IOWA STATE UNIVERSITY
Digital Repository

Graduate Theses and Dissertations

Iowa State University Capstones, Theses and Dissertations

2016

A novel resource sharing algorithm based on distributed construction

Peter Samuel Broen Finzell
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/etd

 Part of the Mechanical Engineering Commons

Recommended Citation
Finzell, Peter Samuel Broen, "A novel resource sharing algorithm based on distributed construction" (2016). *Graduate Theses and Dissertations*. 16053.
https://lib.dr.iastate.edu/etd/16053

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

A novel resource sharing algorithm based on distributed construction

by

Peter Samuel Broen Finzell

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Program of Study Committee:
Kenneth Mark Bryden, Major Professor
Xinwei Wang
Atul Kelkar
Halil Ceylan
Umesh Vaidya

Iowa State University

Ames, Iowa

2016

Copyright © Peter Samuel Broen Finzell, 2016. All rights reserved.

TABLE OF CONTENTS

NOMENCLATURE

i	Index of design surfaces
j	Index of heater surfaces
A	Surface area
F_{i-j}	Exchange factor between heater and design surfaces
N	Number of surfaces
T	Temperature
T^*	Desired temperature
X	Steady state error
Q	Heat transfer
α	Magnitude of scaling factor
σ	Stefan-Boltzman constant

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This thesis develops a novel resource sharing approach for solving inverse radiant enclosure problems based on distributed construction which is then extended to provide a new controls algorithm for complex energy systems. Specifically, the problem of determining the temperature distribution needed on the heater surfaces to achieve a desired design surface temperature profile is recast as a distributed construction problem. This algorithm creates computational agents and the distribution and redistribution of a shared resource (blocks) allows these computational agents to manipulate a shared environment. The sharing of blocks between agents enables them to achieve their desired local state, which in turn achieves the desired global state. Each agent uses the current state of their local environment and a simple set of rules to determine when to exchange blocks, each block representing a discrete unit of change. By gradually building up a solution, this algorithm can be applied to any situation where the output is specified but the required inputs are unclear. In this problem the shared resource is temperature, which is distributed among the various surfaces by computational agents moving blocks. The algorithm is demonstrated using the established two-dimensional inverse radiation enclosure problem. The temperature profile on the heater surfaces is adjusted to achieve a desired temperature profile on the design surfaces. The resource sharing algorithm was able to determine the needed temperatures on the heater surfaces to obtain the desired temperature distribution on the design surfaces in the nine cases examined. This resource sharing algorithm is then extended to two controls problems—a simple physical realization of the radiant enclosure problem and a fuel cell-gas turbine hybrid power system at the National Energy Technology Laboratory's Hyper

facility. In both cases, the controls problem was posed as resource sharing problem, which was solved by incrementally building a solution using distribution construction. The primary strength of the algorithm is the ability to control a system without detailed analysis or development of transfer functions. In the case of the physical realization of the radiant heater problem, the new controls algorithm performed as well as a PID controller without the detailed system knowledge required to setup and tune a PID controller. In the case of the Hyper system, the controls algorithm performed as well as a MIMO controller. Future work will focus on extending the algorithm for load following applications in complex hybrid power systems, which typically require a large number of configurations at different power levels.

.

CHAPTER 1. INTRODUCTION

Advanced power generation facilities are becoming more ubiquitous as more energy is generated from complicated fossil generation facilities, standalone renewable resources and advanced hybrid systems. Control strategies often encounter difficulties in managing these advanced power systems, since they can be composed of disparate and frequently highly coupled components. Simultaneously, advances in sensor technology have increased the capability and intelligence of sensors, making it easier than ever to collect large amounts of data from operating systems (DOE, 2015). Existing control strategies often fail to take advantage of these sensor advances in any meaningful way due to the twin challenges of leveraging sensor intelligence and processing large datasets to yield meaningful control insights. The motivation for this research is to create a new control strategy that can overcome the existing challenges faced by traditional control strategies for advanced energy systems and make use of improvements in sensor technology.

Improving power generation is a prevalent topic in political and economic policy specifically in regards to increasing domestic energy production, decreasing reliance on fossil fuels and reducing emissions while getting greater efficiencies out of existing generation facilities. Traditional power generation facilities, including petroleum based production, natural gas, coal, and nuclear power, operate using well understood thermodynamic cycles, such as the Rankine Cycle, Brayton cycle, etc (Flynn, 2003). Improvements to conventional fossil system have frequently been accomplished by adding on additional processes for recuperation, combining thermodynamic cycles, or making use of unused or underused byproducts. While these improvements can have some effect on lowering emissions by increasing system efficiencies, most

fossil generation facilities reduce emissions primarily by adding or improving upon a cleaning or capture process. These additional and combined processes can make fossil generation systems complicated, but they can still be effectively controlled by traditional methods.

While renewable resources show great promise for reducing the use of fossil fuels, they have challenges with storage and intermittent production. Hybrid facilities combine fossil generation process with one or more renewable energy conversion or energy storage methods (Burch, 2001). Systems combining conversion processes such as gas turbines, fuel cells, wind, or solar generation with thermal storage, electrolysis for hydrogen storage, and battery storage have all been proposed (Bizon et al., 2013). Combining renewable and fossil energy generation or conversion methods offers the potential to increase efficiencies while reducing emissions. However, hybrid systems frequently use new and emerging technologies that are still being actively researched and not fully understood even when operating independently, e.g. fuels cells, supercapacitors or electrolyzers (Bajpai and Dash, 2012). This makes the systems more complex, with disparate components that require very intelligent control strategies for effective control.

Complicated interactions are those that are challenging but possible to explain or understand, while complex interaction are not well understood and may not be possible to fully characterize, at least using conventional approaches (Åström et al., 2012; Bakule, 2008). System coupling occurs when a change to an input has cascading effects throughout several of the systems outputs, with unknown or difficult to characterize consequences (Siljak, 2013).

Highly coupled components can be found in any power generation system with a great number of components or processes, but hybrid systems are much more prone to complex interactions because of their highly disparate components (Bizon et al., 2013). Management of these different components is often accomplished through a supervisory or hierarchical controller

that oversees and coordinates the actions between components (Torres-Hernandez et al., 2008; Valenciaga and Puleston, 2005). At the component level centralized (multivariable) and distributed controllers are typically used, which are considered an industry standard for conventional power generation (Carmeli et al., 2012). When applied to hybrid systems they may become cumbersome to accurately model or cause unwanted and interfering control actions. In these hybrid systems, the challenge is to create a control strategy that optimizes performance under many different and changing conditions for systems composed of many disparate components, operating at very different time scales.

