

Controlling Networks with Collaborative Nets

**A Dissertation
Submitted to the Graduate School
in Partial Fulfillment of the Requirements
for the Degree of**

**Doctor of Philosophy
in
Electrical and Computer Engineering**

by

Eduardo Camponogara

**Carnegie Mellon University
Pittsburgh, Pennsylvania**

September 2000

Copyright © 2000 by Eduardo Camponogara

All Rights Reserved

Abstract

Networks, such as the electric grid, are operated by sets of agents that are heterogeneous, local and distributed. (By “heterogeneous” we mean that the agents can range from simple devices, like relays, to very intelligent entities, like committees of humans. By “local and distributed” we mean that each agent can sense only a few of the network’s state variables and influence only a few of its control variables.)

We are concerned with two issues: the quality and speed of decision-making by heterogeneous, local and distributed agents. For quality, our standard of comparison is an ideal, centralized agent, which senses the state of the entire network and makes globally optimal decisions. (Of course, such a centralized agent is impractical for large networks. Even for today’s best hardware and software, the decision-making task is unmanageably large, let alone the real-time controls.) For speed, our goal is to allow completely asynchronous decision-making, so that all the agents can work in parallel, each at its own speed, and in particular, the agents that need to make decisions quickly can do so without having to wait for the slower agents.

Thus, the two questions of concern here are:

- Can heterogeneous, local and distributed agents arrive at decisions comparable in quality to those of an ideal, centralized agent?
- Can these agents arrive at these decisions by working asynchronously?

This dissertation provides evidence (some analytical, some empirical) for affirmative answers to both questions.

For labor division, we extend the rolling horizon approach (model predictive control) to break up the overall control task, leaving each agent with a series of much smaller and localized, static optimization problems, each approximating a part of the global task over a time horizon.

For teamwork, we assemble the agents in collaborative nets or C-Nets (organizations of agents devoid of supervision), whereby the agents communicate locally, and use a collaboration protocol to steer their efforts toward solutions to the overall task. To this end, proximate exchange stands as a basic, yet powerful protocol that can be elaborated (in a number of ways) to best suit the needs: it can, in one instantiation, be shown to guarantee convergence to a solution to the overall task.

For analysis of the research questions, we put collaborative nets to the test in small, but prototypical networks (arrays of pendulums) and standard electric grids. Therein, in simulated incidents, the local and distributed agents consistently arrive at good solutions, thus, corroborating our beliefs.

For the assembly of C-Nets, we put together models for maximizing the agents’ regions of influence, while minimizing communication, and assess their merit empirically.

Acknowledgments

I thank my advisor, Sarosh Talukdar, for taking me as a graduate student and believing that I would successfully complete a doctoral degree. Sarosh has emphasized the importance of general, abstract questions in research and allowed me a lot of freedom to search for their answers. Without him, this work would not have been completed.

I am also indebted to the other members of the thesis committee: Bruce Krogh (ECE), Andrew Moore (CS), and Granger Morgan (EPP). Bruce introduced me to the field of controls, guided my first steps, and has been always available for discussions. Andrew was enthusiastic about my work and brought forth important issues that open up new research directions. Granger has been a father's hand and confident about the success of this work.

I want to express my gratitude to Pedro de Souza, my former advisor and friend from State University of Campinas, Brazil, who was very influential to my personal and professional lives. He encouraged me to pursue a doctoral degree and, without his help, I would not have come this far.

I will always be thankful to my best friend, Kathie Porsche. She made my stay in the US much smoother, introduced me to cool people, was a mother's hand, and transformed me into an enthusiast of outdoor sports.

Stephen Chen, my former colleague and friend, was a source of inspiration, who showed me what graduate school is all about and lifted my morale in critical moments. To him, I extend my thanks.

My roommates were positive influences on my journey through life in the US, too. From each of them, I learned a great deal about life, cultures, and science. I thank Priyadarshree Mathur, Lars Baerentzen (in memory), Sanjay Sachdev, and Jeremy O'Dell for helping me to become a better person.

I could not forget to thank Lynn Philibin and Elaine Lawrence from ECE, and Christina Cowan and Rhonda Moyer from ICES. They are very competent professionals and always happy to help the students.

Finally, I extend my gratitude to friends and colleagues for their unlimited supported: Haoyu Zhou and Song Zhang, Michael Cumming and Cornelia Peckart, Marge Widmeyer and Rudy, Miroslav Velej,

Brian Robick, Kathy Miskinis, Matt Drown, Alexandre Cunha, Evandro Gouvea, Sheyla Pontes, Marcel Bergerman and family, and Hao-Chi Wong.

This research was supported in part by the Institute for Complex Engineered Systems (ICES) at Carnegie Mellon University (CMU), the Pennsylvania Infrastructure Technology Alliance (PITA), and the Engineering Power Research Institute (EPRI) and the Army Research Office (ARO) under contract number EPRI-W08333-05. The author was supported in part by a graduate fellowship from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil, under grant number 20.0047/96-5, the Institute for Complex Engineered Systems, and EPRI/ARO.

The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the funding agencies.

Table of Contents

1. Introduction.....	1
1.1 Motivation	1
1.2 Overview	1
1.2.1 A Framework for Specifying the Agents' Tasks.....	2
1.2.2 Collaboration Protocols	4
1.2.3 Experiments and Applications	6
1.2.4 Matching Agents to Subproblems	8
1.3 Outline.....	9
2. Collaborative Nets	11
2.1 Networks and Dynamic Control Problems.....	11
2.1.1 Traffic Networks.....	12
2.1.2 Power Networks.....	13
2.2 The Rolling Horizon Formulation	13
2.2.1 The Static Optimization Problem	15
2.3 Decomposing the Static Optimization Problem.....	16
2.4 Collaborative Nets.....	18
2.4.1 Terminology	18
2.4.2 Definition	19
2.5 Related Domains.....	19
2.5.1 Asynchronous Teams	19
2.5.1.1 Similarities between A-Teams and C-Nets	21
2.5.2 Natural Organizations.....	22
2.6 Graphical Representation for Collaborative Nets	24
2.7 Summary	27
3. Collaboration Protocols.....	29
3.1 Attributes of Collaboration Protocols	30
3.1.1 Types of Iterative Solutions	30
3.1.2 Types of Decision-variable Updates.....	32
3.1.3 Types of Subproblem Modifications	33
3.1.4 Use of Automatic Learning.....	33
3.2 A Taxonomy of Collaboration Protocols	34
3.3 The Voting Protocol.....	35
3.3.1 A Simple Best-Case Analysis of Voting	35

3.4 The (Serial) Proximate-Exchange Protocol.....	37
3.5 The (Serial) Proximate-Exchange Protocol with Mutual-Help.....	41
3.6 The Asynchronous Proximate-Exchange Protocol.....	44
3.7 Summary.....	44
4. Using Resource Margins.....	47
4.1 Introduction.....	48
4.1.1 The Implementation of Resource Margins.....	48
4.2 Sufficient Conditions for Feasibility.....	49
4.3 Resource Margins for Ellipsoids.....	51
4.4 Computing Resource Margins.....	53
4.5 Examples.....	56
4.5.1 Example 1.....	57
4.5.2 Example 2.....	59
4.6 Resource Margins for Speed-Dissimilar Agents.....	60
4.7 Summary.....	63
5. Experiments in Forests of Pendulums.....	65
5.1 Forests of Pendulums as Prototypical Networks.....	66
5.1.1 The Dynamic Equations of Pendulum Motion.....	68
5.1.2 The Dynamic Control Problem.....	70
5.2 Relaxing the Sufficient Conditions: Experiments.....	71
5.2.1 Overview.....	71
5.2.2 Specifics.....	71
5.2.3 Results.....	73
5.2.4 Computer-Implementation Details.....	75
5.3 Improving Performance with Mutual-Help.....	76
5.3.1 Implementation Details.....	76
5.3.2 Experimental Results.....	77
5.4 Contrasting C-Nets with Traditional Control.....	78
5.4.1 Feedback Linearization.....	78
5.4.2 Experimental Results.....	79
5.5 Summary.....	81
6. Experiments in Electric Power Systems.....	83
6.1 The Transient Stability Problem.....	84
6.2 The Experimental Set-up.....	85
6.3 Experiments in the IEEE 14-bus Power Network.....	92
6.4 Experiments in the IEEE 30-bus Power Network.....	94
6.5 Discussion.....	96
6.6 Summary.....	97

7. Matching Agents to Subproblems	99
7.1 Model Assumptions.....	100
7.2 Maximizing the Influences on Proximate Variables	101
7.3 Maximizing the Influences on Proximate and Neighborhood Variables	103
7.3.1 The Hardness of Computing Representative Subsets of the Noninferior Matches ...	104
7.3.2 Implications of the Computational Hardness.....	107
7.3.3 Accounting for the Direct Communication between Agents	108
7.3.4 Accounting for the Influences on Control Variables.....	108
7.4 Putting a Matching Model into Action.....	108
7.4.1 The Experimental Set-Up	109
7.4.2 Computing Matches	111
7.5 Summary	114
8. Conclusions.....	115
8.1 Contributions	116
8.2 Extensions and Future Research.....	117
Appendix A. More on Collaboration Protocols.....	119
A.1 The (Serial) Proximate-Exchange Protocol	119
A.2 The Proximate-Exchange Protocol with Mutual Help.....	121
Bibliography.....	123

List of Figures

Figure 1.1: An illustration of the solution approach to dynamic control problems.....3

Figure 2.1: The structural representation of an A-Team designed to solve the traveling salesman problem. The A-Team is made up of two memories (rectangles) and various agents (arcs). Agents read objects from the memory at their tails, and write the results in to the memories at their heads. One memory accumulates complete solutions (permutations of the cities to be visited by the salesman), and the other accumulates partial tours (partial permutations of the cities). The agents LK and CLK are versions of the Lin-Kernighan algorithm. OR is the OR-Opt algorithm. Dec creates a partial tour by picking only the edges that are present in two complete tours. AI is the arbitrary insertion algorithm. D_1 and D_2 are the destruction agents that keep the size of the memories under limits.21

Figure 2.2: Two snapshots of a flock of birds flying in formation. The birds swerve around the obstacle, Fig. 2.2a, and carry on their flight, Fig. 2.2b. Each arrow indicates the direction of a bird's velocity. The dotted lines illustrate the local perception of all birds—a bird sees another when a dotted line links the head of its arrow to the other bird's. The snapshots are from a flock simulation and animation that we carried out with Reynolds's software package (Boid Simulation Package).24

Figure 2.3: The variable-connection graph. The nodes correspond to the state and control variables of the static optimization problem, (P). The edges model the coupling between these variables, that is, two nodes are linked when their variables appear in one constraint of (P), or in one term of its objective function.26

Figure 2.4: The variable-perception graph of exact and near matches. The nodes correspond to agents of the C-Nets and variables of (P). The *solid, undirected edges* represent the communications among the agents. The *solid, directed edges* point to the proximate variables. The *dashed, directed edges* point to the neighborhood variables. The *dotted, directed edges* point to the remote variables (that are relevant for the C-Net to induce an exact match).....26

Figure 3.1: A tree of attributes for collaboration protocols and some plausible values.....30

Figure 3.2: A partial tree representation of the taxonomy of collaboration protocols.....34

Figure 3.3: The collective decision-making ability, $p^*(n)$, for $n = 1, 3, 5,$ and 9 collaborative agents. The plot shows that $p^*(n)$ increases monotonically as the number of collaborative agents increases.....37

Figure 4.1: The diagram of the procedure for computing resource margins.54

Figure 4.2: The feasible region, and the contour lines of the objective function, of the overall problem. The inner polytope encloses agent-2's feasible space, when its resource margins are in place.58

Figure 4.3: The trajectories of three solutions found by the C-Net. The agents run proximate-exchange asynchronously and implement resource margins. Beginning with a feasible solution, they consistently converged to the optimal solution without incurring constraint violations.....58

Figure 4.4: The feasible region and the trajectory of the solutions traced by the C-Net. The agents run proximate-exchange asynchronously, help one other explicitly, and use resource margins to retain the feasibility of (P).60

Figure 4.5: The minimum, expected usage of resource R (by either agent-1 or agent-2) as a function of p_2 (the probability that agent-2 finishes its computations ahead of the due date). The minimum, expected resource-usage under the suggested margins, $R^*(p_2)$, is always higher than that under

identical margins, $R(p_2)$. That is, the collective performance is higher when the agents use the suggested margins.	62
Figure 4.6: The suggested resource margins, $\alpha_1^*(p_2)$ and $\alpha_2^*(p_2)$, as functions of agent-2's speed (the sluggish agent). The resource margin of agent-2 decreases, and agent-1's increases, as the speed of agent-2 drops. Therefore, agent-2 compensates for its low speed with the allocation of large chunks of the resource.	62
Figure 5.1: A typical forest of pendulums. Nine pendulums are distributed over a 3x3 uniform grid and coupled by springs.	67
Figure 5.2: The illustration of an incident in a forest of pendulums. A disturbance at 5s throws the pendulums off balance, forcing them to oscillate helplessly when the agents are dormant. The plots trace the angles of the rods over time.	68
Figure 5.3: The generalized coordinates and force-decomposing directions. The coordinates are spherical, and given by the swing (ϕ) and rotational angle (θ). These angles and their first derivatives constitute the state of the pendulums. The dynamic equations are obtained by a) projecting the acting forces onto the decomposing directions $\{n_\phi, n_\theta, n_0\}$, b) evaluating the resulting torques, and c) invoking Newton's second law.	69
Figure 5.4: An inexact match of agents to subproblems in a forest of four pendulums. Fig. (a) shows the arrangement of pendulums in a uniform grid and the springs that link all pairs of pendulums. Fig. (b) depicts the variable perception graph: 1) the <i>solid, undirected edges</i> represent communication links; 2) the <i>solid, directed edges</i> point to the proximate variables; 3) the <i>dashed, directed edges</i> point to the neighborhood variables; and 4) the <i>dotted, directed edges</i> point to the remote variables (that are relevant for the C-Net to induce an exact match). Each subproblem (P_m) is dependent on variables that neither agent- m nor its neighbors can measure or set.	72
Figure 5.5: The mean value of the C-Net penalty versus the number of pendulums for three amounts of resource R . The agents work asynchronously, impose margins on the constraints that divide the resources evenly, exchange information with agents in the vicinity, and do not induce an exact match. The plots show that the C-Net penalty is low when the resources are abundant, or moderate, but high when they are scarce.	75
Figure 5.6: The mean value of the C-Net penalty for an amount $R = \sqrt{5}$ of resource units versus the number of pendulums. The agents display mutual-help behavior: the least troubled agents (those facing easier problems or needing less resources) tighten their margins and allow the most troubled ones (those facing hard problems) to allocate more resources.	77
Figure 5.7: The performance contrast between collaborative nets and feedback-linearization controllers. The plot shows the ratio of the performance of the C-Net to that of the FLC, $f(\text{C-Net})/f(\text{FLC})$, as a function of the control-input cost, b , which is given in log-scale. The plot shows that C-Nets outperform FLCs when energy is costly.	80
Figure 5.8: The recovery of the synchronous mode under feedback-linearization control. The figure plots the angle between each pendulum's rod and its vertical axis as a function of time. The FLC immediately brings the pendulums back on the synchronous trajectory.	80
Figure 5.9: The recovery of the synchronous mode under the control of a collaborative net. The figure plots the angle between each pendulum's rod and its vertical axis as a function of time. The agents hold off their control actions until the angles become small enough for the inputs to be cost-effective.	81
Figure 6.1: The steps to formulate dynamic control problems and assess the quality of the solutions.	86

Figure 6.2: The simplified circuit of power-network PN-5. The FACTS are modeled as current sources whose phase angles (δ 's) are adjustable. PN-5 was designed by adding the FACTS devices to the network of Example 7.9 of [GS93, 7.4, pp. 271-275].	87
Figure 6.3: The transient response of network PN-5 to a non-severe contingency: the disconnection of line 4-5 from 1 s to 1.10 s. The incident knocks the generators off balance; however, they recover synchronization and gracefully approach another operating point.	90
Figure 6.4: The unstable response of power network PN-5 to a severe disturbance: the temporary disruption of line 4-5 and the short circuit of bus-2, both from 1 s to 1.5 s. Gen-1 speeds up faster than Gen-2 does; they lose synchronization and are eventually shut down.....	91
Figure 6.5: The one-line diagram of the IEEE 14-bus circuit with five FACTS devices. The diagram illustrates the locations of the generators, FACTS devices, loads, and transformers. With exception of buses 1, 7, and 8, all buses have loads plugged in.....	93
Figure 6.6: The performances of the collaborative net, relative to the centralized controller's, in the IEEE 14-bus network. The evaluation of the solution found by the C-Net, $f(\text{C-Net})$, exceeded the one found by C_1 , $f(C_1)$, in each contingency. This excess is called cumulative C-Net penalty, and defined as $[f(\text{C-Net}) - f(C_1)] / f(C_1)$. The mean value of the cumulative C-Net penalty is under 10%.	94
Figure 6.7: The one-line diagram of the IEEE 30-bus circuit with six FACTS devices. The diagram shows the sites of the generators, FACTS devices, and transformers. Almost all buses have loads hooked up to.	95
Figure 6.8: The performances of the collaborative net and the centralized controller in the IEEE 30-bus network. The chart shows the cumulative C-Net penalty, $[f(\text{C-Net}) - f(C_1)] / f(C_1)$, for each contingency.....	95
Figure 7.1: A view of the fully connected six-pendulum forest. Only the springs linked to pendulum-3 are depicted however.	111
Figure 7.2: The values of the objectives of dominated and nondominated matches. Each point p_m corresponds to a nondominated match, that is, one whose objectives cannot be improved simultaneously. Each point r_m corresponds to a dominated, randomly generated match.	112
Figure 7.3: The penalties incurred by the C-Nets that were induced by the nondominated matches and dominated, randomly generated ones. The C-Net penalty is defined as follows: $[f(X_{\text{CN}}, U_{\text{CN}}) - f(X_{\text{C}_1}, U_{\text{C}_1})] / f(X_{\text{C}_1}, U_{\text{C}_1})$, where $(X_{\text{C}_1}, U_{\text{C}_1})$ is the solution found by C_1 , $(X_{\text{CN}}, U_{\text{CN}})$ is the solution found by the C-Net, and f is the objective function in the rolling horizon formulation.....	113

List of Tables

Table 7.1: The stiffness of the springs in the forest of pendulums. The coupling between network components, as modeled by the stiffness of the springs, diminishes with distance. The magnitude of the stiffness ranges from a maximum of one unit (when the pendulums are adjacent) to a minimum of two tenths of unit (when they are far-off). 110

Table 7.2: The costs of communication from agents to pendulums. Entry (m,n) shows the cost $w(m,x_n)$ to establish a communication link from agent- m (sitting at pendulum- m) to pendulum- n . The value of $w(m,x_n)$ was generated by adding a random number, drawn uniformly from $(-1,1)$, to the “distance” between the pendulums, $|m - n|$ 110

Chapter 1

Introduction

1.1 Motivation

Collaboration and automatic learning are becoming new dimensions in the arsenal of problem-solving techniques, especially in distributed control and optimization [GMT96][TC00]. Today, the interest in collaboration [Gro96][Syc98] and learning [Mit97] during the design of solutions is increasing and, not infrequently, they are implemented in practice. This trend is not incidental, but rather a consequence of the existing theories, algorithms, and tools that have been developed by a great deal of research. There exists, however, need and opportunity to expand the use of collaboration and learning in large, geographically distributed networks. The need originates from the difficulties of coping with the ever changing contexts and the pressure from competitive markets. The opportunity lies in the potential of improving performance, adaptability, robustness, and therefore fulfilling the needs.

Impelled by the needs and optimistic about the opportunities, we have performed research on the use of collaboration in the operation of networks. The emphasis is on providing collaborative behavior (collaboration protocols) to autonomous control agents so that they meet the operating constraints and improve the performance of the network. In these contexts, the agents are geographically distributed over the network, have a partial view of its state, and can only set the values of its local controls. It is our belief that the impact of collaboration will be more pronounced under these circumstances.

1.2 Overview

Large and distributed networks contain hundreds, or even thousands, of devices that are continually controlled to maintain the network under limits and maximize its performance. Traffic networks, packet-switching systems, and the electric power grid are typical instances whose adequate operations are essential to today's life. Their satisfactory operation relies on the solution of dynamic control problems (DCPs)—that is, problems whose variables represent the state of, and the control inputs to, the network, whose constraints model its dynamics and operating requirements, and whose objective

functions measure performance. A standard solution technique to DCPs is model predictive control—also known as the rolling horizon approach. It formulates, instantiates, and solves a series of time-overlapping, static optimization problems, $\langle P \rangle$, where each element (P) approximates the DCP over a time horizon.

This dissertation develops a solution approach to the static optimization problem, (P), that breaks it up into a set of small and localized subproblems, $\{P_m\}$, matches the agents to the subproblems, and then engages the agents in solving $\{P_m\}$ (see Figure 1.1 for an illustration of the solution approach). The agents are autonomous, collaborative, and arranged in a collaborative network (collaborative net or C-Net). By autonomous agent, we mean any entity that can make and implement decisions without supervision. By collaboration, we mean any process by which the agents work together (effectively exchanging information and helping one another) so as to improve collective performance. By collaborative net, we mean a flat organization (having only one layer and no supervision) wherein the agents communicate locally and run a collaboration protocol. By collaboration protocol, we mean the rules or schedule for the exchange of information among agents.

The research focuses on 1) the design of collaboration protocols that promote the solution of $\{P_m\}$, 2) the experimental and analytical analyses of the protocols, including their attributes and appropriateness, and 3) the demonstration that collaborative nets can be effective in controlling networks.

1.2.1 A Framework for Specifying the Agents' Tasks

A framework for specifying the static optimization problem and breaking it up into small subproblems has been proposed in this work [TC00]. The framework formulates the agents' tasks as optimization problems, and introduces the concept of match between agents and subproblems. (The match is near when the agent's subproblem is strongly coupled only to those of its neighbors, and is weakly coupled, if at all, to every other subproblem. Two agents are neighbors if they can communicate directly.) The concepts of agent, collaboration protocol, and organization are more formally introduced and brought together to define collaborative nets. Being flat organizations, C-Nets are potentially fast, open, and fault tolerant. These features are not only desirable, but also critical if collaborative nets operate networks that need a quick response to disturbances, that dynamically change their configuration, and that should be robust to failures.

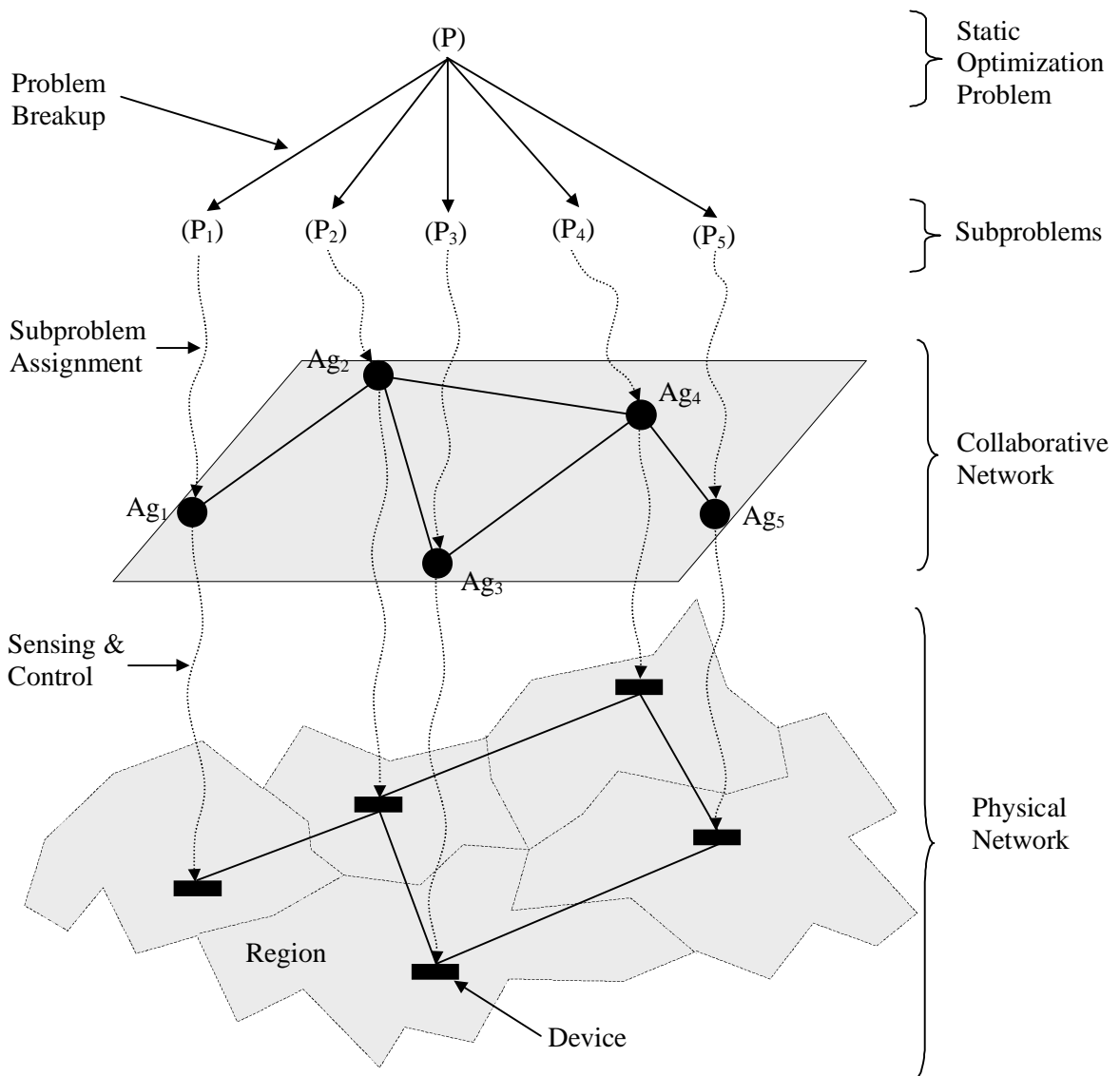


Figure 1.1: An illustration of the solution approach to dynamic control problems.

Asynchronous teams [TBG+98] and flocks of birds [Rey87] are relatives of collaborative nets that are found in artificial and natural domains, respectively. Asynchronous teams (A-Teams) consist of organizations of memories (that accumulate solutions to subproblems) and agents (that read elements from their input memories and generate solutions for the output memories). A-Teams are the precursors of C-Nets and, therefore, the bonds between them are not surprising—such as the similar structure, the emphasis on problem decomposition, and the potential of parallel work. However, their applications are dissimilar in the sense that C-Nets are being developed to solve on-line optimization and control problems, whereas A-Teams have been solving hard, off-line optimization problems. With respect to the

biological domain, C-Nets operate much like the way natural organizations, such as flocks of birds, do. In the latter organizations, the agents are autonomous (there is no indication of centralized control), restricted both in their view of, and control over, the organization, and yet able to solve complex problems.

1.2.2 Collaboration Protocols

Together with problem decomposition and agent-subproblem match, collaboration protocols form the backbone of the suggested approach to solve dynamic control problems. What alternatives are there for the design of protocols? To this end, a short and initial taxonomy has been proposed. It compiles design alternatives—that were found to be relevant in the course of this work—in dimensions of a design space for protocols. These dimensions comprise the following:

- 1) the type of variable update (exclusive or shared),
- 2) the type of iterative solution (with or without precedence constraints), and
- 3) the type of subproblem augmentation (the means by which the subproblems can be modified to a) lessen the side effects of asynchronous work—such as resource margins, and b) make up for a lack of global view—such as the implant of mutual-help reflexes that encourage the agents to pursue the goals of their neighboring peers).

We have also suggested protocols that fall into the categories of the taxonomy, studied their properties and merits qualitatively, and, in some cases, analyzed their performances numerically or analytically. Two instances of protocols are 1) the voting protocol, whereby the agents collect the assessment of their neighbors and implement the decision of the majority, and 2) the proximate-exchange protocol, whereby neighboring agents exchange data and iteratively revise their plans.

The proximate exchange is a basic and yet powerful protocol that can be elaborated in a number of ways. Besides the exchange of information, an agent may optimize the objective and constraints of its neighbors (especially, when these are more disturbed or face harder problems) and also delegate the authority to set the values of its controls to its neighbors. For this protocol, sufficient conditions were identified for the effort of the agents in solving $\{(P_m)\}$ to arrive at a solution to (P). Roughly speaking, the conditions are:

- 1) the collaborative net covers the entire network,
- 2) (P) must be strictly feasible,
- 3) (P) must be convex,
- 4) (P) must be differentiable,

- 5) the agents solve $\{(P_m)\}$ with interior-point methods,
- 6) the match between agents and subproblems must be exact, that is, each (P_m) must be only sensitive to the variables that agent- m senses directly or through its neighbors, and
- 7) the agents work serially within neighborhoods, that is, parallel work is permitted only among non-neighbors.

Although the conditions have analytical merit, they have little practical potential. Consider, for instance, the conditions on exact match and serial work. Even if these conditions could be met and afforded in practice, the response of the collaborative nets could be excessively slow. The purpose of the conditions is to bring forth the issues that need be studied and resolved before collaborative nets are deployed.

How could the sufficient conditions be relaxed and the collaborative nets still find good solutions to (P) ? We demonstrate experimentally that the above conditions are not necessary. In prototypical networks, the agents arrived at a solution to (P) even though the conditions had been ignored. If the conditions were also necessary, convergence would have not been observed. This fact should not be surprising because, not infrequently, sufficient conditions are conservative (accounting for the worst-case outcome) and thereby need not be necessary.

A specific issue is the feasibility of (P) when the agents solve $\{(P_m)\}$ asynchronously. Itself, the issue constitutes a relevant and hard research question whose answer depends on the overall problem (P) , its decomposition into $\{(P_m)\}$, and how agents solve subproblems. This research looks into the tightening of the constraints in $\{(P_m)\}$ with resource margins so as to uphold the feasibility of (P) , when the agents begin with a feasible solution and work asynchronously.

Along these lines, we have formulated an optimization-based procedure that checks whether or not, a given set of resource margins is sufficient for feasibility. The procedure's nature indicates, however, that it is effective only when (P) is convex. Also, we have found conditions that are sufficient, and necessary, to retain the feasibility of a class of constraints—namely, those that define ellipsoids without rotations, that is, balls that are scaled along the axes and displaced from the origin. For the general case, this work offers rules for computing adequate resource margins that, in essence, suggest the following:

- 1) the approximation of equality constraints with inequalities,
- 2) the approximation of the feasible region with convex sets, if possible,
- 3) the bounding of the variables, and
- 4) an iterative procedure that simulates the asynchronous work, or checks the sufficient conditions, to identify adequate resource margins.

By themselves, the resource margins do not promote convergence to solutions of good quality, but only sustain feasibility. The abilities of the agents and the collaboration protocols play important roles in achieving satisfactory quality. If the proximate exchange of information is not satisfactory, then 1) provide each agent with a means to estimate the actions that its neighbors can take to, and the extent to which these actions, improve its performance (improve the ability of the agents), and 2) introduce mutual-help behavior (improve the protocol).

1.2.3 Experiments and Applications

Collaborative nets span a wide range of applications, extending from dynamic networks (where the location of the elements is time-dependent such as in mobile robots) to static ones (where the locations are physically fixed such as in power networks). Prototypical instances stand at the left extreme of the range, including the dynamic network of a flock of artificial birds, and the static network of a forest of pendulums. Typical instances of real-world networks stand at the right extreme, comprising teams of robots and the power grid. Along this range, the dynamics vary from linear to highly nonlinear, the complexity of the control problems increases, and the use of traditional control techniques becomes more challenging. In the coming chapters, we demonstrate the potential of collaborative nets in forests of pendulums and electric-power networks.

Experiments in Forests of Pendulums

Forests of pendulums are arrangements of frictionless pendulums, linear springs, and control devices that influence motion. Typically, the pendulums are distributed over a uniform grid and fully connected by springs. Under normal operation, the pendulums swing uniformly at the same frequency. The DCPs arise from disturbances—arbitrary changes on the position, and speed, of the pendulums—to restore the synchronous operation quickly and cheaply. These forests are prototypical in the sense that their nonlinear dynamics, and associated control problems, are like those encountered in power networks. In the forthcoming experiments, we demonstrate that the sufficient conditions are not necessary for convergence, and contrast the performances of C-Nets with those of traditional control techniques and those of an ideal, centralized agent.

First, we conduct experiments to verify the non-necessity of the conditions. Specifically, we follow the steps below in a number of fully connected forests:

- 1) break up (P) into a set of subproblems, $\{(P_m)\}$, containing one element for each pendulum,
- 2) equip each pendulum with an agent that senses its state and governs its controls,

- 3) set up communication links between physically adjacent agents, and embed the asynchronous proximate-exchange protocol into them, to form C-Nets,
- 4) compute resource margins to inhibit violations of the constraints, and
- 5) disturb the pendulums at random, simulate the operation of the collaborative net, and record the trajectory of the pendulums.

In all instances, the solution of $\{(P_m)\}$ has consistently converged to a solution to (P), despite the noncompliance with the conditions on serial work, convexity of (P), interior-point method, and exact agent-subproblem match.

Second, the performances of the C-Nets were contrasted against those of an omniscient agent, C_1 , which can sense and control the entire network, that is, an agent that solves (P). In the experiments, the performance of each C-Net approached that of C_1 . The good quality of the C-Net solution is due in part to the addition of mutual-help behavior to the protocol. More specifically, the least troubled agents are compelled to tighten their resource margins, allowing the most troubled ones to use more resources. In other words, the agents augment their subproblems to make up for the changing context.

Third, centralized controllers were synthesized with nonlinear-feedback techniques (traditional control) and their performances compared with the ones achieved by the C-Nets. The experiments indicate that the collaborative nets attain comparable performances—inferior in some incidents but superior in the others.

Experiments in Power Networks

The preceding results indicate that the suggested approach has potential to solve dynamic control problems in real-world networks. To further investigate this potential, we ran experiments in representative power networks, wherein the control task is to restore (after major disturbances) the synchronous mode of the generators. Like in forests of pendulums, the performances of the C-Nets were not far from those of ideal, centralized agents.

In the experimental set-up, the networks consisted of IEEE test-grids, turbine-speed governors (that regulate mechanical-power input) and flexible AC transmission devices (that, unlike the on-and-off capability of relays, allow continuous and fast control of network parameters such as line impedance).

In the event of a major disturbance (such as a power surge or lightning strike), the generators undergo oscillations and, very often, recover synchronization, settling down at an acceptable operating point. Sometimes, however, the disturbance triggers a sequence of failures that cascade outwards from the

originating site, forcing the shutdown of generators and leaving wide areas without supply. In this scenario, the DCP—the so-called transient stability problem—consists in adjusting the on-line control devices, and implementing preventive measures, to sustain synchronization. The experimental set-up comprises the solution of transient stability problems, which arise from incidents in two representative power grids.

1.2.4 Matching Agents to Subproblems

What variables should the agents sense and control? What should the neighborhoods be like? Answers to these questions define the proximate variables (those whose values are sensed or set by the agents), the neighborhood variables (those whose values are obtained from the neighbors), and the remote variables (all the other variables). Therefore, they specify the match between agents and subproblems, which plays an important role in the effectiveness of collaborative nets. The match between agent- m and (P_m) is near if (P_m) is weakly sensitive to the remote variables or, equivalently, if (P_m) is strongly coupled only to those of the agent- m 's neighbors. In an attempt to answer the above questions, we have proposed two models for matching, studied their computational hardness, suggested algorithms, and carried out experiments.

In these models, the elements of the network are divided into agents (the control devices) and system components (the controlled devices). The models assume that the following data are given:

- 1) the variables governed by the agents,
- 2) the estimates of the effects that their actions have on the components,
- 3) the costs to establish communication from agents to components and between agents, and
- 4) the limitations on the communications.

Then, the matching boils down to picking the communication links that maximize the overall perception of effects and, at the same time, that minimize the total cost of communication. The problem belongs to the class of multi-objective problems, since it optimizes two conflicting objectives—an increase in effect-perception can incur communication costs. This implies that the problem admits a set of nondominated solutions, the so-called Pareto set, which captures the trade-offs between perception and cost. The computation of the Pareto set constitutes an important step in matching agents to subproblems, especially because dominated solutions are inferior and potentially exponential in number.

The first model maximizes the total effect that the agents perceive directly, not accounting for the effects that they perceive through the neighbors. The model is basic and among the simplest. For this

model, however, a representative subset of the Pareto set can be computed efficiently by solving a polynomial number (on the size of the output) of minimum-cost network-flow problems.

The second model extends the first one to account for the indirect perception of effects. Unfortunately, the model was shown to be computationally hard. This should not be surprising since the most general and interesting problems are hard, nor should it be discouraging since heuristics and other problem-solving techniques can find near-optimal solutions. Indeed, this constitutes an opportunity for research.

The second model was put to the test. In forest of pendulums, a representative subset of the Pareto set of agent-subproblem matches was computed, and the corresponding collaborative nets were instantiated and then deployed to restore the synchronous operation of the pendulums. The experiments indicate that the resulting collaborative nets also belong to the set of nondominated C-Nets in performance space.

1.3 Outline

Chapter 2 introduces the basic concepts of this dissertation. It suggests an approach to the solution of dynamic control problems, describes a framework for specifying the agents' tasks, and provides definitions for agent, organization, and collaborative net. Further, it identifies attributes of C-Nets that are found in artificial and natural organizations, such as asynchronous teams and flocks of birds.

Chapter 3 focuses on collaboration protocols. It begins by enumerating the attributes of protocols that were found to be relevant in the course of this research. It then presents an initial taxonomy that includes the voting protocol, the basic proximate-exchange protocol, and variants of the latter. The material therein provides qualitative and analytical analyses of protocols, especially the proximate exchange for which sufficient conditions for convergence were found.

Chapter 4 addresses the use of resource margins to reduce constraint violation under asynchronous work. It formalizes the implementation of resource margins, presents an optimization-based test to check if they are sufficient to ensure feasibility, and states sufficient and necessary conditions to sustain the feasibility of ellipsoid-like constraints. Finally, it suggests rules for computing resource margins.

Chapter 5 demonstrates experimentally that the sufficient conditions are not necessary for convergence. It suggests forests of pendulums as prototypical networks, sets up and runs experiments in a number of scenarios, and reports the results. The collaborative nets consistently reached solutions to (P) whose qualities approached not only those of a traditional control technique, but also the ones of an ideal agent.

