminal company's handling
of the grain - the end user receives the grain directly from them by
truck, conveyor, or boat. This necessitates demand models for the end
users, again interacting with terminals and ports through delivery
Figure 5.1
System Topology:
Supply, Country Elevators, and Railroads
Table 5.2

Average Transportation Costs

(1973)

<table>
<thead>
<tr>
<th>Mode</th>
<th>Cost (Dollars per Ton-Mile)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air</td>
<td>.242</td>
</tr>
<tr>
<td>Truck</td>
<td>.217</td>
</tr>
<tr>
<td>Rail</td>
<td>.017</td>
</tr>
<tr>
<td>Water</td>
<td>.00389</td>
</tr>
<tr>
<td>Pipeline</td>
<td>.00325</td>
</tr>
</tbody>
</table>
variables. These are similar to supply models as agents operating the system have no control over demand - that is handled by the marketing departments. Thus there is one (or more) demand model for each terminal or port facility, as shown in Figure 5.3. (Notice that as there may be more than one terminal served by a railroad, the system structure is not a hierarchy.)

The final segment of the system is that of sub-terminals (river houses), and barges. Sub-terminals receive rail (and some local truck) shipments, and ship out by barge or another rail carrier. Barges operate freely on southern rivers, but icing of the upper Mississippi and Great Lakes is a major factor to be considered in their operation. Sub-terminal domules are similar to other elevators; the model representing a barge system is a complete system model (all barges, tugs, etc., involved) and is influenced by the meteorological factors affecting supply (again in a purely open-loop fashion). This segment is diagrammed in Figure 5.4.

Thus the terminal company is structured to parallel the flow of grain from source to user. The actual physical entities forming the underlying interactions between elements of it are the deliveries of grain from one to another. Note that the reverse transfer of money in payment is not part of the physical process; in a company operating as a unit such "transfer payments" are part of the accounting system which is part of the coordination scheme.

The system structure just described is illustrated in Figure 5.5. It captures fully the grain handling process; however, extensions are possible to include, for example, market dynamics (as country elevator and terminal/port operators influence supply by setting bid or asking prices)
Figure 5.3

System Topology:
Railroads, Terminals, and Ports

from country elevators

terminal elevator (domestic)

domestic demand

terminal elevator (domestic)

port elevator

foreign demand

barges
Figure 5.4

System Topology:

Sub-terminals, Barges, and River Houses
Figure 5.5
System Topology:
Overview
or the use of multiple modes of transportation in parallel. The former results in a doubly connected topology; the latter an acyclic one. Both can be readily handled by coordination schemes as discussed in Section 4.2, but the remainder of this chapter considers this singly connected structure only.

5.3 Basic Elements

The basic module types represented in Figure 5.5 can be grouped into three classes. The first are the trivial (from a control point of view) supply and demand models from which probabilistic predictions of harvests and needs can be made. Of more interest are the processing and storage facilities: elevators, country, sub-terminal, and terminal. Transportation networks, by rail and barge, form the third group.

The supply models can run from very simple to very complex, depending on the level of detail to which weather is included. The simplest model has a state space of a finite (say 52) number of states representing the various stages of growth of the crop. While, on the average, all states would be cycled through in one year, stochastics would allow the state occupancy to advance or retard in order to model variations away from the seasonal norm. Each state would probabilistically map into a daily harvest (which would be zero for all but those representing the critical period just after the crop matures) associated with that stage of crop growth. This formulation allows the generation of predictions of harvest time, yields, and quality which reflect seasonal factors yet vary from year to year. Note that the maps from crop stage to type, amount, and quality (interaction variables from supply to country elevators) would vary from locale to locale reflecting the progression of harvest time from south to north, climatic and soil conditions, etc.
Demand models can be constructed in a similar vein, although the effects of seasonality and geography are much less pronounced. If a given terminal supplies a single customer, such as a large miller, which has very steady demand, then the model can be quite simple - a few states indicating the operational condition of the miller and the demands made in each. If the demand model is an export model, however, then factors such as world economic conditions, foreign harvest levels, etc., must be included. In either case, and all between, it is important to have a model which can provide predictions of type, quantity, and quality of grain required at given times, and define penalties for lateness of delivery, sub-standard quality, or reduced quantity. The latter allow the coordinators of operations to evaluate possible schedules in terms of reduced or lost income due to delayed or sub-standard delivery as well as in terms of in-house operational expense.

The following two subsections will deal with various types of domules which may represent facilities in the other two classes: elevators and transportation.

5.3.1. Elevators

Elevators are the facilities for processing and storing grain. Thus their state variables describe the type, amount, and quality of grain in them at any given time; controls determine how the grain is processed and what is loaded out for departure; interactions define deliveries and are defined by loading.