Recently, sensor technology has improved tremendously in terms intelligence within the sensor (Malley et al., 2015). At the same time, sensors have gotten more affordable such that the cost of sensors is small compared to other system components (DOE, 2015). This allows power generation systems to include many additional sensors, which can offer a better understanding of the system behavior at any given time. It is challenging to make effective use of these new sensors for control strategies, especially making full use of their increasing computational power. Conventional control strategies either couple a single sensor and actuator pair or create relationships between many sensors and actuators. A new relationship or equation set must be created for every additional sensor or actuator (Massioni and Verhaegen, 2009). Since these relationships are hard-coded and labor-intensive to derive, they tend to include only the sensors and actuators that provide the most obvious, direct impact on the system, avoiding redundant or less obvious ancillary sensors and actuators. Traditional control strategies are not well suited to make use of intelligent sensors. Rather traditional control architectures create a controller connected to passive sensors and the control decision is made at higher level.

To effectively maximize performance outputs, increase efficiencies, and reduce emissions for advanced energy systems, a new control strategy has been developed. This control strategy has a goal of managing these complex and highly-coupled interactions between disparate components while making effective use of advances in sensor technology. Instead of matching complexity of a system with complexity of a control strategy, this control strategy is straightforward and uses simple rule sets, establishing coordination indirectly, through changes made to the system. Contrary to many other control strategies, this operates more effectively with more sensor information and even redundant information. Computational agents are created and each agent is given the means to manipulate their environment based on a simple rule set. The resource sharing algorithm gradually "builds up" a solution using agents that make changes to their environment using a shared resource, or blocks, which serve as an incremental unit of change. The value or size of these blocks can change and can be varied. This resource sharing control strategy can be applied to any situation where a desired global state is specified, but the inputs required to get to that state are unclear. Each agent is not given specific instructions as to how to achieve their local state but will take blocks from and give blocks back to a repository until their desired local state is achieved. It is through this frequent, independent, and random manipulation of the environment by a group of agents acting independently and the use of a shared resource that the desired global state is achieved. Since these agents act as autonomous controllers, additional sensor information can be added dynamically without significant redesign of the system.

1.1 Dissertation Organization

This dissertation consists of seven chapters with chapters 2 through 6 formatted as journal articles focusing on the methodology and results from this research work.

Chapter 2 discusses the development of the resource sharing algorithm and the initial implementation of the algorithm on a computational model for an inverse radiant enclosure problem. This chapter discusses the development and structural foundation of the algorithm. It also serves as a proof of concept, showing that the algorithm can find a solution to a computational problem. In this problem the two sets of parallel plates are divided into ten heater surfaces and design surfaces and the goal is to maintain a given temperature profile along the design surfaces. The two adjustable parameters, block size and probability of action, were held constant and the performance of the algorithm was evaluated as the initial and final conditions of design surfaces were varied.

Chapter 3 applies the algorithm as a control strategy to a radiant heater test rig, which is a physical realization of the radiant enclosure problem discussed in chapter 2. The control strategy is applied to achieve and maintain a desired temperature profile along the design surfaces by controlling the temperature of the radiant heaters. The resource sharing control strategy is compared to a traditional PI controller where the desired temperature distribution is held constant for both controllers. The setpoint response is evaluated in terms of rise time, settling time and overshoot. This comparison shows the response as the gains of the PI controller are varied and the block size and probability of action are changed.

In chapter 4, the concept of dynamic scaling is introduced, where the probability of action or block size is based on the steady state error between the current value and the desired set point. The effect is the block size or probability of action is greater when the current value is further away from the desired value and decreases as the current value approaches the desired value. The setpoint response for dynamic scaling is evaluated compared to the standard implementation of the resource sharing control strategy to determine which one is preferred for this system. Dynamic

scaling is also introduced as tuning mechanism to determine the optimal values for block size and probability of action using the standard implementation.

Chapter 5 discusses the implementation of the resource sharing control strategy on a gas turbine fuel cell hybrid power system. The system setpoint response is evaluated as the tunable parameters were adjusted to find the optimal values for different operating conditions. Two agents were created; the first agent controls the gas turbine speed by adjusting the electric load, while the second agent controls the cathode mass flow through the fuel cell using the cold air bypass valve. Changes in system response are examined by holding one of the two adjustable parameters constant–probability of action or block size–and varying the other parameter. The resource sharing controller is compared to a previously developed Multi-Input Multi-Output (MIMO) controller that controlled the same design variables and actuators using approximately the same setpoint changes.

Dynamic scaling is introduced to the hybrid system discussed in chapter 6. The hybrid setpoint response is examined by scaling either probability of action or block size. The response is compared using dynamic scaling, the standard implementation, and the MIMO controller, to determine in what conditions one controller would be preferred over another. It is shown that dynamic scaling can be used as the initial implementation of the resource sharing control strategy on a new system, to simultaneous protect sensitive equipment and find the initial values for the standard implementation.

Finally, chapter 7 summarizes the research finding, benefits, and applicability of this new control strategy. This chapter discusses any future areas of research and opportunities where further investigation is needed.

References

Åström, K.J., Albertos, P., Blanke, M., Isidori, A., Schaufelberger, W., Sanz, R., 2012. Control of Complex Systems. Springer Verlag, London.

Bajpai, P., Dash, V., 2012. Hybrid renewable energy systems for power generation in stand-alone applications: A review. Renewable and Sustainable Energy Reviews 16, 2926–2939. doi:10.1016/j.rser.2012.02.009

Bakule, L., 2008. Decentralized control: An overview. Annu. Rev. Control 32(1), 87–98.

Bizon, N., Shayeghi, H., Mahdavi Tabatabaei, N., 2013. Analysis, control and optimal operations in hybrid power systems. Springer Verlag, London. doi:10.1007/978-1-4471-5538-6

Burch, G.D., 2001. Hybrid Renewable Energy Systems. Presented at: U.S. DOE Natural Gas/Renewable Energy Workshops

Carmeli, M.S., Castelli-Dezza, F., Mauri, M., Marchegiani, G., Rosati, D., 2012. Control strategies and configurations of hybrid distributed generation systems. Renew. Energ. 41, 294–305. doi:10.1016/j.renene.2011.11.010

DOE, 2015. Quadrennial Energy Review: An assessment of energy technologies and research opportunities. Available at: http://energy.gov/sites/prod/files/2015/09/f26/Quadrennial-Technology-Review-2015_0.pdf

Flynn, D., 2003. Thermal power plant simulation and control. Institution of Electrical Engineers, London.

Levine, W.S., 2010. The control handbook. CRC Press, Boca Raton.

Maley, S., Rawls, P., 2014. Crosscutting Research Sensors and Controls Project Portfolio.

Massioni, P., Verhaegen, M., 2009. Distributed control for identical dynamically coupled systems: a decomposition approach. IEEE T. Automat. Contr. 54(1), 124–135. doi:10.1109/TAC.2008.2009574

Siljak, D.D., 2013. Decentralized control of complex systems. Dover, Mineola.