Chapter 6 employs the suggested approach to solve transient stability problems in representative power networks. The networks consist of IEEE test-grids, turbine-speed governors, and solid-state control devices. The results indicate that the collaborative nets are effective in these highly complex scenarios; specifically, they restored the synchronous operation of generators after major disturbances.

Chapter 7 tackles the problem of matching agents to subproblems. It studies matching models that maximize the agents' perception of the effects of their actions on the controlled components, while minimizing the total cost of communication. Therein, the computational difficulty of the associated problems is assessed analytically, and their efficacy is verified in forest of pendulums.

Chapter 2

Collaborative Nets

This chapter constitutes the foundations of the dissertation. It introduces the terminology, presents the framework for solving dynamic control problems, DCPs, and defines collaborative nets together with their underlying concepts.

The chapter begins with a review of DCPs that covers instances from traffic networks and the power grid. It then presents the rolling horizon approach, that is, the formulation and posterior solution of a series of time-overlapping, static optimization problems, $\langle P \rangle$, which approximate the DCP over a time horizon. The subsequent material suggests the breakup of (P) into a set of small subproblems, $\{P_m\}$, and their solution by a collaborative net—a flat organization of autonomous agents. The chapter also elaborates on the concept of agent, organization, and collaboration protocol so as to define collaborative nets. Therein, it identifies similarities between collaborative nets and their relatives in artificial and natural domains, namely, asynchronous teams and flocks of birds. The remainder of the chapter proposes a graphical representation for C-Nets and provides illustrations. We remark that a substantial part of the text was taken from Talukdar et al. [TC00][TCZ00], especially the rolling horizon formulation and the problem decomposition.

2.1 Networks and Dynamic Control Problems

When is the operation of a network satisfactory? It is satisfactory when potentially multiple (and often conflicting) goals are achieved, and several operating constraints are met. Typical goals are high performance, low operation cost, and adequate robustness. Constraints consist of technical limitations, regulations, and safety measures. Two standard approaches for operating networks are:

- 1) the combination of the conflicting goals into a single objective that captures the desired trade-offs, and
- 2) the delegation of the goals and constraints to several independent entities.

In either approach, we are left with the design and solution of single objective, dynamic control problems—that is, problems whose variables model the state of, and control inputs to, the network,

whose constraints express its operating requirements and simulate its dynamics, and whose objective functions measure the quality of the solutions. In short, the design is the translation of the specifications into a DCP and, invariably, calls for expertise and specific knowledge of the physical network. Single objective DCPs can be expressed as follows:

$$\begin{aligned} & \text{(DCP) Minimize } f(x, dx/dt, u, t) \\ & \text{Subject to:} \\ & \quad H(x, dx/dt, u, t) = 0 \\ & \quad G(x, dx/dt, u, t) \leq 0, \end{aligned}$$

where f is the objective function, H is the set of equations that simulate the dynamics of the network, and G is the set of operating constraints.

This dissertation is not about the design of DCPs, but rather devoted to their solution and, henceforth, refers to DCP as a single objective, dynamic control problem. Below, we look into two real-world networks, the DCPs that are encountered therein, and the status quo of the in-place solutions.

2.1.1 Traffic Networks

The satisfactory operation of traffic networks lies in setting traffic-signal switching schedules that maximize throughput and drivers' satisfaction, while ensuring safety and enforcing the network's limitations. Instances of the dynamic control problem combine these goals into a performance criterion, and implement operating constraints that capture the limits on traffic flow, enforce feasible signal switching, and guarantee compliance with the traffic regulations.

As of today, the standard solution to the above DCPs is an off-line centralized procedure: the road network, the stochastic and time-varying route patterns, and the operating constraints are fed into an optimizer to compute (traffic-signal switching) schedules that maximize the performance criterion [Che98]. The in-place solution is inflexible in the sense that it prevents rapid adaptation and circumvention of contingencies. These inflexibilities have been recognized by field scientists who have deployed autonomous, distributed agents to control traffic signals in experimental networks [MF80]. Unfortunately, the results have been unsatisfactory. In these experiments, however, the agents do not communicate with others in their geographic vicinity. We believe that collaboration and learning are promising alternatives to boost collective performance and, ultimately, promote satisfactory operation of traffic networks.

2.1.2 Power Networks

The goals and constraints in operating the power grid are dissimilar and drawn from various categories such as costs, regulations, and safety [TC00]. Rather than combining them in a single dynamic control problem, they have been delegated to different organizations. Therefore, several DCPs are solved in parallel, and at different speeds, by various organizations that handle a subset of the goals and constraints. Examples of organizations are:

- **The Protection System:** the goal is to maintain the equipment under their operation limits. The dynamic control problem lies in switching the protection devices, such as relays, to prevent equipment damage.
- **The Generator Control System:** the dynamic control problem is to find a power-generation schedule that meets the dynamically changing demand at minimum production cost.
- **The Security System:** the objective is to anticipate contingencies, make plans to handle them, and prevent cascading failures. The plans might prescribe the use of spinning reserve in anticipation of immediate need of power, and reduce the power transfer of the transmission lines to prevent severe disturbances in case of their disruption.

2.2 The Rolling Horizon Formulation

We are given a physical network, Θ , its current state, and a dynamic control problem that specifies the goals and constraints in operating Θ . How can the DCP be solved? The “rolling horizon approach” [MP93] approximates the DCP with a static optimization problem (where time is not present), and uses an optimization procedure to calculate the control actions to be taken at a set of discrete points in time called horizon: $[t_0, t_1, \dots, t_N]$, where t_0 is the present time and N is the number of intervals. (The discrete points in time are not necessarily equidistant; they can be chosen so that the interval between them becomes longer towards the end of the horizon.)

The disadvantages of relying on a single approximation of the DCP to calculate the real-time controls of Θ are twofold. First, the calculations are “open loop” (they use model-based predictions of future states, and these predictions become less accurate, the further one goes into the future). Second, the approximation is typically too large (it has N -times as many variables as the DCP).

The first disadvantage is overcome with the introduction of feedback by periodically “rolling the horizon forwards,” that is, by recalculating the static approximation for a series of overlapping time-intervals. (Only the controls of the first interval obtained by solving the static approximation are implemented; after a short time has elapsed, a new static approximation is formulated, initialized with current state measurements and solved to obtain a new control plan; the process is then repeated.)

The second disadvantage (the unmanageably large size of the static approximations) can be alleviated by breaking the approximation down to a set of small, local approximations. This is the stage where this dissertation begins. It addresses the means to solve the set of approximations, by enumerating solution alternatives and providing an account of their limitations.

The rolling horizon approach is also called “model predictive control” [Mos95][Raw99], because of the predictive nature and the use of a behavioral model for Θ . As described in [GPM89], model predictive control (MPC) has been widely used in the process industry for its considerable advantages: the ability to handle constraints in a systematic way, the capability of operating without intervention for long periods, and the ease of adaptation to new contexts.

The steps in a rolling horizon algorithm are:

- 1) Choose a time horizon: $[t_0, t_1, \dots, t_N]$, where t_0 is the present time and t_N is the length of the horizon (t_N may be finite or infinite).
- 2) Instantiate the static optimization problem, (P), whose objective and constraints reflect the specifications on how the network Θ must behave. The decision variables are the control actions to be taken at each discrete point of the time horizon.
- 3) Solve (P) to obtain the values of the control actions.
- 4) Implement the first of these control actions, that is, the action for the present time, t_0 .
- 5) Pause (allowing the horizon to roll forward, i.e., allowing t_0 to increase) and repeat from step 1.

Thus, a rolling horizon algorithm reduces a dynamic control problem to a series of time-overlapping, static optimization problems. The approach is adaptive in the sense that the control plans are revised every time problem (P) is instantiated. Take, for instance, the predicted state and planned control action for time t_1 as calculated at time t_0 , $x(t_1|t_0)$ and $u(t_1|t_0)$, and the actual state and control action at time t_1 , $x(t_1|t_1)$ and $u(t_1|t_1)$. The discrepancies between the model and the network translate into differences between $x(t_1|t_0)$ and $x(t_1|t_1)$. They can, however, be compensated by revising the control plans, from $u(t_1|t_0)$ to $u(t_1|t_1)$, at time t_1 .

The above comments suggest that a quick revision of the control plans (in the rolling horizon approach) can make up for the inaccuracies of the model of Θ . Stochastic models are other, and perhaps more powerful, ways of handling inaccuracies. They represent the network state as stochastic variables and express the dynamics (and constraints) in terms of probability distributions. If so chosen, the rolling horizon approach solves a series of stochastic optimization problems [KW94][Had64, §4] in place of the deterministic ones.

The work reported hereafter addresses only deterministic models for the sake of simplicity. This does not seem to impose limitations to the forthcoming framework because stochastic optimization problems can often be recast as, or approximated by, deterministic ones [Beasley]. The solution of the resulting problems is, however, potentially hard due to the possible introduction of integer variables and the substantial increase in the problem's size. To this end, the continuation of this dissertation will extend the framework to stochastic problems, especially those whose parameters are random variables and whose constraints should be met with a given probability.

2.2.1 The Static Optimization Problem

Consider the network Θ over a time horizon: $[t_0, t_1, \dots, t_N]$. Let:

- $x(t_n)$ and $u(t_n)$ be vectors of the state and control variables of Θ at time t_n ,
- $H(x(t_{n+1}), x(t_n), u(t_n)) = 0$ be a set of difference equations that model the dynamics of Θ ,
- $X = [x(t_0), x(t_1), \dots, x(t_N)]$ be a set of state vectors over the horizon,
- $U = [u(t_0), u(t_1), \dots, u(t_N)]$ be a set of control vectors over the horizon,
- $f(X, U)$ be a scalar function representing the objective in operating Θ over the horizon, and
- $G(X, U) \leq 0$ be a set of inequalities representing the operating constraints of the network over the horizon.

A general form of the rolling horizon optimization problem is the following: given the current state of the network, $x(t_0)$, find U so as to:

$$\begin{aligned}
 & \text{(P) Minimize } f(X, U) \\
 & \text{Subject to:} \\
 & \quad H(X, U) = 0 \\
 & \quad G(X, U) \leq 0.
 \end{aligned}$$

2.3 Decomposing the Static Optimization Problem

Herein, the focus is on the decomposition of the static optimization problem, (P) , into a set of small and localized subproblems, $\{(P_m)\}$, and the assignment of the subproblems to the agents of a collaborative net. The material addresses this decomposition, the solution of $\{(P_m)\}$ by the C-Net, and its relationship with the solutions to (P) .

Consider a set of M agents that are distributed over the physical network Θ . The agents are limited both in their perception of, and authority over, the network Θ . More precisely, they sense only a subset of the state variables of Θ , and can set the values of only a few of its control variables. To form a collaborative net, we establish communication links between these agents, match each agent to an element of $\{(P_m)\}$, and embed a collaboration protocol into the agents. Two agents are neighbors if they can communicate directly, that is, if they are connected by a link. By means of this decomposition, the collaborative net can solve $\{(P_m)\}$ in place of (P) .

From each agent's view-point, the state and control variables of Θ can be clustered into three groups:

- 1) **Proximate Variables:** the variables that the agent can sense or set.
- 2) **Neighborhood Variables:** the variables that its neighbors can sense or set, that is, the proximate variables of its neighbors.
- 3) **Remote Variables:** all the other variables—those that neither the agent nor its neighbors can sense or set.

The match between agent- m and its subproblem is *exact* when (P_m) contains only agent- m 's proximate and neighborhood variables, and *near* when (P_m) is weakly sensitive to agent- m 's remote variables. A match can, however, always be made exact by expanding agent- m 's neighborhood to cover its remote variables.

Although large networks, such as the power grid and traffic networks, can contain a sizeable number of constraints, they are typically local and sparse. We say that a constraint is sparse if it is expressed in terms of a few variables, and local if the variables come from a small geographic area. In the power grid, for instance, the flow limits on transmission lines are sparse and usually local—they depend only on the voltage magnitude and phase angle at the extremes of the transmission lines. In traffic networks, the car-flow limits on a street segment are functions of the speed limits and the traffic signal settings at the intersections.

The sparse, and typically local, nature of the constraints in (P) suggests that the match between agents and subproblems can be made near with small neighborhoods. We use the terminology below to more formally state the decomposition. Let:

- $x_m(t_n)$ be the components of $x(t_n)$ that agent- m can sense,
- $u_m(t_n)$ be the components of $u(t_n)$ that agent- m must calculate and set,
- $X_m = [x_m(t_0), x_m(t_1), \dots, x_m(t_N)]$ be the set of values of $x_m(t_n)$ over the horizon,
- $U_m = [u_m(t_0), u_m(t_1), \dots, u_m(t_N)]$ be the set of values of $u_m(t_n)$ over the horizon,
- $Y_m = [X_k, U_k \mid k \text{ is a neighbor of agent-}m]$ be agent- m 's neighborhood variables over the horizon,
- Z_m be the set of agent- m 's remote variables over the horizon,
- f_m be the objective function f , or a suitable approximation,
- G_m be the subset of G that contains all elements that depend on agent- m 's proximate variables,
- H_m be the subset of H that contains all elements that depend on agent- m 's proximate variables.

The above terminology is used to formulate each element of $\{(P_m)\}$ as an optimization problem. Given $x_m(t_0)$ and $x_k(t_0)$, for each neighbor k of agent- m , the subproblem (P_m) consists in finding U_m so as to:

$$\begin{aligned} (P_m) \text{ Minimize } & f_m(X_m, U_m, Y_m, Z_m) \\ \text{Subject to: } & \\ & H_m(X_m, U_m, Y_m, Z_m) = 0 \\ & G_m(X_m, U_m, Y_m, Z_m) \leq 0. \end{aligned}$$

The purpose of the decomposition is to solve $\{(P_m)\}$ by a collaborative net, rather than by a single agent that senses the entire network and sets the values of all its controls (a centralized controller). In the collaborative net, the agents are autonomous (they are not supervised), and may work asynchronously and in parallel.

When do solutions to $\{(P_m)\}$ result into solutions to (P)? What penalty do we incur by solving $\{(P_m)\}$ instead of (P)? Answers to these questions are, of course, dependent on (P), its decomposition in $\{(P_m)\}$, and the collaboration protocol that the agents use to solve their subproblems. The work reported in this dissertation addresses these questions experimentally and, in some cases, analytically. The next chapter, for instance, sets down conditions for the solution of $\{(P_m)\}$ to reach a solution to (P).

A relevant issue concerns the equivalence between (P) and $\{(P_m)\}$, that is, the conditions under which a trial solution (X, U) solves both (P) and $\{(P_m)\}$. A set of sufficient conditions for the equivalence between the solutions to $\{(P_m)\}$ and (P) will be given in a moment. (The conditions do not, however, say

anything about the process for obtaining these solutions by solving $\{(P_m)\}$ with a network of agents that work asynchronously and in parallel. Much of this process is embedded in the collaboration protocols, which constitute the topic of the next chapter.)

Theorem 2.1. If:

- 1) The collaborative net covers the entire physical network Θ and has authority over all its control variables: $\cup_{m=1, \dots, M} \{x_m(t_n)\} = x(t_n)$, and $\cup_{m=1, \dots, M} \{u_m(t_n)\} = u(t_n)$.
- 2) The aggregation of the objective function of all subproblems is identical to the overall objective function: $\sum_{m=1, \dots, M} \{f_m\} = f$.

Then (X^*, U^*) is a simultaneous solution to the set of subproblems, $\{(P_m)\}$, if and only if, (X^*, U^*) is a solution to the overall problem (P).

Proof. Since $\sum_{m=1, \dots, M} \{f_m\} = f$, and since condition (1) ensures that every constraint (in G and H) appears in at least one element of $\{(P_m)\}$, it follows that $\{(P_m)\}$ is merely a restatement of (P) with some redundancy. (Some constraints of (P) may be present in more than one element of $\{(P_m)\}$.) Therefore, every simultaneous solution to $\{(P_m)\}$ is a solution to (P), and vice-versa. ■

2.4 Collaborative Nets

2.4.1 Terminology

The following terms appear in the definition of collaborative nets:

- **Agent:** any entity capable of making and implementing decisions. Agents can be “intelligent,” “inaccurate,” and slow such as human beings. They can also be “non-intelligent,” “highly accurate,” and fast such as switching devices in communication networks. Thus, the definition encompasses a large spectrum of agents that are found in natural and artificial systems.
- **Organization:** any network of agents and communication links. The rationale for this definition rests on the basic elements that are usually, if not always, present in alternative definitions of organizations [PCG98]. These elements are a) the members of the organization (the agents), and the structural relations among them (the communication links).
- **Flat Organization:** any organization whose members are not supervised. More precisely, the communication links do not enforce command and the agents are autonomous (they make and implement the decisions at their own pace). The agents use the links to exchange information, collaborate, and help one another to accomplish their goals.

- **Collaboration Protocol:** a set of rules that dictate the interaction between neighboring agents (collaborators), that is, those that are joined by a communication link. The rules set down the data exchanged by the agents, the way they communicate, and the use of the data in solving their subproblems.

2.4.2 Definition

A collaborative net is a flat organization of dissimilar agents, such as computer programs and human beings, whose purpose is to solve on-line optimization and control problems. To accomplish this task, the agents solve a set of subproblems, $\{P_m\}$, exchange information, and help one another as dictated by the collaboration protocol.

Being flat organizations, collaborative nets have the potential to be fast, open and fault tolerant. These are desired properties, especially in large and distributed networks that dynamically change their structure, need a quick response to disturbances, and should be robust to failure of the underlying components.

2.5 Related Domains

Collaborative nets can be thought of as, and were inspired by, asynchronous teams (A-Teams). The purpose of C-Nets is to solve on-line optimization (and control) problems, rather than the off-line optimization problems that are routinely solved by A-Teams. In essence, the target problem is the main distinction between C-Nets and A-Teams.

In the material below, we briefly review asynchronous teams, list some of their applications, and identify parallels with collaborative nets. We also look into natural organizations that perform tasks in the same way that collaborative nets do, namely schools of fish and flocks of birds.

2.5.1 Asynchronous Teams

Talukdar et al. [TBG+98] define an asynchronous team as “a set of autonomous agents and a set of memories, interconnected to form a strongly cyclic network, that is, a network in which every agent, or almost every agent, is in a closed loop.”

The memories accumulate solutions to related problems, such as relaxations, and partial or candidate solutions of an off-line optimization problem. The agents are of two types: construction or destruction.

A construction agent reads a solution from its input memory, runs an operator on the input object, and writes the resulting solution in to its output memory. The destruction agents remove solutions from the memories and, therefore, keep the size of the memories under limits. The components that characterize agents are 1) the input memory, 2) the output memory, 3) the operator, and 4) the control engine. The operator transforms objects from the input memory into items of the output memory. The control engine decides which object the agent should work on (selection), and when it should execute this task (scheduling). Agents are autonomous, that is, they are equipped with control engines, and work asynchronously and in parallel. The ability of a construction agent is mostly concentrated on its operator component, whereas that of a destruction agent is fully concentrated on its control engine (its operator component is empty).

Structurally, A-Teams are represented by hypergraphs whose nodes model memories, and whose arcs model agents. Figure 2.1 illustrates one of the A-Teams that were designed by de Souza [dS93] to solve the traveling salesman problem (TSP). A-Teams were shown to be effective at solving various off-line optimization problems. Especially, the problems for which there are various inadequate algorithms, but there does not exist an adequate algorithm—one that finds an acceptable solution in the available time. Applications of A-Teams include spatial layout [Sac98], train scheduling [Tse95], control of electric networks [Ram94], and traveling salesman problems [dS93]. In all these applications, the results showed that the A-Teams found better solutions, and also quicker, than any of its constituent agents.

A-Teams encompass other optimization techniques, such as genetic algorithms (GA) and simulated annealing (SA) [dS93, pp.19-25][TBG+98]. A-Teams are population-based and form cyclic networks like genetic algorithms. However, GAs enforce synchronization to evolve the population from one generation to the next, and are often limited to a single memory of candidate solutions. A-Teams and simulated annealing allow iterative work, that is, the continuous improvement and modification of the current solution. Nevertheless, SA is a local search procedure that works on a single solution, rather than on a population.

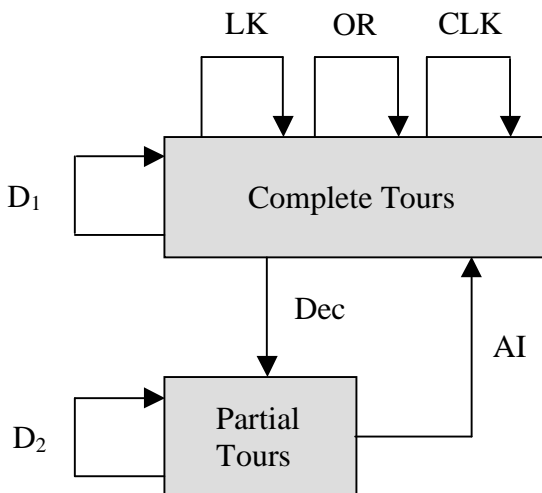


Figure 2.1: The structural representation of an A-Team designed to solve the traveling salesman problem. The A-Team is made up of two memories (rectangles) and various agents (arcs). Agents read objects from the memory at their tails, and write the results in to the memories at their heads. One memory accumulates complete solutions (permutations of the cities to be visited by the salesman), and the other accumulates partial tours (partial permutations of the cities). The agents LK and CLK are versions of the Lin-Kernighan algorithm. OR is the OR-Opt algorithm. Dec creates a partial tour by picking only the edges that are present in two complete tours. AI is the arbitrary insertion algorithm. D_1 and D_2 are the destruction agents that keep the size of the memories under limits.

2.5.1.1 Similarities between A-Teams and C-Nets

Asynchronous teams have inspired the conception of collaborative nets. Therefore, the parallels between them, as will be seen in a moment, are not incidental.

1) Problem Decomposition

The decomposition of an optimization problem into a set of related subproblems (such as relaxations and partial solutions), and the subsequent design of algorithms that transform solutions to a subproblem into another's, have invariably improved the effectiveness of asynchronous teams. Problem decomposition also plays an important role in collaborative nets. However, the decomposition is more critical because the static optimization problem, (P), must be broken up into a set of small and localized subproblems—it is not a matter of choice, but mandatory. The problem decomposition and the match (between agents and subproblems) can greatly affect the overall performances of C-Nets, in the same way that the number and diversity of related subproblems and algorithms improve solution quality, and often convergence speed, of A-Teams.

2) Structure

Like asynchronous teams, collaborative nets can be viewed as networks of agents and memories. More precisely, we can instantiate an A-Team to function like a C-Net as follows:

- a) let the memories of the A-Team store the values of state variables, control variables, and the information exchanged by the agents,
- b) let the input memories of each agent- m (in the A-Team) match the proximate and neighborhood variables of its subproblem (P_m), and
- c) let the output memory of each agent- m match its proximate variables in (P_m).

3) Autonomous Agents and Parallel Work

The agents of A-Teams are autonomous, asynchronous and work in parallel. These features make A-Teams attractive to bring together dissimilar agents and solve large optimization problems. What can be said about the agents of C-Nets? The agents are also autonomous; however, the extent of asynchronous and parallel work depends upon the collaboration protocol.

On the one hand, a protocol may allow asynchronous and parallel work, but offer no guarantee that the solution to $\{P_m\}$ approaches a solution to (P). On the other hand, another protocol may force the agents work serially, allow partial parallelism, and provide guarantees, but be excessively slow for practical use. Inevitably, the selection of one protocol to another incurs trade-offs that depend on (P), its breakup into $\{P_m\}$, and the agent-subproblem match.

2.5.2 Natural Organizations

Schools of fish and flocks of birds are natural, leaderless organizations that operate much like the way collaborative nets do. The agents of these organizations are autonomous (there is no indication of centralized control), restricted both in their view of, and control over, the organization, and yet capable of solving complex problems. For instance, flocking birds can fly in formation (close to each other) along complicated acrobatic paths and still they never collide. Effectively, the birds solve the dynamic control problem of flying in formation. They do so by breaking it up into small and localized subproblems—one for each bird—and then solving them asynchronously.

The behavior of flocking birds seems to be characterized by two conflicting forces: the desire to stay close to flockmates, and the urge to avoid collision. The collision-avoidance behavior is easily understood. The desire to flock can be attributed to the evolutionary pressure from factors such as protection from predators, more effective search for food, and advantages for social and mating

activities. The evidence also suggests that each bird has local perception—it is aware of itself and its nearest neighbors. Further, there is no indication that flocks ever become overcrowded as new birds join in. Schools of fish, for instance, can stretch for several miles and contain millions of fish during migration periods [Rey87][Pat82][TdS94].

Using behavioral rules, Reynolds [Rey87] simulated and animated the aggregate motion patterns, such as formation, of flocking birds and schooling fish. Figure 2.2 depicts a flock of birds flying in formation and swerving around an obstacle. In parallel with collaborative nets, the rules constitute the mechanisms by which the agents—birds and fish—solve their subproblems, that is, the collaboration protocol. Specifically, the position and velocity of the birds constitute the state of the flock, the ability to gradually adjust velocity provides control, and the dynamics arise from linear equations that simulate in-air motion. Further, the birds perceive only the state of surrounding flockmates and act according to the following control rules:

1) Collision Avoidance

The birds move away from obstacles and flockmates that are too close so as to prevent in-air collision. This rule produces v_{avoid} , a velocity vector that points away from the obstacles.

2) Velocity Matching

The birds adjust their velocity vectors to match the average of nearby birds. This rule produces v_{match} , the vector with the average velocity of the nearby birds.

3) Flock Centering

The birds minimize external exposure, and create formation, by moving towards the center of their nearby birds. This rule produces v_{center} , a velocity vector that points towards the center.

The birds continuously perceive their surroundings, apply the above rules, and then fuse the resulting velocities to compute the next velocity vector. It is remarkable that the complex behavior of flying in formation arises from these simple rules. See Figure 2.2 for two snapshots of a flock of birds.

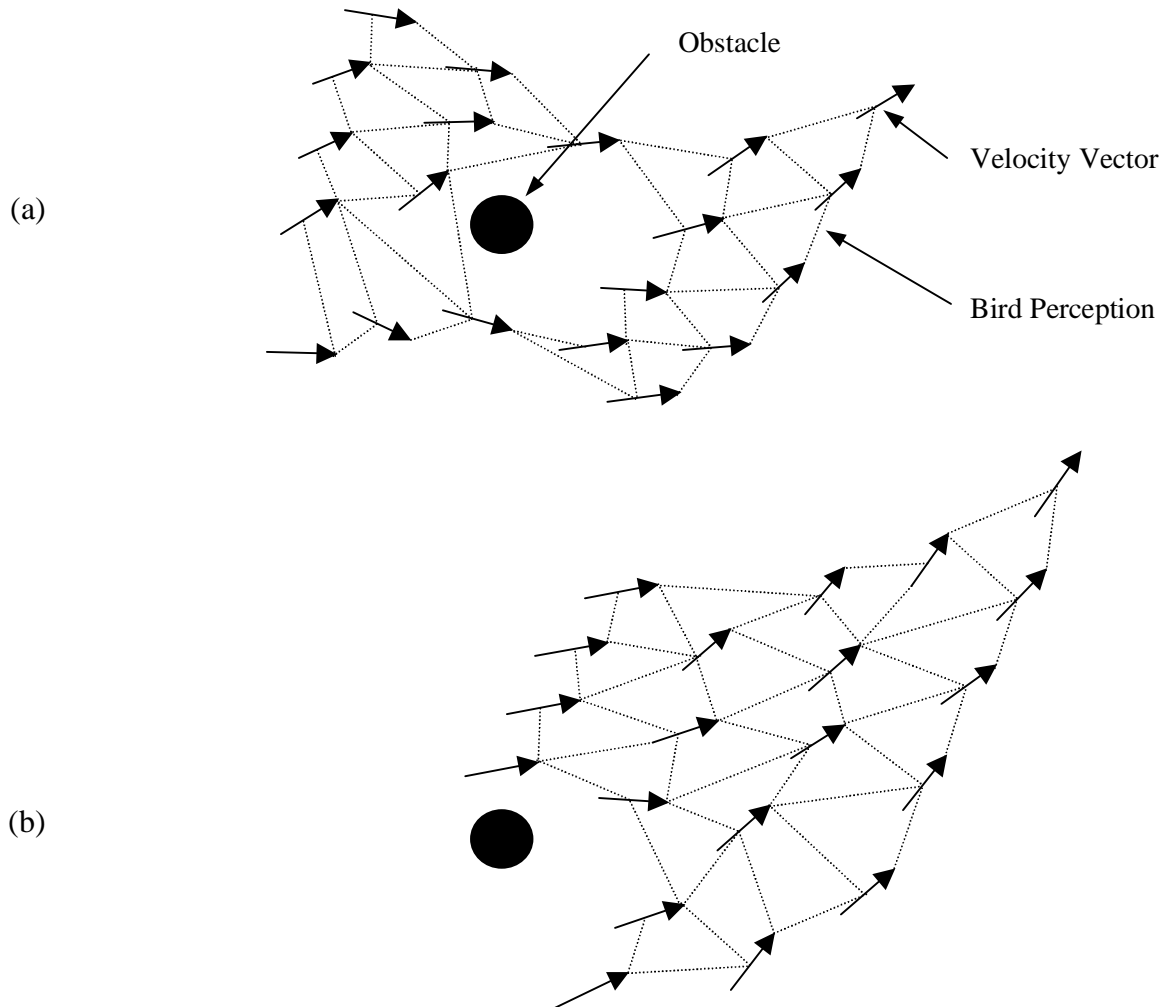


Figure 2.2: Two snapshots of a flock of birds flying in formation. The birds swerve around the obstacle, Fig. 2.2a, and carry on their flight, Fig. 2.2b. Each arrow indicates the direction of a bird’s velocity. The dotted lines illustrate the local perception of all birds—a bird sees another when a dotted line links the head of its arrow to the other bird’s. The snapshots are from a flock simulation and animation that we carried out with Reynolds’s software package (Boid Simulation Package).

2.6 Graphical Representation for Collaborative Nets

The structural representations of C-Nets capture the agents and the neighborhood relations. They consist of a graph whose nodes correspond to the agents, and whose edges display the communications among them. The preceding sections bear out (and the forthcoming ones will reinforce) that problem breakup and agent-subproblem match can greatly influence the performances of C-Nets. It then becomes pertinent to enrich the structural representations with illustrations of breakup and match. To this end, this section proposes the following diagrams:

1) The Variable-Connection Graph

It illustrates the coupling between variables in both the objective function and constraints of the static optimization problem, (P). The state variables correspond to the nodes labeled with x_k . The control ones correspond to those labeled with u_k . An edge joins two nodes if, and only if, their corresponding variables (labels) appear in one constraint of (P), or in one term of its objective function. Notice that the graph is independent of the C-Net and likely to be sparse in typical networks.

2) The Variable-Perception Graph

The graph extends the structural representation of C-Nets to show the proximate, neighborhood, and remote variables of the agents. The agents correspond to the nodes labeled with Ag_m and the variables of (P), to those labeled with u_k and x_k . There are four sets of edges:

- a) *The Solid, Undirected Edges*: they represent the communication (collaboration) links.
- b) *The Solid, Directed Edges*: they point to the agents' proximate variables.
- c) *The Dashed, Directed Edges*: they point to the neighborhood variables.
- d) *The Dotted, Directed Edges*: they point to an agent's remote variables that should be in the agent's set of neighborhood (or proximate) variables for the match to be exact.

The graph offers a view of the C-Net and the perception that the agents have of the network state (proximate and neighborhood variables), which is intimately related to the agent-subproblem match.

We illustrate the graphical representation of C-Nets in a small, hypothetical scenario. Figure 2.3 depicts the variable-connection graph of the physical network. Specifically, it depicts the state variables, $\{x_1, \dots, x_4\}$, the control variables, $\{u_1, \dots, u_4\}$, and their coupling in (P).

Figure 2.4 shows the variable-perception graph of two C-Nets. The C-Net in Figure 2.4a induces an exact match. (Not one *dotted, direct edge* appears in the graph and, therefore, each (P_m) is insensitive to agent-m's remote variables). In Figure 2.4b, however, the C-Net does not induce an exact match. (The graph contains two *dotted, direct edges* that indicate subproblem sensitivity to remote variables. The edge (Ag_3, x_4) indicates that (P_3) is sensitive to x_4 , a variable that Ag_3 cannot sense. Assuming that u_3 and x_4 are weakly coupled, the match becomes near.)

The graphs can be elaborated to display estimates of the coupling between variables. For instance, the influence of agent-m on variable x_n could label the edge between their nodes. We believe that these graphs can be helpful to break up (P), into $\{(P_m)\}$, and match the agents to the subproblems.

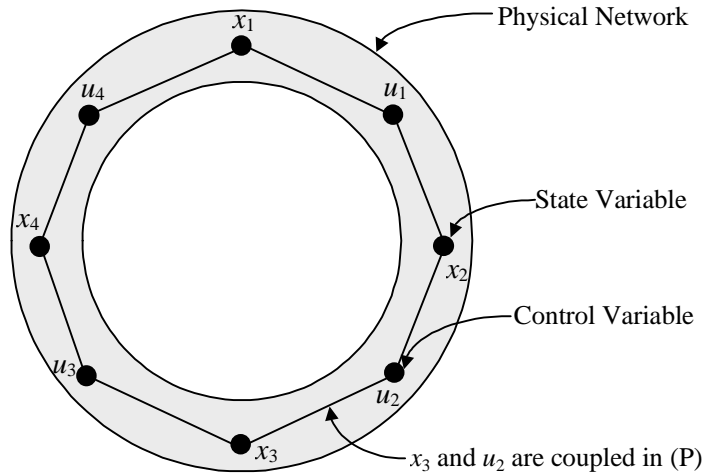


Figure 2.3: The variable-connection graph. The nodes correspond to the state and control variables of the static optimization problem, (P). The edges model the coupling between these variables, that is, two nodes are linked when their variables appear in one constraint of (P), or in one term of its objective function.

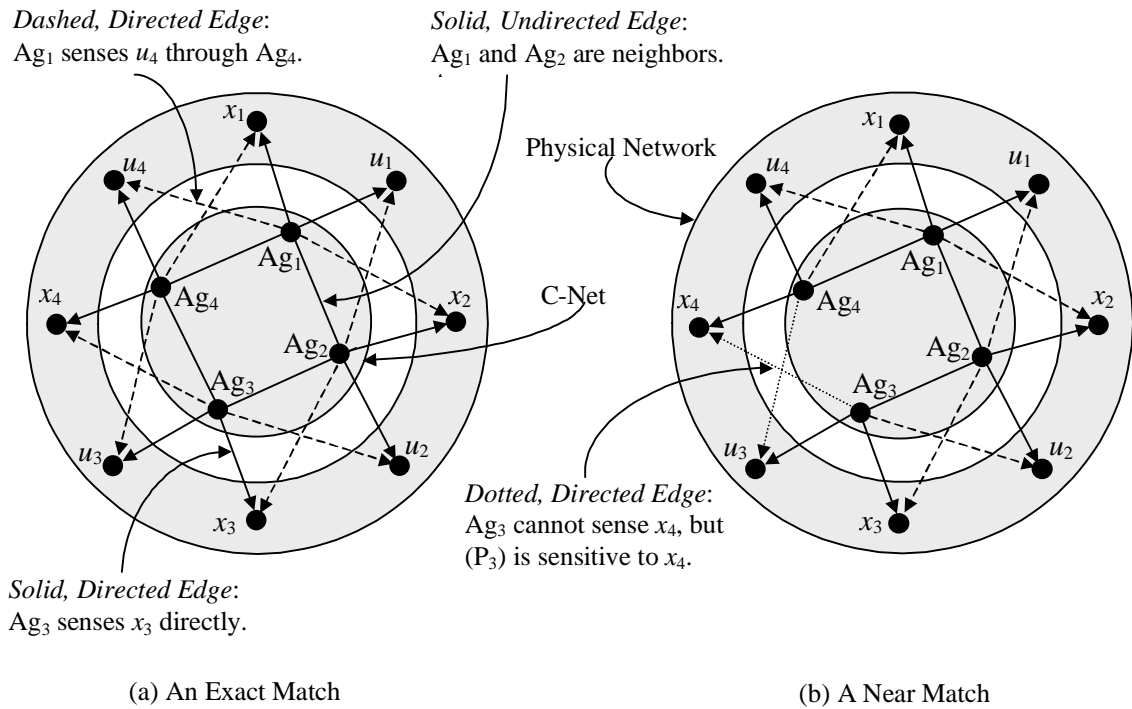


Figure 2.4: The variable-perception graph of exact and near matches. The nodes correspond to agents of the C-Nets and variables of (P). The *solid, undirected edges* represent the communications among the agents. The *solid, directed edges* point to the proximate variables. The *dashed, directed edges* point to the neighborhood variables. The *dotted, directed edges* point to the remote variables (that are relevant for the C-Net to induce an exact match).

2.7 Summary

This chapter has touched upon the importance of dynamic control problems, DCPs, to the operation of networks, and skimmed the status quo of their solutions in the power grid and traffic networks. It then succinctly reviewed the rolling horizon solution to DCPs, also known as model predictive control, and introduced a formulation for each static optimization problem, (P) , that appears in the rolling horizon approach. At that point, the original contributions of this dissertation began. They include the framework for breaking up (P) into a set of small, local optimization tasks, $\{(P_m)\}$, to be tackled by a network of agents (C-Net). The material introduced the basic terminology (that defines the constituent elements of C-Nets) and also identified the parallels between C-Nets and its relatives, in artificial domains (A-Teams) and natural domains (flocks of birds). Finally, the chapter closed with the proposal of graphical representations for C-Nets.

Chapter 3

Collaboration Protocols

This chapter concentrates on collaboration protocols. (That is, the sets of rules under which the agents exchange information, help one another, and take combined actions to improve collective performance.) Protocols can be decisive to the effectiveness, scope of applications, and ease of implementation of collaborative nets.

The first section presents the attributes that we have found to be relevant in the design of protocols, especially to capture the trade-offs between the simplicity of analytical analysis and their practical use. The trade-offs can be illustrated with the type of iterative solution—perhaps, the most critical of the attributes. While the analytical analysis of protocols becomes much simpler when the agents iterate synchronously, their practical application becomes limited. The attributes form a short, and initial, taxonomy of collaboration protocols, some of which are extensively studied hereafter. The subsequent sections specify these protocols, examine their strengths and weaknesses, and provide analytical insights in some cases. The list of protocols comprises:

- **The Voting Protocol:** each agent brings together recommendations for its decisions (or the values of its controls) from its neighbors, and then implements the decisions (or the actions) suggested by the majority.
- **The (Serial) Proximate-Exchange Protocol:** the agents work serially in each neighborhood and collaborate by exchanging their plans.
- **The Asynchronous Proximate-Exchange Protocol:** the agents never hold off their computations to obtain the latest outcome of the neighbors’—they go ahead and work with the current estimates.
- **The (Serial) Proximate-Exchange Protocol with Mutual-Help:** the agents not only exchange information, but also help each other out by giving up their resources to, and pursuing the goal of, a neighbor even to the detriment of theirs.