Almost all grain elevators contain several bins, either the vertical cylinders commonly seen in the Midwest, or horizontal bays in warehouse-like structures. They are equipped to load and unload the various units of transportation (e.g. trucks, barges, etc.) which apply to their operation.
Conveyor belts move grain from unloading docks (after weighing and inspecting) to bins; from one bin to another; from bins to processors (dryers, screeners, etc.) and back; and often to loading docks (which are sometimes served directly from spouts at the base of bins).

**EXAMPLE 5.1:** Terminal Elevator (St. Louis Grain Corp. A) [41]

Capacity: 1,536,517 bushels upright storage
693,487 bushels flat storage

Receiving facilities: Rail, truck, barge

Loading facilities: None

Services: Drying, cleaning, scalping,
automatic sample

**EXAMPLE 5.2:** Sub-terminal Elevator (Fowles Grain Co., St. Louis) [41]

Capacity: 4,000,000 bushels upright

Receiving facilities: Rail, truck, barge

Loading facilities: Rail, truck, barge

Services: Drying, cleaning

The state space for an elevator with B bins is best represented as

\[
\Sigma_i = (T \times \mathbb{R} \times \mathbb{R}^f)^B
\]

(5.2)

where \( T \times \mathbb{R} \times \mathbb{R}^f \) is the same set used in interactions to describe the type, weight, and quality factors of the grain stored in an individual bin.

Thus \( \sigma_i(t) \) describes the contents of domule i, if it represents an elevator, at 12:01 AM of day t.
The dynamical model of the elevator maps the state at the end of one day, plus the effect of controls (movement of grain, processing if any, loadings) and incoming interactions (unloading grain) into the state at the end of the next day. Note that the flow of grain is from unloader to bin (to processor to bin) (to bin) to loading. For an elevator with no processors, the controls can be divided into three classes:

1. Those representing flow from unloader to bins
2. Those representing flow from bins to bins
3. Those of flows from bins to loader.

If the elevator can handle a maximum of $N$ unloadings (e.g. $N$ trucks per day for a country elevator) and $M$ loadings (e.g. $M$ railroad cars per day) then the control can be specified as a set of three matrices

$$\omega(t) \in \mathbb{R}^{N \times N \times N \times M} \quad (5.3)$$

where the first matrix represents transfer fractions from unloads to bins:

$$[\omega^1(t)]_{n,b} \triangleq \begin{cases} \text{fraction of delivery } n \text{ in bin } b \\ \text{at end of day} \end{cases} \quad (5.4)$$

the second is bin to bin transfers:

$$[\omega^2(t)]_{b,b'} \triangleq \begin{cases} \text{fraction of contents of bin } b \\ \text{transferred to bin } b' \text{ during day} \end{cases} \quad (5.5)$$

and the third is loadings:

$$[\omega^3(t)]_{b,m} \triangleq \begin{cases} \text{fraction of contents of bin } b \\ \text{loaded into car } M \end{cases} \quad (5.6)$$

These are, of course, subject to constraints

$$\sum_b [\omega^1_n]_{n,b} = 1 \quad n = 1, \ldots, N \quad (5.7)$$
\[ \sum_b [u^*_b(t) + u^*_b(t)] = 1 \quad b = 1, \ldots, B \] (5.8)

which represent the conservation of grain.

With this control formulation, the state at \( t + 1 \) can be determined from that at \( t \) by combining appropriate linear combinations of weight and quality variables, with the \( \omega \)'s as coefficients, of incoming and outgoing shipments and bins. The type variable associated with each bin is determinable from the 90% pure rule - anything less than 90% of one type is "mixed".

The local cost is a function of the \( \omega \)'s and amount of grain in each bin - state information. Costs are incurred every time grain is moved, as grain is stored (for ventilation to prevent deterioration by heat and danger of dust explosions), and as it is loaded or unloaded. While these costs might be approximated as a linear function of the \( \omega \)'s, it is probably more complicated than this. Data for a typical, individual elevator operation is not available, but Table 5.6 shows average storage costs (excluding fixed or capital costs) for country, terminal, and port elevators. [42]

This is the basic model for a simple country elevator - states describing bin contents, controls describing grain transfers, dynamics modelling the effects on type, quantity, and quality due to mixing, and costs based on transfers, inspection, storage, etc. Extensions of it to other, more complex facilities include:

1. Controls, costs, and state transition functions to model special processors.
Table 5.6

Average Grain Storage Costs

<table>
<thead>
<tr>
<th></th>
<th>Cost per Bushel per Year (Cents)</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country</td>
<td>14.34</td>
<td>55.5%</td>
</tr>
<tr>
<td>Terminal</td>
<td>19.68</td>
<td>51.7%</td>
</tr>
<tr>
<td>Port</td>
<td>24.58</td>
<td>67.4%</td>
</tr>
</tbody>
</table>
2. The effects of overflow - forcing storage of grain outdoors when all bins are full, and the resulting losses due to weather, rodents, and insects.