Torres-Hernandez, M.E., Velez-Reyes, M., 2008. Hierarchical control of Hybrid Power Systems. In: Proceedings of the 11th IEEE International Power Electronics Congress. pp. 169–176. doi:10.1109/CIEP.2008.4653837

Valenciaga, F., Puleston, P.F., 2005. Supervisor control for a stand-alone hybrid generation system using wind and photovoltaic energy. IEEE T. Energy Conver. 20(2), 398–405. doi:10.1109/TEC.2005.845524

CHAPTER 2. A NOVEL RESOURCE SHARING ALGORITHM BASED ON DISTRIBUTED CONSTRUCTION FOR RADIANT ENCLOSURE PROBLEMS

Peter Finzell, Kenneth M. Bryden[1]
Simulation Modeling and Decision Science Program
Ames Laboratory
1620 Howe Hall, Ames, Iowa, 50011

Abstract

This paper demonstrates a novel approach to solving inverse radiant enclosure problems based on distributed construction. Specifically, the problem of determining the temperature distribution needed on the heater surfaces to achieve a desired design surface temperature profile is recast as a distributed construction problem in which a shared resource, temperature, is distributed by computational agents moving blocks. The sharing of blocks between agents enables them to achieve their desired local state, which in turn achieves the desired global state. Each agent uses the current state of their local environment and a simple set of rules to determine when to exchange blocks, each block representing a discrete unit of temperature change. This algorithm is demonstrated using the established two-dimensional inverse radiation enclosure problem. The temperature profile on the heater surfaces is adjusted to achieve a desired temperature profile on the design surfaces. The resource sharing algorithm was able to determine the needed temperatures on the heater surfaces to obtain the desired temperature distribution on the design surfaces in the nine cases examined.

Keywords: Stigmergy; Multi-Agent Systems; Bio-inspired; Inverse Heat Transfer; Distributed Construction

[1] Corresponding author. tel +15154600875
Email address: kmbryden@iastate.edu

2.1 Introduction

Radiant enclosure problems are encountered in many design problems, for example, in annealing, industrial process ovens, and combustion chambers (Daun and Howell, 2005; Franca et al., 2003). These are problems where radiant heat transfer from several *heater surfaces* is used to establish a specified temperature distribution over several *design surfaces*. These types of problems are concerned with the geometric design of the enclosure or the heater surface inputs to produce a desired temperature profile along the design surface or both (Howell et al., 2000). Radiant enclosure problems require developing a detailed analysis model, which can then be used in the design process. In the case considered here, the design process involves selecting the needed inputs to the heater surfaces to achieve the desired temperature profile along the design surfaces.

Frequently in engineering design, the analysis problem to be solved is an inverse problem, in which the desired outcome is specified and the goal is to specify the inputs required to obtain the desired outcome, i.e. the design variables. For example, in many radiant enclosure design problems the temperatures of the design surfaces are specified and the temperatures of the heater surfaces need to be determined. Direct solutions to these inverse problems, (e.g., using known design surface temperatures to determine the needed heater surface temperature) is not a tractable solution method because there may be multiple solutions that yield the desired design outcome, many or all of which may be physically infeasible, or there may be no solutions. These types of inverse problems are mathematically ill-posed (Kabanikhin, 2011). A problem is considered well posed when the problem is unique, a solution exists, and the solution depends on the data (Hadamard, 1923; Samarskii and Vabishchevich, 2007). The ill-posed nature of these problems makes them sensitive to errors, and small changes in the input may significantly change the output (Özisik and Orlande, 2000).

To overcome this, three approaches are generally taken: trial and error, optimization, and regularization. In trial and error the design engineer uses information from the system, known or desired constraints, and experience to make an educated guess about what inputs are needed to achieve the desired design. That educated guess can be then iteratively refined to achieve the desired design. However, in many cases trial and error is too time consuming and optimization or regularization are used to find the needed design inputs (Hansen, 1998).

In optimization, an objective function *f(x),* is used to minimize the difference between the actual solution and the desired design (Colaço and Orlande, 2006). The objective function incorporates all the design variables and is frequently subject to design constraints, which can either be equality constraints *g(x)* or inequality constraints *h(x)*. The search space is the domain of the objective function *f(x),* and computing the value of the objective function at each iteration, given that it satisfies the constraints, is called a feasible solution (Vanderplaats, 1984). Depending on the nature of the desired solution either the design variables or the constraints can be varied until the objective function is sufficiently minimized to find an acceptable solution. This can be expressed as

$$\min f(x) \tag{2.1}$$

is subject to

$$g(x) = c \tag{2.2}$$

and

$$h(x) \leq d \tag{2.3}$$

When optimization is used to solve radiant enclosure problems, the objective function takes the form of the equation below. The value of the objective function is found by computing the variance for desired temperature or heat flux profiles on the design surfaces.

$$f = \frac{1}{N}\sum_{i=1}^{N}\left[T_i - T_i^*\right]^2 \tag{2.4}$$

Where N is the number if surfaces, T is the current temperature of each design surface and T^* is the desired temperature and f is the value of the objective function. Derivative-based optimization methods move iteratively through the search space, using derivative information to guide the search (Daun and Morton, 2003b; Federov et al., 1998). This procedure is repeated until the stopping criteria have been reached, e.g., a specified number of time steps or the objective function has been minimized below a certain value (Mera et al., 2003). These methods include conjugate gradient, Newton-Rapson, golden section, and steepest descent, which have all been used to solve radiant enclosure problems (Daun et al., 2004; Duan and Howell, 2005; Ertürk et al., 2002b; Park and Yoon, 2000). Heuristic optimization methods have also been used in radiant enclosure problems and include: genetic algorithms, tabu search, simulated annealing and particle swarm optimization (Amiri et al., 2011; Porter et al., 2006; Safavinejad et al., 2009). Heuristic optimization often requires a large amount of sampling of the search space, which means the search may take longer.

Regularization attempts to make the ill-posed portion of the problem tractable at the expense of accuracy. Finding an accurate and stable solution means striking a balance by reducing the fluctuations associated with the ill-posed nature of the problem without producing an over smoothed solution (Tikhonov and Arsenin, 1977; Bakushinsky and Goncharsky, 1994). Regularization techniques, such as truncated singular value decomposition, modified truncated singular value decomposition, and Tikhonov regularization have been used to solve radiant enclosure problems using different geometric configurations and initial conditions (Howell et al., 2000; Daun et al., 2003a; Ertürk et al., 2002a). Singular value decomposition (SVD) is an algebraic manipulation wherein a matrix of known parameters (A) is broken into three linearly independent

matrices, an orthonormal unitary matrix, an orthonormal conjugate transpose matrix, and a diagonal matrix of singular values. This matrix of singular values are used to determine how invertible the matrix is and if the matrix is well posed or ill posed. Truncated singular value decomposition and modified truncated singular value decomposition are based off of singular value decomposition. By truncating some of these singular values, the matrix becomes well posed, and a realizable solution can be found. Modified truncated singular value decomposition adds a correcting term for the remaining singular values and corresponding singular vectors (Franca et al., 2003). Tikhonov's regularization procedure attempts to reduce unstable effects by adding smoothing terms to the least squares equation (Tikhonov et al., 1995). These methods regularize the system by minimizing the residual and as such are resistant to errors in input data.