3.1 Attributes of Collaboration Protocols

Collaboration protocols specify the information that the agents exchange, the way they communicate, and how they use this information to solve the subproblems. Our intent is to look into the attributes that influence (if not determine) the scope of applications, and the properties, of the solution to $\{(P_m)\}$ by collaborative nets. In what follows, we list the attributes that have been studied in this research, and then compile them in a taxonomy. Figure 3.1 displays a tree of these attributes and some plausible values.

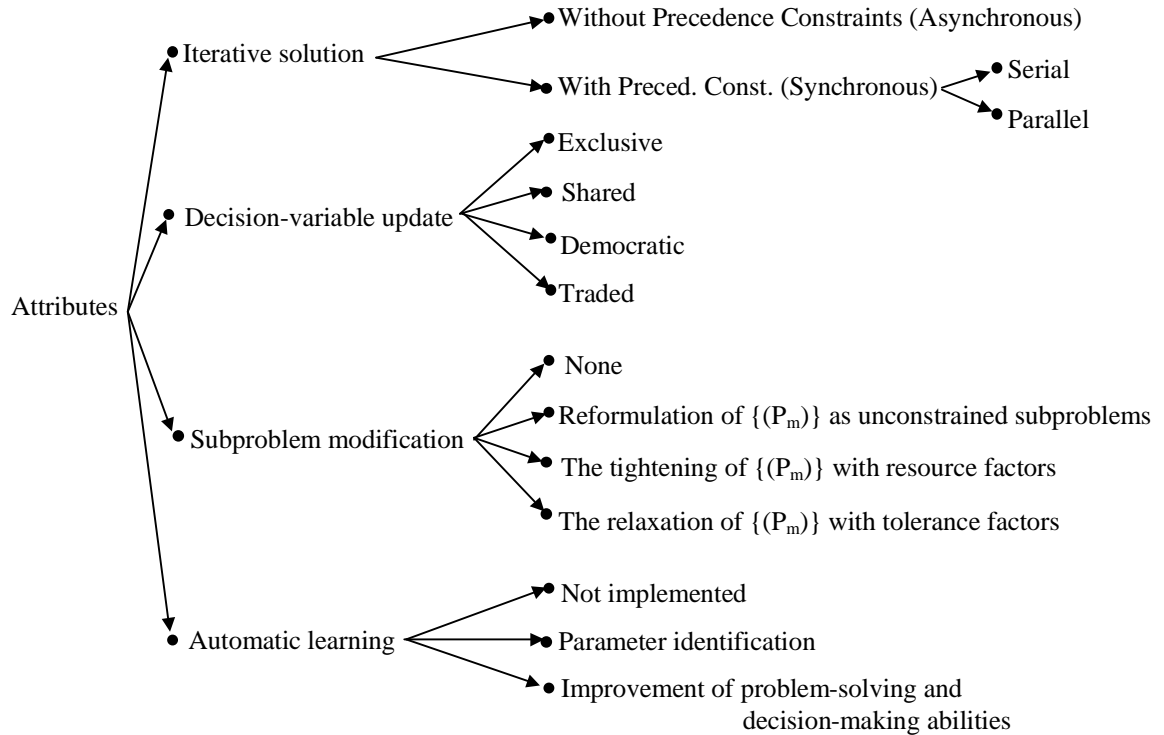


Figure 3.1: A tree of attributes for collaboration protocols and some plausible values.

3.1.1 Types of Iterative Solutions

By iterative solution, we mean any process that solves a problem in steps (or iterations) such that the outcome at each step- k is a function of the outcome at step- $(k-1)$. By synchronization, we mean any mechanism for ensuring that the computations at step- k are carried out only after the ones at step- $(k-1)$ are finished (or after the values necessary for the computations at step- k are ready) [Kun76][Pyo85]. That is, any mechanism that enforces the precedence constraints on the computations, from one iteration to the next. Synchronization, or its absence, has profound implications to how the agents exchange information and solve subproblems. Below, we present the types of iterative solutions in a general setting, and discuss the implications to collaboration protocols.

For the sake of simplicity, assume that (P) has only control variables, and that all sets of remote variables are empty. Then, the behavior of each agent- m can be expressed by the following iterative function: $U_m = S_m(U_1, \dots, U_M)$, where S_m returns the U_m that solves (P _{m}) for the given (U_1, \dots, U_M) . Using this notation, the types of iterative solutions can be described as follows:

- **Synchronous, Serial Solution:** the agents work one at time and use the current values of all variables to compute their controls. For the permutation $(1, \dots, M)$, the agents iterate as follows:

$$\begin{aligned} U_1(t_{k+1}) &= S_1(U_1(t_k), U_2(t_k), \dots, U_M(t_k)), \\ U_2(t_{k+2}) &= S_2(U_1(t_{k+1}), U_2(t_k), \dots, U_M(t_k)), \\ &\vdots \\ U_M(t_{k+M}) &= S_M(U_1(t_{k+1}), \dots, U_{M-1}(t_{k+M-1}), U_M(t_k)). \end{aligned}$$

(If the problem is to find a fixed point--that is, $U = (U_1, \dots, U_M)$ such that $U = (S_1(U), \dots, S_M(U))$ --for a system of M linear equations, then the serial, iterative solution is known as Gauss-Seidel method. It can be shown to converge in some cases, and diverge in others [GS93, pp.261-266]. Refer to [GL83, pp. 352-362] for an account of theoretical results on convergence.)

- **Synchronous, Parallel Solution:** all agents iterate concurrently, in lock-step, on the current values as follows:

$$\begin{aligned} U_1(t_{k+1}) &= S_1(U_1(t_k), U_2(t_k), \dots, U_M(t_k)), \\ U_2(t_{k+1}) &= S_2(U_1(t_k), U_2(t_k), \dots, U_M(t_k)), \\ &\vdots \\ U_M(t_{k+1}) &= S_M(U_1(t_k), U_2(t_k), \dots, U_M(t_k)). \end{aligned}$$

(For the problem of finding a fixed point for a system of linear equations, the parallel, iterative solution is known as Gauss-Jacobi method. It converges to a fixed point in some instances, but diverges in others [GS93, pp. 261-266]. There exists sufficient conditions for convergence of general mappings, that is, $U = S(U)$ where $U = (U_1, \dots, U_M)$ and $S = (S_1, \dots, S_M)$. In short, if S is a norm-contraction map for any vector norm $\|\bullet\|$ --that is, if there exists $0 \leq \alpha < 1$ such that $\|S(U) - S(V)\| \leq \alpha \|U - V\|$ for any U and V --then S defines a unique fixed point, U° , and the iterative solution converges to it [OR70].)

- **Asynchronous, Parallel Solution:** the agents ignore the precedence constraints and, therefore, can work in parallel, at their own pace. Each agent- m uses the latest values (probably not the current ones) to compute its controls as follows: $U_m = S_m(U_1^*, \dots, U_{m-1}^*, U_m, U_{m+1}^*, \dots, U_M^*)$, where U_k^* is the latest value (or an estimate) of U_k . (For a general map, the asynchronous solution arrives at a unique fixed point, U° , if S is an ∞ -norm contraction map [Ber83][Pyo85].)

In the context of collaboration protocols, we have considered the following types of iterative solutions:

- **Serial Iteration:** the agents work in lock-step by utilizing a synchronization (coordination) mechanism that guarantees the use of up-to-date information. On the one hand, the serial mode allows the agents to retain feasible constraints, if they choose to. Further, it facilitates, as will be seen in a moment, the analytical analysis of the protocol. On the other hand, the C-Net solution of $\{(P_m)\}$ is likely to be too sluggish for practical applications—only a small number of the agents might enjoy parallel work, while the majority agonize as they hold off their computations. Another drawback is the potentially dissimilar speed of agents that might compromise collaboration between slow and fast agents, such as humans and computer-based agents.
- **Asynchronous Iteration:** the agents work freely, without enforcing the precedence relations. This mode is very desirable because the agents work in parallel (at their own speed) and can respond quickly to disturbances. The difficulties arise in maintaining feasible coupling constraints, and promoting convergence of the solution to $\{(P_m)\}$ to an acceptable solution to the overall problem (P).

3.1.2 Types of Decision-variable Updates

In the framework for specifying the tasks of the agents (see Section 2.2 and 2.3), only agent- m has authority to set the control variables of problem (P_m) . However, agent- m can accept suggestions from, delegate its authority to, or trade values with, its neighbors. A few types of decision-variable update schemes follow below:

- **Exclusive Update:** the agents are self-interested, that is, they exclusively set the values of the control variables so as to solve their subproblems.
- **Shared Update:** the agents are not completely self-interested and let others update the control variables—they behave altruistically during catastrophes. Many possibilities exist for shared update. It could, for instance, be realized directly or indirectly. When the update is indirect, the helping agents simply let their neighbors update the proximate variables. The extent of the update may, however, be subject to limits and under conditions—the helping agent restricts the update within a region and imposes limits on the degradation of its own objective. When the update is direct, the helping agents act as suggested by the neighbors.
- **Democratic Update:** the agents are altruistic. They collect recommendations for the values of the control variables, use a mechanism to combine them, and finally implement the control actions that incorporate the assessment of their neighbors.

- **Traded Update:** the agents propose deals to their neighbors. For example, agent-m allows another agent, let us say agent-n, to set agent-m's control variables within a space, provided that agent-n sets its variables within a space suggested by agent-m. Alternatively, agent-m promises to optimize a function f_m (which is not necessarily its goal) as long as agent-n optimizes a function f_n . The trading could take place in an auction whereby the agents post bids and take offers.

3.1.3 Types of Subproblem Modifications

Collaboration protocols may prescribe, or not, the modification of the set of subproblems, $\{(P_m)\}$. Here, only creativity and insight bound the extent and form of the modifications. So far, we have explored the modifications listed below:

- **The reformulation of the subproblems in $\{(P_m)\}$ as unconstrained subproblems:** like in the rolling horizon approach, a solution to $\{(P_m)\}$ can be reached by solving a sequence of unconstrained problems; in particular, this strategy will prove to be fundamental in attaining global convergence under the proximate-exchange protocol.
- **The tightening of the constraints in $\{(P_m)\}$ with resource factors:** the factors may be implemented to make up for the uncertainties of asynchronous updates, and prevent the solution to $\{(P_m)\}$ from becoming infeasible for (P); resource factors and their applications in asynchronous protocols will be investigated in the forthcoming chapter.
- **The relaxation of the constraints in $\{(P_m)\}$ with tolerance factors:** the factors could be utilized in contexts that admit resource violations, perhaps for a limited time length, to speed up solution convergence.

3.1.4 Use of Automatic Learning

There exists opportunity to 1) boost the effectiveness, 2) widen the scope of applications, and 3) improve the adaptability of collaboration protocols with automatic learning techniques [Mit97]. The dimensions of the opportunities do not end here, but rather span the diverse space of learning techniques (such as Bayesian nets, neural nets, and reinforcement learning). Below, we speculate about this opportunity and enumerate two broad applications:

- **The automatic identification of parameters:** often, parameters appear explicitly in, or buried in the implementation of, collaboration protocols; examples of parameters are a) the values of the resource factors, and b) the weights for fusing together the recommendations in the voting protocol.
- **The transformation of past experience into improved problem-solving and decision-making abilities:** learning techniques could help the agents to infer the right decisions, in a global sense, from historical records and simulated events; additionally, they could speed up the solution of $\{(P_m)\}$ by providing the agents with a means to anticipate the values of neighborhood variables.

3.2 A Taxonomy of Collaboration Protocols

Here, we bring together the above attributes to assemble a taxonomy of collaboration protocols. The taxonomy corresponds to a space whose dimensions match these attributes. Figure 3.2 offers a partial view of the taxonomy, showing the projection of a few protocols onto the subspace spanned by the dimensions associated with 1) the type of iterative solution, and 2) the type of decision-variable update.

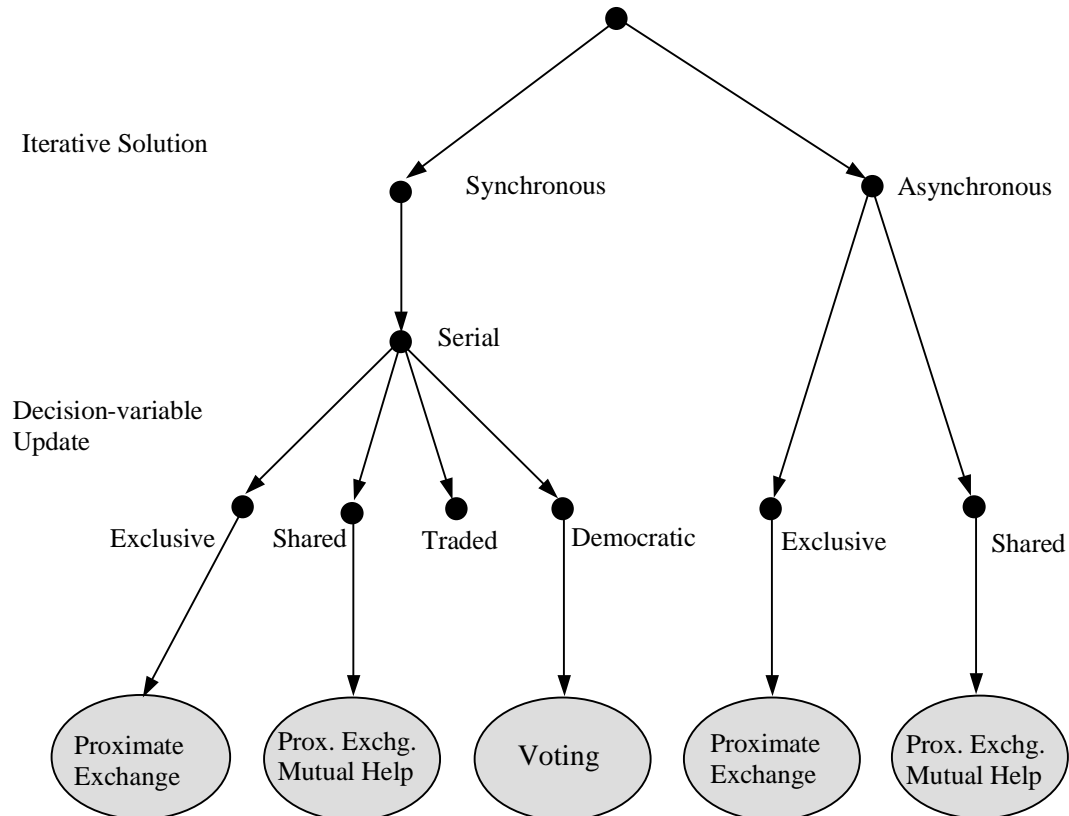


Figure 3.2: A partial tree representation of the taxonomy of collaboration protocols.

3.3 The Voting Protocol

The aggregated performance of the agents (that is, the quality of the solutions found to the subproblems) can be viewed as a function of their decision-making abilities. These abilities can be measured by the probability p_m that each agent- m makes, in a global sense, correct decisions. For instance, p_m could be the probability that agent- m correctly trips a breaker to de-energize a power transmission line after lightning.

How could the agents improve collective performance? This can be accomplished in two ways: 1) by improving the individual, innate decision-making abilities of the agents, or 2) by providing the agents with a means to collaborate so that they help one another to reach better collective decisions, even though they retain their innate abilities. The benefits of the first approach are usually limited, in the sense that the agents are physically constrained in their view of the overall network [CT99].

In the second approach, however, the agents can assess the context, provide their view of the network, and recommend (to their neighboring peers) what they believe is the correct decision. This potential is attractive and will be elaborated in this section.

3.3.1 A Simple Best-Case Analysis of Voting

The potential of collaboration can be explored by:

- 1) boosting each agent's abilities to recommend values for the variables controlled by its peers,
- 2) promoting the broadcast of the recommendations, and
- 3) encouraging the agents to use voting to synthesize a decision from the recommendations.

Our intent is to 1) illustrate the potential benefit of the voting protocol in a broad sense, rather than focusing on the specifics of boosting and voting, and 2) adopt the view of the optimist, carrying out a best-case analysis. Thus, we assume that:

- 1) the variables set by the agents are binary—they reflect the decisions necessary to solve the subproblems, and
- 2) the agents make independent and correct decisions with probability p .

Under these circumstances, simple majority, perhaps the simplest of all voting protocols, improves the collective decision-making abilities of the agents. (Even though the individual abilities of the agents remain the same, they help one another to reach better overall decisions.) The analysis of the voting protocol follows below.

Definition 3.1. p is the probability that an agent is correct when it makes a decision. The probability that the agent is incorrect is $q = (1 - p)$.

Definition 3.2. $p^*(n)$ is the probability that the majority of n independent agents is correct.

Definition 3.3. $p^*(n) = \sum_{j=\lceil n/2 \rceil}^n \binom{n}{j} p^j q^{(n-j)}$ is the probability that the majority of an odd number of collaborative agents, n , is correct.

Definition 3.4. $p^*(n) = \sum_{j=(n/2+1)}^n \binom{n}{j} p^j q^{(n-j)} + \frac{1}{2} \binom{n}{n/2} p^{(n/2)} q^{(n/2)}$ is the probability that the majority of an even number of collaborative agents, n , is correct.

Theorem 3.1. $p^*(n+1) = p^*(n)$ for n odd.

Proof. By induction on n . ■

Theorem 3.2. $p^*(n+1) = p^*(n) + \binom{n-1}{\lfloor (n-1)/2 \rfloor} p^{\lfloor (n-1)/2 \rfloor} q^{\lfloor (n-1)/2 \rfloor} (2p-1)$ for n even.

Proof. By induction on n . ■

Corollary 3.1. $p^*(n+1) > p^*(n)$ for n even if, and only if, $p > \frac{1}{2}$.

Proof. According to Theorem 3.2, $\binom{n-1}{\lfloor (n-1)/2 \rfloor} p^{\lfloor (n-1)/2 \rfloor} q^{\lfloor (n-1)/2 \rfloor} (2p-1) > 0$ if, and only, if $p > \frac{1}{2}$. ■

In synthesis, the probability that the majority of n agents is correct, $p^*(n)$, increases with the number of collaborative agents when $p > \frac{1}{2}$. This fact follows directly from Theorem 3.1 and Corollary 3.1. See Figure 3.3 for an illustration of the potential benefit of collaboration.

The above analysis shows the potential benefit of the voting protocol in a simple scenario. This bears out the application of voting, or its variations, to improve decision-making in real-world networks. Therein, however, the agents can:

- 1) exhibit dependencies, and often do,
- 2) be more (or less) competent than other possibly dissimilar agents, and
- 3) be able to improve their decision-making abilities with experience (self-learning).

Therefore, the crafting of voting protocols that account for these factors is essential to fully exploit the benefits of collaboration in real-world networks.

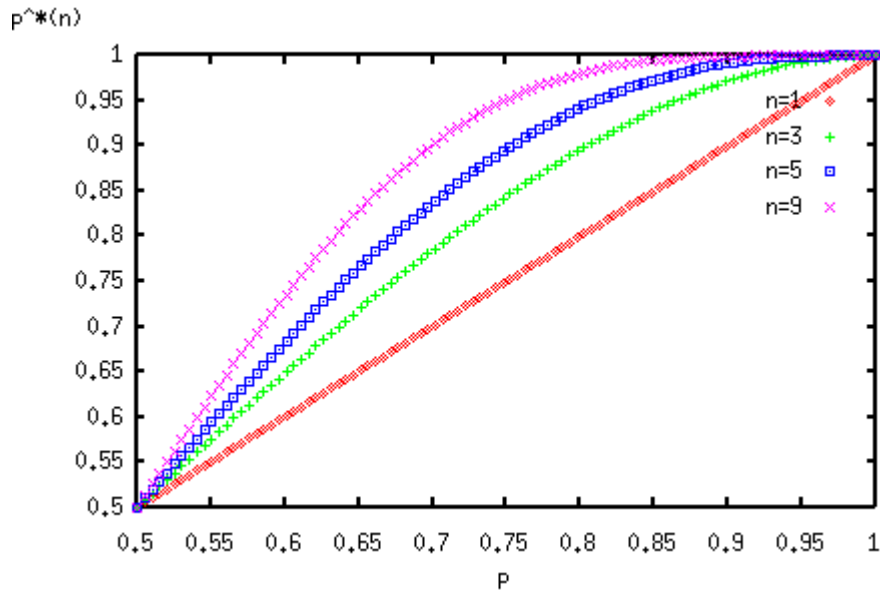


Figure 3.3: The collective decision-making ability, $p^*(n)$, for $n = 1, 3, 5,$ and 9 collaborative agents. The plot shows that $p^*(n)$ increases monotonically as the number of collaborative agents increases.

3.4 The (Serial) Proximate-Exchange Protocol

The power of the proximate-exchange protocol rests on the communication of relevant information among neighboring agents—an essential vehicle for collaboration in any domain. The agents iteratively, and sequentially, broadcast their plans to the neighboring ones, and revise these plans to account for the neighbors’ decisions, until they converge or time is up. At that point, the agents put into action the control plans that solve $\{(P_m)\}$.

When is the combined effort of these agents effective in solving $\{(P_m)\}$? It would appear that the self-interested effort of the agents hinders, rather than promotes effective collaboration. It turns out that the agents can collaborate effectively despite their self-interested behavior—that is, the one emerging from the optimization of the individual objective functions. The effectiveness of the protocol will be verified experimentally (in forests of pendulums and power grids). In the meantime, we focus on the analytical analysis of the protocol, that is, we formalize its rules and spell out sufficient conditions for the collaborations to be effective.

The steps of the proximate-exchange protocol are:

- 1) agent- m uses default values for its remote variables, the up-to-date values of its proximate variables, and the latest values of its neighborhood variables, to run an optimization procedure that attempts to solve (P_m) , or improve the current solution,
- 2) agent- m passes the newest values of its proximate variables to its neighbors,
- 3) agent- m repeats from step-1 until convergence is obtained or time is up, and
- 4) agent- m implements its decisions.

Herein, the agents are restricted to work serially in each neighborhood. They synchronize their actions to permit parallel work only among non-neighbors. This assumption facilitates the analytical analysis that culminates in sufficient conditions for the solution of $\{(P_m)\}$ to converge to a solution to (P) . In essence, the conditions state that:

- a) the collaborative net must provide a complete cover of the physical network Θ . This means that its agents must observe all state variables, and set all control variables, of the network Θ ,
- b) (P) must be convex [NW99],
- c) (P) must be strictly feasible,
- d) (P) must be differentiable,
- e) the collaborative net must be sufficiently dense to induce an exact match between each agent- m and its subproblem (P_m) ; in other words, the variables that appear in each subproblem (P_m) must be either in the proximate, or neighborhood, variable set of agent- m ,
- f) the agents must use an interior-point-method [NN94], and
- g) the agents must work serially within each neighborhood.

A more formal and complete list of conditions is given below.

Theorem 3.3. If:

- 1) The agents of the collaborative net provide a complete cover for the physical network, and they have exclusive authority over the control variables, that is,

$$\bigcup_{m=1}^M \{x_m(t_n)\} = x(t_n), \bigcup_{m=1}^M \{u_m(t_n)\} = u(t_n), \text{ and } u_i(t_n) \cap u_j(t_n) = \emptyset \text{ for all } i \neq j.$$

- 2) The match between each agent- m and its subproblem, (P_m) , is exact. That is,

$$\begin{aligned} f_m(X_m, U_m, Y_m, Z_m) &= f_m^*(X_m, U_m, Y_m) + f_m^*(Y_m, Z_m) = f, \\ G_m(X_m, U_m, Y_m, Z_m) &= G_m^*(X_m, U_m, Y_m), \text{ and} \\ H_m(X_m, U_m, Y_m, Z_m) &= H_m^*(X_m, U_m, Y_m). \end{aligned}$$

- 3) The functions f_m^* , f_m^* , and G_m^* are convex, continuously differentiable, and finite in their domain and H_m^* is linear.
- 4) The space of feasible solutions, S , can be relaxed by a factor $\delta > 0$ and its interior, $\text{int}(S)$, is non-empty. More precisely,

$$\begin{aligned} S &\equiv \{(X, U) \mid G(X, U) \leq 0 \text{ and } |H(X, U)| \leq \delta\}, \\ \text{int}(S) &\equiv \{(X, U) \mid G(X, U) < 0 \text{ and } |H(X, U)| < \delta\}, \text{ and} \\ \text{int}(S) &\neq \emptyset. \end{aligned}$$

- 5) The initial, overall solution belongs to the interior of the feasible set, that is, $(X^{(0)}, U^{(0)}) \in \text{int}(S)$.
- 6) Each agent- m uses an iterative barrier algorithm to solve its subproblem (P_m) . More formally, the agent runs an optimization procedure (such as Newton's method) to solve the following problem, or improve the current solution until the Wolfe conditions are satisfied [NW99, pp.39]:

$$\text{Minimize } \lambda_m(\alpha, X_m, U_m, Y_m) = \{f_m^*(X_m, U_m, Y_m) + \alpha B_m(X_m, U_m, Y_m)\},$$

where $\alpha > 0$ is a penalty factor, and B_m is a barrier function that is continuously differentiable and strongly convex (in the interior of the constraints), and also tending to infinity as one approaches the boundary of any constraint. (A plausible barrier function is

$$B_m(X_m, U_m, Y_m) = -\sum_{\forall i} \log(-G_{mi}^*(X_m, U_m, Y_m)) - \sum_{\forall i} \log(-H_{mi}^*(X_m, U_m, Y_m) + \delta) - \sum_{\forall i} \log(H_{mi}^*(X_m, U_m, Y_m) + \delta),$$

where $G_{mi}^*(X_m, U_m, Y_m)$ and $H_{mi}^*(X_m, U_m, Y_m)$ are the i -th elements of the inequality and equality constraint sets, respectively.)

- 7) The agents work serially within each neighborhood. Two agents may work in parallel only if they are non-neighbors. (Since the match is exact, two agents are neighbors—they are joined by a communication link—when their subproblems are coupled.)

Then: the overall solution obtained by the agents, $(X^{(0)}, U^{(0)})$, will converge to an optimal solution to the barrier version of problem (P) , that is, the problem of minimizing $\lambda(\alpha, X, U) = \{f(X, U) + \alpha B(X, U)\}$.

Proof. Let $(X^{(t)}, U^{(t)})$ be the overall solution at time t , and $(X^{(t+\Delta)}, U^{(t+\Delta)})$ be the next overall solution obtained after agent- m updates its proximate variables.

First, we show that $(X^{(t+\Delta)}, U^{(t+\Delta)}) \in \text{int}(S)$. From condition (4), from condition (5), and by induction on the update time t , solution $(X^{(t)}, U^{(t)}) \in \text{int}(S)$. Under condition (7), agent- m used the current values of its proximate variables, $(X_m^{(t)}, U_m^{(t)})$, and neighborhood variables, $Y_m^{(t)}$, to compute an improved solution $(X_m^{(t+\Delta)}, U_m^{(t+\Delta)})$; thereby, $(X_m^{(t)}, U_m^{(t)}, Y_m^{(t)}) \in \text{int}(S_m)$ (the interior of problem (P_m) 's space of feasible solutions). Since agent- m does not update the variables in Y_m , it follows that $Y_m^{(t)} = Y_m^{(t+\Delta)}$ and $(X_m^{(t+\Delta)}, U_m^{(t+\Delta)}, Y_m^{(t+\Delta)}) \in \text{int}(S_m)$. Under condition (2), none of the constraints in (P_m) depends on Z_m and, therefore, $(X^{(t+\Delta)}, U^{(t+\Delta)}) \in \text{int}(S)$.

Second, we show that $\lambda(\alpha, X^{(t+\Delta)}, U^{(t+\Delta)}) < \lambda(\alpha, X^{(t)}, U^{(t)})$. According to the above arguments and under condition (6), agent- m computes solution $(X_m^{(t+\Delta)}, U_m^{(t+\Delta)}, Y_m^{(t)})$, from solution $(X_m^{(t)}, U_m^{(t)}, Y_m^{(t)})$, such that $\lambda_m(\alpha, X_m^{(t+\Delta)}, U_m^{(t+\Delta)}, Y_m^{(t)}) < \lambda_m(\alpha, X_m^{(t)}, U_m^{(t)}, Y_m^{(t)})$. Since $Y_m^{(t)} = Y_m^{(t+\Delta)}$, and since all terms in $\lambda(\alpha, X, U)$ that depend on (X_m, U_m) also appear in $\lambda_m(\alpha, X_m, U_m, Y_m)$ and do not depend on the remote variables, Z_m , it follows that $\lambda(\alpha, X_m^{(t+\Delta)}, U_m^{(t+\Delta)}) < \lambda(\alpha, X_m^{(t)}, U_m^{(t)})$.

Third, we show that $(X^{(t)}, U^{(t)})$ converges to an optimal solution to the problem of minimizing $\lambda(\alpha, X, U)$. According to the above arguments, $(X^{(t)}, U^{(t)}) \in \text{int}(S)$ and $\lambda(\alpha, X^{(t)}, U^{(t)})$ decreases whenever an agent updates its results. Since a solution $(X^{(t)}, U^{(t)})$ is a local minimum for the overall barrier problem if and only if $(X^{(t)}, U^{(t)})$ defines a local minimum for the barrier problem of each agent- m (dropping the terms in $\lambda(\alpha, X, U)$ that do not depend on (X_m, U_m) results in $\lambda_m(\alpha, X_m, U_m, Y_m)$), there is always at least one agent capable of decreasing the value of $\lambda(\alpha, X^{(t)}, U^{(t)})$ if $(X^{(t)}, U^{(t)})$ is not a local minimum. According to Lemmas A.1 and A.2 (see Appendix A), the decrease in $\lambda(\alpha, X^{(t)}, U^{(t)})$ satisfies the Wolfe conditions [NW99, pp. 39-41] after the update of each agent, that is, each agent provides sufficient decrease in the overall barrier-objective-function for global convergence. This fact together with Theorem 3.2 of [NW99, pp. 43-45] imply that the sequence of solutions, $(X^{(n_0)}, U^{(n_0)})$, $(X^{(n_1)}, U^{(n_1)})$, ..., $(X^{(n_T)}, U^{(n_T)})$, is convergent to a local minimum (X^*, U^*) for the overall barrier problem. Indeed, the solution (X^*, U^*) is optimal because $\lambda(\alpha, X, U)$ is convex. ■

Corollary 3.2. If the conditions of Theorem 3.3 hold and the collaborative net solves a sequence of barrier problems, that is, if the agents minimize $\lambda(\alpha^k, X, U)$ such that $0 < \alpha^{(k+1)} < \alpha^k$ for all k , then the overall solution, $(X^{(k)}, U^{(k)})$, will arrive at an optimal solution to problem (P).

Proof. The proof follows directly from Theorem 3.3 and Proposition 4.1.1 of [Ber95a, page 314]. ■

Talukdar et al. [TC00] have emphasized that the above conditions set forth the issues that need further investigation, and must be resolved, before collaborative nets are deployed to real-world networks. The upshot is that the majority of the sufficient conditions, as will be seen in a moment, are difficult to meet and undesirable in practical applications. A qualitative analysis of these conditions follows below:

- **Condition-a** (the complete coverage of the network) could be implemented in practice.
- **Condition-b** (the convexity of the objective function and constraints) cannot be guaranteed in practice. The dynamics of real-world networks can be highly nonlinear and nonconvex, such as the power grid's.
- **Condition-c** (the feasibility of the initial solution) is also hard to be met. The specifications on how the network should behave in the future can introduce conflicts and make the problem infeasible. The resolution of the conflicts stands as a hard problem. Which constraints should be dropped, if any? Which constraints should be relaxed? These are open research questions.
- **Condition-d** (the differentiability of the objective and constraint functions) cannot be expected in all real-world problems. Not infrequently, the decisions are a mix of discrete and continuous variables that introduce non-differentiability in the functions, such as those found in mixed-integer optimization problems.
- **Condition-e** (the exact match of agents to subproblems) is impractical. Collaborative nets could become too dense to induce an exact match.
- **Condition-f** (the use of interior-point methods) is not convenient since interior-point methods are too brittle, and less robust than algorithms such as sequential-quadratic-programming (SQP) [BT96], which is one the best off-the-shelf algorithms for nonlinear programming [LZT94].
- **Condition-g** (the enforcement of serial work within neighborhoods) is quite impractical and unattractive. The convergence speed of C-Nets would be excessively slow and, therefore, they would not respond promptly to disturbances.

3.5 The (Serial) Proximate-Exchange Protocol with Mutual-Help

Equipment failures sporadically cripple network sites, tugging them to operate in the emergency mode. These incidents dramatically change the specifications on how the crippled subnetworks should operate. In the power grid, for instance, a lightning strike can inflict disturbances on nearby equipment that leaves no alternative to the control devices, but to switch from cost-minimization to synchronization-recovery mode.

Technically speaking, the incidents translate into a new set of subproblems, $\{(P_m)\}$, that reflect the operating modes of the subnetworks. The mode landscape of the network then becomes uneven, where unlucky agents find themselves matched to hard subproblems, while others enjoy the task of solving easy ones. The outcome could turn out quite undesirable, especially if the agents are self-interested and the incident is catastrophic. Potentially, the least troubled agents (those facing easy problems) could sacrifice their performances and relinquish resources to the most troubled agents (those working on hard problems) for the overall good.

Proximate exchange with mutual-help encourages the agents to help one another. The form and extent of the help can be molded into the agents in a number of ways. In this regard, this section imparts the essentials of help by direct, shared update (Section 3.1.2), and dwells on to speculate about sufficient conditions for convergence. The protocol extends basic proximate-exchange by implanting in each agent- m the abilities to:

- 1) recognize that a neighboring agent needs help,
- 2) determine the extent to which the neighbor should be free to modify agent- m 's proximate variables, and
- 3) transfer the rights to update the proximate variables.

To keep the material short, we provide only the steps of a troubled agent- m as follows:

- 1) agent- m enlarges (P_m) to encompass the variables over which its neighbors delegated authority,
- 2) agent- m uses the default values of its remote variables, and the latest values of its proximate and neighborhood variables, to solve or improve the solution to (P_m) ,
- 3) agent- m hands to its neighbors the values of the variables it was given authority to set,
- 4) agent- m reiterates from step-2 until convergence is attained, or time is up, and
- 5) agent- m puts into action the control plan.

The steps constitute general guidelines—they do not say how to grow mutual-help abilities. This gap, however, does not limit the use of the protocol. On the contrary, it adds flexibilities and therefore allows the crafting of the abilities that best suit the application at hand. In Chapter 5, we explore the heuristic insights of the network problem (synchronization recovery in forests of pendulums) and adapt sensitivity analysis to embed the required abilities (into the agents). A relevant issue is whether or not, the mutual-help abilities promote convergence to a solution to (P) . Below, we speculate about the sufficient conditions for convergence:

- **Subproblem Cover**

The covering condition entails a reduction to the conditions that we have identified for basic proximate-exchange. To explain this point, let agent-m be the troubled agent, and agent-n be the helping one. Suppose that agent-m coherently merges (P_m) and (P_n) into $\{(P_m), (P_n)\}$, that is, agent-m drops the duplicate constraints and terms of the objective function. Further, suppose that agent-m incorporates agent-n's view and takes over agent-n's authority—perhaps with the assistance of agent-n. It is as if agent-m and agent-n were replaced by an agent that solves $\{(P_m), (P_n)\}$. If the sufficient conditions for proximate exchange hold (Theorem 3.3), and if each helping agent is exclusively covered by exactly one troubled agent, then the solution of $\{(P_m)\}$ reaches a solution to (P) .

- **Subproblem Approximation**

This condition bears on the troubled agents extending their subproblems to nearly cover the helping agents' subproblems. Approximate cover may prove to be more practical than full cover.

For one thing, subproblem approximation does not incur as much computational burden as subproblem cover does. For another, the approximations prevent the troubled agents from ignoring the needs of the helping agents. We conjecture that the following conditions are sufficient to ensure convergence:

- 1) the conditions of the proximate-exchange protocol hold (Theorem 3.3),
- 2) each helping agent-n passes a set of feasible updates for its control variables to the troubled agent-m,
- 3) agent-n provides an approximation of how its objective function varies with changes on its control variables, and
- 4) agent-m adds the approximation of f_n to f_m , and covers the neighborhood of agent-n.

The conjecture is more formally stated in Appendix A.

Herein, the sufficient conditions inherit the same disabilities of the ones found for serial proximate-exchange—especially, serial work and convexity. Thus, the issues raised in the preceding section carry over to the mutual-help version of serial proximate-exchange.

3.6 The Asynchronous Proximate-Exchange Protocol

This protocol is identical to basic proximate-exchange, except that the agents are free to work asynchronously—in parallel and at their own pace. We have brought forth the advantages and disadvantages of asynchronous work. On the one hand, they allow quick response of the collaborative nets and facilitate collaboration between dissimilar agents. On the other hand, analytical guarantees seem hard to be established for all but the simplest problems. This scenario should not be discouraging. It is not exceptional, but typical of real-world networks. Often, the lack of analytical guarantees is overcome through extensive experimental analysis that is undertaken prior to implementation.

Talukdar [Tal99a] has brought forward the potential of resource factors to soften the side effects of asynchronous work and, at the same time, promote effective collaboration between dissimilar agents. The next chapter addresses this potential in depth.

3.7 Summary

The chapter listed the attributes that loom large in the design of collaboration protocols. They include the following: 1) the type of iterative solution, 2) the type of decision-variable update, 3) the problem modification, and 4) the use of automatic learning. The list provided plausible instantiations for these attributes, and distilled their implications to the protocols (and the resulting solutions of $\{(P_m)\}$ by C-Nets). The study therein shows that the type of iterative solution is very influential to the merit, and can seal the fate, of any protocol. For instance, a protocol that enforces serial, iterative work is impractical in large, widespread networks.

The above attributes were arranged together to assemble a taxonomy of protocols, some of which have been studied in this research. These protocols comprise 1) voting, 2) serial proximate-exchange, 3) serial proximate-exchange with mutual-help, and 4) asynchronous proximate-exchange.

The voting protocol illustrated the potential benefit of collaboration—rather than improving the agents' abilities, which can be difficult in face of their limited views, provide the agents with a means to collaborate and improve collective performance.