3. Stochastic modelling to allow for equipment failure, disasters, etc., which are of low probability but could cause severe problems in a tightly coordinated system.

A final point of note is that this formulation, which gives elevators the power to determine loading schedules and transportation units control over unloading times, is consistent with standard grain practice of giving the seller discretion to control transfers. Technically, this will force the transportation units to have costs defined for the case where an elevator tries to load a car or barge when none is present at the loading bay. Such costs will always be avoided in practice by (free) communication between the rail or barge domule and the elevator as the elevator operator can always visually check for the presence of a recipient. This is essentially a form of abstract model cost prediction with two states - present and not present. Although a technical point, this shows how the interactions can truly be defined as being unidirectional if coordinating communication of all types is suppressed from the basic dynamical system models. The rest of the transportation models will be discussed in the next subsection.

5.3.2. Transportation

Each transportation model of a railroad or barge system is similar to an elevator in that the contents of each unit must be described in the state, but location must be included too. Dynamics describe the movement
of units through the transportation network; since it involves movements of both full cars and empties, each network must be modelled by a single agent if a cyclic topology is to be avoided.

Rail is the most efficient overland mode of transportation and hence is used almost exclusively for hauls of over 200 miles. Grain can be moved either in boxcars (roughly 60 ton capacity) or hopper cars (100 tons). As mentioned in the discussion of grading, all of the grain in a car is defined to be in one lot, with aggregate type and quality. Trains usually consist of 50 or 100 cars, engines, and caboose.

The major problems faced by rail operators in moving grain are those of scheduling. It is expensive to haul empty cars long distances; therefore it is best to find cargo which moves in both directions. This is often difficult to do in rural, grain producing areas. Also, maximum movement of grain takes place during and just after harvest, then tapers off through the rest of the year, creating intense pressure on the supply of cars in the fall, but requiring alternate uses in other seasons.

Barges are the preferred way to move large quantities of grain, partly due to the fact that capital investments in waterways are made by the government, not private companies. A typical barge is 195 feet long, carries 1500 tons of grain, and might be a member of a 30 barge convoy pushed by a single 6000 horsepower tug [43]. Typical management of a downstream convoy includes adding barges as they progress, and switching to more powerful tugs at appropriate points.

Aside from the same backhaul problems as experience by railroads, barges face a more serious difficulty: freezing of waterways (particularly the Mississippi above St. Louis and the Great Lakes) from mid-December to mid-April. Combined with spring floods and occasional summer drought,
this mode of transportation faces many more constraints than railroads.
(In fact, the elevator of Example 5.2 apparently exists primarily to
receive grain shipped from the upper Midwest between harvest and the
river freezing, which is transported further downstream when barges
cannot operate further north. [44])

The state variables of either rail or barge subsystem models must
include information on the location and contents of each piece of equip-
ment owned or leased by the company. This can be described by a point
in the space

\[ \mathbb{R}^3 \times T \times \mathbb{R} \times \mathbb{R}^7 \]  \hspace{1cm} (5.9)

where the first two components are geographical location, and the remaining
seven describe the contents in the usual way. All equipment can thus be
specified using a state space of

\[ \Sigma_i = (\mathbb{R}^3 \times T \times \mathbb{R} \times \mathbb{R}^7)^N \]  \hspace{1cm} (5.10)

where \( N \) is the number of pieces of equipment (boxcars, engines, barges,
tugs, etc.) in the subsystem.

Controls available to an agent for these elements are those of move-
ment and orders to unload. Thus

\[ u_i(t) \in (\mathbb{R}^1 \times \{0, 1\})^N \]  \hspace{1cm} (5.11)

where the first two components describe the desired position at the end
of the period \( t \), and for the last component

\[ 0 \Rightarrow \text{don't unload} \]

\[ 1 \Rightarrow \text{unload} \]  \hspace{1cm} (5.12)
for each of the \( N \) units. Constraints on admissible \( \omega_i(t) \)'s can be derived from the locations of cars (states) and the system structure such as routes, regulations, bottlenecks, etc.

Note that unloading authority rests with the carriers, while loading authority lies with the elevators. The precise interfaces between elevators and carriers are thus

1. Loading: the end of the loading spouts above loading bays

2. Unloading: the output of the unloading mechanism

A railroad, for example, can deliver an empty car to a country elevator but it will not be loaded until the elevator is ready; however, a full car delivered to a facility must be unloaded whenever the railroad desires (providing capacity exists to do so) so the car can be returned to circulation. Once unloaded, the grain is the responsibility of the receiving facility.