In this paper we describe a novel approach to solving a radiant enclosure design problem by posing it as a resource sharing problem which can be solved using distribution construction. Specifically, the distribution and redistribution of a shared resource (temperature) by computational agents allows them to manipulate a shared environment. Unlike optimization, where all of the design variables are used in a single objective function, each agent acts independently and observes a single design variable. Each agent will continue to take action until their local state is met, and when all of the agents meet their local state, the desired global state is met as well.

2.2 Background

Biological systems have been used as the source of inspiration for many optimization algorithms, e.g., particle swarm optimization, genetic algorithms, and ant colony optimization (Yang et al., 2013). Self-organization is an area of study, that draws its origins in biological systems and seeks to establish how organisms can react, adapt, and interact with their environment

and each other to create macroscopic level behaviors from microscopic interactions (Halley and Winkler, 2008). Flocking behavior of birds and schooling behavior of fish are both examples of how order can spontaneously be created from disorder and how global behaviors are created from local interactions (Camazine et al., 2003).

Stigmergy, like self-organization, uses local information and simple instruction sets to make decisions without global guidance. Stigmergy is how social insects (e.g., bees, ants, termites and wasps) coordinate their behaviors based on indirect communication methods. Coordination is established by making small changes to the insects' environment, which other insects can interpret, and triggering actions or responses, which further alter their environment (Bonabeau et al., 1997). Actions reinforce each other and can lead to the construction of complex structures without the need for direct communication between the individuals or a centralized coordinator. Insects use only local information and simple instruction sets to solve complex problems based on emergent behavior. The stigmergic construction processes shown by social insects have been studied extensively. For example, papers wasps and African termites and their construction methods have been modeled (Karsai, 1999; Bonabeau, 1998; Theraulaz and Bonabeau, 1995a; Downing and Jeanne, 1988). This construction process has inspired the development of a number of computational techniques (Petersen et al., 2011; Feltell et al., 2005; Bullock et al., 2012). Construction of paper wasp nests involves wasps gradually depositing their own building material based only on the current state of the nest (Karsai and Penzes, 1993). Each wasp evaluates the current state of construction and based on a simple set of rules determines where to build. In the same way African termites build complicated colonies with complex systems for heating, ventilation, cooling, and separate chambers for nurseries and farming (Theraulaz et al., 1998). Each termite will perform a task based on their preference, experience, and abilities (Bonabeau et

al., 1999). Termites operate continuously and efficiently without a centralized coordinator or a task list. Each worker is allowed to contribute and although they use roughly the same rules, slight differences in preferences and abilities leads to emergent behavior.

The computational application of stigmergic construction is *distributed construction*. In distributed construction *Agents* are independent actors, i.e., computational insects. These agents can sense changes to their environment and based on simple rule sets, manipulate uniform construction materials or blocks to change their environment (Theraulaz and Bonabeau, 1995b). Distributed construction has been used to examine the applicability of stigmergic processes to the construction of complex two- and three-dimensional structures and lattices using simple blocks (Ladley and Bullock, 2005; Bonabeau et al., 2000). No one agent has any knowledge of the final goals but only an understanding of their current local conditions. If each individual agent meets their goal, the global goal will be attained without any agent having complete knowledge of the system. Additional agents can contribute if needed or leave without consequence, allowing these systems to be distributed, robust, and scalable. The common themes in distributed construction processes are

- Agents perform tasks using a shared resource in a shared environment.
- Each agent is given the means to manipulate their environment.
- Actions taken are based on some set of simple instructions and the current state of the environment.
- The resource may or may not be conserved.

2.2.1 Radiant Heater Enclosure Design Problem

The resource sharing algorithm developed here will be demonstrated using the well-studied two-dimensional inverse radiation enclosure problem with two parallel surfaces separated by a given distance. As implemented here, the goal is to achieve a desired spatial temperature profile on the design surfaces by adjusting the temperature profile along the heater surfaces (Fig. 1). The spatial temperature profile of the design surfaces $i_1 \ldots i_n$ is the desired output, and the spatial temperature profile of the heater surfaces $j_1 \ldots j_n$ is the input. The temperatures and radiative heat flux on design surface are known, but the temperatures needed on the heater surfaces are unknown, making this problem ill-posed. Since this is a radiant enclosure problem, the radiation heat transfer from the heater surfaces will either be transferred to the design surfaces or to the surrounding environment. The environment can be thought of as another heater surface with a constant temperature of 26°C. Similar inverse radiant enclosure problems have been examined in (Daun et al., 2002; Franca et al., 2003; Howell et al., 2000). For the purpose of this problem, the surfaces are assumed to be perfectly insulated and the only external losses are radiant losses to the environment. Radiation is assumed to be the only mode of heat transfer, and all surfaces are two dimensional and black so the transferred thermal radiation is perfectly absorbed and emitted. The temperature is uniform across each heater surface and design surface. The equation for a single design surfaces is then.