Proximate exchange served as a basic protocol, which gives insights into issues that need further analysis. For its serial version, sufficient conditions were found for the labor of the agents to reach a solution to the overall problem. The majority of these conditions are, however, impractical such as the

convexity of (P), exact match, and serial work. On the optimistic side, the asynchronous version of proximate exchange (combined with mutual-help behavior) stands as a promising protocol, whose merit is still to be confirmed in the coming chapters.

Chapter 4

Using Resource Margins

The preceding chapter presented sufficient conditions for the collaborative net solution of the subproblems, $\{(P_m)\}$, to reach a solution to the static optimization problem, (P) . The conditions fence in the agents to work serially in each neighborhood, and refuse any imperfection in the match of agents to subproblems. We need to alleviate the pain of serial work; otherwise, C-Nets may become inapt to recover subnetworks that have been crippled by disturbances.

What are the impediments to asynchronous work? They are twofold. First, being devoid of synchronization, the agents can concurrently change the solutions to the subproblems, potentially violating the coupling constraints, and making the solution to $\{(P_m)\}$ infeasible to (P) . Second, guarantees of convergence, let alone convergence to good (and feasible) solutions to (P) , seem very hard to come up with. Further, the two obstacles are entangled with the sufficient conditions, such as the convexity of (P) and exact match.

How could the agents turn loose, work asynchronously, and still reach good, feasible solutions to (P) ? The facets of this research question (feasibility and convergence) appear intimately related to the dynamic control problem (DCP), the implementation issues, and the trade-offs between solution speed and ease of constraint satisfaction. This chapter sheds light on the feasibility facet of the research question. Specifically, it focuses on the use of resource margins, as suggested by Talukdar [Tal99a] [Tal00a], to keep the solution to $\{(P_m)\}$ feasible to (P) when the agents iterate asynchronously. In the next chapter, we turn the attention to the convergence facet and complement the material herein. We show experimentally (in a class of control problems) that the agents find good solutions to (P) , even though they work asynchronously and cannot afford an exact match.

This chapter is organized as follows. Section 4.1 provides the specifics of the feasibility research question, and then formalizes the implementation of resource margins in $\{(P_m)\}$. Section 4.2 describes an optimization-based procedure that checks whether or not, the given resource margins are sufficient to sustain the feasibility of (P) . Section 4.3 presents conditions that are necessary and sufficient, to ensure that the asynchronous solution retains the feasibility of special types of constraints—namely, those

defining ellipsoids without rotation. Section 4.4 suggests a procedure to identify resource margins that guarantee feasibility and, together with mutual-help behavior, promote convergence to good solutions. Section 4.5 illustrates the use of the procedure in two problems, one convex and the other nonconvex. Section 4.6 touches upon the use of resource margins to improve the aggregated performance of agents whose speeds are dissimilar.

4.1 Introduction

Ideally, the agents would solve the subproblems asynchronously—at their own pace and in parallel—and still find good, feasible solutions to each element of the series of static problems, $\langle(P)\rangle$. This behavior is hard, if not impossible, to achieve in its entirety for all classes of (P) . The hardness is even more pronounced when the elements of $\langle(P)\rangle$ change continually as constraints are added (or dropped) to reflect the modes of the subnetworks.

While the asynchronous work displays the ideal behavior in some classes of problems, it can become helpless in others if measures are not taken (to account for the lack of synchronization). In forests of pendulums, as will be shown in Chapter 5, the agents consistently yield good solutions even though they boldly ignore the conditions for convergence—they cover small areas of influence, and never wait for a reply from nearby agents. In problems that contain equalities however, one has to approximate each equality with two inequalities and, thereby, provide “maneuvering room” for the asynchronous work to find near-feasible solutions—that is, one has to trade off speed against near feasibility. Therefore, this chapter assumes that the constraints in (P) are, or can be approximated with, inequalities.

How do we mitigate the undesirable side effects of asynchronous work? (That is, how do we prevent infeasibility and promote convergence?) Hereafter, the attention is on modifications in $\{(P_m)\}$, and behavior implants in the agents that pull them toward good, feasible solutions to (P) . In short, the recipe squeezes the feasible space of $\{(P_m)\}$ with resource margins that 1) anticipate the danger of asynchronous update and 2) retain a feasible solution. It also embeds mutual-help attitudes in the agents to promote convergence to good solutions. This attitude encourages each agent to sacrifice its goal if it believes, or is told by nearby agents, that an action improves collective performance.

4.1.1 The Implementation of Resource Margins

The resource margins consist of a set of vectors, $\{\alpha_m\}$, one for each subproblem (P_m) , that tighten the inequality constraints in $\{(P_m)\}$. The resource margins of agent- m have precisely one non-negative entry

for each constraint of (P_m) . After the implementation of the resource margins, agent- m 's subproblem becomes:

$$\begin{aligned} (P_m) \text{ Min } & f_m(X_m, U_m, Y_m, Z_m) \\ \text{s.t.} & \\ & G_m(X_m, U_m, Y_m, Z_m) \leq -\alpha_m. \end{aligned}$$

4.2 Sufficient Conditions for Feasibility

Consider problem (P) , its decomposition into $\{(P_m)\}$, and the set of resource margins $\{\alpha_m\}$. If the agents begin with a feasible solution, what conditions would be sufficient to keep it feasible when the agents solve $\{(P_m)\}$ asynchronously? This section offers an optimization-based procedure to test the sufficiency of $\{\alpha_m\}$. The procedure prescribes a) the solution of optimization problems to compute bounds on the values of the state and control variables, and b) the feasibility check of each constraint in (P) for the most adversarial circumstances under the bounds.

To keep the description of the procedure succinct, we omit the test of the control variables and the time index, t_n , of the discrete points in the time horizon (i.e., we ignore $u_i(t_n)$ and denote $x_j(t_n)$ as simply x_j). Let $lb(x_j)$ and $ub(x_j)$ be the lower and upper bound on variable x_j , respectively. Let also $\phi(m,j)$ be 1 if agent- m can set the value of x_j (if x_j is a proximate variable of agent- m) and 0 otherwise. The steps of the procedure are given below:

Step 1: Compute bounds on each variable x_j .

For each agent- m that can set the value of x_j , $\phi(m,j) = 1$, solve the optimization problems below:

$$\begin{aligned} lb(m,x_j) = \text{Min } & x_j \\ \text{s.t.} & \\ & G_m(X_m, U_m, Y_m, Z_m) \leq -\alpha_m \\ & X_m, U_m, Y_m, \text{ and } Z_m \text{ are the decision variables;} \end{aligned}$$

$$\begin{aligned} ub(m,x_j) = \text{Max } & x_j \\ \text{s.t.} & \\ & G_m(X_m, U_m, Y_m, Z_m) \leq -\alpha_m \\ & X_m, U_m, Y_m, \text{ and } Z_m \text{ are the decision variables.} \end{aligned}$$

Then, the bounds on x_j can be computed as: $lb(x_j) = \text{Min}\{lb(m,x_j) : \phi(m,j) = 1\}$ and $ub(x_j) = \text{Max}\{ub(m,x_j) : \phi(m,j) = 1\}$.

Step 2: Check whether or not it is possible to violate each constraint in (P) .

For each constraint g of G , solve the optimization problem below:

$$\begin{aligned}
g^* &= \text{Max } g(X,U) \\
&\text{s.t.} \\
&\quad lb(x_j) \leq x_j \leq ub(x_j) \quad \text{for all } j \\
&\quad lb(u_i) \leq u_i \leq ub(u_i) \quad \text{for all } i.
\end{aligned}$$

If $g^* > 0$ for any g , then report that $\{\alpha_m\}$ does not pass the test. Or else, report that $\{\alpha_m\}$ preserves the feasibility of (P).

Two nice aspects of the test are 1) its potential of being effective when (P) is convex, and 2) the fact that $\{\alpha_m\}$ preserves the feasibility of (P) under any collaboration protocol. The test, however, is conservative and probably inapplicable when the constraints are nonconvex. For one thing, the test would rely on global optimization and, in addition, the resource margins could excessively squeeze the constraints and leave $\{(P_m)\}$ infeasible.

Theorem 4.1. If the set of resource margins, $\{\alpha_m\}$, passes the test in Section 4.2, then $\{\alpha_m\}$ is sufficient to keep (P) feasible when the agents solve $\{(P_m)\}$ asynchronously.

Proof. (By contradiction) Suppose the converse and let (X^*, U^*) be the first infeasible solution to (P) found by the agents. Then, there must be an element g of G such that $g(X^*, U^*) > 0$. According to step-2 of the test, there exists (without loss of generality) j such that $x_j^* > ub(x_j)$. Clearly, the value of x_j was set to x_j^* by an agent- m such that $x_j \in X_m$, that is, x_j is proximate to agent- m . Immediately before that moment, agent- m used some values (Y_m^\perp, Z_m^\perp) for (Y_m, Z_m) to compute the values (X_m^*, U_m^*) for (X_m, U_m) . If agent- m did so, then the following subproblem is feasible:

$$\begin{aligned}
&\text{Min } f_m(X_m, U_m, Y_m^\perp, Z_m^\perp) \\
&\text{s.t.} \\
&\quad G_m(X_m, U_m, Y_m^\perp, Z_m^\perp) \leq -\alpha_m.
\end{aligned}$$

Since agent- m updates its proximate variables only if (P_m) is feasible, and according to step-1 of the test, it follows that the inequalities below are valid:

$$\begin{aligned}
x_j^* &\leq \text{Max } x_j && \leq ub(m, x_j) \leq ub(x_j). \\
&\text{s.t.} \\
&\quad G_m(X_m, U_m, Y_m^\perp, Z_m^\perp) \leq -\alpha_m \\
&\quad X_m \text{ and } U_m \text{ are the decision variables.}
\end{aligned}$$

This implies the contradiction: $ub(x_j) < x_j^* \leq ub(x_j)$. ■

4.3 Resource Margins for Ellipsoids

This section identifies sufficient and necessary conditions on $\{\alpha_m\}$ to ensure the feasibility of (P) when the constraints define multiple ellipsoids without rotation—balls that are stretched along the axes and displaced from the origin. The conditions are gradually developed from one ball centered at the origin, to one ellipsoid, and, finally, to multiple ellipsoids.

Let $\{\sum_{m=1,\dots,M}(u_m)^2 + \sum_{n=1,\dots,N}(x_n)^2 \leq R\}$ be the only constraint that binds a network of M agents. The conditions state that $\{\alpha_m\}$ keeps the ball-type constraint feasible under asynchronous work if, and only if, $\{\sum_{m=1,\dots,M}(\alpha_m) \geq (M-1)R\}$. These conditions can be readily extended to one and several ellipsoids without rotation. A formal statement of the conditions is provided for a ball in Theorem 4.2, for one ellipsoid in Corollary 4.1, and for multiple ellipsoids in Proposition 4.1.

Theorem 4.2. If:

- 1) The only constraint of (P) is: $\sum_{m=1}^M (u_m)^2 + \sum_{n=1}^N (x_n)^2 \leq R$.
- 2) Agent- m has exclusive authority over the control variable u_m .
- 3) The value of state variable x_n is concurrently set by at least one agent. Agent- m can set its value only if $\phi(m,n) = 1$. Otherwise, $\phi(m,n) = 0$.
- 4) Agent- m solves the following subproblem:

$$(P_m) \text{ Minimize } f_m$$

$$\text{Subject to: } (u_m)^2 + \sum_{\substack{i=1 \\ i \neq m}}^M (u_i^*)^2 + \sum_{\substack{n=1 \\ \forall n, \phi(m,n)=1}}^N (x_n)^2 + \sum_{\substack{n=1 \\ \forall n, \phi(m,n)=0}}^N (x_n^*)^2 \leq R - \alpha_m,$$

where:

- a) (u_i^*) is the latest value of control variable u_i , which agent- i sent to agent- m ,
 - b) (x_n^*) is the latest value of state variable x_n , which agent- m received from a neighbor, and
 - c) α_m is the resource margin for agent- m .
- 5) The initial values of the variables are feasible for (P).
 - 6) When the constraint in (P_m) is infeasible, agent- m a) leaves its proximate variables untouched (it delays its actions until the constraint becomes feasible), or b) sets their values to zero.

Then: the asynchronous solution of $\{(P_m)\}$ is feasible to (P) if, and only if, $\{\sum_{m=1,\dots,M}(\alpha_m) \geq (M-1)R\}$.

Proof. (\Leftarrow) According to condition (5), the constraint in (P) is initially feasible and, therefore, we have to show that it remains feasible. Let $\gamma(m,n)$ be 1 if agent- m was the last agent to set the value of state

variable x_n . Otherwise, let $\gamma(m,n)$ be 0. (Notice that $\phi(m,n) = 0$ implies $\gamma(m,n) = 0$.) At the beginning, assume that, for each state variable x_n , $\gamma(m,n) = 1$ for exactly one agent- m . From these observations and condition (6), the inequalities below follow:

$$(u_m)^2 + \sum_{\forall n, \gamma(m,n)=1} (x_n)^2 \leq R - \alpha_m \text{ is a valid inequality for } m = 1, \dots, M.$$

Because precisely one agent made the last change on the value of x_n , each state variable x_n must appear exactly once in the above constraints. Adding up these constraints over all agents, yields the following inequalities:

$$\begin{aligned} \sum_{m=1}^M (u_m)^2 + \sum_{n=1}^N (x_n)^2 &\leq MR - \sum_{m=1}^M (\alpha_m) \Rightarrow \sum_{m=1}^M (u_m)^2 + \sum_{n=1}^N (x_n)^2 \leq MR - (M-1)R \\ &\Rightarrow \sum_{m=1}^M (u_m)^2 + \sum_{n=1}^N (x_n)^2 \leq R. \end{aligned}$$

The last inequality implies the feasibility of (P).

(\Rightarrow) By contradiction, assume that $\sum_{m=1, \dots, M} (\alpha_m)^2 < (M-1)R$. It is possible that the latest values of each agent- m 's neighborhood variables are all zero ($u_i^* = 0$ and $x_n^* = 0$) and that agent- m 's proximate state variables are all set to zero ($x_n = 0$). If each agent- m chooses to set its control variables to the maximum possible values, $(u_m)^2 = R - \alpha_m$, then the following inequalities result:

$$\begin{aligned} (u_m)^2 = R - \alpha_m \text{ for } m = 1, \dots, M &\Rightarrow \sum_{m=1}^M (u_m)^2 = MR - \sum_{m=1}^M (\alpha_m) > MR - (M-1)R \\ &\Rightarrow \sum_{m=1}^M (u_m)^2 > R \Rightarrow \sum_{m=1}^M (u_m)^2 + \sum_{n=1}^N (x_n)^2 > R. \end{aligned}$$

The last inequality implies the infeasibility of (P) and, consequently, a contradiction. ■

Corollary 4.1. Let the constraint in Theorem 4.2 be of the form: $\sum_{m=1}^M [a_m(u_m - u_{0,m})]^2 + \sum_{n=1}^N [b_n(x_n - x_{0,n})]^2 \leq R$,

where a_m and b_n are nonzero scaling constants, and $u_{0,m}$ and $x_{0,n}$ are displacement constants. Then, the asynchronous solution of $\{(P_m)\}$ is feasible to (P) if, and only if, $\{\sum_{m=1, \dots, M} (\alpha_m) \geq (M-1)R\}$.

Proof. The demonstration is based on a change of variables. Let $u_{1,m} = a_m(u_m - u_{0,m})$ for $m = 1, \dots, M$, and let $x_{1,n} = b_n(x_n - x_{0,n})$ for $n = 1, \dots, N$. Then the constraint in the overall problem becomes the following:

$$\sum_{m=1}^M (u_{1,m})^2 + \sum_{n=1}^N (x_{1,n})^2 \leq R.$$

The result follows from the application of Theorem 4.2 to the above constraint, and the fact that there is bijective function between each pair of control variables (u_m and $u_{1,m}$) and each pair of state variables (x_n and $x_{1,n}$). ■

Proposition 4.1. If the constraints in (P) are of the form in Corollary 4.1, then the asynchronous solution of $\{(P_m)\}$ is feasible to (P) if, and only if, the resource margins satisfy the conditions of Corollary 4.1 for each constraint.

4.4 Computing Resource Margins

This section works out the steps that we have found helpful for calculating resource margins. It arranges the steps together in a procedure that records our insights in setting margins to uphold the feasibility of $\{(P_m)\}$. The steps are displayed schematically in Figure 4.1 and described below.

Step 1: Approximate Equality Constraints with Two Inequalities

Approximate the equality constraints $\{H(X,U) = 0\}$ with the inequalities $\{|H(X,U)| \leq \epsilon\}$ or, equivalently, $\{H(X,U) \geq -\epsilon \text{ and } H(X,U) \leq \epsilon\}$. The looser the approximation—that is, the larger the entries of vector ϵ , the easier the search for suitable resource margins. The following steps assume that:

- a) the constraints are inequalities of the form $\{G(X,U) \leq 0\}$, and
- b) the interior of the feasible set, $\text{int}(S) = \{(X,U) : G(X,U) < 0\}$, is non-empty.

(If an equality h of H cannot be relaxed, then the coupled agents must form a neighborhood and confine themselves to work serially or, alternatively, they can merge into a single agent. Unless h is not a hard constraint such as those used for prediction.)

Step 2: Approximate the Space of Feasible Solutions with Convex Sets

The chores in the subsequent steps are much simpler when G delimits a convex set, or G is adequately approximated with convex inequalities G^\perp . By adequately, we mean that the “volume” mismatch between G and G^\perp is small, that is, $\{\text{Vol}[\text{Sol}(G) \cap \text{Sol}(G^\perp)] \gg \text{Vol}[\text{Sol}(G) - \text{Sol}(G^\perp)] + \text{Vol}[\text{Sol}(G^\perp) - \text{Sol}(G)]\}$ where $\text{Sol}(G) = \{(X,U) \mid G(X,U) \leq 0\}$. How could a convex approximation G^\perp be found? The algorithm proposed by Director et al. [DMS90, §2.1, pp. 46-57] is a candidate for this task. How could the volume mismatch between G and G^\perp be calculated? It can be estimated through sampling [DFK91].

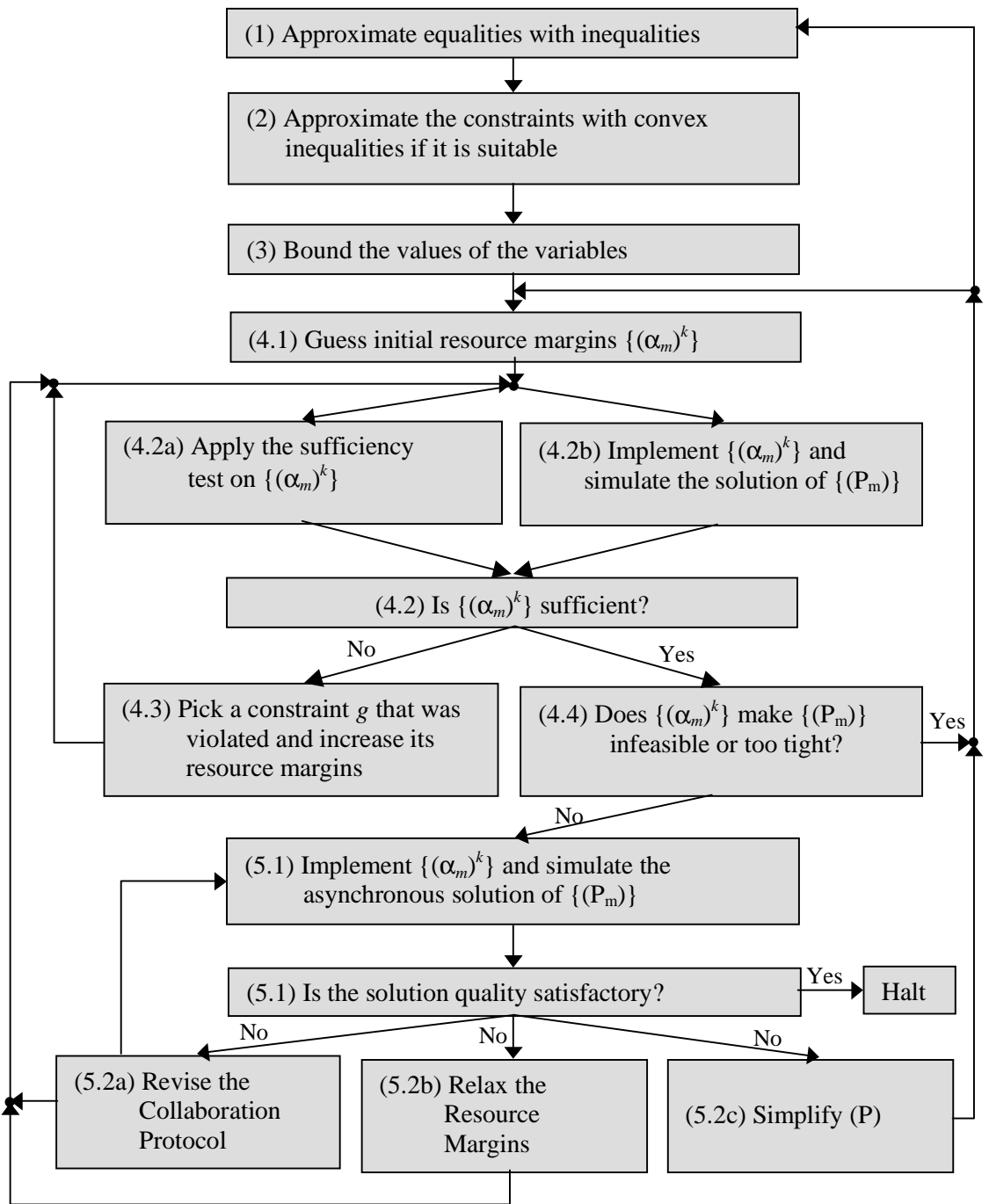


Figure 4.1: The diagram of the procedure for computing resource margins.

Step 3: Set Bounds for the Space of Feasible Solutions

The boundedness of the feasible space facilitates the search for adequate resource margins. If the feasible space is unbounded, then set lower and upper bounds for the variables. It should not be hard to figure out bounds because state and control variables are often, if not always, subject to limits.

Step 4: Find Resource Margins That Ensure Feasibility

This step begins by guessing an initial set of resource margins, $\{(\alpha_m)^{(k)}\}$, and then it iteratively checks whether or not $\{(\alpha_m)^{(k)}\}$ retains the feasibility of (P) and revises $\{(\alpha_m)^{(k)}\}$ accordingly. It checks $\{(\alpha_m)^{(k)}\}$ analytically when the constraints are ellipsoids, numerically when (P) is convex, and through simulation when the constraints are general. The steps within this step are provided below.

Step 4.1: Guess an initial set of resource margins, $\{(\alpha_m)^{(0)}\}$, and set $k = 0$.

Step 4.2: Does $\{(\alpha_m)^{(k)}\}$ keep (P) feasible? The procedure suggests two methods to figure it out. Use the first method when (P) is convex, otherwise use the second one.

- **Method-1 (Sufficient Conditions).** Check if $\{(\alpha_m)^{(k)}\}$ passes the sufficiency test proposed in Section 4.2 or (when the constraints are ellipsoids without rotation) if the conditions of Proposition 4.1 hold. Recall that:
 - 1) Method-1 is probably inapplicable when (P) is nonconvex,
 - 2) the tests are conservative in the sense that they anticipate the worst-cases, and
 - 3) the tests rely on the solvability of global optimization problems.
- **Method-2 (Simulation).** Start with the current solution, $(X^{(k)}, U^{(k)})$, simulate the asynchronous solution of $\{(P_m)\}$ with $\{(\alpha_m)^{(k)}\}$ in place, and report any violation of the constraints in (P). To extend the validity of $\{(\alpha_m)^{(k)}\}$ from (P) to $\langle(P)\rangle$, begin the asynchronous solution from several initial solutions, covering the most likely states of the network. Notice that $\{(\alpha_m)^{(k)}\}$ is sufficient to the extent to which the simulations are comprehensive and accurate.

Step 4.3: If a constraint g does not pass the sufficiency test (Method-1) or if its violation was reported (Method-2), then increase the resource margins for g in all elements of $\{(P_m)\}$ in which it appears, increment k by 1, and go back to Step 4.2.

Step 4.4: If $\{(\alpha_m)^{(k)}\}$ is overly tight—if $\{(P_m)\}$ becomes infeasible even for the current solution, then either simplify (P) and go back to Step 1, or relax the resource margins and restart from Step 4.1.

Step 5: Evaluate and Improve the Quality of the Solutions

This step amounts to 1) simulate the asynchronous work of the agents, 2) evaluate the quality of the resulting solutions, and then 3) revise the collaboration protocol, and the resource margins, if the quality is not satisfactory. The steps within this step are given below.

Step 5.1: Evaluate solution quality. Start with the current solution, or several solutions, that are feasible for $\{(P_m)\}$ with $\{(\alpha_m)^{(k)}\}$, simulate the asynchronous work, and evaluate the quality of the solutions that the agents produce.

Step 5.2: Is the quality satisfactory? If so, then quit. Or else, follow one of the recommendations below:

- **Revise the Collaboration Protocol.** By simply running proximate exchange and implementing resource margins, the agents may stall at dissatisfactory solutions. This can be attributed to their intrinsic behavior—namely, the agents chew up the available “resources” to achieve their goals. In an effort to discourage greediness and improve collective performance, embed (or revise) a means for the agents to help one another explicitly. More specifically, provide the agents with estimators of the overall (or local) benefits of changes in the neighborhood variables (or resource margins), and compel them to use the estimations to help each other.
- **Loosen Up the Resource Margins.** Loosen up the resource margins and go back to step 4.2.
- **Relax the Problem.** Relax (P) by dropping, or simplifying constraints, and return to step 1.

4.5 Examples

This section illustrates the use of resource margins in two problems: one convex and the other nonconvex. Specifically, the above procedure was followed to find margins (that ensure the feasibility of (P)) and tune a proximate-exchange protocol (to promote convergence to good solutions). The convexity of the first problem facilitated the search for resource margins. The nonconvexity of the second problem, however, prevented the use of the sufficiency test. Instead, simulation was used to find margins and mutual-help behavior was embedded in the agents.

4.5.1 Example 1

The problem is formulated in mathematical programming as follows:

$$(P) \text{ Minimize } f(x_1, x_2) = 3x_1^2 - 22x_1 + 51 - 2x_1x_2 + 2x_2 + x_2^2$$

Subject to:

$$-2x_1 + 3x_2 - 16 \leq 0$$

$$x_1 + 3x_2 - 28 \leq 0$$

$$3x_1 + x_2 - 28 \leq 0$$

$$4x_1 - x_2 - 28 \leq 0$$

$$-x_1 - 4x_2 + 7 \leq 0$$

$$-x_1 - x_2 + 4 \leq 0$$

$$-2x_1 + x_2 - 4 \leq 0.$$

The specifics of the collaborative net that solves (P) are:

- 1) the C-Net is made up of two agents: agent-1 controls x_1 and agent-2 controls x_2 ; and
- 2) the agents work asynchronously and under the proximate-exchange protocol—that is, each agent- m initializes (P_m) with the latest value of its neighborhood variable, solves (P_m) , and then sends a message to its neighbor, carrying its decision.

To compute resource margins, we walked through the procedure (delineated in Section 4.4), and found $\alpha_1 = (1/2, 1/2, 3, 3, 1/2, 4, 4)$ and $\alpha_2 = (9/2, 9/2, 1/2, 3, 5, 1/2, 2)$ to be adequate. Figure 4.2 illustrates the feasible space, and the contour lines of the objective function, of problem (P). The figure also shows agent-2's feasible region after implementing α_2 in (P_2) .

In step 5 of the procedure, the asynchronous work was simulated, and the trajectory of the solution was recorded, for several starting conditions. Figure 4.3 offers the trajectories of three runs, together with the region of the solutions that the agents can reach when $\{\alpha_1, \alpha_2\}$ is in place. In all runs, consistent convergence to the optimal solution was observed and, as anticipated, constraints violations never occurred. (The inner region was estimated by letting the agents optimize random, linear objective functions.)

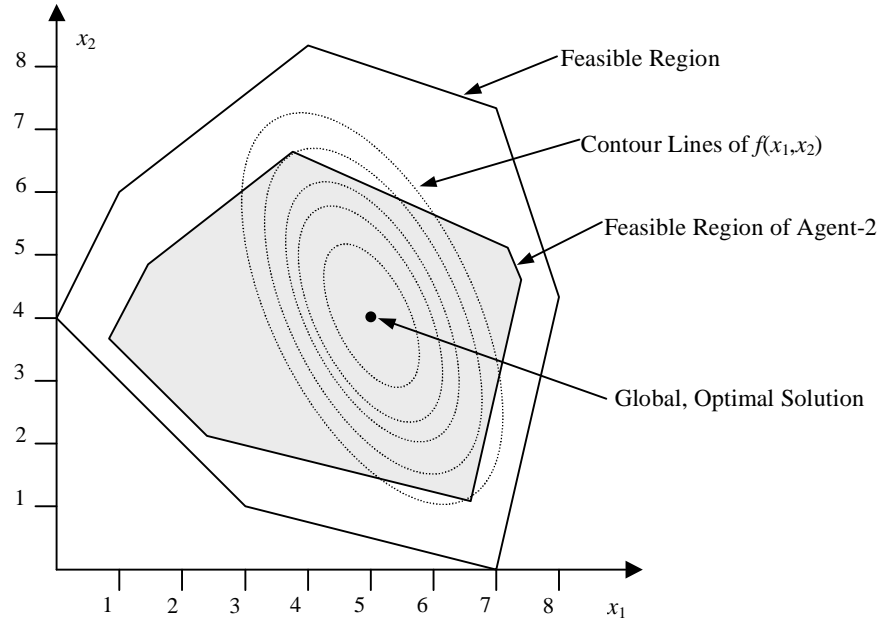


Figure 4.2: The feasible region, and the contour lines of the objective function, of the overall problem. The inner polytope encloses agent-2's feasible space, when its resource margins are in place.

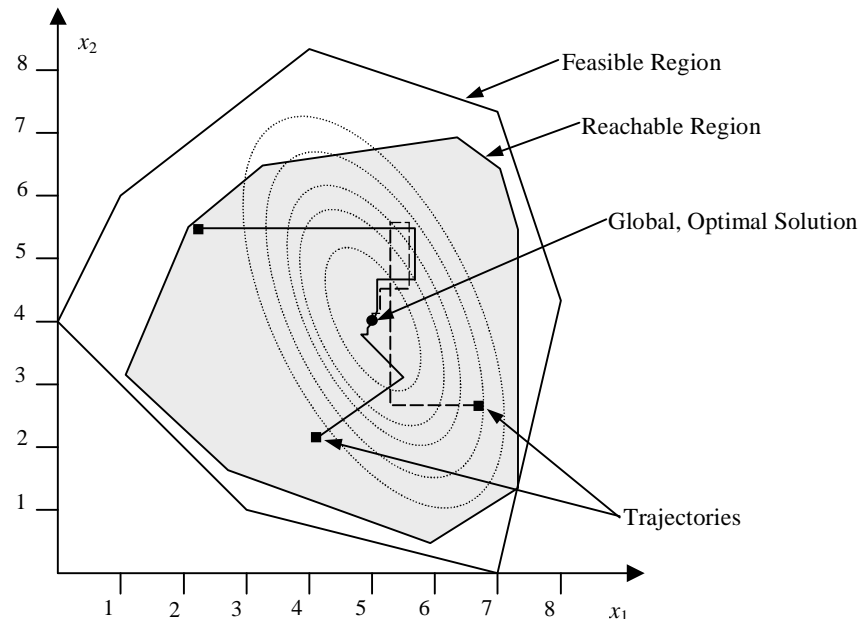


Figure 4.3: The trajectories of three solutions found by the C-Net. The agents run proximate-exchange asynchronously and implement resource margins. Beginning with a feasible solution, they consistently converged to the optimal solution without incurring constraint violations.

4.5.2 Example 2

The problem of focus is expressed in mathematical programming as follows:

$$\begin{aligned} \text{(P) Minimize } & f(x_1, x_2) = -x_2 \\ \text{Subject to:} & \\ & -x_1 + 2x_2 - 1 \leq 0 \\ & x_1^2 + x_2^2 - 1 - \varepsilon \leq 0 \\ & -x_1^2 - x_2^2 + 1 - \varepsilon \leq 0 \\ & -x_1 \leq 0, \text{ where } \varepsilon = 1/10. \end{aligned}$$

Problem (P) corresponds to test-case 217 of [Sch87, page 41], except that two inequalities play the role of the constraint $\{x_1^2 + x_2^2 = 1\}$, as prescribed by the procedure (given in Section 4.4). We followed the procedure and found the resource margins $\alpha_1 = (0.1, 0.9, 0.9)$ and $\alpha_2 = (0.1, 0.9, 0.9)$ to be adequate. In particular, the margins' guarantees of feasibility were verified by simulating the asynchronous work, whereby the agents optimize random, linear objective functions. The specifics of the C-Net in the previous section carry over to this one.

If the agents stick to basic proximate-exchange, then they do not steer toward good solutions (because the overall objective, f , is not directly dependent on agent-1's variable, x_1). The agents need a mutual-help attitude to avoid getting trapped at suboptimal solutions. Along these lines, we have implemented this attitude as follows:

- 1) agent-2 calculates the sensitivity $\partial f_2 / \partial x_1$ of its objective function to changes in x_1 ,
- 2) agent-2 appends $\partial f_2 / \partial x_1$ to the messages it sends out to agent-1, and
- 3) agent-1 adds $\partial f_2 / \partial x_1$ to its objective function, that is, agent-1 minimizes $\{f_1 + (\partial f_2 / \partial x_1) \Delta x_1\}$.

Then, agent-1 becomes sympathetic to the needs of agent-2 and, invariably, promotes good overall performance. Figure 4.4 illustrates the benefits of mutual-help behavior and resource margins. The agents trace a solution trajectory that progresses toward the optimal solution, without incurring constraint violation while working asynchronously.

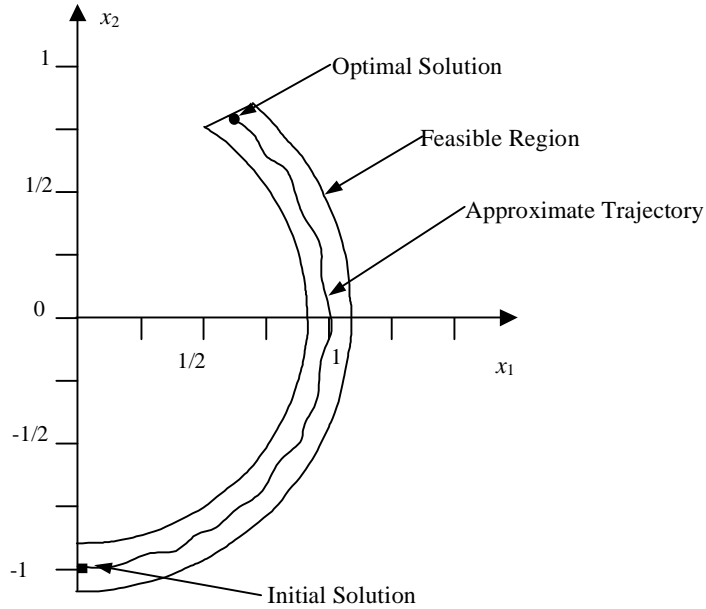


Figure 4.4: The feasible region and the trajectory of the solutions traced by the C-Net. The agents run proximate-exchange asynchronously, help one other explicitly, and use resource margins to retain the feasibility of (P).

4.6 Resource Margins for Speed-Dissimilar Agents

The previous section illustrated the use of resource margins to ensure feasibility (when the agents work asynchronously). Herein, we narrow the attention to their use in promoting collaboration between agents that are dissimilar in speed, such as human and computer-based agents. In essence, margins can promote progress toward good solutions by better allotting the resources.

Consider a scenario where the agents ought to solve $\{(P_m)\}$ and put the solutions into action by a due date—in typical networks, the elements of $\langle(P)\rangle$ need to be solved, and the solutions implemented, before they become outdated. Therein, the sluggish agents seldom solve the subproblems and have no alternative, but to execute the default, previous, or preliminary solutions. The speedy agents, however, often finish their tasks before the due date. That is, speedy agents display high probability of solving the subproblems, whereas the sluggish ones display low probability. Below, we illustrate the role of margins to adequately allot the resources in a simple, hypothetical experiment. The assumptions of the experimental set-up are the following:

- 1) (P) is decomposed into $\{(P_1), (P_2)\}$.
- 2) Agent-1 and agent-2 form the collaborative net, and run asynchronous proximate-exchange.

- 3) The probability that agent- m solves (P_m) ahead of the due date is p_m .
- 4) The agents are bound by the constraint $\{(u_1)^2 + (u_2)^2 \leq R^2\}$ that models a scarce resource.
- 5) Agent- m inputs no control ($u_m = 0$) if it does not finish its computations.
- 6) Agent-1 is very fast ($p_1 \cong 1$), but agent-2 is very slow ($p_2 \ll 1$).

The introduction of resource margins is mandatory to prevent constraint violations and, according to Theorem 4.2, the condition $\{(\alpha_1)^2 + (\alpha_2)^2 \geq R^2\}$ must hold.

What values should we pick for the resource margins? The background knowledge of adequate resource-allocation and the speed of the agents (as modeled by p_m) are fundamental to the answer. Suppose that 1) the resource R should be allocated fairly for the overall good, and 2) the agents greedily chew up the available resources. Then, the intuition says that α_1 should be larger than α_2 because agent-1 is faster than agent-2 (that is, because agent-1 is more likely to consume resources, it must do so in smaller chunks). More technically, the optimal margins, $\{\alpha_1^*, \alpha_2^*\}$, should maximize the minimum resource-usage by either agent, for the allocation to be fair. Notice that the resource-usage is a stochastic quantity, which depends on the speed probabilities and the margins. Near-optimal margins can be found by 1) accounting for the most likely outcomes, and 2) solving a stochastic optimization problem [Had64, §4] that maximizes the minimum, expected resource-usage.

For the above set-up, we have computed the margins that maximize the minimum resource-usage. Let:

- 1) $\alpha_1^*(p_2)$ and $\alpha_2^*(p_2)$ be the values of the optimal margins as functions of agent-2's speed,
- 2) $R(p_2)$ be the minimum, expected resource-usage when the margins are $\alpha_1 = \alpha_2 = (\sqrt{2}/2)R$, and
- 3) $R^*(p_2)$ be the minimum, expected resource-usage if the agents implement $\alpha_1^*(p_2)$ and $\alpha_2^*(p_2)$.