This elementary description of a transportation subsystem model, for either barge or rail can capture

1. The structure of the transportation network
   (through constraints on admissible \( \omega_i \))

2. The problem of handling empties

3. Routing, scheduling, and capacity constraints

4. Building up or breaking down trains or tows.

This latter is encouraged by the structure of the local operating cost function which is non-linear in the total distance moved by all units in a time period. The non-linearities reflect the cost advantages of moving multiple units (up to a point) over moving many single units.

Other factors which could be included in a more complex model are
1. Meteorological effects - stochastic behavior causing floods (and bridge failures), icing, etc.

2. Inter-relation between grain and other cargoes (fertilizer, chemicals, coal, etc.)

3. The use of alternate modes of transportation between points (e.g. barge in summer, rail in winter) although this might force the topology to be acyclic but other than singly connected.

4. The relationships between the rail subsystems operating in different areas of the country, as in sharing equipment. This would also introduce topological modifications, this time in the form of cycles.

As in all control system design methodologies, one can always refine and extend the system model to greater levels of detail. The formulation presented here allows consideration of far more detail in the state spaces of all domules than could be practically considered in any large scale, centralized approach to system management. Since each facility is considered with all its local idiosyncracies, a much more realistic model exists than is possible with the standardization of form that is often required in centralized approaches. The remainder of this chapter will evaluate the way this model structure reflects on control system design and management.

5.4 Evaluation of Coordination Strategies

The whole purpose of integration in a terminal company for grain handling is to be able to exert more control over the movement of the grain from farm to user, by-passing the problems that can arise when other organizations must be relied upon to supply services. As stated in [43],
"... shippers own their barges and contract with barge operators for towing service. Others acquire equipment through a lease agreement. Similar ownership and lease contracts exist in the railroad industry. With these methods, shippers can more nearly control when and where the shipments are to take place."

Thus the coordination mechanism used with the domular models must provide this advantage: each unit working towards the organizational goal, choosing its controls to best mesh with the activities of others.

The problem is compounded as the dynamic nature of the grain handling process is quite important. This precludes coordination of day-to-day activities based on aggregate flow models (although these [38, 39] are quite useful in looking at longer term planning problems). Again [43] explains,

"Timeliness of service, while in part dependent on the speed travelled between points, relies heavily on the scheduling of departures to meet shippers needs, especially in peak periods. However, there are times when arrival schedules dictate abrupt revisions in departure schedules and in choices of modes. For example, particular shipments may be required to complete the loading of an ocean vessel destined for a market having specific quality requirements."

This requirement for dynamic fine-tuning of the system to meet varying needs effectively rules out some of the coordination strategies described in Chapter 4. For example, the silent strategy would have each
agent minimizing its own operating expenses regardless of its effect on the rest of the system. This results in the ludicrous situation of

1. All elevator operators pass grain on down the system as fast as possible - thus not storing it at all.

2. All transportation managers move grain to the closest unloading point, then stop operating (thereby reducing operating cost to zero for most of the year).

3. Thus terminal and port managers receive an initial influx of much grain, forcing outdoor storage and losses, not matched to demand at all. Excess is wasted; un- filled orders lead to a tremendous loss of revenue.

Thus communication and coordination is necessary; the company is a shambles without it.

The selfish strategy is no more reasonable. With it, behavior is essentially the same, but each agent receives predictions from those that affect it of the highly suboptimal things they will do.

The static strategy can be eliminated from consideration for other reasons. Generation and transmission of production/response cost functions is prohibitively expensive for anything beyond a few weeks horizon due to the large number of possible delivery sequences that must be considered (each sequence describing the types, amounts, and qualities of grain delivered in each day). In addition, supply and demand prediction is highly uncertain beyond a limited horizon, especially in the months just preceding harvest, due to the vagaries of weather, export ship schedules, etc.

These are implementation as opposed to performance considerations, but serve to prohibit the full static coordination scheme.
This leaves the more reasonable, but suboptimal, coordination strategies: finite look-ahead, abstraction, and hybrid. Each will be discussed in succeeding sub-sections.

5.4.1. Short Term Coordination

As mentioned in the discussion of the elevator/transport interface, a certain amount of look-ahead (or abstraction) is unavoidable as the elevator operator will always be able to see whether or not a car or barge is at the loading dock, thus receiving some communication from an agent it affects. "Common knowledge" will inform him that it is not appropriate to send grain to the loading dock when railroad cars are not there. This section will assume a horizon of the order of one to four weeks, significantly longer than that required to assure the above behavior.