$$Q_i = A_i \sigma \sum_{j=1}^{N} F_{i-j} (T_i^4 - T_j^4) \tag{2.5}$$

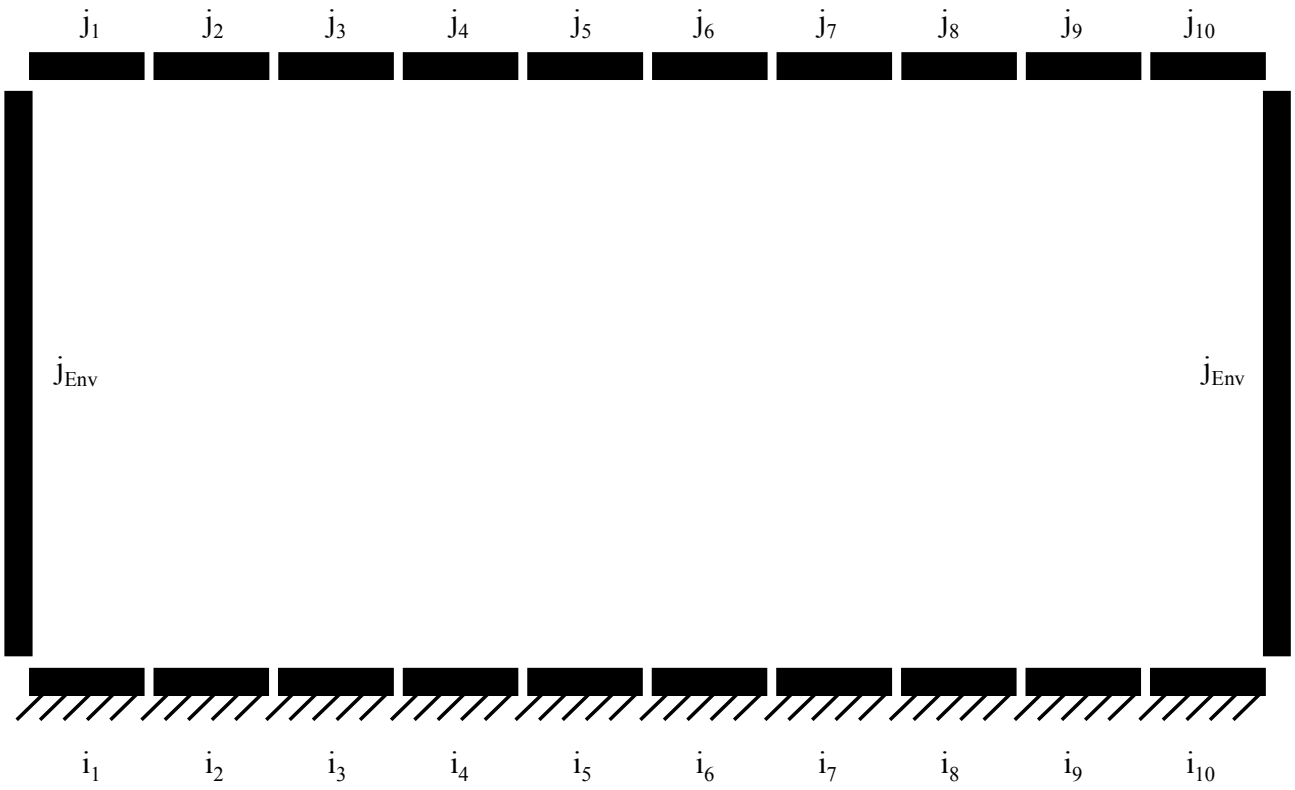


Figure 2.1. Radiation heat transfer between heater surfaces, design surfaces and the surrounding environment.

This problem is subject to several constraints because it was designed to simulate conditions faced in a physical system, which restricts the temperature range of the heater and design surfaces. The constraints are

$$T_i \geq 26°C \tag{2.6}$$

$$T_j \geq 26°C \tag{2.7}$$

$$T_j \leq 300°C \tag{2.8}$$

2.3 Algorithm Setup

Previous implementations of distributed construction use mobile agents, either computational agents or autonomous robots (Beckers et al., 1994; Holland and Melhuish 1999; Petersen et al., 2011). Each agent is given a very simple set of rules, a uniform building material (blocks), and a means to manipulate their environment using blocks. The agents move about their environment, positioning these blocks according to the current state of the environment and their own rule set. Actions among agents do not need to be directly coordinated because the environment serves as a means for indirect coordination. For example, as shown in Fig. 2.2, the red agents are constructing a grid by evaluating the current state of construction, applying their rule sets to determine where to place the next block, and taking action by placing a block in the appropriate location.

This algorithm uses the same methodology except the agents are stationary. The agents still use the current state of their environment and a simple set of rules to place blocks. Although the changes made by the agents effect their global environment, each agent makes decisions based on local information from a single design surface. In this instance each block is a representation of a single unit of temperature change to the heater surface. Each agent senses the current state of their local environment, i.e. a single design surface, and takes action to change that environment, based on a simple rule set, by transferring blocks to or from a single heater surface, shown in Fig. 2.3.

The algorithm developed here has three primary parts—the agents, the blocks, and the block repository. As stated previously, each block represents a single unit of temperature change to the heater surface. The value or size of these blocks can change and is called the block size. The creation, distribution and redistribution of blocks is facilitated by the block repository, which is not a controller and has a very limited amount of information at any given time. The only knowledge the repository has is the number of agents in the system, how many of those agents are requesting blocks, and how many blocks are available at any given time.

During each time step, each agent sends their current state to the block repository based on if their design surface is above, below, or at the desired setpoint (Fig. 2.4). The block repository determines whether it needs to create more blocks based on how many block requests are currently being made compared to the total amount of agents. If the number of agents requesting blocks is more than half of the total number of agents, the block repository begins to distribute new blocks. If half or more of the agents are at or above their setpoint, the block repository only distributes blocks that have been given back. An agent is considered to be at its setpoint when the current value is within a user specified tolerance of the desired value. At each time step two random numbers are created, one to randomly select an agent out of the group and one to determine if that randomly selected agent can take an action based on a user defined probability. This random probability of an action being taken ensures that not every agent requesting a block or attempting to return a block is allowed to do so immediately. This helps ensure that the behavior of the system is emergent, and safeguards against unintended imposition of predefined behaviors. The probability of action can be raised or lowered based on how often the user wants action to be taken.