Figure 4.5 shows that the resource-allocation is more balanced when the agents use $\alpha_1^*(p_2)$ and $\alpha_2^*(p_2)$, translating into higher collective performance. Figure 4.6 depicts the suggested resource margins, and shows that $\alpha_1^*(p_2)$ increases, and $\alpha_2^*(p_2)$ decreases, as agent-2's speed drops. This means that agent-2 compensates for its low speed with the allocation of large chunks of resources.

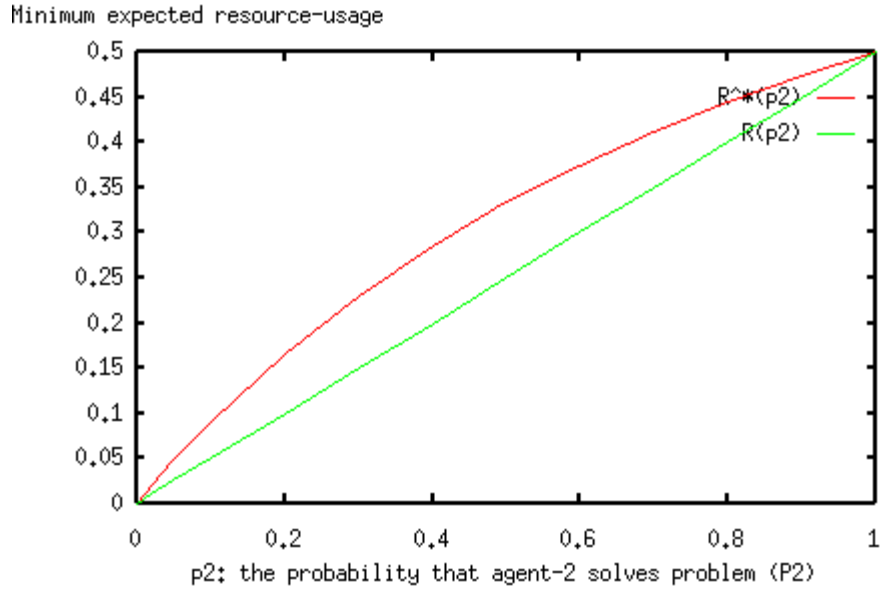


Figure 4.5: The minimum, expected usage of resource R (by either agent-1 or agent-2) as a function of p_2 (the probability that agent-2 finishes its computations ahead of the due date). The minimum, expected resource-usage under the suggested margins, $R^*(p_2)$, is always higher than that under identical margins, $R(p_2)$. That is, the collective performance is higher when the agents use the suggested margins.

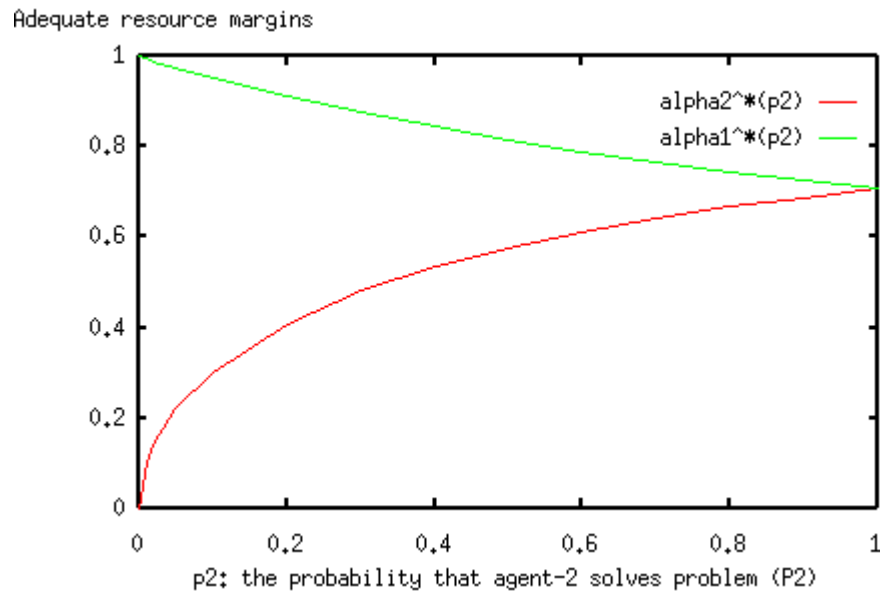


Figure 4.6: The suggested resource margins, $\alpha_1^*(p_2)$ and $\alpha_2^*(p_2)$, as functions of agent-2's speed (the sluggish agent). The resource margin of agent-2 decreases, and agent-1's increases, as the speed of agent-2 drops. Therefore, agent-2 compensates for its low speed with the allocation of large chunks of the resource.

4.7 Summary

The material brought forth two impediments to asynchronous work: 1) the potential of violating constraints, and 2) the limited, analytical guarantees for convergence. In view of the impracticality of serial work, ways around these impediments are badly needed. To this end, we have proposed the use of resource margins to mitigate constraint violations, and speculated that mutual-help behavior promotes convergence to good solutions.

The study begins with a formalism for implementing resource margins, and goes on to develop an optimization-based procedure to test whether or not the margins, $\{\alpha_m\}$, one for each (P_m) , guarantee the feasibility of (P) when the agents solve $\{(P_m)\}$ asynchronously. The test is, however, very conservative and seems applicable only to convex constraint sets. For a narrower class of constraints, namely that defining ellipsoids without rotation, sufficient and necessary conditions were found for the asynchronous work to retain a feasible solution.

We have condensed relevant issues into steps of a procedure for computing resource margins. The steps comprise 1) the approximation of equality constraints with inequalities, 2) the introduction of bounds for the variables, 3) the iteration over the sufficiency test (for convex constraints) or the simulation of the asynchronous work (for general constraints), and 4) the addition of mutual-help behavior to steer the agents toward good solutions. The procedure, however, does not provide the allocation of the resources, but only a way to go about allocating them. This chore is left to the designer, who can draw insights from the problem at hand and the background knowledge. The intuition behind the resource margins is that increasing them repels the agents from the constraint boundaries and, thereby, reduces constraint violations at the price of a smaller solution space.

The final part illustrates the roles of resource margins in simple, synthetic problems. In the first two problems (one convex and the other nonconvex), it illustrates the use of resource margins and mutual-help reflexes in solving the subproblems asynchronously. In the last problem, a scenario is synthesized to explain how one can use background knowledge to allocate the resources, specifically in an environment where the agents are speed-dissimilar.

Chapter 5

Experiments in Forests of Pendulums

So far, we have dwelled on the framework for solving dynamic control problems (DCPs), collaboration protocols, and resource margins. Much of the attention has been centered on the proximate-exchange protocol. The interest is not incidental, but grows out of its simplicity, and the opportunity to shape it up in a number of ways. Further, the existence of sufficient conditions for convergence is a pivotal feature of proximate-exchange, which has brought about reflections on the limitations of this work and the issues that need further research. Recalling from the preceding chapters, the conditions insist on:

- a) a complete cover of the physical network by the collaborative net,
- b) the convexity of problem (P),
- c) the feasibility of (P),
- d) the differentiability of (P),
- e) an exact match between agents and subproblems,
- f) the use of interior-point methods, and
- g) serial work within neighborhoods.

This chapter provides evidence that the conditions are not necessary. That is, it shows experimentally that the conditions on convexity, exact match, interior-point method, and serial work may be relaxed. In small, but prototypical networks, the agents pulled toward stationary solutions to $\{(P_m)\}$ even though they did not conform to the conditions. The agents run proximate-exchange, and impose margins on the resource constraints to prevent their overuse. Except when the resources are very scarce, the agents consistently arrive at solutions of high quality. To get around the scarcity and make up for the varying context, we have implanted mutual-help reflexes (in the agents) and observed substantial improvement in the solution quality. Further, we have contrasted the performances of traditional control techniques with those of collaborative nets, and noticed that they are comparable in terms of solution quality.

The material hereafter is organized as follows. Section 5.1 suggests forests of pendulums, arrays of pendulums connected by springs, as prototypical networks. It presents the structure, the dynamics, and

the dynamic control problems thereof. Section 5.2 develops the experimental set-up (for investigating the necessity of the above conditions) and reports the results. Section 5.3 provides the specifics of the resource margins and the mutual-help behavior, and then gives an account of the improvements obtained with mutual-help. Section 5.4 details the implementation of feedback-linearization controllers (traditional control techniques) and contrasts their performances with those of collaborative nets. The results show that C-Nets attain performances comparable, and sometimes superior, to those attained by the feedback-linearization controllers.

5.1 Forests of Pendulums as Prototypical Networks

Structurally, forests of pendulums are arrangements of frictionless pendulums, linear springs, and control devices that exert forces onto the pendulums. The pendulums are disposed in a uniform grid and joined by springs. Figure 5.1 shows a typical forest of nine pendulums distributed over a 3x3 grid.

The pendulums swing uniformly when they operate in the synchronous or normal mode, that is, the distance between their lower-ends match that between their upper-ends. In the event of a disturbance—an arbitrary and long-range displacement of pendulums, the network becomes crippled and its operation switches to the emergency mode. The DCPs arise out of these incidents, and their mission is to drive the pendulums back to the synchronous mode cheaply and quickly, while not infringing on the constraints. Figure 5.2 illustrates an incident that jostles the pendulums, forcing them to oscillate helplessly and lose synchronization.

Why are forests of pendulums prototypical networks? They can be viewed as mechanical analogies of electric-power grids, which are representative instances of large, widespread networks. The pendulums and the springs act much like the way power generators and transmission lines do. For one thing, the pendulums operate in the synchronous mode when they swing uniformly, like power generators that rotate at the nominal frequency.

For another, the stiffness of the springs captures the direct coupling between pendulums, in parallel with the impedance of transmission lines captures the direct linkage between generators. (Of course, the analogy could be made more accurate. For instance, the pendulums could be dissimilar in weight and length, and also subject to friction. These dissimilarities would model the many types of generators and require control effort to not only recover synchronization, but also to sustain it.)

Other representative features of forests of pendulums are:

- 1) the similarities between the pendulum's motion dynamics and the power-generator's,
- 2) the highly nonlinear nature of the dynamics that stands as a challenge to the control design,
- 3) the ease of computation and short simulation time of the dynamics (relative to those of electric-power networks), and
- 4) the ease of putting pendulums together in a forest, and systematically formulating the dynamic control problems thereof.

The above analogy and features are attractive, making forests of pendulums representative test-beds for collaboration protocols. The remainder of the section details the dynamic equations and the DCPs that arise in forests of pendulums. Thereafter, collaborative nets are put into action and their performances are measured.

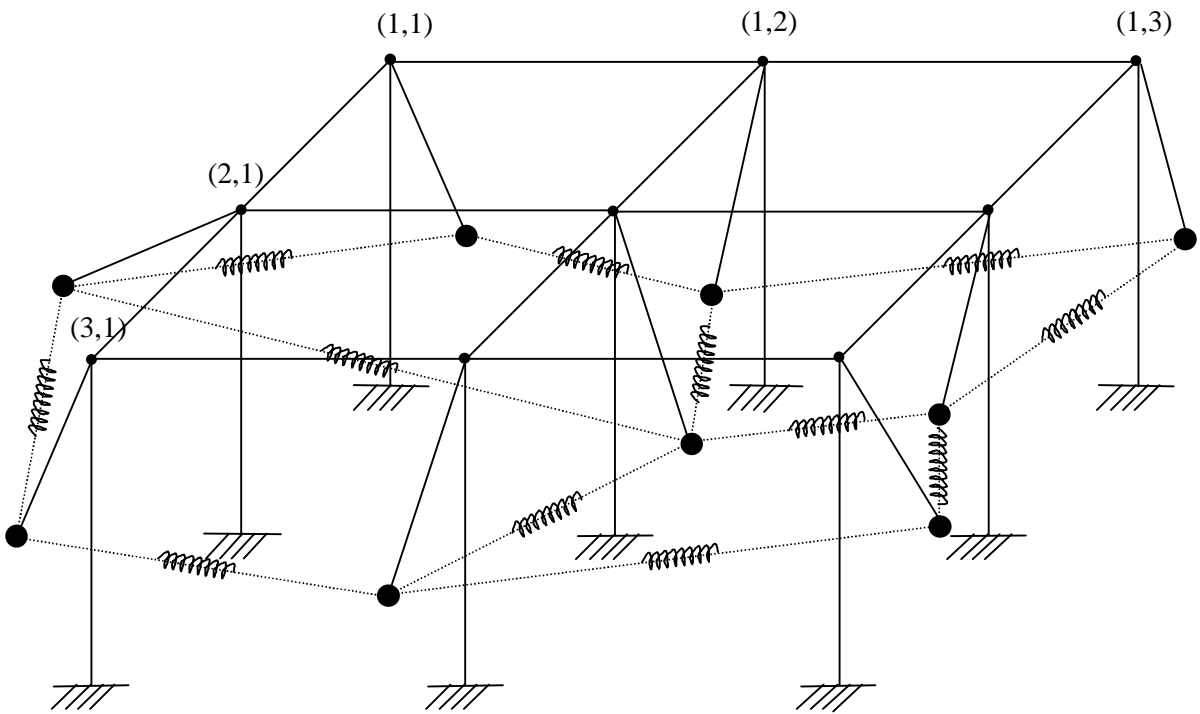


Figure 5.1: A typical forest of pendulums. Nine pendulums are distributed over a 3x3 uniform grid and coupled by springs.

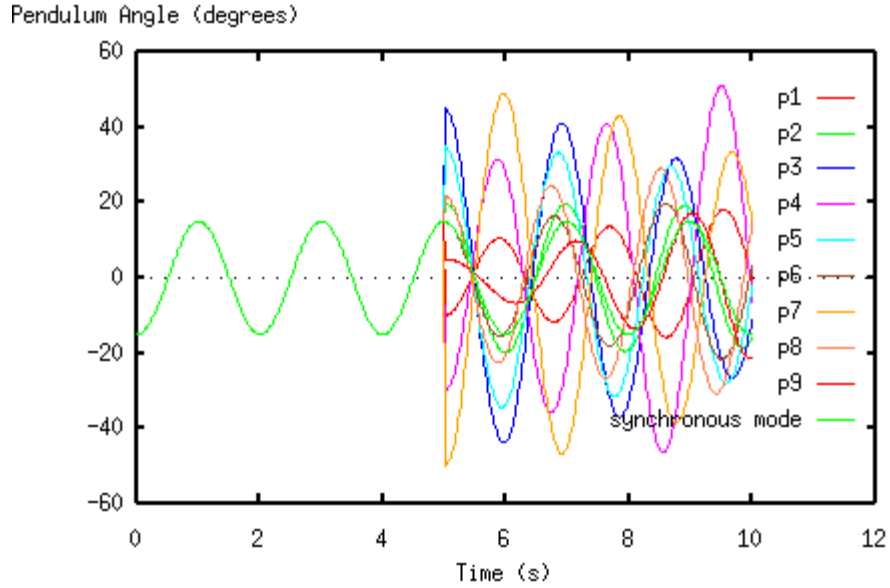


Figure 5.2: The illustration of an incident in a forest of pendulums. A disturbance at 5s throws the pendulums off balance, forcing them to oscillate helplessly when the agents are dormant. The plots trace the angles of the rods over time.

5.1.1 The Dynamic Equations of Pendulum Motion

We have invoked Newton's second law to derive the dynamic equations of a canonical pendulum. The derivation is systematic and much simpler than that with Lagrange's equations [HH59, §9, pp. 293-354]. With ease, we can bring canonical equations together and assemble the motion dynamics of any configuration of pendulums.

Spherical coordinates have been chosen as the generalized coordinates of pendulums, and depicted in Figure 5.3. Then, the state of the pendulums becomes a vector with the following variables:

- 1) the swing angle, ϕ , the angle between the rod and the z -axis,
- 2) the rotational angle, θ , the angle between the x -axis and the projection of the rod onto the xy -plane,
- 3) the swing velocity, $d\phi/dt$, and
- 4) the rotational velocity, $d\theta/dt$.

In essence, we have obtained the dynamic equations by a) projecting the acting forces onto the force-decomposing directions, $\{n_\phi, n_\theta, n_o\}$, b) calculating the resulting torques, and c) plugging them in Newton's second law. The dynamic equations for pendulum i can be expressed as follows:

$$J \frac{d^2 \phi}{dt^2} = -mgl \sin(\phi) - \sum_{j \in N(i)} l \langle f(i,j), n_\phi \rangle + l \cos(\phi) u_\phi$$

$$J \frac{d^2 \theta}{dt^2} = - \sum_{j \in N(i)} l \langle f(i,j), n_\theta \rangle - l u_\theta.$$

Where:

- $N(i)$ is the set of indices corresponding to the neighbors of pendulum i , that is, those connected to pendulum i by a spring,
- $f(i,j)$ is the force exerted along the spring that links pendulums i and j ,
- $\langle x,y \rangle$ is the dot product of vectors x and y ,
- m is the mass, l is the length, and J is the inertia (ml^2) of the pendulum,
- g is the gravitational acceleration, and
- u_ϕ and u_θ are the control inputs, that is, the external forces that act on the pendulums.

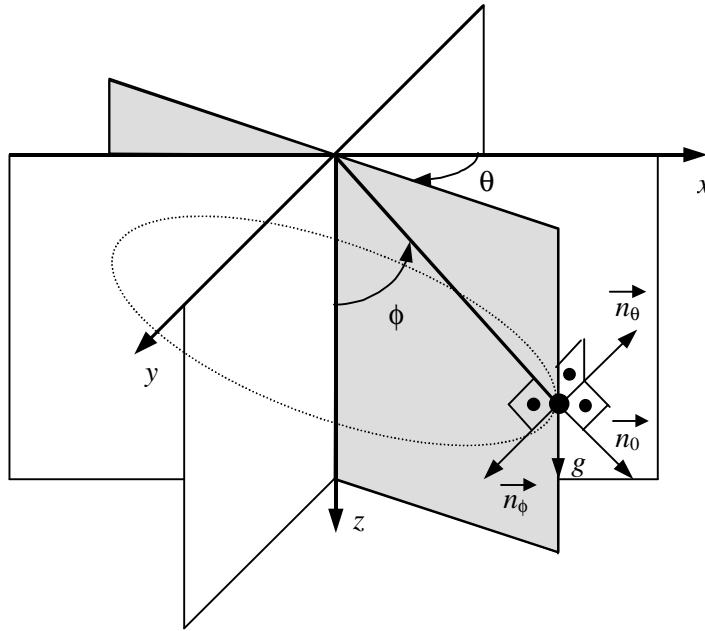


Figure 5.3: The generalized coordinates and force-decomposing directions. The coordinates are spherical, and given by the swing (ϕ) and rotational angle (θ). These angles and their first derivatives constitute the state of the pendulums. The dynamic equations are obtained by a) projecting the acting forces onto the decomposing directions $\{n_\phi, n_\theta, n_0\}$, b) evaluating the resulting torques, and c) invoking Newton's second law.

5.1.2 The Dynamic Control Problem

The DCP calls for the quick and low-cost recovery of the pre-disturbance, synchronous mode of the pendulums. These goals are conflicting, that is, decreasing synchronization time invariably incurs extra cost. To cope with these conflicts, we combined the goals into the single-objective of minimizing a) the cumulative distance from the synchronous trajectory, and b) the cumulative control-input cost. More formally, the problem is stated as follows:

$$\begin{aligned} & \text{Minimize } \int_{t=0}^{t=\infty} \|x(t) - x'(t)\|^2 dt + b \int_{t=0}^{t=\infty} \|u(t)\|^2 dt \\ & \text{s.t.} \\ & \quad H(x(t), dx(t)/dt, u(t)) = 0 \\ & \quad G(u(t)) \leq 0. \end{aligned}$$

Where:

- a) $x(t)$ is the state vector of the pendulums at time t ,
- b) $x'(t)$ is the state vector of synchronous pendulums (it tracks the trajectory of free pendulums),
- c) $u(t)$ is the vector of the external forces acting on the pendulums,
- d) b is the rate at which the control-input cost gets converted into error units (the trade-off rate between the goals),
- e) H is the set of dynamic equations,
- f) G is the set of inequality constraints, and
- g) $\| \circ \|$ stands for the 2-norm.

The inequality constraints limit the concurrent usage of a scarce resource, such as energy. They are quadratic functions that bound the simultaneous control-input to pairs of pendulums. For instance, $\{\|u_m(t)\|^2 + \|u_n(t)\|^2 \leq R^2\}$ limits the forces exerted concurrently on pendulums m and n .

The constraints serve as a mechanism to illustrate the need of resource margins and the potential benefit of mutual-help behavior, especially when the resources are scarce. They add difficulties to the DCP, but do not lead us off track—which is the experimental assessment of the non-necessity of the conditions for convergence. In the forthcoming experiments, we use Proposition 4.1 to come up with resource margins (that rule out constraint violations) and sensitivity analysis to embed mutual-help reflexes (in the agents).

5.2 Relaxing the Sufficient Conditions: Experiments

5.2.1 Overview

The experimental scenarios consist of forests ranging from two to nine pendulums, which are disposed along a one-dimensional grid and fully connected by springs. The stiffness of the springs joining nearby pendulums is more pronounced than that joining far-off ones; they capture the high coupling between geographically adjacent components, relative to those between distant ones. (Specifically, the stiffness ranges from a maximum of one, to a minimum of one tenth of, unit). Further, the amount of resources varies from one scenario to another, being drawn from the set $\{\sqrt{5}, \sqrt{25}, \infty\}$, to bring about insightful behavior.

In each of the forests and for each amount of resources, we initiate an incident, instantiate the DCP, and then carry out the solution of the series of static optimization problems, $\langle(P)\rangle$, by a collaborative net and a centralized controller. For each element of $\langle(P)\rangle$, the following steps are executed:

- 1) Decompose (P) into $\{(P_m)\}$.
- 2) Engage the C-Net in solving $\{(P_m)\}$, and record the solution (X_{CN}, U_{CN}) that it arrives at. The agents run the proximate-exchange protocol asynchronously to solve $\{(P_m)\}$.
- 3) Engage the centralized controller, C_1 , in solving (P) and record the solution (X_{C_1}, U_{C_1}) that it yields. C_1 is an omniscient agent (that senses the entire state of, and has full authority over, the forest). It starts with the same, initial solution that was given to the C-Net, and uses a nonlinear-optimization algorithm to solve (P) [LZT94].
- 4) Implement the control plan U_{C_1} .

5.2.2 Specifics

In what follows, we explain how the sufficient conditions have been relaxed in the experiments. The material gives an account of the relaxation, the static optimization problem, (P) , and the way the C-Net solves (P) .

The Condition on Convexity

The nonlinear, nonconvex nature of (P) arises from the cosine terms appearing in the dynamic equations (that simulate pendulum motion).

The Condition on Exact Match

The number of agents (and subproblems) is precisely the number of pendulums. Consider pendulum- m and its intrinsic subproblem (P_m). Only agent- m can sense the state of, and exert forces on, pendulum- m . Further, two agents can collaborate only if they are physically adjacent. Since every pair of pendulums is joined by a spring, all subproblems are coupled and, therefore, the match is imperfect for forests of size three and higher (that is, each (P_m) is sensitive to variables that neither agent- m nor its neighbors can sense or set). Figure 5.4 illustrates an inexact match in a forest of four pendulums.

The Condition on Interior-Point Method

The agents do not recast $\{(P_m)\}$ as barrier subproblems, nor do they use interior-point algorithms. Instead, they manipulate the constraints explicitly and run a sequential-quadratic-programming (SQP) algorithm [NW99, pp. 529-575]. As the agents proceed to solve $\{(P_m)\}$, they can arrive at infeasible and boundary solutions.

(SQP is among the most effective methods for nonlinearly constrained optimization. The approach yields a sequence of candidate solutions, by solving a quadratic approximation at each iteration k . More specifically, SQP a) linearizes the constraints, and approximates the objective function with a quadratic expression about the current iterate, x_k , b) finds a solution to the quadratic-approximation problem, y_k , and c) executes a line search along the direction defined by x_k and y_k to compute the next iterate, x_{k+1} .)

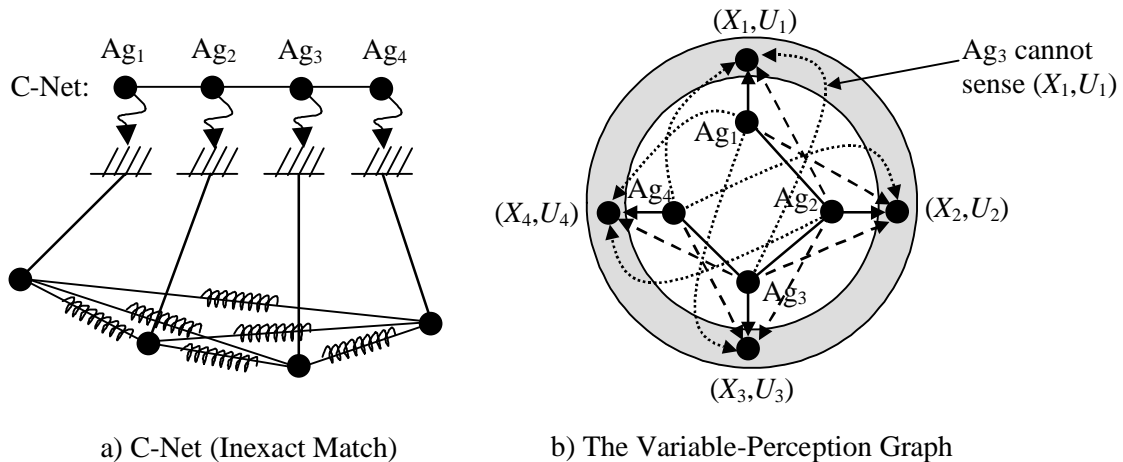


Figure 5.4: An inexact match of agents to subproblems in a forest of four pendulums. Fig. (a) shows the arrangement of pendulums in a uniform grid and the springs that link all pairs of pendulums. Fig. (b) depicts the variable perception graph: 1) the *solid, undirected edges* represent communication links; 2) the *solid, directed edges* point to the proximate variables; 3) the *dashed, directed edges* point to the neighborhood variables; and 4) the *dotted, directed edges* point to the remote variables (that are relevant for the C-Net to induce an exact match). Each subproblem (P_m) is dependent on variables that neither agent- m nor its neighbors can measure or set.

The Condition on Serial Work

The agents do not serialize their efforts, but rather work asynchronously. To better understand the asynchronous solution of $\{(P_m)\}$, consider an agent- m and let:

- Y_m^* be the latest values of the proximate variables (the ones stored in agent- m 's memory, which are likely to differ from the actual values, Y_m , before convergence is attained),
- Z_m^\perp be the values of the remote variables (which are predicted as if the remote agents were operating in the synchronous mode), and
- S_m be agent- m 's reaction function, that is, a mapping from (Y_m, Z_m) to agent- m 's proximate variables, (X_m, U_m) , which corresponds to the solution to (P_m) with the given (Y_m, Z_m) .

Then, agent- m continually revises its decisions, $(X_m, U_m) = S_m(Y_m^*, Z_m^\perp)$, immediately after the arrival of new estimates for agent- m 's neighborhood variables, Y_m^* . Once the optimization procedure is finished, agent- m sends out messages to its neighbors carrying the up-to-date values of its proximate variables, (X_m, U_m) .

We simulated the asynchronous work on a single-processor, Unix-based workstation. In doing so, each agent was equipped with a queue of on-coming messages (that mimics the communication delays that are typical of actual networks). Further, the agents were picked at random, with uniform probability, to solve their subproblems and, therefore, they ran at about the same speed.

The Resource Constraints

The resource constraints are active for each pair of neighboring agents, and over the time horizon. More formally, the constraint set is $\{\|u_m(t_n)\|^2 + \|u_{m+1}(t_n)\|^2 \leq R^2 \mid \text{for } n = 0, \dots, N \text{ and } m = 1, \dots, M-1\}$. Agent- m and its succeeding neighbor agent- $(m+1)$ impose margins, $\{\alpha_m, \alpha_{m+1}\}$, on the constraints to prevent the depletion of the resources. These margins split the resources evenly and, according to Theorem 4.2, must both take on the value $(\sqrt{2}/2)R$.

5.2.3 Results

The experiments consist of incidents in forests whose sizes range from two to nine pendulums, and whose amounts of resources are drawn from the set $\{\sqrt{5}, \sqrt{25}, \infty\}$. In almost all incidents (arbitrary perturbations of the pendulums) and elements of $\langle(P)\rangle$ therein, the C-Nets converged to a solution to $\{(P_m)\}$. (The experiments also showed that the failures to convergence occur when the resources are meager). By convergence, we mean that the values of the neighborhood variables of each agent- m , Y_m^* , eventually become identical to their actual values, Y_m , at time t^* , and henceforth agent- m does not

change its proximate variables, that is, $(X_m(t), U_m(t)) = (X_m(t^*), U_m(t^*))$, $Y_m = Y_m^*$, and $(X_m(t), U_m(t)) = S_m(Y_m^*, Z_m^\perp)$ for all $t \geq t^*$. This state is equivalent to a Nash equilibrium point [BO82] (one at which each agent- m expresses no desire at all to change the solution to its subproblem, (P_m) , as long as the other agents stick to their solutions).

Notice that, when the agents arrive at a solution to $\{(P_m)\}$, they indeed arrive at a solution to (P) . Putting together the individual solutions to $\{(P_m)\}$ results in a feasible solution to the equality constraints of (P) . Further, the inequality constraints are guaranteed, by the resource margins, to be feasible.

As the incident unfolds, the collaborative net and the centralized controller, C_1 , solve hundreds of problems, $\langle(P)\rangle$. The evaluation of the solution found by the C-Net, for each element of $\langle(P)\rangle$, was always greater than that found by C_1 . This excess is called C-Net penalty, and defined as: $[f(X_{CN}, U_{CN}) - f(X_{C1}, U_{C1})] / f(X_{C1}, U_{C1})$, where $f(X, U)$ is the objective function in the rolling horizon formulation. The mean value of the C-Net penalty is used to contrast the performances of the collaborative nets with those of controller C_1 . Figure 5.5 shows the mean value of the C-Net penalty (%) for three amounts of resource, R , as a function of the number of pendulums.

The plots show that the C-Net penalty is low when the resources are plentiful, or moderate, but quite high when they are scarce. The low performance can be attributed to inflexible margins: they split the resources evenly among the agents, even though some agents agonize with highly disturbed pendulums, while others enjoy the ease of synchronizing slightly disturbed ones. The agents could, however, adjust the margins dynamically to make up for the varying context and, invariably, improve performance. This potential remedy is the subject of the next section.

Intuitively, one would expect the C-Net penalty to increase as the forest becomes larger. Why has this tendency not been pronounced? We believe that it has not been pronounced in part because the incidents are random, and in part because the relative quality of the centralized solution drops as the number of pendulum increases, that is, the hardness of (P) increases at a faster rate than the availability of computational resources).

5.2.4 Computer-Implementation Details

This subsection briefly discusses the computer implementation of the experiments and the calculation of the C-Net penalty. We have coded up the software in C, C++ and Fortran programming languages for Unix-based workstations. The programming effort was greatly reduced by the extensive use of the following libraries:

- 1) the CFSQP package [LZT94], which offers a robust implementation of sequential-quadratic-programming,
- 2) the LEDA class library [MN99] that implements several data structures, and
- 3) the LAPACK library [ABB+99] that provides routines for several numerical problems.

To further simplify the implementation, we simulated the asynchronous work (and communication) of the agents in one workstation (rather than distributing the agents over a computer network). In regards to the results, the plots of C-Net penalty are free of outliers [SM95, §1.4.4, pp. 29-30]. The spurious samples that resulted from numerical instability, and implementation glitches, have been removed from the datasets.

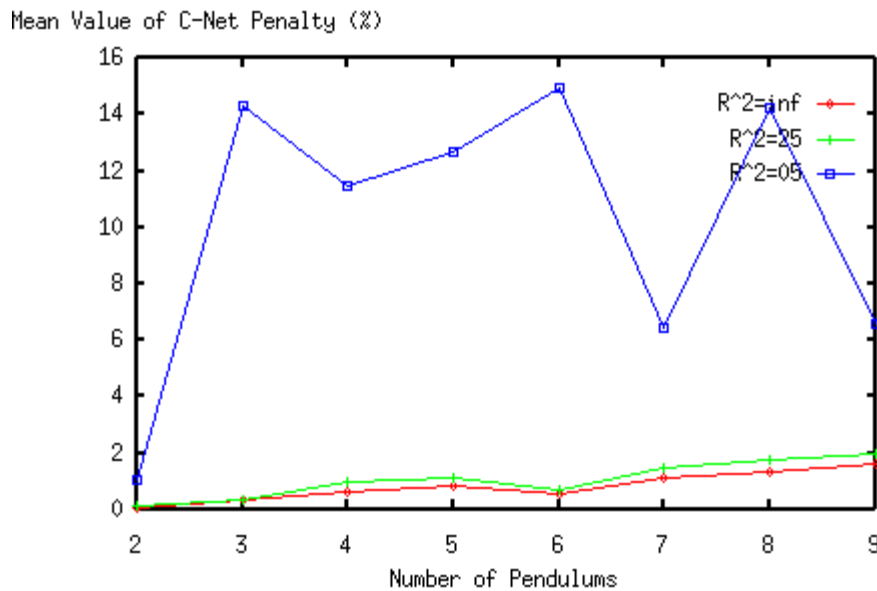


Figure 5.5: The mean value of the C-Net penalty versus the number of pendulums for three amounts of resource R . The agents work asynchronously, impose margins on the constraints that divide the resources evenly, exchange information with agents in the vicinity, and do not induce an exact match. The plots show that the C-Net penalty is low when the resources are abundant, or moderate, but high when they are scarce.

5.3 Improving Performance with Mutual-Help

The preceding experiments exposed a side effect of static resource allocation: the infliction of penalties in performance, especially when the context changes drastically from one incident to another. A remedy is the implant of mutual-help attitude to encourage the agents to dynamically tweak their resource margins. With this attitude, the least disturbed agents (those facing easier problems or needing little resources) increase their margins and provide more latitude to the severely disturbed ones (those facing difficult problems).

Dreadful incidents, such as the ones above, can lead some agents to “panic” and leave others unaffected in large networks. In power networks, for instance, faults can tug nearby devices to operate under the emergency mode; meanwhile, other devices experience harmless disturbances. The unaffected devices could join the troubled ones to damp down the disturbance and, hopefully, improve performance.

5.3.1 Implementation Details

Agent- m can exhibit mutual-help by:

- 1) estimating and passing around its contributions to the overall objective with respect to changes in α_m ,
- 2) appraising its contributions against that of its neighbors, and
- 3) proactively increasing α_m and coordinating the decrease of its neighbors' margins accordingly.

The specifics of this attitude are reported below.

Estimating the Overall Contributions

Agent- m calculates $\partial f_m(X_m, U_m)/\partial \alpha_m$ about the current solution, (X_m, U_m) , to estimate its contributions to the overall objective with respect to changes in its resource margins. The agent derives the estimates by applying the chain rule to the active constraints (those without slack). (Notice $\partial f_m/\partial \alpha_m \geq 0$ and $\partial f_m/\partial \alpha_{m,r} = 0$ when a resource r is not exhausted.)

Assessing the Contributions

Consider agent- m , resource r , its contributions, and those of its neighbors. Agent- m chooses to increase its margin on r if $\partial f_m/\partial \alpha_{m,r}$ is smaller than $\{\sum_{n \in N(m)} [\partial f_n/\partial \alpha_{n,r}]\}$, where $N(m)$ is the index set of agent- m 's neighbors. That is, agent- m increases its margin when the total contribution of its neighbors outweighs its own contribution.

Coordinating Changes in the Resource Margins

Suppose that agent- m is willing to increase its margins on resource r . First, it increases $\alpha_{m,r}$ by an amount Δ_r and resolves (P_m) . Second, it coordinates the changes on its neighbors' margins so that the invariant of Proposition 4.1 holds.

The coordination can be realized by two asynchronous mechanisms that do not pose any bottleneck to the parallel solution of $\{(P_m)\}$. The first one sequences the coupled agents and let them pass around a token to change the resource margins. The second mechanism implements a distributed synchronization algorithm to sustain the invariant. A recent and comprehensive account of these algorithms is [Lyn96]. A shorter and easy to follow reference is [Ray88, §1 and §5].

5.3.2 Experimental Results

The hypothesis that mutual-help reflexes significantly improve performance is verified in Figure 5.6. The plots show a substantial drop in the mean value of the C-Net penalty (%), which is on the order of 10%. The results demonstrate the potential benefit of dynamically tuning resource margins to better suit the incident.

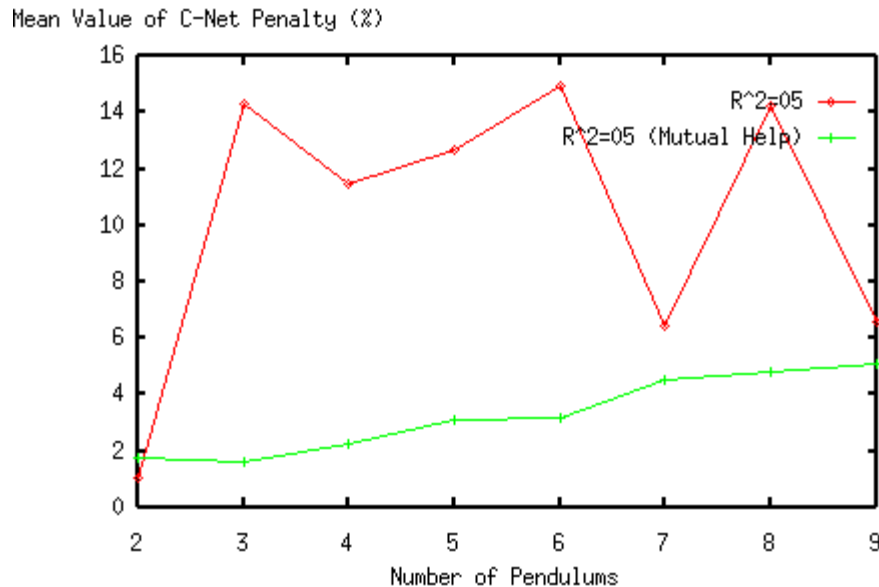


Figure 5.6: The mean value of the C-Net penalty for an amount $R = \sqrt{5}$ of resource units versus the number of pendulums. The agents display mutual-help behavior: the least troubled agents (those facing easier problems or needing less resources) tighten their margins and allow the most troubled ones (those facing hard problems) to allocate more resources.