The use of finite-horizon coordination with the interaction variables defined in Section 5.2 leads essentially to cost-to-produce/respond functions which are defined over a set of loading or delivery schedules for the horizon of consideration. Each day, tentative schedules for the next T days are set up, assuming the worst possible behavior beyond then. The first day's plans are implemented, then a day later the next T-1 day plans are modified based on the possibilities of excluding that worst case behavior on the T-1st day. For the various element types, this translates into:

1. Supply agents: Since these exert no control over anything, incur no cost, and represent only outside influence on the team, their cost-to-produce function is just a prediction of what short term supplies are likely to be.
(Currently, this role is filled by local, state, and federal agriculture departments.)

2. Demand agents: Similarly, as these represent efforts of the marketing arm of the company, and thereby supplier's needs, they produce a cost-to-respond function that is a list of penalties charged for each delivery schedule from the associated terminal or port (or equivalently, revenues expressed as negative costs).

3. Country elevators: The worst case situation they see is a sudden influx of grain from local farms. Hence they will generate cost-to-produce functions which favor having the elevator empty at the end of the planning horizon. This will favor actions which force grain towards terminals, though not nearly as drastically as in the silent or selfish cases. If the horizon is long enough that the elevator can always be emptied in the later part of it, then needs for specific grain types and qualities can be factored in through its early part. As the horizon grows longer, this effect improves - grain will tend to stay in the country elevators as they are the least expensive storage available.

4. Railroads: These will have to coordinate activities to handle the large flow of grain encouraged by the country elevators at the end of the horizon. Due to the long transit/handling times between points, plus the additional worst-case assumption of continued high supply from
elevators, this will mean the early parts of the planning interval will probably be devoted to getting empties to the country elevators. These empties will be first filled with grain in most demand at later stages of the system, as cost-to-respond functions from terminals favor those.

5. Barges: These operate much the same as railroads, except that the urgings of the country elevators to be empty by day T will propagate through railroads and sub-terminals, thus entering cost-to-produce functions from sub-terminals later than from country elevators to railroads, leading to a longer effective horizon of reasonable planning on the part of the barge agent.

6. Terminals and port elevators: These assume worst case possibilities of a sudden influx of grain beyond T, so will favor making less profitable (more heavily penalized) deliveries in order to get stocks low by day T. This will generate cost-to-respond functions from them favoring no deliveries (except for those meeting very short term, low penalty needs), helping keep grain away from their expensive storage facilities. They can consider processing stocks on hand, also.

7. Sub-terminals: These face essentially the same considerations as country elevators.

Thus the overall effect of the use of the short term coordination strategy is for agents to match demand as best they can over the planning
horizon while continually bracing themselves for the possibility of huge influxes of grain. As the horizon postulated above is short compared to seasonal effects, this leads to poor behavior compared with strategies in which agents know such surplusses will not occur except near harvest time.

An interesting modification to this strategy might be some use of multiple time scales - daily operations planning as above, subject to constraints imposed by monthly aggregate planning to factor in seasonal considerations.

5.4.2. Abstraction

The main function of abstraction in this system would be to introduce knowledge of the seasonality which finite horizon coordination lacks. In return, however, capacity for detailed planning of delivery schedules would be severely impaired.

Much of the performance of abstraction as a coordination mechanism would depend on the size of the abstract models constructed and shared. While this is difficult to quantify for this specific application, it will be assumed that the abstraction is complex enough to at least contain a 365 day "clock" with which seasonal factors can be synchronized.

Each agent would use abstraction as follows:

1. Supply agents: Generate a model which predicts bounds on the quantity and type of harvests which can be expected at each time of year for each local model.

2. Demand agents: Generate seasonal bounds on quantities in demand. Though less complex than supply factors, these could reflect seasonal shutdowns of facilities, export demand, etc.
3. Country elevators: Using the supply models passed to them, plus their own capability to buffer grain flows, models can be formed which show relatively low production costs for all interaction sequences generated except for those which exceed total supply of given types of wheat. Also, they may show that the elevator is forced to send out some grain during harvest time if supply exceeds storage capacity.

4. Railroads: Given production models from country elevators which show no great surge of grain output except during harvest, the railroad agents can construct distribution models which average out irregularities due to geographical factors. In use, these agents will no longer plan to get empties to country elevators as soon as possible, there to wait until harvest, but use them more evenly to shuttle grain from source to sink as dictated by average supply and demand.

5. Barges: These act as railroads, with additional seasonal effects due to the expected freezing season.

6. Terminals and ports: These can accept production models generated by railroads and barges, and plan operations to have a variety of grain on hand before the freezing of rivers while little is in storage between thaw and harvest. Alternatively, they can generate response models which indicate generally even demands as seasonality in demand patterns is tempered by their storage.
7. Sub-terminals: Act as country elevators and terminals depending on their relative locations to barges.