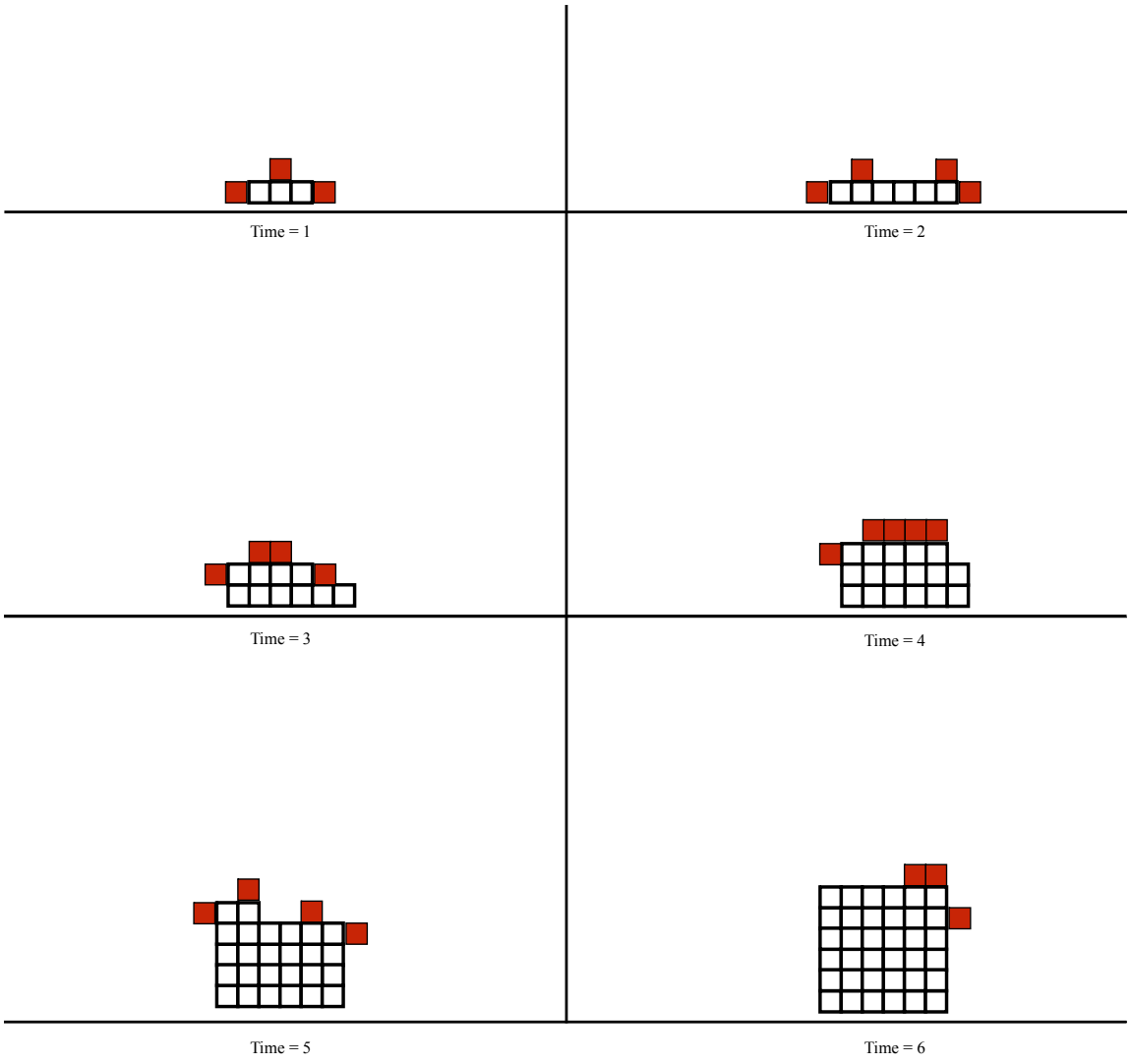


Figure 2.2. Agent-based distributed construction of a grid.

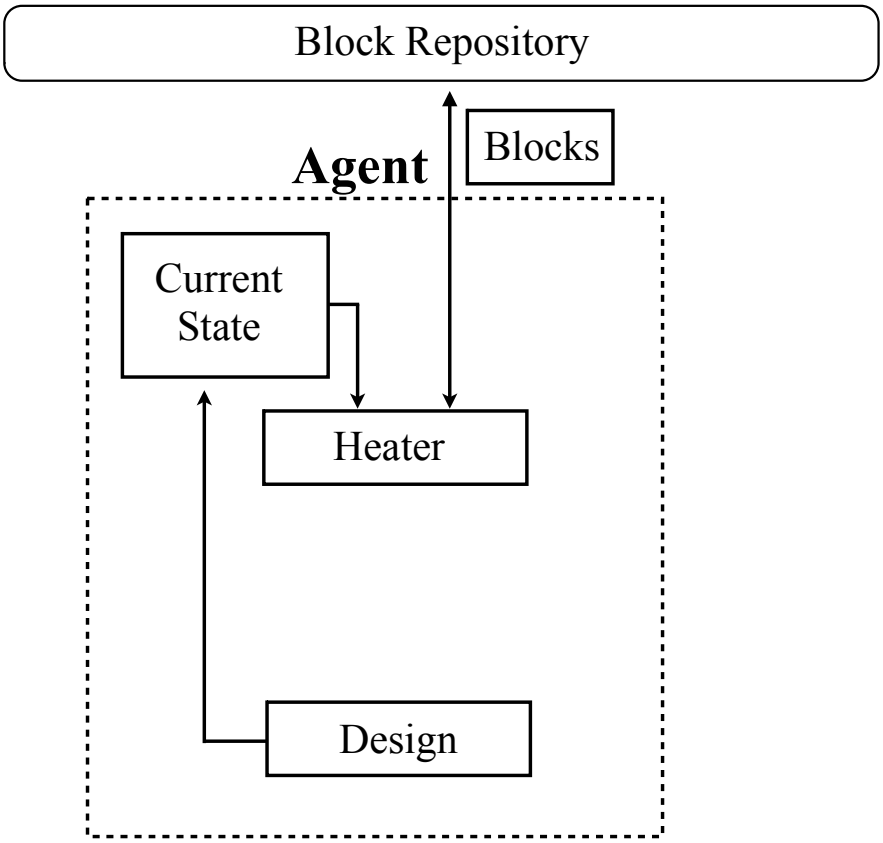


Figure 2.3. Agent composition.

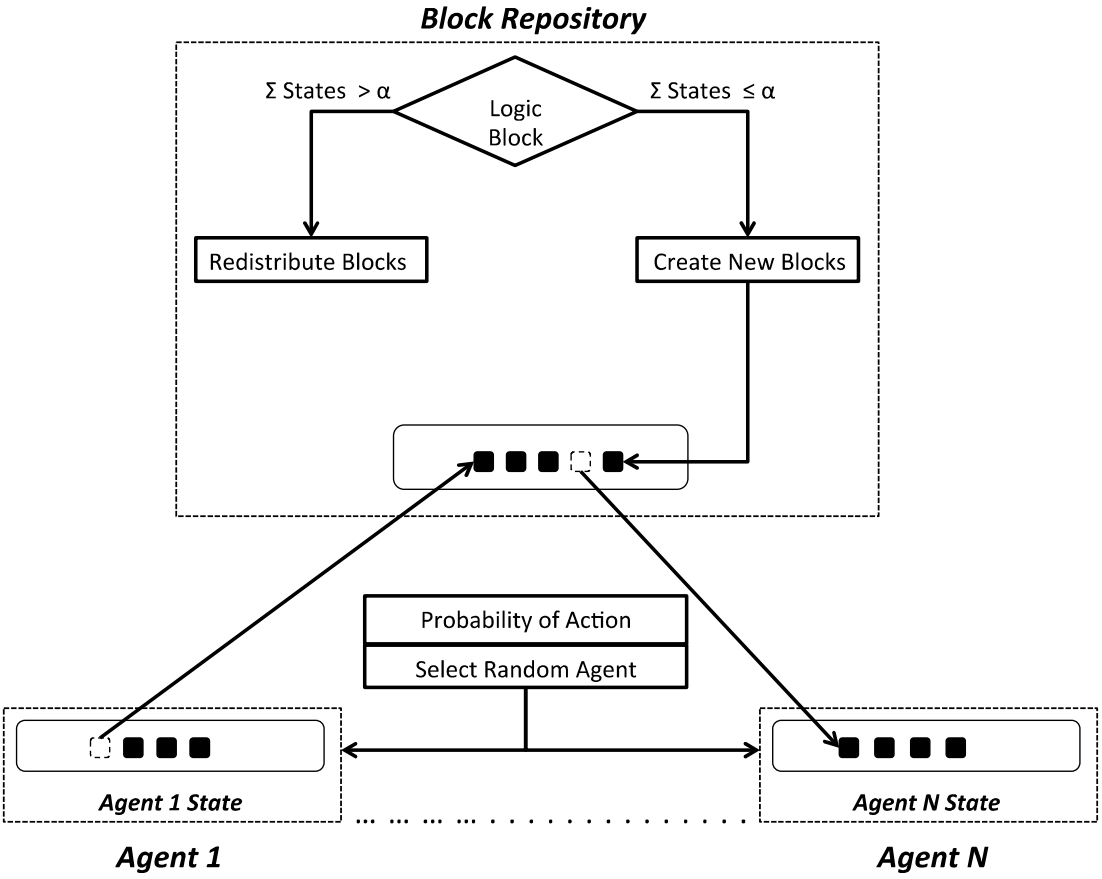


Figure 2.4. Agent and block repository interactions.