5.4 Contrasting C-Nets with Traditional Control

The arsenal of techniques to solve dynamic control problems is quite large. Typically, the techniques are limited in scope but, altogether, they handle several classes of problems. By dissecting the attributes of these classes and categorizing them, one can put together a taxonomy. Control engineers often traverse a taxonomy, perhaps implicitly in their thoughts, to identify the problem class and solution alternatives.

Which traditional control technique would cope with the DCPs encountered in forests of pendulums? How well would they perform? The remainder of this chapter attempts to answer these questions. The first step consists in isolating the properties of the problem at hand. These properties are:

- 1) the dynamics are nonlinear and time-invariant,
- 2) the network has multiple inputs and outputs (MIMO), and
- 3) the task is trajectory tracking.

The problem, therefore, belongs to the class of nonlinear control problems. Traditional techniques often, if not always, neglect constraints in the design stage and handle them in an ad-hoc way at the implementation stage [GPM89]. Along these lines, we have eliminated the resource constraints in our networks to facilitate the design of feedback-linearization controllers. The material below introduces feedback linearization, provides the specifics of its implementation, synthesizes feedback-linearization controllers, and contrasts their performances with those attained by collaborative nets.

5.4.1 Feedback Linearization

Feedback linearization is a powerful technique for controlling nonlinear dynamics. It comprises the algebraic transformation of the nonlinear dynamics into linear ones, by a change of the variables, and the subsequent use of linear control techniques [SL91, §6]. The approach is unlike conventional or Jacobian linearization that approximates the dynamics, about the operating point, with linear functions.

The approach is best understood with the following example. Consider the dynamic system $\{dx/dt = a(x) + b(x)u\}$, where $x \in R$ is the state variable, $u \in R$ is the control variable, and $a: R \rightarrow R$ and $b: R \rightarrow R$ are arbitrary nonlinear functions of x . Provided that $b(x)^{-1}$ exists, the feedback-control law $\{u(x,v) = [v - a(x)]/b(x)\}$ transforms the nonlinear dynamics into the linear dynamics $\{dx/dt = v\}$. The approach leads to the trivial design of a linear-feedback law in the transformed dynamics, $v(x) = -kx$, and also yields the control-input to the nonlinear dynamics by just plugging $v(x)$ into $u(x,v)$. Thus, the nonlinear problem has been reduced to a linear control problem.

In principle, one can find an algebraic feedback that linearizes any nonlinear, affine dynamics. That is, a system of the form: $dx/dt = a(x) + \sum_{i=1}^m b_i(x)u_i$, where $x \in R^n$ is the state vector, $u \in R^m$ is the control vector, and $a: R^n \rightarrow R^n$ and $b_i: R^n \rightarrow R^n$ are field vectors. In systems of this form, the task becomes to find state and feedback transformations that linearize the dynamics. More precisely, are there $z = \gamma(x)$ and $u = \alpha(x) + \beta(x)v$ that transform the dynamics into $dz/dt = Az + Bv$? This question constitutes the so-called feedback-linearization problem. Necessary and sufficient conditions for their existence, and algorithms for computing them, can be found in [NvdS90, pp. 176-210].

The dynamics of an array of pendulums happen to be in the nonlinear, affine form, and can be readily linearized. The steps for the design of a feedback-linearization controller are:

- 1) linearize the dynamics with the state feedback $u = \alpha(x) + \beta(x)v$,
- 2) reduce the tracking problem to a stabilization problem [SL91, page 196] by changing the variables of the linear dynamics,
- 3) synthesize a feedback law, $v(x) = -Kx$, with the linear-quadratic-regulation technique [Ber95b, pp. 129-143], to solve the linear problem that arises in step-2, and
- 4) combine the linear feedback law, $v(x) = -Kx$, with the algebraic state feedback, $u = \alpha(x) + \beta(x)v$, to assemble the feedback-linearization controller.

5.4.2 Experimental Results

The experiments consist of an incident in the forest of Figure 5.1 and control-input costs from the set $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. For each of the four experiments, we designed a feedback-linearization controller (FLC) and let it drive the pendulums back to the synchronous mode. (The syntheses of the linear-feedback matrices were performed with MATLAB-5 [Pra98].) A collaborative net also had the opportunity to restore pendulum synchronization and, as in the preceding experiments, its agents 1) work asynchronously, 2) induce an inexact match, and 3) exchange information locally. The control-input cost and the distance from the synchronous trajectory were accumulated over the simulation of each incident to evaluate the performances of the C-Net and FLC: $f(\text{C-Net})$ and $f(\text{FLC})$.

Figure 5.7 plots the ratio $f(\text{C-Net})/f(\text{FLC})$ against the control-input cost (in log-scale). The results show that the C-Nets outperform the FLCs when energy becomes costly. Why have we observed this trend? The influence of the control-input (torque) is proportional to the cosine of ϕ , that is, the angle between the pendulum's rod and its vertical axis (see Figure 5.3). Thus, the higher the angle, the smaller the influence on the pendulum motion. However, this nonlinear trend vanishes when the dynamics are

linearized and are not accounted for in the design of the linear feedback law. Figures 5.8 and 5.9 illustrate this point for the highest control-input cost ($b = 10^{-1}$). While the FLC exerts intense force at the beginning (rapidly driving the pendulums near to the synchronous trajectory), the C-Net holds off its action until the angles are small enough for the inputs to be cost-effective.

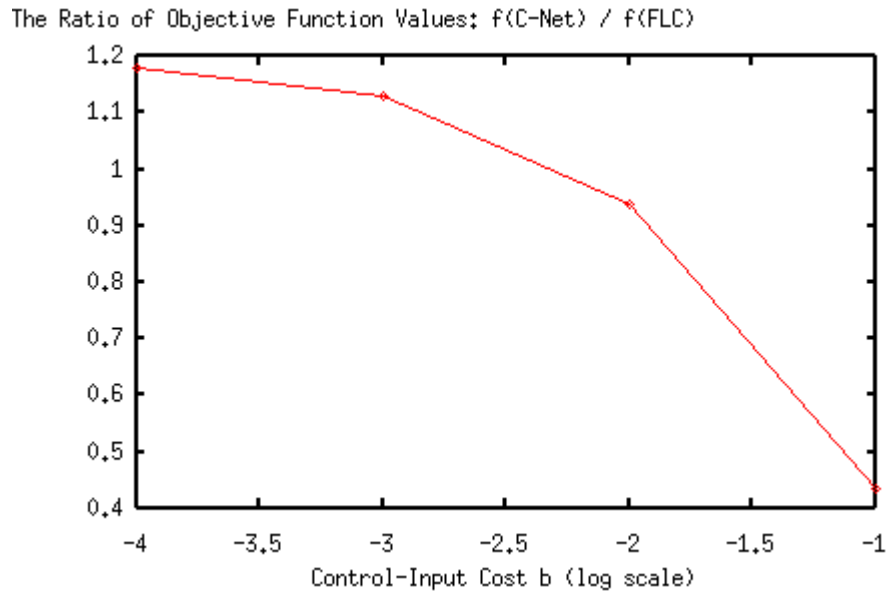


Figure 5.7: The performance contrast between collaborative nets and feedback-linearization controllers. The plot shows the ratio of the performance of the C-Net to that of the FLC, $f(\text{C-Net})/f(\text{FLC})$, as a function of the control-input cost, b , which is given in log-scale. The plot shows that C-Nets outperform FLCs when energy is costly.

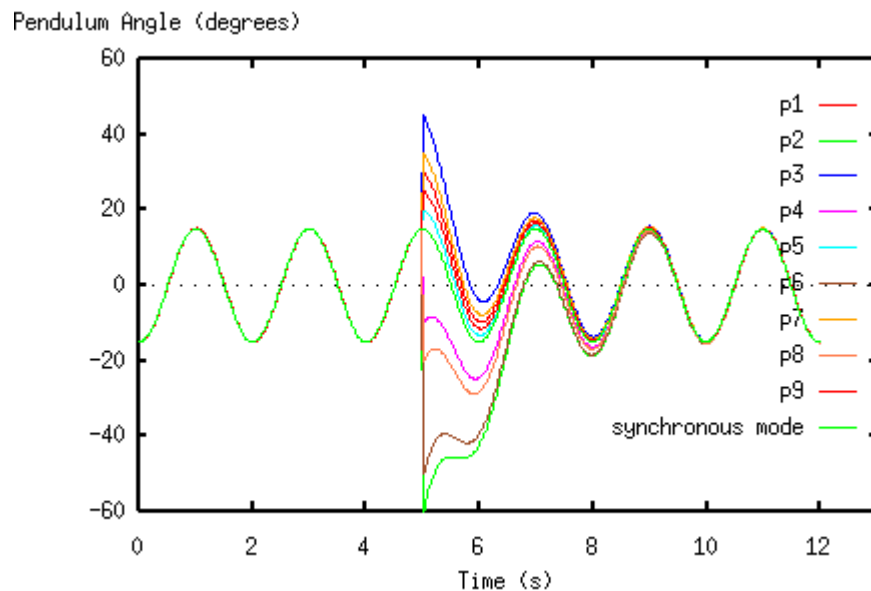


Figure 5.8: The recovery of the synchronous mode under feedback-linearization control. The figure plots the angle between each pendulum's rod and its vertical axis as a function of time. The FLC immediately brings the pendulums back on the synchronous trajectory.

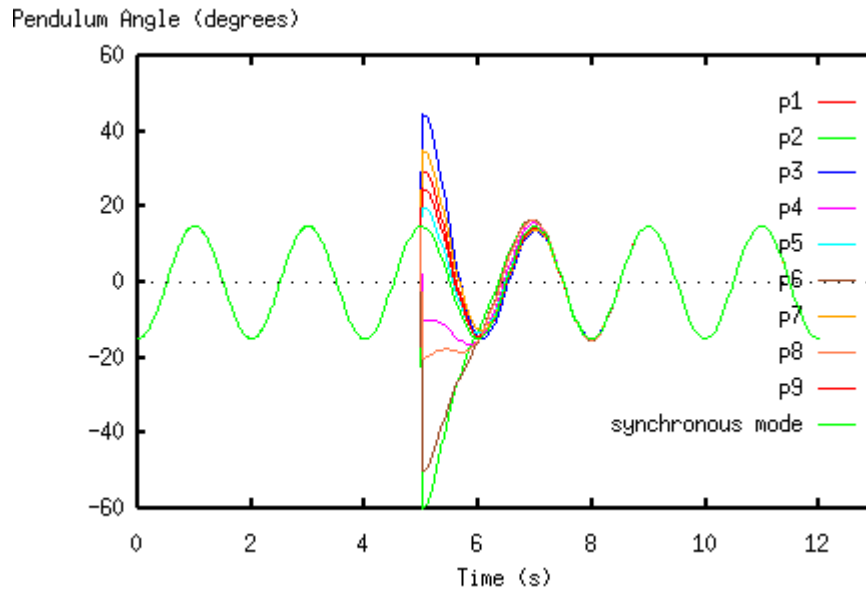


Figure 5.9: The recovery of the synchronous mode under the control of a collaborative net. The figure plots the angle between each pendulum’s rod and its vertical axis as a function of time. The agents hold off their control actions until the angles become small enough for the inputs to be cost-effective.

5.5 Summary

Large, widespread networks are operated by a multitude of dissimilar, local and distributed agents. (By dissimilar, we mean that the agents can vary in ability from simple devices, like relays, to very capable agents, such as humans. By local and distributed, we mean that each agent can sense only a few of the network’s state variables, and affect only a few of its controls.)

Two fundamental questions of the dissertation are:

- Can dissimilar, local and distributed agents operate large networks with a solution quality comparable to that of an ideal, centralized agent?
- Can these agents sustain a comparable quality while operating asynchronously?

This chapter provided experimental evidence to answer these questions in the affirmative. In small, but prototypical networks (arrays of pendulums), the asynchronous effort of local, distributed agents consistently arrived at solutions comparable in quality to those of an omniscient agent.

On the more technical side, the chapter demonstrated experimentally that the sufficient conditions for convergence are not necessary. The conditions include 1) the convexity of the static optimization problem (P), 2) serial work, 3) exact match between agents and subproblems, and 4) interior-point method.

In forests of pendulums, we initiated incidents and let the collaborative network of agents drive the pendulums back to the synchronous mode. Therein, the mentioned conditions are relaxed, that is, the agents are local and distributed (they do not induce an exact match), work asynchronously, and run an off-the-shelf optimization method. During the incidents, the agents not only frequently arrived at a solution to each element of $\langle P \rangle$, but also attained a quality comparable to that of an ideal, centralized agent.

The above scenarios also illustrated the use of margins in retaining feasible resource constraints. When the resources are scarce however, the agents perform poorly if they split the resources evenly. In this regard, mutual-help behavior proved to be helpful to improve solution quality: the agents dynamically allocate the resources (by tweaking their margins) in a way that suits the incident, making up for the varying context.

Finally, the performances of collaborative nets were contrasted with those of traditional, centralized controllers (feedback-linearization techniques). The contrast indicates that C-Nets are competitive.

Chapter 6

Experiments in Electric Power Systems

The operation of large and distributed networks, such as the electric grid, relies on the solution of dynamic control problems (DCPs). Given a) the mathematical model of the network (the equations that simulate its dynamics), b) the constraints on its limits and desired behavior, and c) the objective function, the problem reduces to finding the control actions that meet the operating constraints and minimize the objective function.

This dissertation concentrates on network-type, DCPs and its ultimate goal is to demonstrate that the suggested solution approach can be effective in large, distributed networks. The approach prescribes:

- a) the rolling horizon formulation of the DCP—the instantiation and solution of a series of static optimization problems, $\langle P \rangle$, where time is not present,
- b) the adoption of a common framework for specifying agent tasks, which shatters problem (P) into pieces, $\{P_m\}$, and delegates the task of solving (P_m) to agent- m ,
- c) the deployment of a collaborative net (a network of agents and communication links) and the match of the agents to their subproblems (the specification of the proximate, neighborhood and remote variables), and
- d) the crafting of a collaboration protocol to promote effective exchange of information, encourage joint or altruistic actions, and lead to satisfactory overall performance.

The purpose of this chapter is to show that the solution approach can be effective in controlling real-world networks of high complexity. In particular, it demonstrates this potential by solving control problems that are found in electric power networks. In this context, the problem demands a quick synchronization of generators that have been knocked off balance by random contingencies—the so-called transient stability problem [GS93, pp. 529-535]. The suggested solution approach was tested in a simplified version of the problem. The results indicate that the collaborative nets are effective, i.e., the quality of their solutions is comparable to that attained by a single agent that senses and controls the entire network. The results, however, are preliminary due to the simplifications made in the experimental scenarios. At the end, we address the extensions to this nascent research that need to be realized to fully demonstrate the potential.

Section 6.1 introduces basic concepts of power systems, reviews the problem of transient stability, and addresses its connections with the new economic regime and technologies that are driving the power industry [Whi88][GMT96]. Section 6.2 presents the experimental set-up, which is made up of the following: a) power networks that are equipped with diverse control apparatus (such as turbine-speed governors and solid-state control devices [Bal91]), b) major disturbances that force generators to oscillate and may cause instability, and c) DCPs that are instantiated to damp down the oscillations and recover synchronization. Sections 6.3 and 6.4 report the results achieved by the collaborative nets and centralized controllers, and contrast their performances, in two standard networks. Finally, Section 6.5 brings forth elements of real-world networks that are missing in our scenarios, addresses some implementation and research issues, and outlines future research efforts.

6.1 The Transient Stability Problem

The power grid consists of a sizable number of diverse equipment (such as generators, controllers, and protection devices) and thousands of miles of transmission lines. Synchronous machines generate power that is transmitted through the network and delivered to remote sites. When the network is under normal operation (in steady-state), its generating units run at the nominal frequency; they are synchronized and should remain in this state for safety and technical reasons [TF82]. (Roughly speaking, the frequency acts as a control signal that indicates shortage of power input when it is low and, conversely, excess when it is high.)

Contingencies, such as the surge of power demand and lightning strikes, create imbalances between the mechanical-power input and the electric-power drawn at the generating units. When the disturbance is severe, the generators fluctuate drastically—some speed up while others slow down—and might be shut off to prevent equipment damage. These events can, and sporadically do, cascade outwards from the faulty sites, ceasing only after a blackout.

The transient stability problem encompasses the implementation of preventive and on-line control measures to maintain synchronized generators and, thereby, sustain uninterrupted supply of power to customers [PM94, pp. 1-14]. The physical limitations to achieve this goal lie in the inability to quickly adjust mechanical-power input to match the electric-power output (the reaction time of mechanical equipment is much slower than that of the electric equipment). Today, the approach to sustaining stability is mostly preventive and involves the following measures: a) the execution of extensive simulations to spot weaknesses and set safety margins that increase fault tolerance, b) the provision of spinning reserve whereby generators are kept on standby to immediately act against a power surge, and

c) the employment of qualified personnel to monitor the network and respond to emergencies around-the-clock.

The continuous shift of demand patterns and the unbundling of the power industry are pushing the grid to its limits, leaving less space for error, and making blackouts more likely [CND+00]. It has become more difficult for today's preventive measures to sustain stability. Fortunately, new technologies are under development, both in hardware and software, to provide quick on-line control and improve stability. For instance, solid-state power control devices—the so-called flexible AC transmission systems (FACTS)—are being developed to quickly act in response to disturbances [Hin88][Hin91][Hin93]. FACTS constitute a family of devices that can quickly, reliably, and precisely adjust parameters of the physical network, such as line impedance. These devices can be switched at high speed so as to influence electric-power output and, thereby, compensate for its imbalance with mechanical-power input, absorb oscillations, and uphold the synchronization of the generators.

6.2 The Experimental Set-up

The constituents of the experimental set-up are a) the model of the physical network (its state variables, controls, and dynamic equations), b) the operating constraints, c) the disturbances, and d) the control problems that arise from the disturbances. Once these problems are instantiated, the suggested solution approach can be put into action and its effectiveness assessed. A tremendous, tedious effort is indispensable to formulate these dynamic control problems from the basic elements (especially, in data gathering, experiment design, and computer implementation). This section puts together a procedure to carry out transient stability studies by a) enumerating the steps to formulate the control problems, b) providing the details of the suggested solution approach, and c) delivering a means to evaluate the effectiveness of the solution. The steps are depicted in Figure 6.1 and are fully described below.

1) Pick a Power Network and Associated Equipment

The IEEE 14-bus and 30-bus power networks [CR73b] were chosen for the experiments. Their specification defines the topology, the transmission lines, and the properties of the associated equipment (e.g., generating units, transformers, and loads). We have added turbine-speed governors [CR73a] (that tune the speed of the turbines and thereby match mechanical-power input to electric-power output) and FACTS devices on the power lines.

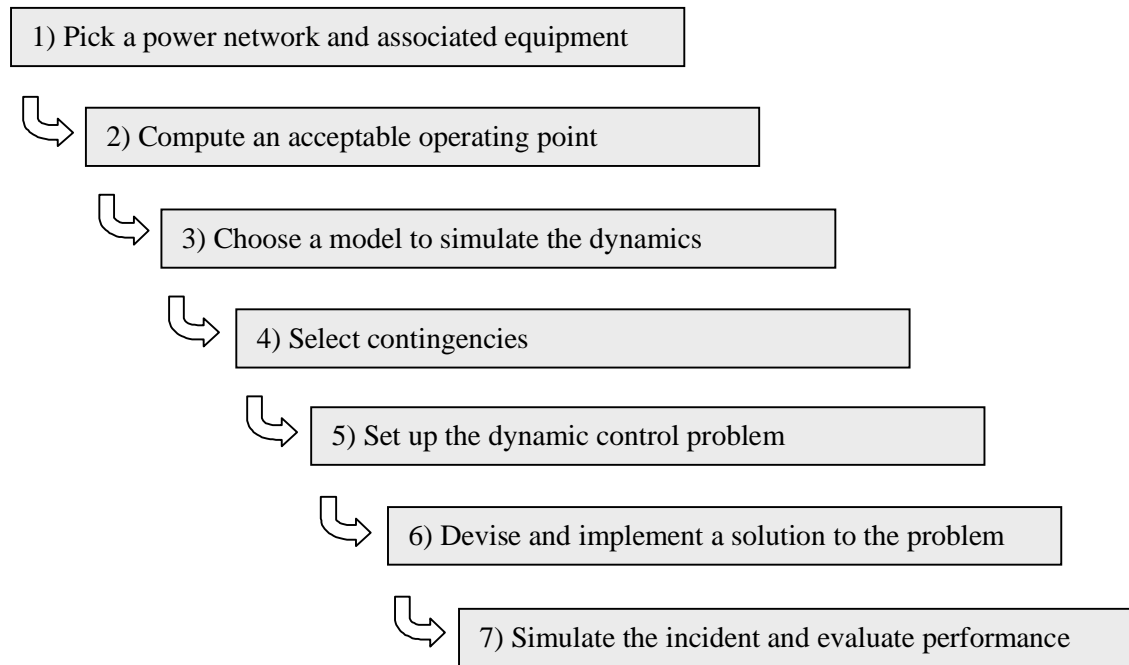


Figure 6.1: The steps to formulate dynamic control problems and assess the quality of the solutions.

2) Compute an Acceptable Operating Point

Before computing an operating point, we convert the power network into its equivalent electric circuit. This is accomplished by replacing load and generation buses with voltage sources, FACTS devices with adjustable current sources, and the other equipment with their electric equivalents.

Given the equivalent electric circuit, the computation of an acceptable operating point boils down to finding the voltage magnitude and phase angle for each circuit node or bus—locations where the voltage is identical—that meet the constraints. Examples of constraints are a) the rated voltage at the buses, b) the power delivered at the load buses, and c) the maximum power transfer in the lines.

In its simplest form, the above problem is called the load flow problem, and is equivalent to finding a solution to nonlinear equations [GS93, pp. 253-285]. Usually, two parameters are given and two are unknown at each bus. At generating buses, for instance, the voltage magnitude and real power are given, but the phase angle and reactive power are unknown. A solution to these equations constitutes an acceptable operating point.

In our implementation, the task of finding operating points was facilitated by the use of GAMS modeling language [BKM92] and MINOS 5.3 optimization package [MS83]. GAMS permitted the

development of a template to instantiate these optimization problems, and subsequently tackle them with MINOS 5.3.

3) Choose a Model to Simulate the Dynamics

The solution to a load-flow problem yields an operating (equilibrium or steady-state) point that complies with the constraints. The solution itself does not say how the network, Θ , arrives at a steady-state point, nor does it say how Θ jumps from one point to another as it experiences exogenous influences. The missing element is the dynamics of Θ . This section touches upon the modeling of its dynamics by differential equations.

A few models have appeared in the literature to simulate power-network dynamics [PM94, pp. 49-118]. The most accurate models utilize several state variables of the synchronous machines—power generators—and spell out relationships and time-dependencies in the form of differential equations. Other models, however, aggregate state variables or simply neglect the least influential ones. Our model, perhaps the simplest of all models, represents machines as voltage sources, loads as constant impedances, and FACTS devices as adjustable current sources [GAT+96]. Figure 6.2 depicts the circuit of network PN-5. (To assemble PN-5, we installed two FACTS devices—one in line 1-5 and the other in line 3-4—in the grid of Example 7.9 from [GS93, 7.4, pp. 271-275].)

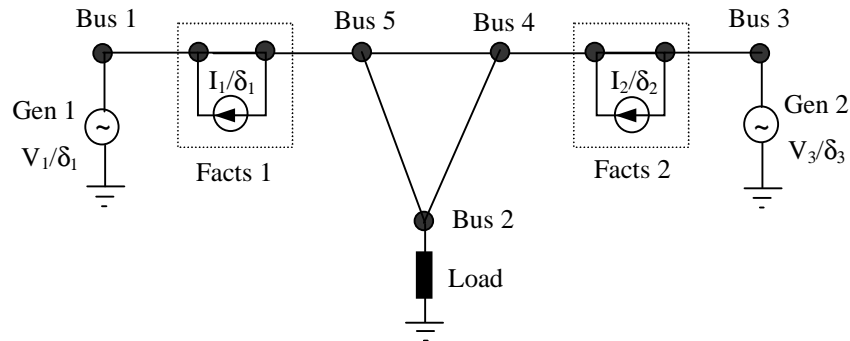


Figure 6.2: The simplified circuit of power-network PN-5. The FACTS are modeled as current sources whose phase angles (δ 's) are adjustable. PN-5 was designed by adding the FACTS devices to the network of Example 7.9 of [GS93, 7.4, pp. 271-275].

Suppose that M machines supply power, and N FACTS provide on-line control, to network Θ . In our model, the state of Θ encompasses the phase angle (δ_m) and speed (ω_m) of each machine- m , and the control input encompasses the phase angle of each device- n (u_n). Let $\delta = (\delta_1, \dots, \delta_M)$ be the vector of phase angles, $\omega = (\omega_1, \dots, \omega_M)$ be the speed vector, and $u = (u_1, \dots, u_N)$ be the control-input vector. Then,

the state of Θ becomes $x = (\delta, \omega)$ and the control input, u . With this notation, the rotor dynamics of machine- m can be simulated (by the so-called swing equations) as follows:

$$\frac{d\delta_m}{dt} = \omega_m - \omega_{\text{sync}}$$

$$\frac{d\omega_m}{dt} = [p_m(\delta_m, \omega_m, t) - pe_m(x, u, t)]/J_m - D_m[\omega_m - \omega_{\text{sync}}]/J_m$$

The list below offers a description of the parameters and functions appearing in the swing equation:

- a) $p_m(\delta_m, \omega_m, t)$ is the mechanical-power input to machine- m .

This quantity is also subject to dynamics in the sense that a turbine-speed governor [CR73a] continually struggles to match p_m to the electric-power drawn from machine- m . In gas turbines, the governor raises the steam flow (and thereby, turbine speed) to compensate for a shortage of power.

In our experimental set-up, control agents play the role of turbine-speed governors. The slow reaction-time of governors—relative to that of electric equipment—is imposed by constraints on how quickly agent- m can adjust p_m .

- b) $pe_m(x, u, t)$ is the electric-power output at machine- m .

This quantity is not only dependent on the state of machine- m , but also on the global state, non-local control inputs, and the network itself. To compute pe_m for all machines, we first calculate the voltage and current phasors (complex numbers with magnitude and angle) for all buses, and then readily obtain the power generated or absorbed by the buses.

The network state and control inputs (x and u) determine the voltage, or exclusively, the current phasor at each bus and, therefore, the missing quantities can be calculated using Kirchoff's law. (Let V be the voltage-phasor vector, I be the current-phasor vector, and Z be the impedance matrix. Then, we calculate the missing values with the equation $V = ZI$.) Once the voltage and current phasors are available, we compute the power produced or absorbed at bus- k , p_k , with the expression $p_k = \text{Real}\{V_k I_k^*\}$ (where $\text{Real}\{z\}$ is the real part, and z^* is the complex conjugate, of z). The computation of p_k can be carried out numerically (by solving the above linear equations) or algebraically (by expressing p_k as a function of x and u). The electric-power output is a highly nonlinear function of the state and control variables, involving several sinusoidal terms.

- c) J_m stands for the inertia of machine- m .
d) D_m stands for the damping factor.

- e) ω_{sync} is the nominal speed (usually, $2\pi 60$ rad/s).

4) Select Contingencies

Contingencies are random incidents that inflict disturbances in the network. Typical contingencies are lightning strikes, short circuits, and power surges. Often, they trigger countermeasures that attempt to isolate the fault, prevent equipment damage, and damp down the disturbance. A dramatic incident is that of cascading failures: a disturbance initiates a sequence of reactions that spreads outwards from the initiating site; system operators watch helplessly as their network breaks into pieces, leaving many customers without service [CT99]. These incidents bring the network to its knees, often ending with its collapse, a blackout, such as the one that occurred in the WSCC (Western Systems Coordinate Council) system in 1996. In the aftermath, millions of households were left without supply and further penalized with increased costs.

Today, the control measures are mostly preventive and are composed of the following steps:

- 1) simulate the likely incidents to single out the threatening ones,
- 2) analyze the dangerous incidents and determine corrective control actions, and
- 3) implement the control actions.

These steps are known as transient stability studies. We now describe them more precisely. Let $dx/dt = h(x,u,t)$ be the equations that simulate an incident (perhaps a series of them). Let $(x(t_0), u(t_0))$ be the initial operating point at time t_0 . The network Θ remains at the operating point from t_0 until t_1 , when the incident begins to unfold. Thereafter, the structure of Θ changes, the machines undergo oscillations, and they eventually settle down at another operating point or lose synchronization (the collapse of Θ). The incidents that hurl Θ into an unacceptable operating point, or cause its collapse, need further analysis and countermeasures to prevent reoccurrence. Often, the countermeasures are implemented as safety margins that reduce power generation and flow in the lines.

Figures 6.3 and 6.4 illustrate two incidents, one harmless and the other catastrophic, in network PN-5. Figure 6.3 shows the stable response to the harmless incident (the temporary disconnection of line 4-5 from 1 s to 1.10 s): the generators experience oscillations, but sustain synchronization and gracefully settle down at another operating point. Figure 6.4 shows the unstable response to the catastrophic incident (the disruption of line 4-5 and the short circuit at bus 2, both from 1 s to 1.5 s): the generators are helpless; they lose synchronization and are eventually shut down.

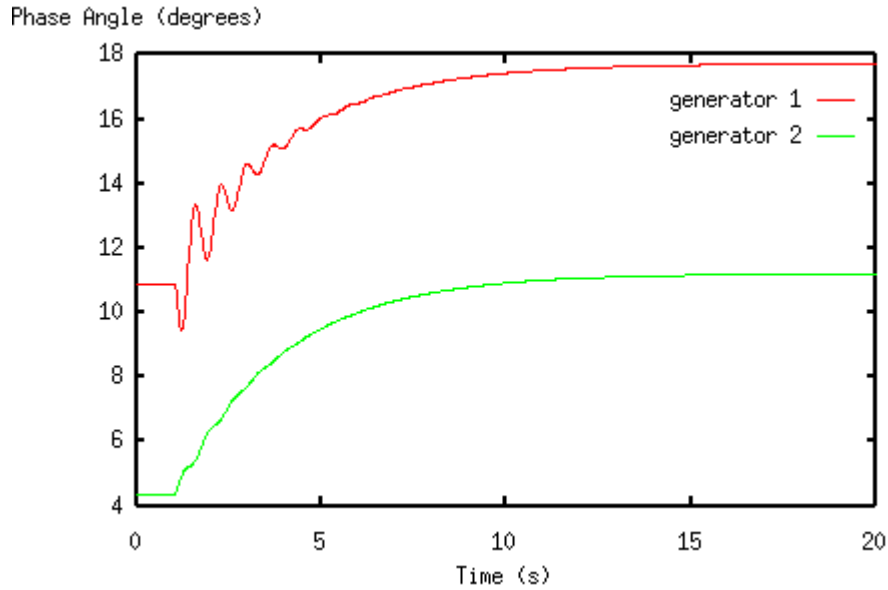


Figure 6.3: The transient response of network PN-5 to a non-severe contingency: the disconnection of line 4-5 from 1 s to 1.10 s. The incident knocks the generators off balance; however, they recover synchronization and gracefully approach another operating point.

As previously mentioned, solid-state devices are being developed to mitigate the inability to act quickly against disturbances. These FACTS devices are fast enough to be utilized on-line and, thereby, improve robustness and prevent collapse. Our experimental set-up makes use of FACTS devices and considers two types of contingencies: 1) the trigger of breakers as modeled by the disruption of lines, and 2) power surges as modeled by the abrupt, arbitrary change of load impedances.

We have decided to model disturbances as initial states that are away from the operating point, as opposed to injecting contingencies, to facilitate the implementation of experiments. For instance, the dynamic equations become time-invariant and the control task is reduced to driving the network back to its operating point. The disturbances were obtained by inflicting contingencies, simulating the dynamics (without on-line control) for a few seconds, and recording the final state.

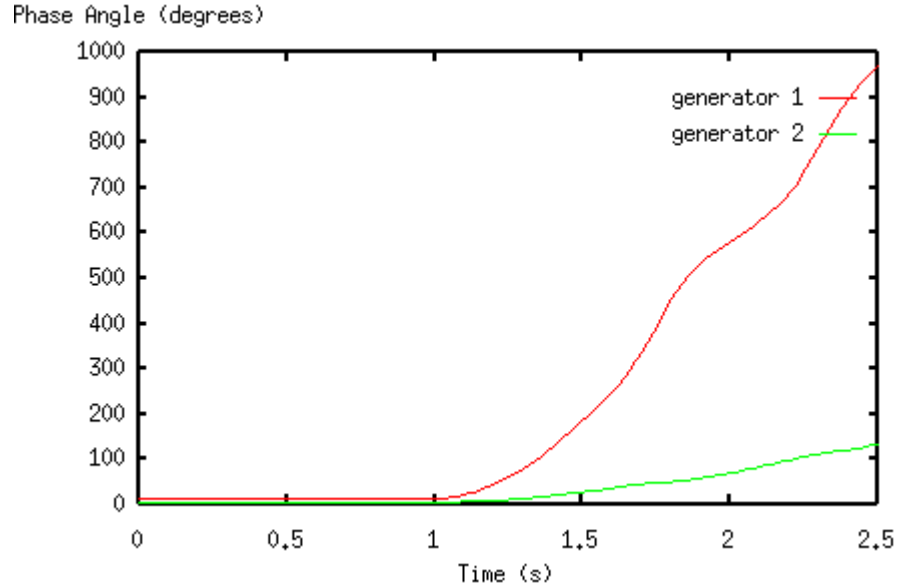


Figure 6.4: The unstable response of power network PN-5 to a severe disturbance: the temporary disruption of line 4-5 and the short circuit of bus-2, both from 1 s to 1.5 s. Gen-1 speeds up faster than Gen-2 does; they lose synchronization and are eventually shut down.

5) Set Up the Dynamic Control Problem

This step chooses an objective function, identifies the operating constraints, and combines them with the deliverables of the previous steps (e.g., the dynamic equations, the operating point, and the disturbances) to set up the DCP. This problem is formally stated below:

$$\begin{aligned} & \text{Minimize } f(x,u) \\ & \text{Subject to: } H(x,dx/dt,u) = 0 \\ & \quad G(x,dx/dt,u) \leq 0. \end{aligned}$$

Where:

- the state, $x = (\delta, \omega)$, is the vector with the phase angle and angular speed of all generators,
- the control input, $u = (\mu, v)$, consists of the angles of the FACTS, $\mu = (\mu_1, \dots, \mu_N)$, and the mechanical-power input to the generators, $v = (v_1, \dots, v_M)$,
- the dynamic equations are buried in H ,
- the constraints on mechanical-power control, and other limits, appear in G , and
- the objective function is the cumulative distance from the operating point and cost of the control effort; that is, $f = \int_{t=0}^{t=\infty} \|x - x_0\|^2 dt + \beta_\mu \int_{t=0}^{t=\infty} \|\mu - \mu_0\|^2 dt + \beta_v \int_{t=0}^{t=\infty} \|v - v_0\|^2 dt$, where a) x_0 , μ_0 and v_0 define the pre-fault operating point, and b) β_μ and β_v are conversion rates.

6) Devise and Implement a Solution to the Problem

A series of static optimization problems, $\langle(P)\rangle$, is solved in place of the DCP, whose elements are solved by the following control approaches:

a) The centralized controller (C_1)

C_1 is an agent that senses the entire network Θ , and sets the values of all its controls by solving each (P) with the help of an SQP optimization package [LZT94].

b) The collaborative net (C-Net)

The agents of the C-Net are deployed to exclusively control the FACTS devices and turbine-speed governors. The agents induce an exact match and run the proximate-exchange protocol synchronously and in parallel to solve $\{(P_m)\}$. To better understand the way they work, let:

- $Y_m(t_k)$ be the values of agent-m's proximate variables at time t_k , and
- S_m be agent-m's reaction function, that is, a mapping from Y_m to agent-m's proximate variables, (X_m, U_m) , corresponding to the solution to (P_m) with the given Y_m .

Then, the work of the agents can be described as follows:

$$\begin{aligned} (U_1(t_{k+1}), X_1(t_{k+1})) &= S_1(Y_1(t_k)), \\ (U_2(t_{k+1}), X_2(t_{k+1})) &= S_2(Y_2(t_k)), \\ &\vdots \\ (U_{M+N}(t_{k+1}), X_{M+N}(t_{k+1})) &= S_{M+N}(Y_{M+N}(t_k)). \end{aligned}$$

7) Simulate the Incident and Evaluate Performance

A complete experiment comes out of the above steps. We are left with the task of simulating the incident, recording the state-control trajectory under the control of C_1 or the C-Net, and evaluating their performances.

6.3 Experiments in the IEEE 14-bus Power Network

Figure 6.6 shows the performances of the C-Net relative to C_1 's, C-Net penalty, for incidents that take place in the IEEE 14-bus network (Figure 6.5). The chart shows that the C-Net incurred low penalty in most incidents, sustaining a C-Net penalty mean whose value is under 10%. The results are encouraging—they bear out that C-Nets can be effective in controlling networks of high complexity. The agents were left alone to grapple with the incidents and still managed to pull the network back to its

operating point. The remainder of this section reports the details of the studies herein, which carry over to the next section.

The experiments were conducted in the manner delineated in the preceding section. In particular, the C-Net is made up of ten agents: a) five of them are in charge of the turbine governors and located at the generating sites, and b) the others are in charge of the FACTS devices. Further, the perception of the agents and the communication links induce an exact match. The incidents consist of a mix of abrupt changes in power demand (load impedances) and the disruption of lines.

The evaluation of the solution found by the C-Net, $f(C\text{-Net})$, exceeded the one found by C_1 , $f(C_1)$, in all incidents. This excess is called cumulative C-Net penalty, and defined as $[f(C\text{-Net}) - f(C_1)] / f(C_1)$. Notice that cumulative C-Net penalty differs from C-Net penalty (in the sense that the former uses the objective function of the dynamic control problem—not that of the rolling horizon formulation—and therefore accounts for the entire incident).