Thus abstraction provides the general coordination of the system required of it to match supply at harvest to even demands, but cannot provide the specific kind required to handle special needs which arise as described at the beginning of this chapter. Since in the high-volume world of grain marketing it is essential to coordinate activities tightly something other than abstraction must be used.

5.4.3. Hybrid Coordination

As expected, use of both finite horizon prediction and abstraction yields performance better than either alone. As the abstraction modifies uncertainty as to worst case events beyond the horizon, eliminating the possibility of a surge in supply except during harvest time, the system is more able to allow decisions to be dominated by short term opportunities.

This is best seen in the case of country elevators. Given predictions of the local supply model of the next one to four weeks production, plus the abstract model of general seasonal effects, it can decide either to send remaining grain down the system to prepare for the new harvest, or to hold grain unless cost-to-respond data from a railroad indicates that shipment of a particular type and quality of wheat will lower expenses by supplying a certain customer's needs.

The other agents act in a way similar to that when finite horizon coordination is used, except that they are not always preparing for a surge of shipments at the end of the period, and seasonal factors such as river icing are predicted and prepared for.
Thus the most suitable coordination mechanism for the automatic
operations management of an integrated terminal company taking ad-
vantage of its modular structure is the hybrid strategy. It shows
promise of producing short term near-optimal decisions which take
advantage of, or prepare for, longer term seasonal effects.
Chapter 6. Conclusion

This concludes the research done to date on this approach to the study of distributed decision systems. This chapter will briefly summarize the major points and results obtained thus far, and sketch some of the many existing lines of work which remain to be explored.

6.1 Summary

The major contributions of this work are:

1. The formulation of a distributed, dynamic decision problem using only knowledge of the system structure local to each agent.

2. The use of uncertainty, and the related best worst-case decision strategy, to supplement knowledge provided from other agents about future capabilities.

3. The collection of several classes of coordination strategies, including some which use "high level" abstracted models of a system external to a decision agent.

4. An analysis of the system topologies for which each class is appropriate, and

5. An example of a dynamic resource management system to illustrate the structure described in (1) and the effects of the strategies of (3).

In slightly more detail, the foundation of this work is the belief that decentralization of control strategies is necessitated in practice by the fact that the system to be controlled is far more complex than that which can be handled by any available decision making mechanism. This
leads to a formulation where each decision maker is an expert on some subsystem, and coordinates its decisions with those of others by evaluating the effects of interactions generated by them on its subsystem, and vice versa.

Formally, it is assumed that each agent's model is exclusive to it; no knowledge is shared unless transmitted through the coordination scheme. The possibility of one subsystem affecting another is captured in a graph relating affectors to affectees. Each agent's model consists of a local "state" space, and a state transition function that allows future states (and outputs) to be determined given a sequence of local controls and external interactions.

Two consequences of this formalization are that
1. There is always a centralized model which is equivalent to the distributed model, and
2. The formulation can be extended to include stochastic as well as deterministic models, provided conditional probabilities of sequences of interactions are used in local state prediction.

The next step in the development of this problem was the examination of techniques to solve the local control problems which arise when particular interaction sequences affecting an agent are assumed. The realization that such sequences are increasingly numerous as their length increases led to a search for techniques which work forward in time and hence require as little specification of interactions as possible. By comparing two decision alternatives and the cost-to-reach functions associated with them,
and selecting the one which has the lowest cost-to-reach every state at
some point in the future, an optimal decision can be made. The same
approach can be used in stochastic problems also.

A final step in the development of techniques to aid coordination
was to investigate ways of constructing simplified, incomplete, yet still
useful models of a decision problem which could be used by others ex-
ternal to it to predict the behavior of the agent solving the problem
without having to simulate it completely. This idea was developed in
stages: for static, dynamic, and modular problems. An interesting fact
to emerge is that if one agent affects another, then a non-trivial abstract
model of the latter can be constructed for the former only if the former
can generate interactions which force the latter into a proper subset of
its state space.

With these techniques in hand, several coordination mechanisms
were proposed and examined. The most general system structure amenable
to use of all of them is that of a singly connected team, where removal of
any interaction path produces two independent subsystems and all local
costs are added to form the overall objective.

The baseline coordination strategy which achieves the optimal cen-
tralized performance is one which involves generation of cost functions
for the production, or response, to all infinite length interaction se-
quenues. By passing these cost-to-produce or cost-to-respond functions
from agent to agent, optimal interaction sequences can be selected by
a series of local optimizations in a reverse sweep. While leading to
an optimal solution, communication and processing requirements are
relatively high unless the system possesses special structure.
Another strategy at the other pole of the spectrum assumes no communication is possible between agents. Then each must select local controls to minimize local costs assuming worst case interactions impinge upon it from its neighbors, and without any way to consider its effect on any models affected by it.