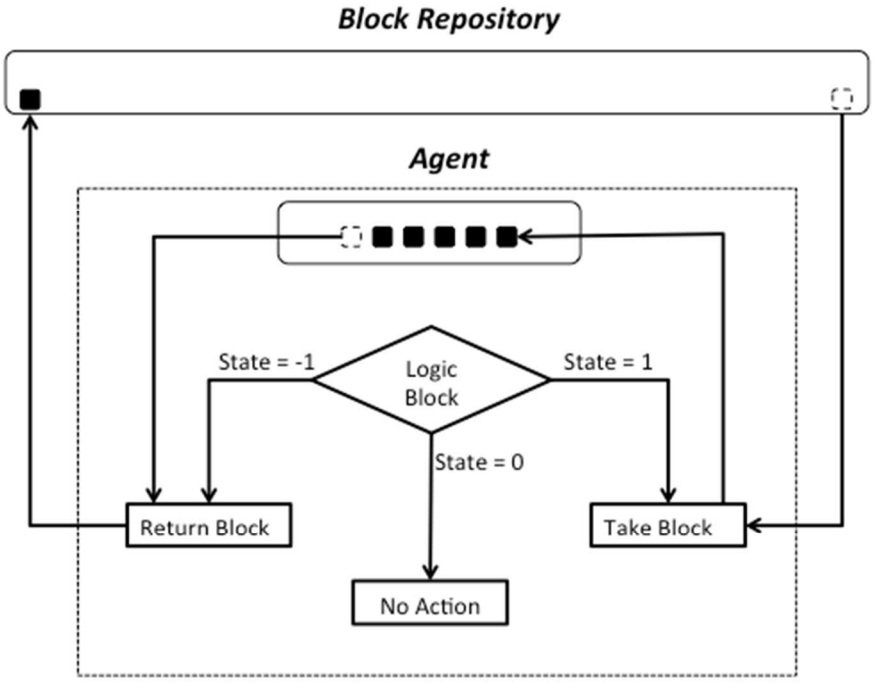


Figure 2.5. Agent rule set.

If an agent is selected and allowed to make a decision, they have the choice to take a block, give back a block, or take no action. This is shown in Fig. 2.5. The rules governing the behavior of the agents are

- Each agent is only capable of detecting the current state in their zone.
- There are three current states—high, low, and as desired.
- If the current state is less than the desired state, the agent requests additional blocks.
- If the current state is greater than the desired state, the agent returns blocks.
- If the current state is at the desired state, the agent takes no action.

Consider the enclosure shown in Fig. 2.6; the goal is to find the required temperature for each heater surface that will produce the desired temperature distribution along the design surfaces. Because each heater surface will have some effect on each design surface, any change in a single heater surface temperature will have some effect on all of the design surfaces. In this case temperature, like building blocks used in distributed construction, becomes the discrete unit of change, which is used to build the solution over time. These building blocks are not a conserved quantity, but can be created as needed, distributed and redistributed among agents as the temperature distribution along design surfaces approaches the desired temperature distribution.

As stated previously, each agent's local environment is a single design surface and uses a decision protocol that decides when to request or return blocks (discrete temperature units). The problem as posed here is then composed of ten agents and one block repository. The problem is:

- Each of the design and heater surfaces is set to an initial temperature.
- Each of the agents is provided with the desired temperature (setpoint) for their assigned design surface.

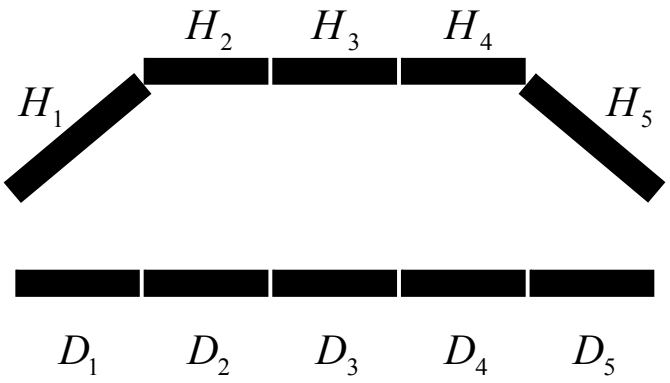


Figure 2.6. Radiant enclosure example.

- The block repository is initialized with requests from agents for additional blocks. If more than half of the agents are requesting blocks, the block repository begins to create blocks and distribution occurs. If more than half of the agents are at or above their setpoint, redistribution occurs and the block repository will temporarily store blocks to be redistributed.

To better understand the algorithm, consider the example shown in Fig. 2.7. Initially (iteration 0) the block repository receives requests from all the agents because none of them are at their setpoint (50°C). At iteration i the block repository begins to randomly distribute blocks, which are randomly accepted by all the agents that requested them. As the agents acquire more blocks, the temperature of each heater surface increases and the temperature of the design surfaces begins to rise. Iteration j shows the results of multiple iterations and the subsequent temperature rise. At some point one or more of the agent's design surface temperatures will exceed their setpoint and they will begin to give blocks back, as shown at iteration k. Once half of the agents have exceeded their setpoint, the block repository will no longer generate additional blocks, and it is left to the agents to redistribute the blocks accordingly (iteration m) until all of the states are met (iteration n).

2.4 Results

Nine test cases were used to evaluate the resource sharing algorithm. Before the inverse problem was solved, the forward problem was solved to determine the radiative flux on the design surfaces. As previously mentioned, in every test case examined, the radiative heat flux on the design surfaces and the desired design surface temperature profile are specified.