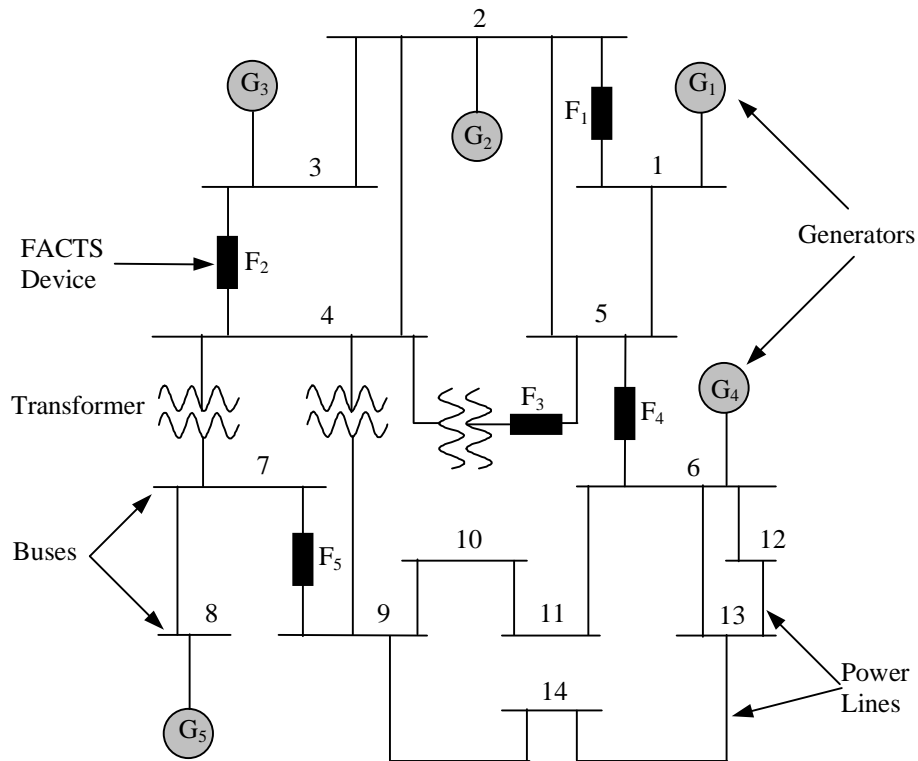


Figure 6.5: The one-line diagram of the IEEE 14-bus circuit with five FACTS devices. The diagram illustrates the locations of the generators, FACTS devices, loads, and transformers. With exception of buses 1, 7, and 8, all buses have loads plugged in.

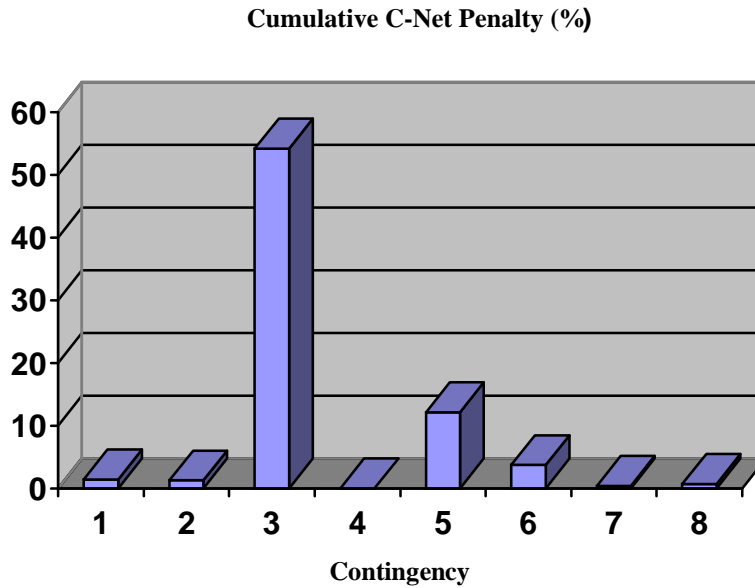


Figure 6.6: The performances of the collaborative net, relative to the centralized controller's, in the IEEE 14-bus network. The evaluation of the solution found by the C-Net, $f(\text{C-Net})$, exceeded the one found by C_1 , $f(C_1)$, in each contingency. This excess is called cumulative C-Net penalty, and defined as $[f(\text{C-Net}) - f(C_1)] / f(C_1)$. The mean value of the cumulative C-Net penalty is under 10%.

6.4 Experiments in the IEEE 30-bus Power Network

This section contrasts the performances of the C-Net with those of the centralized controller, C_1 , in incidents that take place in the IEEE 30-bus network (Figure 6.7). Both the C-Net and C_1 succeeded in pulling the network back to its operating point in all incidents. Figure 6.8 shows the cumulative C-Net penalty (%) that we incur when the authority (over the network) is transferred from C_1 to the C-Net. The results in the IEEE 14- and 30-bus networks are alike: the C-Nets attain good performances in most contingencies and sustain penalty mean-values of approximately 10%.

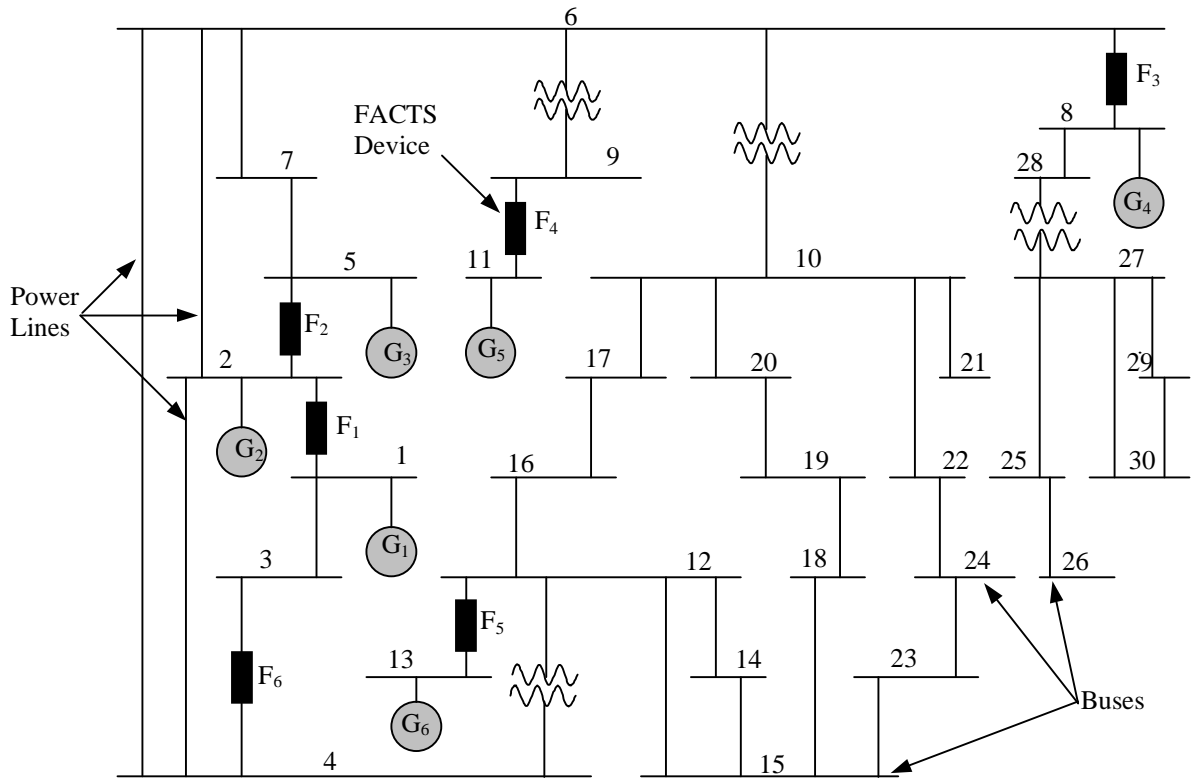


Figure 6.7: The one-line diagram of the IEEE 30-bus circuit with six FACTS devices. The diagram shows the sites of the generators, FACTS devices, and transformers. Almost all buses have loads hooked up to.

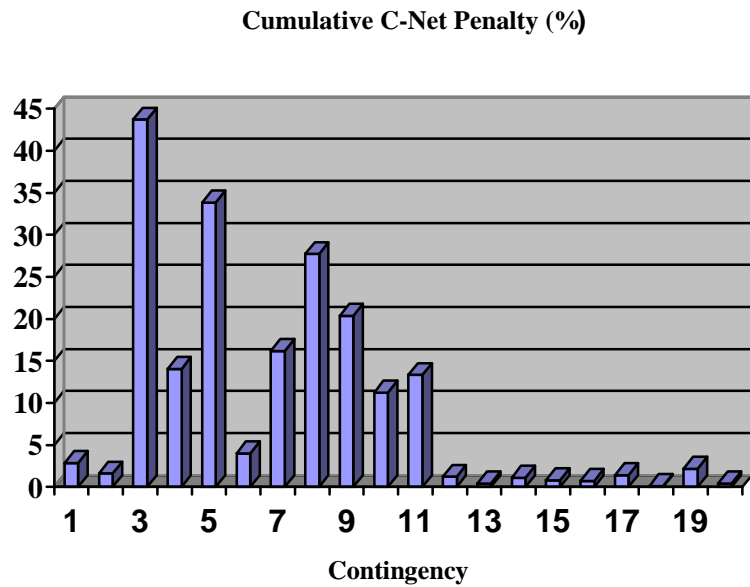


Figure 6.8: The performances of the collaborative net and the centralized controller in the IEEE 30-bus network. The chart shows the cumulative C-Net penalty, $[f(C\text{-Net}) - f(C_i)] / f(C_i)$, for each contingency.

6.5 Discussion

The results reported in this chapter are preliminary: both the problem statement and the solution approach have been simplified by omitting elements of real-world networks. The work, however, establishes the basic ground for further developments, indicates that the suggested solution approach has merit, and provides insights into the difficulties that lie ahead. Below, we address the simplifications and comment on future investigations:

- **Transient Stability Studies**

Power networks are continually jumping from one operating point to another, as they experience changes in the configuration and disturbances. However, our studies neglected this behavior to keep the implementation effort to a minimum: the contingencies are mere perturbations of the operating point and the problem rests on bringing the network back to its operating point. Accounting for the varying configuration will definitely improve the fidelity of the results, but this does not come without the expense of substantial implementation effort.

- **Exact Match**

An exact match is neither desirable, nor affordable in real-world networks. Therefore, alternatives need to be found to reduce the degree of matching in the C-Nets that were deployed to the standard power networks. The difficulty of low degree matching lies in the high coupling between the subproblems, i.e., the influence of FACTS devices on the generators is about the same order and nonlinearly dependent on their state. The studies indicate that adequate prediction of the values of the remote variables, rather than the use of the default values, is imperative to promote convergence to good solutions. The C-Nets do not always sustain synchronization when the match is inexact.

- **Asynchronous Work**

In actual networks, the agents do not solve one element of $\langle(P)\rangle$ after another. Rather, they tackle subproblems of different complexities and sizes, have different abilities, and work asynchronously and in parallel. Agent-m, for instance, might implement the solution to (P_m) every second, while agent-n might implement the solution to (P_n) every minute. Therefore, the variable work-cycle needs to be examined before C-Nets are deployed.

6.6 Summary

This chapter has provided experimental evidence that collaborative nets can be effective in complex real-world networks. It touched upon the status quo of transient stability in power systems and enumerated the basic steps for the analyses thereof. Then, these steps were followed to evaluate the performances of the collaborative nets and those of centralized controllers in two standard power grids: IEEE 14-bus and 30-bus networks. Finally, it addressed the limitations of the studies herein—namely, the use of synchronous work and the simplifications of the case studies—and suggested future investigations to improve the fidelity of our results.

With respect to the core questions of the dissertation, the chapter offers empirical evidence that local and distributed agents can arrive at solutions that approximate in quality those of a centralized agent. Further, the results suggest that the asynchronous, parallel work may produce solutions of good quality, in light of the fact that the synchronous, parallel work has yielded good solutions.

Chapter 7

Matching Agents to Subproblems

The preceding chapters were centered on the means that push the solution of $\{(P_m)\}$, by a collaborative net, toward a solution to (P). These means consist of collaboration protocols and resource margins. The subproblems, the agents, the variables measured or governed by them, and their communication links were given. How do we break up (P)? What variables should each agent sense or regulate? What should the neighborhoods be like? These design issues play a role in the effectiveness of collaborative nets and, if inadequately resolved, can seal their fate.

This chapter recognizes the relevance of the above questions and offers a preliminary study on the matching of agents to subproblems—that is, the design of the proximate, neighborhood, and remote variables of the agents. The focus is on the specification of the variables that each agent can sense and the make-up of neighborhoods that, together, maximize an estimate of the overall performance.

We bring together the guidelines that we have used to break up (P), into $\{(P_m)\}$, and present two models that assist in matching agents to subproblems. We believe that the models will help to design C-Nets, provide insights on the benefits of the design and the difficulties thereof, and serve as prototypes for more elaborate matching models. In essence, the inputs to the models are:

- 1) the agents,
- 2) the variables that the agents govern, $[U_1, \dots, U_M]$, where U_m is the set of agent- m 's control variables,
- 3) the influence of control variable u_m on state variable x_n , $\delta(u_m, x_n)$,
- 4) the communication cost for agent- m to sense state variable x_n , $w(m, x_n)$, and
- 5) the communication requirements and bottlenecks.

The first model addresses the specification of proximate variables so that the overall influence of the agents on these variables is maximized, while simultaneously minimizing the total communication cost. That is, it specifies X_m for each agent- m so as to maximize $\sum_{m=1, \dots, M} \delta(U_m, X_m)$ and, simultaneously, minimize $\sum_{m=1, \dots, M} w(m, X_m)$.

The second model extends the first one to account for the influences of each agent on its neighborhood variables while minimizing the total cost of communication—that is, it minimizes the influences of the agents on their remote variables. More formally, the model specifies X_m and Y_m for each agent- m so as to maximize $\sum_{m=1,\dots,M} \delta(U_m, \{X_m \cup Y_m\})$ and, at the same time, minimize $\sum_{m=1,\dots,M} w(m, X_m)$.

The chapter is organized as follows. Section 7.1 states the assumptions of our matching models. Section 7.2 presents the model that maximizes the influences on proximate variables, and develops an efficient algorithm to compute optimal matches. Section 7.3 formalizes the model that maximizes the influences on proximate and neighborhood variables, and also demonstrates the computational hardness of finding representative matches. Section 7.4 illustrates the use of the second model in a forest of pendulums.

7.1 Model Assumptions

Instances of the forthcoming models are made up of the following data:

- The set of subproblems, $\{(P_m)\}$, which originated from the decomposition of (P) .
- $G = (A \cup B, E)$, the bipartite graph [BM79] that models the structural influences of the agents on state variables: $A = \{1, \dots, M\}$ represents the agents (the control variables), and $B = \{1, \dots, N\}$, the state variables. (Notice that $|A| = |\{(P_m)\}|$ if the model is consistent with the subproblems.)
- The set of control variables whose values are set by agent- m , U_m .
- The influence of control variable u_m on state variable x_n , $\delta(u_m, x_n) \in R_+$. (For example, $\delta(u_m, x_n)$ could be an estimate, or average, of the sensitivity of state variable x_n to changes in control variable u_m , $\|\partial(dx_n/dt)/\partial u_m\|$, about the operating point of the network.) The influence of agent- m on state variable x_n is $\delta(U_m, x_n) = \sum_{u \in U_m} \delta(u, x_n)$.
- The cost $w(m, x_n)$ that will be incurred to establish a communication link from agent- m 's location to state variable x_n 's.
- The minimum number of communication links that each agent- m must establish, $lb(m)$, and the maximum number of links that it can handle, $ub(m)$.
- The minimum number of agents that must sense state variable x_n , $lb(n)$, and the maximum number of agents that can sense x_n , $ub(n)$.

The assumptions are the following:

- Two agents are neighbors if they sense the values of a common state variable, meaning that they can exchange data at the state variable's site. This assumption is not limiting however, since the introduction of an artificial state variable can capture the direct communication between two agents.
- The influences of the agents on one another are omitted to facilitate the analysis hereafter and, as in the preceding assumption, can be accounted for through the use of artificial state variables.

7.2 Maximizing the Influences on Proximate Variables

The problem comes down to picking the communication links (from agents' locations to state variables') that maximize the overall influence of the agents on their proximate state variables while, at the same time, keeping the total communication cost at its minimum. That is, the specification of X_m for each agent- m that maximizes $\sum_{m=1, \dots, M} \delta(U_m, X_m)$ and, simultaneously, minimizes $\sum_{m=1, \dots, M} w(m, X_m)$. (Notice that it is straightforward to elaborate the model to encompass proximate state and control variables.)

The two objectives are conflicting—an increase in sensing can incur additional communication costs. Thus, the problem falls into the class of bicriteria problems [Mie99]. Typically, they do not admit equivalent, optimal solutions, but rather contain several nondominated solutions—those whose improvement in one of the objectives invariably causes the degradation of another. These solutions constitute the so-called Pareto set [BO82] (curve or surface in continuous problems), which displays the trade-offs that we make when one objective is preferred to the other. In what follows, we show how to efficiently compute a representative subset of the Pareto set.

Theorem 7.1. Let Ω be set of nondominated solutions to the problem of maximizing the overall influence of the agents on their proximate state variables, while minimizing the total communication cost. Further, let Ψ be a representative subset of Ω , that is, one subset containing at least one nondominated solution for each possible trade-off between influence coverage and communication cost. A representative subset Ψ can be computed by solving $O(|\Psi| \log(\gamma))$ minimum-cost network-flow problems [AMO93], where γ is the number of bits used to encode numbers.

Proof. The demonstration provides an algorithm to compute Ψ .

Initialization:

- Let $G = (A \cup B, E)$ be a complete bipartite graph whose vertex sets are $A = \{m \mid m = 1, \dots, M\}$ (for the agents) and $B = \{n \mid n = 1, \dots, N\}$ (for the state variables).
- Represent each agent by a source node whose lower and upper bounds reflect its communication constraints.
- Similarly, represent each state variable by a sink node whose bounds are equivalent to its communication limits.
- Set the iteration counter to zero, $t = 0$.
- Let $(\alpha_t, \beta_t) \in (R_+)^2$ be the trade-off pair between influence sensing and communication cost, starting with $(\alpha_t, \beta_t) = (1, 0)$.
- Let the capacity of each edge (m, n) be one unit, $ub(m, n) = 1$.
- Set the cost of each edge (m, n) to the combination of agent- m 's influence on x_n and its communication cost as defined by the trade-off: $c(m, n) = -[\alpha_t \delta(U_m, x_n) - \beta_t w(m, x_n)]$.
- Let the representative subset of the Pareto set be the empty set, $\Psi = \emptyset$.

Main Procedure:

- Solve the minimum-cost network-flow problem in G ; and let ψ_t be its solution, that is, the set of links connecting agents' sites to those of state variables: ψ_t defines the proximate state variables for all agents, $[X_1, \dots, X_M]$, and is a member of the Pareto set with trade-off (α_t, β_t) . (The neighborhood variables are identified through the overlaps of agents' proximate variables.)
- Add ψ_t to Ψ .
- If the evaluation of ψ_t is identical to that of a solution to the problem where $(\alpha, \beta) = (0, \beta)$, then stop.
- Set $\alpha_{t+1} = \alpha_t$.
- Execute binary expansion and search on the parameter β_{t+1} , solving flow problems that minimize the mix of the two objectives as defined by $(\alpha_{t+1}, \beta_{t+1})$, to find the β_{t+1} that induces the "next solution," ψ_{t+1} . By the "next solution" we mean a solution ψ_{t+1} , different from ψ_t , obtained by the smallest increase on the value of β .
- Increment t by 1 and repeat the above steps.

Since number encoding is bounded by γ bits, the search for the “next solution,” ψ_{t+1} , solves at most $O(\log(\gamma))$ flow problems. Therefore, Ψ can be computed by solving $O(|\Psi|\log(\gamma))$ minimum-cost network-flow problems. ■

Corollary 7.1. Let the influence of control variables on state variables, $\{\delta(u_m, x_n)\}$, and the communication costs, $\{w(m, x_n)\}$, be random variables. A representative subset of the nondominated solutions with respect to the expected influence of the agents on their proximate state variables and communication costs, Ψ , can be found by solving $O(|\Psi|\log(\gamma))$ minimum-cost network-flow problems.

Proof. Consider a trade-off (α_t, β_t) , as defined in Theorem 7.1, and a solution ψ_t to the corresponding network-flow problem. The expected influences of the agents on their proximate variables and communication costs, denoted by $E[\psi_t]$, is precisely the weighted sum of the expected influences and expected communication costs of the links appearing in ψ_t . More formally,

$$E[\psi_t] = \sum_{(m,n) \in \psi_t} E[\alpha_t \delta(U_m, x_n) - \beta_t w(m, x_n)] = \alpha_t \sum_{(m,n) \in \psi_t} E[\delta(U_m, x_n)] - \beta_t \sum_{(m,n) \in \psi_t} E[w(m, x_n)].$$

(The expected value of the sum of random variables—not necessarily independent—is equal to the sum of their expected values [Leo94, pp. 232-239].) Hence, a representative subset of the nondominated solutions, Ψ , can be found as in Theorem 7.1 by using $E[\delta(U_m, x_n)]$ and $E[w(m, x_n)]$ in place of $\delta(U_m, x_n)$ and $w(m, x_n)$, respectively. ■

7.3 Maximizing the Influences on Proximate and Neighborhood Variables

This section brings the influences of the agents on neighborhood state variables (those whose values are obtained from their neighbors) to the previous model. The extended model seeks the communication links that maximize the overall influence of the agents on their proximate and neighborhood state variables, while keeping the total communication cost at a minimum—equivalently, it minimizes the influences of the agents on their remote state variables.

More formally, the model specifies X_m and Y_m for each agent- m (assuming that Y_m is restricted to the state variables) so as to maximize $\sum_{m=1, \dots, M} \delta(U_m, \{X_m \cup Y_m\})$ and, at the same time, minimize $\sum_{m=1, \dots, M} w(m, X_m)$ under the assumption that agent- m and agent- n are neighbors when their sets of proximate state variables overlap, that is, $X_m \cap X_n \neq \emptyset$. (We have considered only the influences on state variables for the sake of simplicity; the model, however, can be easily extended to account for the influences on the control variables.)

Like the previous model, this one accounts for two conflicting objectives—*influence maximization* and *communication cost minimization*—and admits a set of nondominated or Pareto solutions. The problem of finding a Pareto solution for a given trade-off (between influence coverage and communication cost) can be formulated in mathematical programming, using integer variables and only linear constraints [Wil78]. Thus, this problem lies in the domain of integer programming [Wol98] and, in principle, off-the-shelf packages (such as CPLEX) can be put into action to find Pareto solutions. The not so good news is that the computation of a representative subset of the Pareto set can be shown to be NP-Hard [GJ79]. In what follows, we show this computational hardness through a reduction from the clique problem [BM79, §7.2, pp. 101-116] and discuss its implications.

7.3.1 The Hardness of Computing Representative Subsets of the Noninferior Matches

Let $G_1 = (V_1, E_1)$ be an undirected graph that does not contain a clique (a complete subgraph) of size $s+1$. Is there a clique of size s in G_1 covering a given vertex u ? The question can be reduced to the problem that maximizes the overall influence of the agents (on proximate and neighborhood state variables) while minimizing the total communication cost. The steps of the reduction are:

- 1) Let $G_2 = (V_2, E_2)$ be the subgraph of G_1 induced by u , $G_1[\{u\}]$: the subgraph that consists of u , its adjacent vertices, and all the edges with both ends in these vertices.
- 2) Generate graph $G = (A \cup B, E)$ for the matching problem as follows:
 - Let $A = \{1, \dots, M\}$ be the set of vertices that correspond to the agents, where $|V_2| = M$.
 - Let $B = \{1, \dots, M\}$ be the set of vertices that correspond to the state variables.
 - Let ϕ be a bijective mapping from the elements of A onto those of V_2 .
 - Assume that $\phi(1) = u$.
 - Let φ be the inverse of ϕ (the mapping from the elements of V_2 onto the elements of A).
 - Let $E = \{(m, m) \mid m \in A\} \cup \{(\varphi(p), \varphi(q)), (\varphi(q), \varphi(p)) \mid (p, q) \in E_2\}$.
- 3) Let $U_m = \{u_m\}$ for each m in A .
- 4) Set $w(m, x_m) = 0$ and $\delta(u_m, x_m) = 1$ for each vertex m in A .
- 5) Set $w(m, x_n) = s - \varepsilon$ and $\delta(u_m, x_n) = 1$ for all $(m, n) \in E$ such that $m \neq n$. Parameter ε is a small, positive value drawn from the open interval $(0, 1/(2M))$, which facilitates the forthcoming demonstration. Essentially, it allows the counting of all links from agents' locations to those of state variables, directly from the value of the objective function and without including the links that connect each agent- m 's location to x_m 's. Also, parameter ε makes the evaluation of a match

corresponding to a clique of size s , higher than that of a trivial match (one that connects each agent- m to state variable x_m).

- 6) Set the trade-off pair (α, β) to $(1, 1)$.
- 7) Set $lb(n) = 0$ and $ub(n) = M$ for all n in B .
- 8) Set $lb(m) = 0$ for all m in A .
- 9) Set $ub(m) = 1$ for all m in $A \setminus \{1\}$, and set $ub(1) = s$. (These bounds prevent the identification of multiple cliques of size s and force the “pricy” links—edges of the original graph, G_2 —to emanate only from vertex 1, which corresponds to vertex u .)
- 10) Let ψ^* be a nondominated match for the trade-off pair $(\alpha, \beta) = (1, 1)$, that is, a set of links that maximizes the following function:

$$f(\psi^*) = \alpha \cdot \sum_{(m,n) \in \psi^*} \delta(U_m, x_n) + \alpha \cdot \sum_{(m,n) \notin \psi^*, \text{ s.t. } (m,p), (k,p), \text{ and } (k,n) \in \psi^*} \delta(U_m, x_n) - \beta \cdot \sum w(m, x_n)$$

- 11) Graph G_2 contains a clique of size s covering vertex u if, and only if, the value of the objective function of ψ^* , $f(\psi^*)$, is equal to $M + (s - 1)\epsilon$. (Essentially, $f(\psi^*)$ takes on value M if G_2 does not contain a clique of size s , and $M + (s - 1)\epsilon$ otherwise.)

Below, we demonstrate that the statement in step-11 is valid.

Theorem 7.2. The graph G_2 has a clique of size s covering vertex u if, and only if, the value of the objective function of ψ^* for the trade-off pair $(\alpha, \beta) = (1, 1)$, $f(\psi^*)$, is equal to $M + (s - 1)\epsilon$.

Proof. (\Rightarrow) Let C be a clique of size s that contains vertex u . Compute the following sets of links:

- The *trivial proximate* links: the links in $N_1 = \{(m, m) \mid m \in A\}$.
- The *nontrivial proximate* links: the links in $N_2 = \{(\phi(u), \phi(v)) \mid (u, v) \text{ is in } E(C)\}$.

Let $\psi^* = N_1 \cup N_2$ be a match. Notice that $|N_1| = M$ and $|N_2| = s - 1$. Let the *neighborhood* links constitute the set $N_3 = \{(m, n) \mid (m, n) \notin N_2, (\phi(m), \phi(n)) \in E(C), \text{ and } \exists \text{ agent-}k \text{ and state variable } x_p \text{ such that } (m, p), (k, p), \text{ and } (k, n) \text{ belong to } \psi^*\}$. That is, link (m, n) belongs to N_3 when agent- m senses state variable x_n indirectly, through its neighboring agent- k . Since C defines a clique of size s , it follows that $|N_3| = s^2 - s - (s - 1) = (s - 1)^2$. Thus, $f(\psi^*) = |N_1| + [|N_2| - |N_2|(s - \epsilon)] + |N_3| = M + [(s - 1) - (s - 1)(s - \epsilon)] + (s - 1)^2 = M + (s - 1)\epsilon$. Notice that ψ^* is indeed a nondominated solution. The connection of an agent- m , other than agent-1, to a state variable x_n , other than state variable x_m , would potentially increase the value of $f(\psi^*)$ by ϵ only if G_2 contained a clique of size $s + 1$. Therefore, the *nontrivial proximate* links (the ones

emanating from agent-1) and the *trivial proximate* links induce a nondominated match ψ^* for the trade-off pair $(\alpha, \beta) = (1, 1)$.

(\Leftarrow) Let ψ^* be a nondominated match for $(\alpha, \beta) = (1, 1)$ such that $f(\psi^*) = M + (s - 1)\epsilon$. Distribute the elements of ψ^* into the sets of *trivial proximate* links (N_1) and *nontrivial proximate* links (N_2). Further, compute the *neighborhood* links (N_3) induced by ψ^* : $N_3 = \{(m, n) \mid (m, n) \notin \psi^* \text{ and } \exists \text{ agent-}k \text{ and state variable } x_p \text{ such that } (m, p), (k, p), \text{ and } (k, n) \text{ belong to } \psi^*\}$. It follows that $f(\psi^*) = |N_1| + [|N_2| - |N_2|(s - \epsilon)] + |N_3| = M + (s - 1)\epsilon$. Since $0 < \epsilon < 1/(2M)$ and $|N_2| \leq M + (s - 1) < 2M$, it follows that $0 < |N_2|\epsilon < 1$. These equalities and inequalities imply that $|N_2| = (s - 1)$ and that $|N_1| + |N_3| = M + (s - 1)^2$. According to the previous argument, $|N_1| = M$ in any optimal solution and, therefore, and $|N_3| = (s - 1)^2$. From these facts and the bounds on communications, we conclude that all *nontrivial proximate* links, N_2 , emanate from agent-1. Let τ^* be the subset of ψ^* that spans the largest connected component of G containing agent-1. Notice that the set of vertices of the subgraph induced by τ^* , $V(G[\tau^*])$, has cardinality $2|N_2| + 2 = 2s$. Since all *neighborhood* links are induced by τ^* , the number of links in $G[\tau^*]$ is $|E(G[\tau^*])| = (|N_2| + 1) + |N_2| + |N_3| = s + (s - 1) + (s - 1)^2 = s^2$. Therefore, the set of vertices $V_u = \{\phi(m) \mid m \in A \text{ and } m \in V(G[\tau^*])\}$ defines a clique of size s in $G_2(\{u\})$ that covers vertex u . ■

Corollary 7.2. The problem of finding a nondominated match for the trade-off pair $(\alpha, \beta) = (1, 1)$ is computationally hard.

Proof. The problem of finding a clique of size s in a graph $G_1 = (V_1, E_1)$, covering a vertex u , can be reduced to the problem of finding a nondominated match with $(\alpha, \beta) = (1, 1)$ in polynomial time, as described in steps 1 through 11 and demonstrated in Theorem 7.2. Since the maximum clique problem for undirected graphs is NP-Complete [GJ79] and, moreover, can be trivially reduced to a polynomial sequence of the clique problem in consideration, it follows that computing a nondominated match for $(\alpha, \beta) = (1, 1)$ is NP-Hard. ■

Theorem 7.3. Let Ω be the set of nondominated solutions to the problem of maximizing the overall influence of the agents on their proximate and neighborhood state variables, while simultaneously minimizing the total communication cost. Further, let Ψ be a representative subset of Ω , one containing a nondominated match for each trade-off (between influence coverage and communication cost). If the size of the output, $|\Psi|$, is polynomial on the size of the input (the size of the graph $G = (V,E)$), then the computation of Ψ is NP-Hard.

Proof. Suppose that there exists an algorithm capable of computing Ψ in polynomial time on the size of the input ($O(MN)$). Then, compute the convex hull of Ψ , $C(\Psi)$, by sorting the elements of Ψ with respect to communication cost and then control influence. This operation can be performed in polynomial time because, by assumption, $|\Psi|$ is polynomial on the size of the input. A simple binary search over $C(\Psi)$ can identify a match ψ that is nondominated for the trade-off pair $(\alpha,\beta) = (1,1)$. But this contradicts Corollary 7.2. ■

7.3.2 Implications of the Computational Hardness

From the computational hardness of finding a representative subset of the nondominated matches, and the nature of the problem thereof, we draw the following conclusions:

- **The Synthesis of Nondominated Matches Is Likely to Be Computationally Hard in Extensions of This Model**

After generalizing the current model, we invariably expand the class of problems that can be reduced to it and, thereby, the new model remains computationally hard.

- **Near-Optimal Solutions Are Possibly Adequate**

The quality of the match (as measured by the above models) is only indicative of the performance of the collaborative net—there are discrepancies between the match quality predicted by the model and the actual performance of the induced C-Net. As such, near-optimal matches, those close to the Pareto set, are likely to be satisfactory.

- **Heuristics May Find Near-Optimal Solutions for the Hard Matching Problems**

Zanakis and Evans [ZE81] define heuristics as simple procedures, often guided by common sense, that are meant to provide good but not necessarily optimal solutions to difficult problems, easily and quickly. In practice, heuristics find near-optimal solutions even though they seldom have “good,” if any, performance guarantees [ZEV89]. Further, general-purpose

heuristics—the so-called metaheuristics, such as Asynchronous Teams [TBG+98], Genetic Algorithms [Dav90][CT97], and Tabu Search [Glo89]—can tackle the matching problems.

- **Randomization May Lead to the Design of Effective Algorithms**

An algorithm that makes decisions at random is called a randomized algorithm [MR95]. In some applications, they are the simplest, or the fastest, or the only alternative. For the problem of matching agents to subproblems, it is not hard to design a randomized algorithm whose probability of finding a nondominated match, for a given trade-off, is at least $1/M^4$ (assuming that the number of agents and state variables are both M , and that all match sizes are equally likely to be nondominated). This fact is encouraging and opens up a research direction.

7.3.3 Accounting for the Direct Communication between Agents

The model assumes that two agents can exchange data at the site of a state variable that they both reach with communication links, that is, two agents are neighbors if their sets of proximate state variables overlap. These assumptions impose no limitation since the direct communication between agent- m and agent- k —without the mediation of the site of a state variable—can be captured as follows:

- 1) create an artificial state variable $x_{(m,k)}$,
- 2) set the effects of agent- m and agent- k on state variable $x_{(m,k)}$ equal to zero, i.e., $\delta(U_m, x_{(m,k)}) = 0$ and $\delta(U_k, x_{(m,k)}) = 0$, and
- 3) set the cost to establish direct communication from agent- m , and agent- k , to the site of $x_{(m,k)}$, $w(m, x_{(m,k)})$ and $w(k, x_{(m,k)})$, to identically half the cost of communication between the agents.

7.3.4 Accounting for the Influences on Control Variables

The model can be augmented to encompass the influences of the control variables on themselves. To do so, just introduce artificial state variables to represent the control variables, and set values of the influences and communication costs (the δ and w parameters) accordingly.

7.4 Putting a Matching Model into Action

This section illustrates the assistance of the preceding model in matching agents to subproblems. For a forest of six pendulums, we synthesize a representative subset of the nondominated matches, randomly generate dominated matches, and then check the performances of the resulting C-Nets. The results

indicate that the match quality (as estimated by the model) translates into the performance of the induced C-Net. More specifically, the partial order [CLR90, pp. 81-83] of dominance in match quality space appears to be consistent with the partial order of dominance in C-Net performance space. The material below describes the experimental set-up, details on the synthesis of matches, reports the performances of the induced C-Nets, and discusses the results.

7.4.1 The Experimental Set-Up

The experimental scenario is a forest of six pendulums placed along a line. In this arrangement, each pair of pendulums is joined by a spring whose stiffness is inversely proportional to their distance—this stiffness pattern captures the weak bond between components that are far apart, relative to those of close ones, which is typical of large and distributed networks. Figure 7.1 offers a view of the pendulums, Table 7.1 presents the stiffness of their coupling springs, and Table 7.2 shows the costs of communication.

Like in Chapter 5, each pendulum is equipped with an agent that senses its state and sets the values of its control variables. (More specifically, the state of a pendulum consists of its swing angle, rotational angle, and their rates of change. Its controls are two horizontal and orthogonal forces that act on its swing and rotational planes. For more details, see Figure 5.3.) In the event of a disturbance, the control mission becomes to restore the synchronous operation quickly and cheaply—that is, to minimize cumulative deviation from the synchronous trajectory and cumulative control-input cost. Herein, the task consists in identifying the match that induces the best C-Net to complete the control mission, with a given communication budget.

Pendulum Number	Pendulum Number					
	1	2	3	4	5	6
1	-	1.0	0.5	0.4	0.3	0.2
2	1.0	-	1.0	0.5	0.4	0.3
3	0.5	1.0	-	1.0	0.5	0.4
4	0.4	0.5	1.0	-	1.0	0.5
5	0.3	0.4	0.5	1.0	-	1.0
6	0.2	0.3	0.4	0.5	1.0	-

Table 7.1: The stiffness of the springs in the forest of pendulums. The coupling between network components, as modeled by the stiffness of the springs, diminishes with distance. The magnitude of the stiffness ranges from a maximum of one unit (when the pendulums are adjacent) to a minimum of two tenths of unit (when they are far-off).

Pendulum Number	Pendulum Number					
	1	2	3	4	5	6
1	0.00	1.7873	1.1158	2.7057	4.6263	4.0197
2	0.5444	0.00	1.0503	1.4053	3.3443	4.6762
3	1.8578	1.4189	0.00	1.3958	1.7567	3.7200
4	2.6839	2.3205	1.6359	0.00	1.1361	1.7408
5	4.9595	2.3459	2.7603	1.0452	0.00	0.0235
6	4.8658	3.6679	3.0310	1.8470	1.1656	0.00

Table 7.2: The costs of communication from agents to pendulums. Entry (m,n) shows the cost $w(m,x_n)$ to establish a communication link from agent- m (sitting at pendulum- m) to pendulum- n . The value of $w(m,x_n)$ was generated by adding a random number, drawn uniformly from $(-1,1)$, to the “distance” between the pendulums, $|m - n|$.

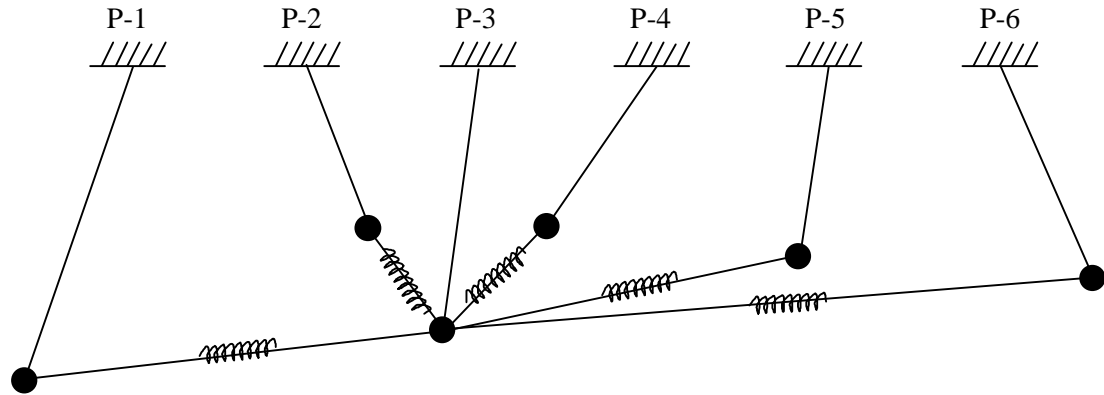


Figure 7.1: A view of the fully connected six-pendulum forest. Only the springs linked to pendulum-3 are depicted however.