A first compromise can be made between the two by allowing communication from affectors to affectees where the former plans its decisions and passes predictions of the resulting sequences to those they affect. This reduces the uncertainty faced by the latter, thus lowering the maximum local cost it can incur and thus improving overall system performance bounds. However, it is important to note that performance bounds get better - actual performance may get worse if the silent strategy happened to be very nearly optimal.

A more practical compromise can be reached by limiting the planning horizon to some finite extent, and optimizing over this near term. This results in the computation and communication of costs over finite length sequences, but interactions beyond this horizon must be assumed to take place in the worst case way in order for the strategy to avoid disaster.

Another way to reduce the uncertainty about the system external to it is to pass an agent simplified models of that system. These allow the agent, if not to predict interactions affecting it, then to eliminate some possibilities and thus improve its behavior. As these models need to be constructed only once, this strategy requires no real-time communication.

Finally, one can combine these last two schemes into one which uses complete cost evaluation over a finite horizon, and then the abstract models to reduce uncertainty beyond it. This provides the advantages of
both while requiring, of course, the communications of both.

For each of the six strategies, one finds that there is a certain class of system topologies on which it can be used. The topological factor which leads to difficulty is that of cycles - although special cases of cycles can sometimes be handled. Applying the silent strategy to any topology works well; the selfish, prediction only scheme leads to difficulties similar to those encountered in games when cycles are present. The fact that cost-to-produce and cost-to-respond functions are not separable in their arguments when they reflect production of or response to interactions of more than one other agent leads to difficulties in any topology other than the doubly connected class. Production of abstract models, however, can be modified so that such models are separable; this leads to use of abstraction in quite complex topologies if iterative refinement of them is permitted.

For other objective structures other than that of a team, any of the above schemes work well within a cooperating group provided the internal topology of the group is amenable to it. Inter-group coordination seems limited to the silent, selfish, or abstraction strategies at this point and require that one group have the proper incentive to provide others with predictions or models of its activities.

These strategies were illustrated in terms of a practical application to a dynamic resource management system. The system was that of an integrated terminal company which transports and processes grain received from diverse suppliers to fulfill the needs of its several customers. Its structure can be represented as a singly connected system, starting with supply models for the farms located near each country elevator owned
by the company and ending with demand models for each customer or market. Intermediate agents are responsible for each facility owned or leased by the company - elevators: country, sub-terminal, and terminal; and transportation facilities: railroad equipment and barges. It was shown that the static, silent, and selfish strategies were not well suited to the coordination of such a system, and that elements of both the finite horizon and abstraction schemes were necessary.

Finally, this work concludes with a discussion of the needs for future work to extend these ideas, which follows in the next section.

6.2. Future Work

As emphasized in Chapter 1, the goal of this effort was to explore new approaches to the idea of the distributed coordination of large, complex systems. As such, it has raised many questions which should be answered for the sake of completeness or for practical demonstration of these concepts. Such questions fall into two classes: those relating to specific issues involving specialization, refinement, or application of the specific ideas presented here, and those on a more general, conceptual level.

6.2.1. Specific

The direction of work dealing with specific concepts presented here which would be most profitable are those of

1. Extending the ideas to systems of specific structure
2. Continuing the development and use of abstraction
3. Delving into a practical example and studying implementation details and issues with it.
4. Improving various technical analyses (e.g. forward search convergence).
As is the case of other modern control theoretic techniques, it is expected that the concepts explored here can be applied at a general level in many systems but the addition of assumptions of special structure to the system models may generate far more practical algorithms for implementation. Obvious assumptions which may provide this advantage are linear dynamics, linear or quadratic cost functions, or Gaussian or Poisson stochastic processes driving the system. The goal of the study of these properties would be mainly to find ways to parameterize the cost to produce/respond functions to reduce the communication required to transmit them in the finite horizon algorithms.

However, there are other assumptions of model structure which achieve this goal in practice, but through a completely different approach. It is highly likely that many natural, and almost all engineered, systems have modular formulations in which many states are not affected by, or generate the same, interactions. Only in "exceptional" circumstances does the state leave this region. For example, if a modular approach to large scale fault monitoring and detection is used, each element would usually be operative or suffer faults of local concern. Only rarely would failure in one element lead to states which produce interactions which increase the likelihood of faults in a neighbor.

Another approach to the same end lies in developing "high level" descriptions of feasible interaction sets and their costs. For example, a grain elevator with no special processors and only one type of wheat, the type grown locally, need not list all possible loading schedules in order to coordinate with a railroad; rather, it only needs to communicate the total amount it has available. This allows the railroad to reconstruct
loading schedules - any which supply more than this amount are infeasible, all others are feasible.

These last two approaches once again lead to ideas involving modelling of elements of the real world and descriptions of activities often appearing in the machine intelligence literature. More will be said about this in the next section where the relationship between domules and frames will be mentioned.

Also strongly related to machine intelligence is the idea of abstraction developed in Chapter 3. The idea of generating a simplified model of a complex task which omits detail but conveys general structure has been appealing for quite a while, but this is a novel approach to generating such models by introducing uncertainty to replace lost detail. Much can be done with this - identifying properties of problems that can be usefully abstracted, exploring ways in which an abstracted model can be linked with detailed sub-models (in a domular structure?) to provide the generalities as well as details when needed. Of course, there are again needs to discover what sort of results abstraction produces when applied to systems of special structure - linearity will probably not be an advantage, but boundedness will.

The third major area of future work should be in applying these concepts to a practical problem far more thoroughly than was possible in Chapter 5. This would reveal processes that could be modelled in this structure through clever definitions of states and interactions that would not seem suitable at first glance; it would also find problems that while seeming to display a domular structure, are completely unsuitable for its application. This insight is necessary if a domular methodology is ever to be widely useful.
Finally, there are many technical details which remain to be developed as they were passed over to get a full set of concepts together - structure, local decision making, abstraction, and global coordination. These include

1. Forward search convergence criteria
2. Stochastic forward search conditions
3. Abstraction of cost-to-respond functions with active feed-back from the responder

6.2.2. General

The general directions of future work are two: use of domular structures for estimation only, and using this work as a stepping stone to a new theory of systems management based on model structures which combine accurate system representation with ease of deriving controls.

One assumption made in the formulation of Section 1.2 is that every agent has perfect knowledge of its local state and those interactions affecting it at each point in time. Relaxing this assumption and retaining emphasis on the control problem produces difficulties even in the centralized case. However, focusing on the estimation problem is likely to be much easier. Applications for which this would be useful even without the decision strategies would be command and control, large system fault detection, and various other monitoring or surveillance systems.

The more fascinating area of endeavor would be one which combined the domular approach of modelling sub-systems and interactions, the strategies for decision making which accompany the, and development of the abstraction concept with the intuitive appeal of frame systems as a
way to represent knowledge about the world in general. This could permit
the addition of real-time model identification into a domular/frame struc-
ture, with components of the model selected for ease of identifiability,
decision support, and usefulness in improving performance. If one be-
lieves that any intelligent entity attempts to control a system (its
environments) to meet some performance objectives, subject to constraints
on the decision making power it can bring to bear on the problem, then
it is clear that the unification of a domular approach to practical
system management coupled with artificial intelligence approaches to
knowledge representation and acquisition could lead to significant pay-
off in the development of automatic, intelligent control of large, complex
systems.
References


17. I. Kant, A Critique of Practical Reason

18. Luce and Raiffa. Games and Decisions,


28. C. E. Lindbom. The Intelligence of Democracy: Decision Making
Through Mutual Adjustment


30. E. A. Feigenbaum and J. Feldman, eds. Computers and Thought,

June, 1974.


34. I. B. Rhodes and D. G. Luenberger. "Stochastic Differential Games
with Constrained State Estimators", IEEE Tran. Aut. Con., Vol. AC 14,

35. N. R. Sandell, Jr., and M. Athans. "A Finite State, Finite Memory

scriptor Variable Approach to Modeling and Optimization of Large

37. L. Besant, D. Kellerman, G. Monroe, eds. Grain: Processing, Production
Marketing, Chicago Board of Trade, 1977.

Food Processing and Distribution Sector, Phase A", Systems Control

Food Processing and Distribution Sector, Phase B", Systems Control


Biographical Sketch

Bob Tenney was born in Cincinnati, Ohio in July, 1954. After living there for seventeen years, he spent a year in Newcastle-upon-Tyne in England. He then went to the Massachusetts Institute of Technology, where he received a B. S. in Computing Science and Engineering and M. S. E. E. simultaneously in 1976. While there, he became a member of Eta Kappa Nu, Tau Beta Pi, Sigma Xi, and several societies of the I.E.E.E. His graduate work was partially supported by a National Science Foundation Graduate Fellowship.

While in school, he joined a cooperative education program with the Naval Surface Weapons Center in Silver Spring, Md. There he did work on acoustic signal processing and target localization which led to his Master's thesis. Later, he spent two summers at the Honeywell Systems and Research Center in Minneapolis, Mn. with the systems and control group. This time was devoted to large scale system theory development and application to command and control systems, and also the the definition of a semiautomated systems engineering methodology. His last year at M.I.T. was spent expanding the effort on command and control systems.

In September, 1979, he will join the faculty of M.I.T. as an assistant professor of electrical engineering. He will continue research into the development of techniques for complex problem solving, structured in a formal framework, as well as other interests in the areas of systems control and estimation.