7.4.2 Computing Matches

The inputs to the matching model are the following:

- The set of subproblems $\{(P_m)\}$ that conforms to the breakup of (P) performed in Chapter 5.
- The graph $G = (A \cup B, E)$, whose agent set is $A = \{1, \dots, 6\}$, whose state variable set is $B = \{1, \dots, 6\}$ and corresponds to the pendulums, and whose edges induce a complete bipartite graph.
- The variables governed by each agent- m correspond to the controls of pendulum- m . We assume that agent- m can measure the state of pendulum- m without incurring any communication cost, and that its influence on pendulum- m is 2 units.
- The effect of agent- m on pendulum- n , $\delta(U_m, x_n)$, is estimated as the stiffness of the spring linking pendulum- m to pendulum- n (Table 7.1).
- The cost to establish a communication link from agent- m to pendulum- n , $w(m, x_n)$, is given in Table 7.2.
- The bounds on communications for each agent- m are $lb(m) = 1$ and $ub(m) = 6$.
- The bounds on communications for each pendulum- n are $lb(n) = 1$ and $ub(n) = 6$.

A binary expansion-and-search was run to vary the trade-off pair, (α, β) , and synthesize nondominated matches, as discussed in Theorem 7.1. However, integer-programming problems [Wol98] were solved for each (α, β) , rather than minimum-cost network-flow problems. Figure 7.2 shows the values of the two objectives—the overall influence of the agents on their proximate and neighborhood variables, and

the total communication cost—of the nondominated matches, $\{p_1, \dots, p_7\}$, together with those of randomly generated matches, $\{r_1, \dots, r_9\}$.

We acknowledge the use of the NEOS Server at Argonne National Lab for solving the above integer-programming problems. They were formulated in AMPL modeling language [FGK93], and then solved with the optimization package MINLP [Ley98].

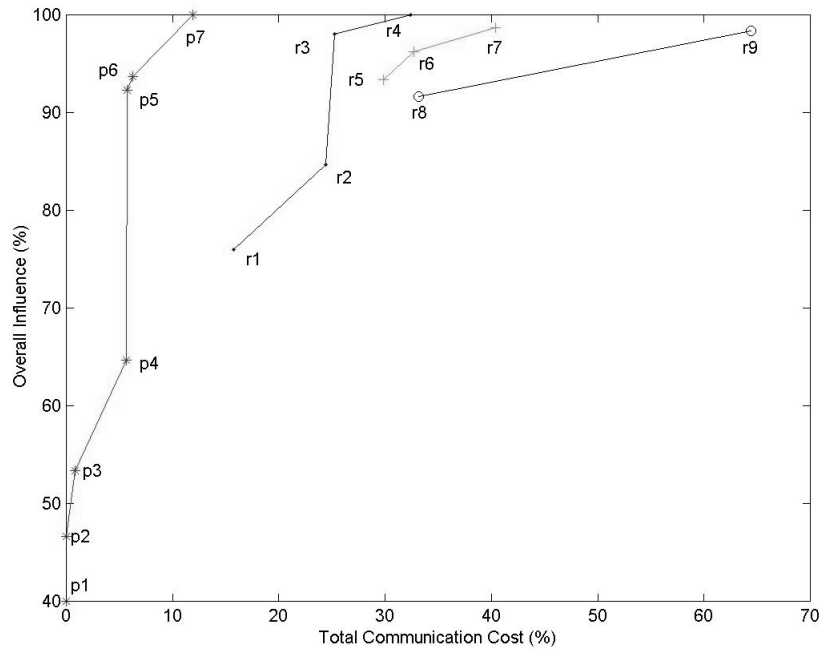


Figure 7.2: The values of the objectives of dominated and nondominated matches. Each point p_m corresponds to a nondominated match, that is, one whose objectives cannot be improved simultaneously. Each point r_m corresponds to a dominated, randomly generated match.

7.4.3 Evaluating C-Net Penalty

A collaborative net was assembled according to the specifications of each match, and subsequently deployed to sustain the synchronous operation of the pendulums. After a random incident, the C-Net and the centralized controller C_1 solved a series of static optimization problems, $\langle(P)\rangle$, containing hundreds of elements, to bring the pendulums back to the synchronous mode. (By centralized controller, we mean an agent that can sense the state, and adjust the controls, of the entire network. Basically, C_1 invokes a nonlinear-optimization package to solve each element of $\langle(P)\rangle$.)

Consider any element (P) of $\langle(P)\rangle$. Let (X_{c1}, U_{c1}) be the solution to (P) found by C_1 , and (X_{cn}, U_{cn}) be the solution to $\{(P_m)\}$ found by the C-Net. In all incidents, the evaluation of the solution attained by the C-

Net in the rolling horizon formulation, $f(X_{CN}, U_{CN})$, always exceeded the one attained by C_1 , $f(X_{C1}, U_{C1})$. The extent of this excess is called C-Net penalty and is defined as $[f(X_{CN}, U_{CN}) - f(X_{C1}, U_{C1})] / f(X_{C1}, U_{C1})$. Figure 7.3 shows the mean value of this excess, over hundreds of experiments, for the C-Nets that were induced by the dominated and nondominated matches.

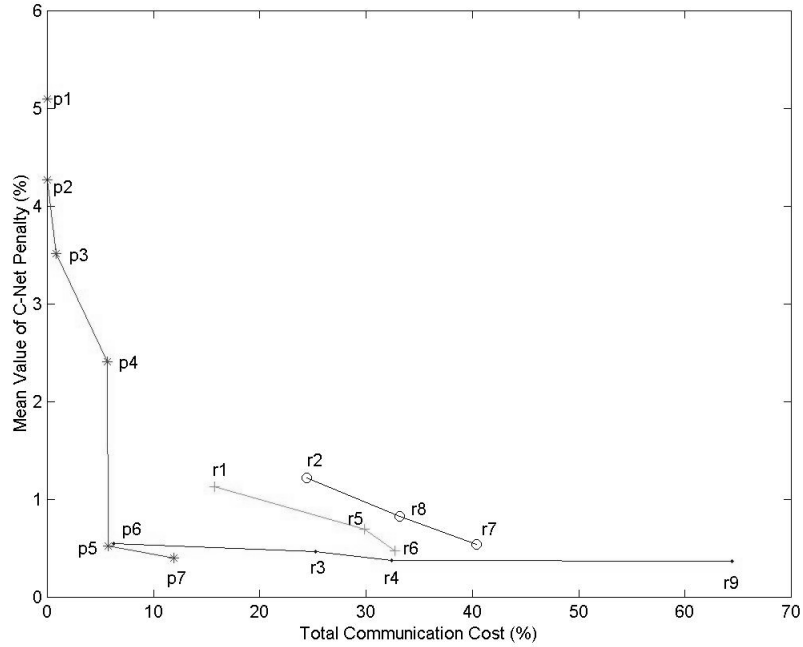


Figure 7.3: The penalties incurred by the C-Nets that were induced by the nondominated matches and dominated, randomly generated ones. The C-Net penalty is defined as follows: $[f(X_{CN}, U_{CN}) - f(X_{C1}, U_{C1})] / f(X_{C1}, U_{C1})$, where (X_{C1}, U_{C1}) is the solution found by C_1 , (X_{CN}, U_{CN}) is the solution found by the C-Net, and f is the objective function in the rolling horizon formulation.

7.4.4 Discussion

The evaluation of the actual quality of a match is laborious and time-consuming—it entails assembling the resulting C-Net, simulating the likely incidents, and measuring performance. Thus, the failure of a direct search for noninferior (nondominated) matches looms large as the size of the network increases. Take, for instance, a network made up of M agents and N state variables; it is easy to check that the number of communication arrangements and therefore potential matches, $\lambda(M, N)$, is bounded below by $\text{Max}\{M^N, N^M\}$ and above by 2^{MN} . To allay the curse of dimensionality, we have to put our bets on a model that, without resorting to simulation, estimates match quality and promotes rapid computation of the noninferior matches.

Along the lines of the first issue (the estimation of match quality), the results reported in Figs. 7.2 and 7.3 appear to suggest the appropriateness of our model. Specifically, the noninferior matches for the model have remained noninferior when their quality is measured in terms of C-Net penalty. Some discrepancy is expected such as the jump of match r_2 (from the 2nd to the 4th layer of dominance) and the non-membership of match p_6 to the actual Pareto set. We have noticed that, despite these discrepancies, the mismatch between a) the dominance relation in the model and b) the actual dominance relation in terms of C-Net penalty is reasonably low. (To understand this point, let A_1 be the matrix where each element (m,n) is zero if, according to the model, match- m is not superior to match- n , and one otherwise. Similarly, let A_2 be the matrix obtained from the performances of the C-Nets. It turns out that the mismatch between A_1 and A_2 —the number of inconsistent entries—is under 4% in our experiments.)

With respect to the second issue (the computation of noninferior matches), the prospects could be pessimistic in light of the fact that the problem is computationally hard. These prospects, however, do not need to be taken as setbacks but rather as opportunities to design heuristics [ZEV89], amplify their abilities with metaheuristics [TBG+98], and perform research on other algorithmic techniques [Hoc97][MR95].

7.5 Summary

The issues in this dissertation are speed and quality of the decisions reached by heterogeneous, local and distributed agents in the operation of networks. For quality, we want the agents' decisions to be comparable to those of centralized controllers. For speed, we want the agents to work asynchronously, all agents in parallel and each at its own pace.

This chapter has recognized that the agents' "local" regions of influence play a role in the quality of their decisions. It has suggested models for defining these regions (matching agents to subproblems), that is, for specifying the variables that each agent can sense (or control) and the neighborhoods. Additionally, it has analyzed the computational effort to solve the problems intrinsic to the matching models, and offered an empirical study whose results corroborate our claims—namely, that the agent-subproblem match influences the quality (through the sensing of the influences that the agents exert on the state and control variables) and speed (through reduced communication) of the decisions found by the collaborative net.

Chapter 8

Conclusions

Large, widespread networks are operated by a multitude of heterogeneous, local and distributed agents. (By “heterogeneous” we mean that the agents can range from simple devices, like relays, to very intelligent entities, like experts. By “local and distributed” we mean that each agent can sense only a few of the network’s state variables and influence only a few of its control variables.)

The fundamental questions are:

- Can heterogeneous, local and distributed agents arrive at solutions comparable in quality to those obtained by centralized, ideal controllers?
- Can these agents yield the same quality while working asynchronously?

This dissertation provided evidence (some analytical, some empirical) to answer these questions in the affirmative.

On the analytical side, we have found sufficient conditions for the effort of the agents to arrive at a globally optimal solution. On the empirical side, we have observed that their asynchronous effort converges consistently to good solutions in small, but prototypical networks (arrays of pendulums). Further, we have observed that the parallel, synchronous effort of the agents produces good solutions in highly complex and coupled networks (power grids), suggesting that the parallel, asynchronous effort may yield good solutions as well.

8.1 Contributions

A Framework for Specifying the Tasks of the Agents

The rolling horizon approach was extended into a framework that divides the overall optimization task into much smaller, localized tasks to be tackled by the heterogeneous, local and distributed agents.

A Taxonomy of Collaboration Protocols

The success in operating a network by distributed agents is due, to a great extent, to the collaboration protocol that promotes effective solution of the agents' tasks. It then becomes pertinent to formalize the design alternatives for protocols. In this regard, we identified relevant attributes, put them together in a taxonomy, and studied a few instances (some analytically, some empirically).

A Demonstration That Collaborative Nets Can Be Effective

We put flat organizations of heterogeneous, local and distributed agents (collaborative nets) to the test in prototypical networks (forests of pendulums) and in more realistic, highly complex networks (standard power grids). In the forests of pendulums, the parallel, asynchronous effort of the agents produced solutions comparable in quality to those produced by an omniscient agent. In power grids, the parallel, synchronous effort yielded good solutions as well.

A Quantitative Study of Agent-Subproblem Matching

We recognized that the specification of the local, influence region of each agent, that is, the variables that it can sense (or set the values of) and its neighborhood, play an important role in the performance of C-Nets. To this end, we have a) suggested models for assisting in matching agents to subproblems, b) analyzed analytically the computational complexity of the problems thereof, and c) offered empirical results.

8.2 Extensions and Future Research

Applications of Collaborative Nets

We believe that the proposed approach (the breakup of the global task into small, local subtasks and their solution by a C-Net) can be beneficial to a number of networks, ranging in structure (from static to dynamic), in decision-making and control (from discrete to continuous), and in operation (from on-line to off-line). Two potential instances are 1) a robot team engaged in surveillance, landscape mapping, or combat, and 2) a network in charge of production planning in an industrial floor. We anticipate research on the following questions:

- What means, such as protocols for agent collaboration and learning, will promote convergence to good solutions in mixed-integer decision-making problems?
- How can constraint-satisfaction techniques be incorporated in the framework?

The Role of Stochastic Optimization in the Framework and Protocols

To date, only deterministic formulations of rolling horizon have been used in our framework. This has proven to be adequate as long as the decisions are revised frequently to compensate for model discrepancies. However, communication can become a burden, limiting the asynchronous, autonomous work. We foresee the switch to stochastic formulations as advantageous in scenarios where prediction is inaccurate, and where there is a greater need of autonomy. We plan to look into questions such as:

- What are the implications of stochastic optimization to convergence?
- How can distributions on the variables be modeled, estimated, or learned?

The Role of Automatic Learning

The behavior of the agents has been programmed as rules in the collaboration protocols. Here, there is opportunity to leap ahead in making tomorrow's networks more autonomous. We mean the use of learning to improve the agents' competence from past experience. More concretely, we see learning capabilities embedded in the agents to improve their predictions and model selection continuously. Indeed, these issues make up our future research questions:

- How can learning be extended to derive distributions within a stochastic framework?
- How can prediction benefit from learning in the operation of networks?

Algorithms and Combinatorics in Self-Organizing C-Nets

We plan to push further the state-of-the-art of the agent-subproblem matching models, and algorithms to solve the problems thereof. One front consists of the development of heuristics—as well as the identification of properties of the integer programs, such as cuts—to expedite the search for noninferior matches. Along the same lines, the other front lives in the crafting of randomized algorithms for matching. We predict that the second front will give rise to self-organization and, thereby, increase the autonomy of networks.

Appendix A

More on Collaboration Protocols

A.1 The (Serial) Proximate-Exchange Protocol

Lemma A.1. If:

- 1) The conditions of Theorem 3.3 hold.
- 2) The overall, current solution belongs to the interior of the feasible set, that is, $(X, U) \in \text{int}(S)$.
- 3) $d_m = (d_{x_m}, d_{u_m})$ is a feasible improving direction for the minimization of λ_m at (X, U) , that is,
 $(\partial\lambda_m/\partial X_m, \partial\lambda_m/\partial U_m)^T d_m < 0$.
- 4) c_1 and c_2 are two positive constants such that $0 < c_1 < c_2 < 1$.

Then, there exist intervals for $\gamma > 0$ that satisfy the so-called Wolfe conditions [NW99, page 39]:

- $\lambda_m(\alpha, X_m + \gamma d_{x_m}, U_m + \gamma d_{u_m}, Y_m) \leq \lambda_m(\alpha, X_m, U_m, Y_m) + c_1 \gamma (\partial\lambda_m/\partial X_m, \partial\lambda_m/\partial U_m)^T d_m$
- $((\partial\lambda_m/\partial X_m)(\alpha, X_m + \gamma d_{x_m}, U_m + \gamma d_{u_m}, Y_m), (\partial\lambda_m/\partial U_m)(\alpha, X_m + \gamma d_{x_m}, U_m + \gamma d_{u_m}, Y_m))^T d_m \geq c_2 (\partial\lambda_m/\partial X_m, \partial\lambda_m/\partial U_m)^T d_m$

Proof. This is an immediate result from Lemma 3.1 of [NW99, 40-41] since all functions are continuously differentiable and bounded below. ■

Lemma A.2. If the conditions of Lemma A.1 hold for an agent-m's unconstrained subproblem, the problem of minimizing $\lambda_m(\alpha, X_m, U_m, Y_m)$, then the Wolfe conditions hold in the overall unconstrained problem, the problem of minimizing $\lambda(\alpha, X, U)$.

Proof. First, we show that a feasible improving direction for agent-m's unconstrained subproblem, $d_m = (d_{x_m}, d_{u_m})$, defines a feasible improving direction for the overall unconstrained problem, d , as follows:
 $(\partial\lambda/\partial X_m, \partial\lambda/\partial U_m, \partial\lambda/\partial Y_m, \partial\lambda/\partial Z_m)^T (d_{x_m}, d_{u_m}, 0, 0) = (\partial\lambda_m/\partial X_m, \partial\lambda_m/\partial U_m, \partial\lambda_m/\partial Y_m, \partial\lambda_m/\partial Z_m)^T (d_{x_m}, d_{u_m}, 0, 0) = (\partial\lambda_m/\partial X_m, \partial\lambda_m/\partial U_m)^T d_m < 0$. Thus, it follows that $(\partial\lambda/\partial X, \partial\lambda/\partial U)^T d < 0$ for a suitable d .

Second, we show that the first Wolfe condition holds as follows: $\lambda(\alpha, X_m + \gamma d_{x_m}, U_m + \gamma d_{u_m}, Y_m, Z_m) = \lambda_m(\alpha, Y_m, Z_m) + \lambda_m(\alpha, X_m + \gamma d_{x_m}, U_m + \gamma d_{u_m}, Y_m) \leq \lambda_m(\alpha, Y_m, Z_m) + \lambda_m(\alpha, X_m, U_m, Y_m) + c_1 \gamma (\partial\lambda_m/\partial X_m, \partial\lambda_m/\partial U_m)^T d_m$
 $= \lambda(\alpha, X, U) + c_1 \gamma (\partial\lambda_m/\partial X_m, \partial\lambda_m/\partial U_m, \partial\lambda/\partial Y_m, \partial\lambda/\partial Z_m)^T (d_{x_m}, d_{u_m}, 0, 0)$
 $= \lambda(\alpha, X, U) + c_1 \gamma (\partial\lambda/\partial X_m, \partial\lambda/\partial U_m, \partial\lambda/\partial Y_m, \partial\lambda/\partial Z_m)^T (d_{x_m}, d_{u_m}, 0, 0)$
 $= \lambda(\alpha, X, U) + c_1 \gamma (\partial\lambda/\partial X, \partial\lambda/\partial U)^T d$ for a $d = (d_x, d_u)$ that is consistent with $d_m = (d_{x_m}, d_{u_m})$.

Thus, $\lambda(\alpha, X + \gamma d_x, U + \gamma d_u) \leq \lambda(\alpha, X, U) + c_1 \gamma (\partial\lambda/\partial X, \partial\lambda/\partial U)^T d$.

Third, we show that the second Wolfe condition holds as follows:

$$((\partial\lambda_m/\partial X_m)(\alpha, X_m + \gamma d_{x_m}, U_m + \gamma d_{u_m}, Y_m), (\partial\lambda_m/\partial U_m)(\alpha, X_m + \gamma d_{x_m}, U_m + \gamma d_{u_m}, Y_m))^T d_m \geq c_2 (\partial\lambda_m/\partial X_m, \partial\lambda_m/\partial U_m)^T d_m \Rightarrow$$

$$((\partial\lambda_m/\partial X_m)(\alpha, X_m + \gamma d_{x_m}, U_m + \gamma d_{u_m}, Y_m), (\partial\lambda_m/\partial U_m)(\alpha, X_m + \gamma d_{x_m}, U_m + \gamma d_{u_m}, Y_m), \partial\lambda/\partial Y_m, \partial\lambda/\partial Z_m)^T v \geq c_2 (\partial\lambda_m/\partial X_m, \partial\lambda_m/\partial U_m, \partial\lambda/\partial Y_m, \partial\lambda/\partial Z_m)^T v \text{ for } v = (d_{x_m}, d_{u_m}, 0, 0) \Rightarrow$$

$$((\partial\lambda/\partial X_m)(\alpha, X_m + \gamma d_{x_m}, U_m + \gamma d_{u_m}, Y_m, Z_m), (\partial\lambda/\partial U_m)(\alpha, X_m + \gamma d_{x_m}, U_m + \gamma d_{u_m}, Y_m, Z_m), \partial\lambda/\partial Y_m, \partial\lambda/\partial Z_m)^T v \geq c_2 (\partial\lambda/\partial X_m, \partial\lambda/\partial U_m, \partial\lambda/\partial Y_m, \partial\lambda/\partial Z_m)^T v \Rightarrow$$

$$((\partial\lambda/\partial X_m)(\alpha, X + \gamma d_x, U + \gamma d_u), (\partial\lambda/\partial U_m)(\alpha, X + \gamma d_x, U + \gamma d_u), \partial\lambda/\partial Y_m, \partial\lambda/\partial Z_m)^T v \geq c_2 (\partial\lambda/\partial X, \partial\lambda/\partial U)^T d \text{ for a } d = (d_x, d_u) \text{ that is consistent with } v = (d_{x_m}, d_{u_m}, 0, 0) \Rightarrow$$

$$((\partial\lambda/\partial X)(\alpha, X + \gamma d_x, U + \gamma d_u), (\partial\lambda/\partial U)(\alpha, X + \gamma d_x, U + \gamma d_u))^T d \geq c_2 (\partial\lambda/\partial X, \partial\lambda/\partial U)^T d. \blacksquare$$

A.2 The Proximate-Exchange Protocol with Mutual Help

Conjecture A.1. If:

- 1) Conditions (1) to (5) of Theorem 3.3 hold.
- 2) To solve its problem (P_m), agent-m uses either:
 - a) the iterative barrier algorithm as given in Theorem 3.3, or
 - b) the forthcoming augmented barrier algorithm to update neighboring agent-n's control variables.

The augmented barrier algorithm and the conditions for its use are described below:

- a) Agent-n provides an approximation $\partial\lambda_n^\dagger/\partial U_n$ of how agent-n's objective function varies with changes in U_n . Herein, $\lambda_n^\dagger(\alpha, X_n, U_n, Y_n)$ encompasses the elements of $\lambda_n(\alpha, X_n, U_n, Y_n)$ that neither depend on X_m , nor on U_m . In other words, $\partial\lambda_n^\dagger/\partial U_n$ captures the effects on the overall objective function, $\lambda(\alpha, X, U)$, that agent-m does not "see."
- b) Agent-n provides a positive parameter ε to i) bound the changes on its control variables, ii) guarantee strictly feasibility, and iii) ensure accuracy of the approximation $\partial\lambda_n^\dagger/\partial U_n$. These conditions are formalized as follows.

Let $G_n^\dagger(X_n, U_n, Y_n)$ be the constraints in $G_n(X_n, U_n, Y_n)$ that do not depend on X_m , nor on U_m . Also, let $H_n^\dagger(X_n, U_n, Y_n)$ be the constraints in $H_n(X_n, U_n, Y_n)$ that do not depend on X_m , nor on U_m . Then, the set of "safe" changes in the control variables of agent-n, S_n^\dagger , is defined as:

$$S_n = \{ \Delta U_n \mid G_n^\dagger(X_n, U_n + \Delta U_n, Y_n) < 0 \text{ and } |H_n^\dagger(X_n, U_n + \Delta U_n, Y_n)| < \delta \}.$$

If G_n^\dagger and H_n^\dagger are empty, then set $S_n^\dagger = R^{U_n}$. The parameter $\varepsilon \geq 0$ must be such that:

- i) The second and higher order terms of the Taylor series expansion of $\lambda_n^\dagger(\alpha, X_n, U_n, Y_n)$ can be dropped, that is, $\lambda_n^\dagger(\alpha, X_n, U_n + \Delta U_n, Y_n) \cong \lambda_n^\dagger(\alpha, X_n, U_n, Y_n) + (\partial\lambda_n^\dagger/\partial U_n)\Delta U_n$.
 - ii) The update of U_n is strictly feasible for S_n^\dagger , that is, if $\|\Delta U_n\| \leq \varepsilon$ then $\Delta U_n \in S_n^\dagger$.
- c) Agent-m extends its neighborhood to cover agent-n's neighborhood.
 - d) Agent-m solves the following augmented barrier problem:

$$\text{Minimize } \lambda_m(\alpha, X_m, U_m, Y_m) + \frac{\partial\lambda_n^\dagger}{\partial U_n} \Delta U_n$$

Subject to:

$$\|\Delta U_n\| \leq \varepsilon$$

U_m is unconstrained.

- 3) Agents work serially within each neighborhood.

Then: the overall solution obtained by the agents, $(X^{(i)}, U^{(i)})$, will converge to an optimal solution to the barrier version of problem (P), that is, the problem of minimizing $\lambda(\alpha, X, U) = \{f(X, U) + \alpha B(X, U)\}$.

Bibliography

- [ABB+99] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, "Lapack User's Guide (Third Edition)," Society for Industrial and Applied Mathematics (SIAM), 1999.
- [AMO93] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, "Network Flows: Theory, Algorithms, and Applications," Prentice-Hall, 1993.
- [Bal91] R.E. Balzhiser, "Human Ingenuity: The Ultimate Resource," IEEE Power Engineering Review, Vol. 11, No. 4, pp. 20-23, April 1991.
- [Beasley] J.E. Beasley, "OR-Notes," (<http://mscmga.ms.ic.ac.uk/jeb/or/rights.html>).
- [Ber83] D.P. Bertsekas, "Distributed Asynchronous Computation of Fixed Points," Mathematical Programming, Vol. 27, pp. 107-120, Sep. 1983.
- [Ber95a] D.P. Bertsekas, "Nonlinear Programming," Athena Scientific, 1995.
- [Ber95b] D.P. Bertsekas, "Dynamic Programming and Optimal Control," Volume I, Athena Scientific, 1995.
- [BKM92] A. Brooke, D. Kendrick, and A. Meeraus, "GAMS Release 2.25: A User's Guide," Course Technology, Inc., 1992.
- [BM79] J.A. Bondy and U.S.R. Murty, "Graph Theory with Applications," Elsevier Science Publishing Co., 1979.
- [BO82] T. Basar and G.J. Olsder, "Dynamic Noncooperative Game Theory," Academic Press, 1982.
- [BT96] P.T. Boggs and J.W. Tolle, "Sequential Quadratic Programming," Acta Numerica, Vol. 4, pp. 1-51, 1996.
- [Che98] O.J. Chen, "Integration of Dynamic Traffic Control and Assignment," Ph.D. Dissertation, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, 1998.
- [CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, "Introduction to Algorithms," MIT Press, 1990.
- [CND+00] B.A. Carreras, D.E. Newman, I. Dobson, and A.B. Poole, "Initial Evidence for Self-Organized Criticality in Electric Power System Blackouts," Proceedings of the 33rd Hawaii International Conference on System Sciences, January 2000.
- [Con99] C. Concordia, "Electric Power Systems: Past, Present, and Future," IEEE Power Engineering Review, pp. 7-8, February 1999.
- [CR73a] IEEE Committee Report, "Dynamic Models for Steam and Hydro Turbines in Power System Studies," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-92, pp. 1904-15, 1973.

- [CR73b] IEEE Committee Report, "Common Format for Exchange of Solved Load Flow Data," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-92, No. 6, pp. 1916-25, December 1973.
- [CT97] E. Camponogara and S.N. Talukdar, "A Genetic Algorithm for Constrained and Multiobjective Optimization," Proceedings of the 3rd Nordic Workshop on Genetic Algorithms, August 1997.
- [CT99] E. Camponogara and S.N. Talukdar, "Agent Cooperation: Distributed Control Applications," Proceedings of the International Conference on Intelligent System Application to Power Systems, April 1999.
- [Dav91] L. Davis, "Handbook of Genetic Algorithms," Van Nostrand Reinhold, 1991.
- [DFK91] M. Dyer, A. Frieze, and R. Kannan, "A Random Polynomial-Time Algorithm for Approximating the Volume of Convex Bodies," Journal of the ACM, Vol. 38, No. 1, pp. 1-17, January 1991.
- [DMS90] S.W. Director, W. Maly, and A.J. Strojwas, "VLSI Design for Manufacturing: Yield Enhancement," Kluwer Academic Publishers, 1990.
- [dS93] P.S. de Souza, "Asynchronous Organizations for Multi-Algorithm Problems," Ph.D. Dissertation, Electrical and Computer Engineering Department, Carnegie Mellon University, April 1993.
- [FGK93] R. Fourer, D.M. Gay, and B.W. Kernighan, "AMPL: A Modeling Language for Mathematical Programming," Duxbury Press, 1993.
- [GAT+96] F.D. Galiana, K. Almeida, M. Toussaint, J. Griffin, D. Atanackovic, B.T. Ooi and D.T. McGillis, "Assessment and Control of the Impact of FACTS Devices on Power System Performance," IEEE Transactions on Power Systems, Vol. 11, No. 4, pp. 1931-36, November 1996.
- [GJ79] M.R. Garey and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman and Company, New York, 1979.
- [GL83] G.H. Golub and C.F. Van Loan, "Matrix Computations," The Johns Hopkins University Press, 1983.
- [Glo89] F. Glover, "Tabu Search—Part I," ORSA Journal on Computing, Vol. 1, pp. 190-206, 1989.
- [GMT96] M. Granger Morgan and S.N. Talukdar, "Nurturing R&D in the New Electric Power Regime," IEEE Spectrum, pp. 32-33, July 1996.
- [GPM89] C.B. Garcia, D.B. Pret, and M. Morari, "Model Predictive Control: Theory and Practice – a Survey," Automatica, Vol. 25, No. 3, pp. 335-348, May 1989.
- [Gro96] B.J. Grosz, "Collaborative Systems," AI Magazine, Vol. 17, No. 2, pp. 67-85, 1996.
- [GS93] J.D. Glover and M.S. Sarma, "Power System Analysis and Design," PWS Publishing Company, 1993.
- [Had64] G. Hadley, "Nonlinear and Dynamic Programming," Addison-Wesley, 1964.
- [Hin88] N.G. Hingorani, "Power Electronics in Electric Utilities: Role of Power Electronics in Future Power Systems," Proceedings of the IEEE, Vol. 76, pp. 481-82, 1988.

- [Hin91] N.G. Hingorani, "FACTS - Flexible AC Transmission Systems," Proceedings of the Fifth Int. Conference on AC and DC Power Transmission, London, September 1991.
- [Hin93] N.G. Hingorani, "Flexible AC Transmission," IEEE Spectrum, pp. 40-45, April 1993.
- [HH59] G.W. Housner and D.E. Hudson, "Applied Mechanics Dynamics," Van Nostrand Reinhold, 1959.
- [Hoc97] D.S. Hochbaum, "Tutorial on Approximation Algorithms," technical report, Copenhagen University, 1997.
- [Kun76] H.T. Kung, "Synchronized and Asynchronous Parallel Algorithms for Multiprocessors," Proceedings of a Symposium on New Directions and Recent Results in Algorithms and Complexity, Carnegie Mellon University, April 1976.
- [KW94] P. Kall and S.W. Wallace, "Stochastic Programming," John Wiley & Sons, 1994.
- [Leo94] A. Leon-Garcia, "Probability and Random Processes for Electrical Engineering," Addison-Wesley, 1994.
- [Ley98] S. Leyffer, "User Manual for MINLP_BB," technical report, University of Dundee, April 1998.
- [Lyn96] N.A. Lynch, "Distributed Algorithms," Morgan Kaufmann Publishers, 1996.
- [LZT94] C. Lawrence, J.L. Zhou, and A.L. Tits, "User's Guide for CFSQP Version 2.5," technical report TR-94-16r1, University of Maryland at College Park, Institute for Systems Research, 1994.
- [MF80] J. MacGowan and I.J. Fullerton, "Development and Testing of Advanced Control Strategies in the Urban Traffic Control System," Public Roads, Vol. 43, 1980.
- [Mie99] K.M. Miettinen, "Nonlinear Multiobjective Optimization," Kluwer Academic Publishers, Boston, 1999.
- [Mit97] T.M. Mitchell, "Machine Learning," McGraw-Hill, 1997.
- [MN99] K. Mehlhorn and S. Naher, "Leda: A Platform for Combinatorial and Geometric Computing," Cambridge University Press, 1999.
- [Mos95] E. Mosca, "Optimal, Predictive, and Adaptive Control," Prentice-Hall, 1995.
- [MP93] T.E. Morton and D.W. Pentico, "Heuristic Scheduling Systems," John Wiley & Sons, New York, 1993.
- [MR94] D.C. Montgomery and G.C. Runger, "Applied Statistics and Probability for Engineers," John Wiley & Sons, 1994.
- [MR95] R. Motwani and P. Raghavan, "Randomized Algorithms," Cambridge University Press, 1995.
- [MS83] B.A. Murtagh and M.A. Saunders, "MINOS 5.5 User's Guide," technical report SOL 83-20R, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1983.
- [NN94] Y. Nesterov and A. Nemirovskii, "Interior Point Polynomial Algorithms in Convex Programming," Society for Industrial and Applied Mathematics (SIAM), 1994.

- [NvdS90] H. Nijmeijer and A. van der Schaft, "Nonlinear Dynamical Control Systems," Springer-Verlag, 1990.
- [NW99] J. Nocedal and S.J. Wright, "Numerical Optimization," Springer, 1999.
- [OR70] J.M. Ortega and W.C. Rheinboldt, "Iterative Solution of Nonlinear Equations in Several Variables," Academic Press, New York, 1970.
- [Pat82] B.L. Patridge, "The Structure and Function of Fish Schools," *Scientific American*, pp. 114-123, June 1982.
- [PCG98] M. Prietula, K. Carley, and L. Gasser, editors, "Simulating Organizations: Computational Models of Institutions and Groups," MIT Press, 1998.
- [PM94] M. Pavella and P.G. Murthy, "Transient Stability of Power Systems: Theory and Practice," John Wiley & Sons, 1994.
- [Pra98] R. Pratap, "Getting Started with MATLAB 5: A Quick Introduction to Scientists and Engineers," Oxford University Press, 1998.
- [Pyo85] S.S. Pyo, "Asynchronous Algorithms for Distributed Processing," Ph.D. Dissertation, Electrical and Computer Engineering Department, Carnegie Mellon University, 1985.
- [Ram94] V.C. Ramesh, "Inertial Search and Asynchronous Decompositions," Ph.D. Dissertation, Electrical and Computer Engineering Department, Carnegie Mellon University, 1994.
- [Raw99] J.B. Rawlings, "Tutorial: Model Predictive Control Technology," *Proceedings of the American Control Conference*, June 1999.
- [Ray88] M. Raynal, "Networks and Distributed Computation: Concepts, Tools and Algorithms," MIT Press, 1988.
- [Rey87] C.W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Proceedings of SIGGRAPH'87*, July 1987.
- [Sac98] S. Sachdev, "Explorations in Asynchronous Teams," Ph.D. Dissertation, Electrical and Computer Engineering Department, Carnegie Mellon University, November 1998.
- [Sch87] K. Schittkowski, "More Test Examples for Nonlinear Programming Codes," Springer-Verlag, 1987.
- [SF94] J.M. Swales and C.B. Feak, "Academic Writing for Graduate Students: Essential Tasks and Skills: A Course for Nonnative Speakers of English," University of Michigan Press, 1994.
- [SL91] J.-J.E. Slotine and W. Li, "Applied Nonlinear Control," Prentice-Hall, 1991.
- [SM95] R.L. Scheaffer and J.T. McClave, "Probability and Statistics for Engineers," Duxbury Press, 1995.
- [Syc98] K.P. Sycara, "Multiagent Systems," *AI Magazine*, Vol. 19, No. 2, pp. 79-92, 1998.
- [Tal99a] S.N. Talukdar, "Resource Margins for Coupling Constraints," Personal Communication, Carnegie Mellon University, October 1999.
- [Tal00a] S.N. Talukdar, "Asynchronous Agents: A Chance-Based Relaxation," Personal Communication, Carnegie Mellon University, March 2000.

- [TBG+98] S.N. Talukdar, L. Baerentzen, A. Gove, and P.S. de Souza, "Asynchronous Teams: Cooperation Schemes for Autonomous Agents," *Journal of Heuristics*, Vol. 4, pp. 295-321, 1998.
- [TC00] S.N. Talukdar and E. Camponogara, "Collaborative Nets," *Proceedings of the 33rd Hawaii International Conference on System Sciences*, January 2000.
- [TCZ00] S.N. Talukdar, E. Camponogara, and H. Zhou, "A Design Space for Enterprises," *Proceedings of the 2nd DARPA-JFACC Symposium on Advances in Enterprise Control*, Minneapolis, July 2000.
- [TdS94] S.N. Talukdar and P.S. de Souza. *Insects*, "Fish and Computer-Based Super-Agents," in "Systems and Control Theory for Power Systems," edited by J.H. Chow, P.V. Kokotovic and R.J. Thomas, Vol. 64 of the *Institute of Mathematics and its Applications*, Springer-Verlag, 1994.
- [TF82] IEEE Task Force on Terms & Definitions, "Proposed Terms & Definitions for Power System Stability," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-101, No. 7, July 1982.
- [Tse95] C.K. Tsen, "Solving Train Scheduling Problems Using A-Teams," Ph.D. Dissertation, Electrical and Computer Engineering Department, Carnegie Mellon University, 1995.
- [Whi88] R.M. White, "Powering Responses to the Global Competitive Challenge," *IEEE Power Engineering Review*, Vol. 8, No. 6, pp. 3-5, June 1988.
- [Wid98] M. Widmeyer, "Potential for the Use of Advanced Technology for Control of AC Transmission in the Deregulated Power Industry," *Engineering and Public Policy Department*, Carnegie Mellon University, 1998.
- [Wil78] H.P. Williams, "Model Building in Mathematical Programming," Wiley, 1978.
- [Wol98] L.A. Wolsey, "Integer Programming," John Wiley & Sons, 1998.
- [ZE81] S.H. Zanakis and J.R. Evans, "Heuristic Optimization: Why, When, and How to Use It," *Interfaces*, Vol. 11, No. 5, pp. 84-91, 1981.
- [ZEV89] S.H. Zanakis, J.R. Evans, and A.A. Vazacopoulos, "Heuristic Methods and Applications: A Categorized Survey," *European Journal of Operational Research*, Vol. 43, pp. 88-110, 1989.
- [Zin98] W.K. Zinsser, "On Writing Well: An Informal Guide to Writing Nonfiction," 6th Edition, HarperPerennial, 1998.