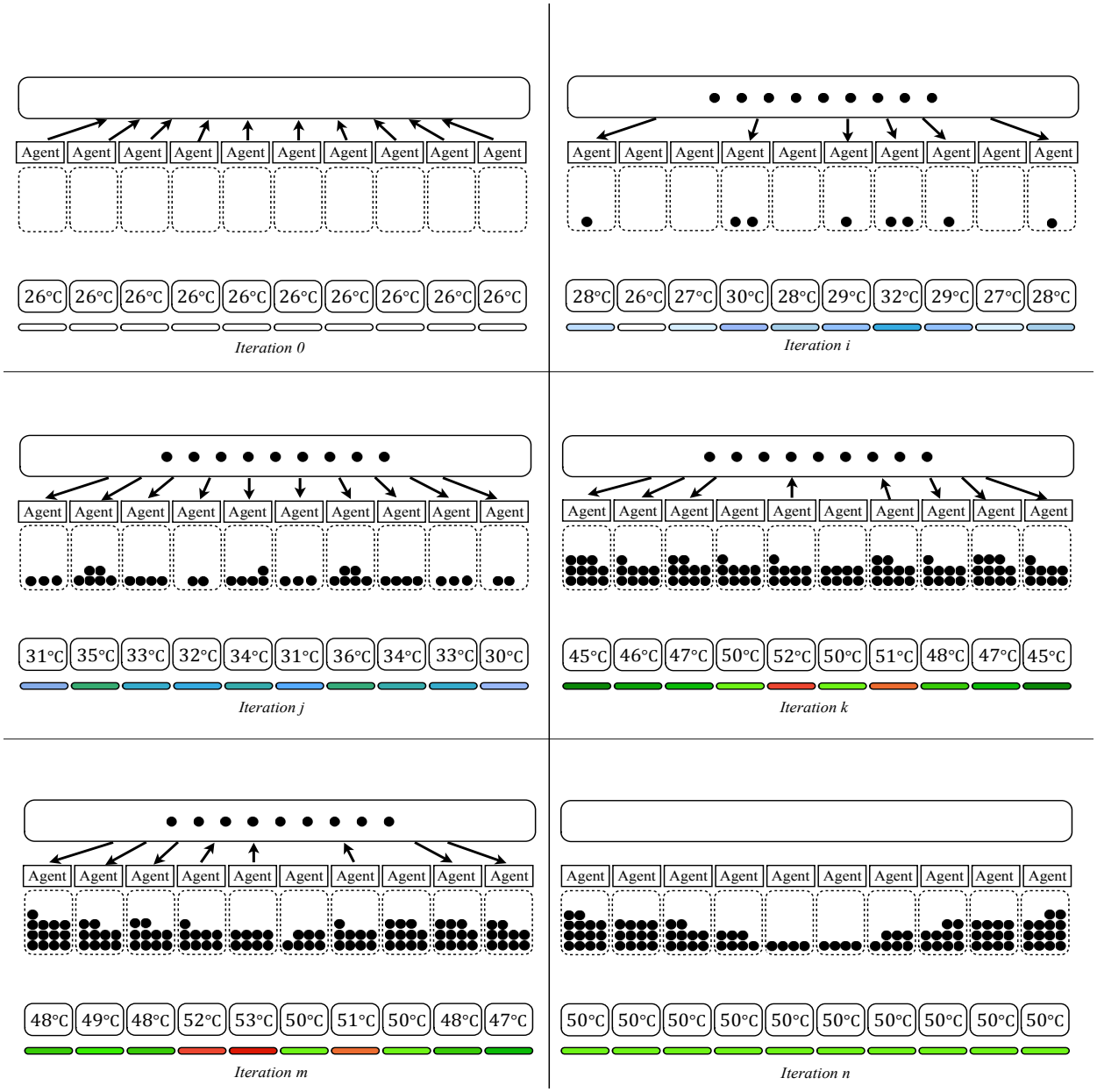


Figure 2.7. Resource sharing algorithm example.

The goal in each test case was to determine the heater surface temperatures needed to reach the desired design surface temperature profile. The initial temperature of the design surfaces was set to 26°C and a setpoint temperature of 50°C was used for all of the design surfaces. Each test case was repeated 100 times and each test was run for 25,000 iterations. The final heater surface temperature profiles were averaged for 100 cases and the design surface temperatures were averaged at every iteration for 100 cases.

The first four test cases were used to examine a problem similar to one found in (Howell et al., 2000) and were used to compare to a known result. The tolerance, block size and probability of action were tuned beforehand to give good results and were set to ±1°C, 0.2°C and 20% respectively. Using a setpoint temperature of 50°C for the design surfaces, the surrounding environment temperature was set to 75°C and the heater surfaces were set to 100°C. Once the radative heat flux for the design surfaces was obtained using these conditions, the inverse problem was then solved to verify that those conditions would produce a uniform heater surface profile of 100°C. Using the same radiative heat flux found when the surrounding environment was 75°C, test cases 1-4 lowered the surrounding environmental temperature from 70°C to 55°C and the resulting heater temperature profile was solved using the resource sharing algorithm. As the environmental temperature approached 55°C, the heater surface profile became more parabolic, with more pronounced temperature differences between the inside and outside surfaces, with the outside surface being hotter than the inside surfaces. These result compares very well with the results found in (Howell et al., 2000). The 95% confidence interval bars are shown for each heater surface in Fig. 2.8-11. Larger bars are associated with larger variability in the mean heater surface temperature.

The next set of test cases used the conditions used in test case 4 to examine the effects of

tolerance and block size on the resource sharing algorithm. The final averaged temperature distributions on the heater surfaces and their respective 95% confidence intervals for test cases 4-9 are given in Table 2.1. The final averaged design surface temperature profiles and their associated error from their setpoint values are given in Table 2.2. The design surface temperature response for test cases 4,6,8 and 9 over the course of 25,000 iterations are given in Figs. 2.12,2.15,2.17 and 2.18, while test cases 5 and 7 required more iterations to converge and are shown in Figs. 2.13 and 2.16. Test cases 4-6 examine how the tolerance affects the response, while test cases 6-9 examine the response when the block size is changed.

2.4.1 Tolerance

In previous methods for solving inverse problems a regularization parameter determines the extent of regularization of the solution. The selection of this parameter involves a trade-off between minimizing oscillations, which can be associated with unstable solutions, and maintaining a certain level of accuracy without over-regularizing the solution. Oscillations are undesirable and depending on the level of oscillations, can lead to unstable or non-physical solutions. A solution is over-regularized when the solution stops oscillating but resulting solution is too inaccurate.

In this instance the regularization parameter is the tolerance, which is used by each agent to determine when the current value is close enough to the desired value to cease taking action. A very small tolerance means that agents will continue to take action despite being very close to the desired solution, which has the potential to produce oscillations and unstable solutions. If the tolerance is very large then the variations of the design surface temperatures between the

Table 2.1. Heater surface temperature profiles for test cases 4–9 (°C).

Element		1	2	3	4	5	6	7	8	9	10
Case 4	Final	142.4	119.5	99.5	98.0	91.9	92.5	98.1	99.1	119.7	142.4
	95% CI	0.24	0.45	0.48	0.50	0.62	0.58	0.54	0.50	0.47	0.26
Case 5	Final	149.4	121.4	86.8	95.0	95.1	96.0	95.2	86.3	121.4	149.6
	95% CI	0.82	0.82	0.82	0.54	0.70	0.72	0.55	0.75	0.82	0.76
Case 6	Final	135.3	115.6	108.1	100.6	93.6	93.3	100.7	108.2	115.4	135.4
	95% CI	0.31	0.49	0.49	0.50	0.74	0.72	0.44	0.40	0.52	0.36
Case 7	Final	142.3	119.7	99.1	98.6	91.8	92.1	98.4	99.4	119.5	142.3
	95% CI	0.16	0.29	0.32	0.37	0.39	0.47	0.42	0.33	0.36	0.18
Case 8	Final	142.9	120.0	98.4	97.1	92.3	93.6	96.7	99.2	119.3	143.0
	95% CI	0.44	0.81	0.79	0.82	0.81	0.68	0.85	0.82	0.73	0.37
Case 9	Final	143.1	120.4	98.6	95.9	92.5	92.1	97.8	97.8	119.7	143.4
	95% CI	0.61	1.07	1.07	1.01	1.27	1.07	1.27	1.12	1.13	0.60

Table 2.2. Design surface temperature profiles for test cases 4–9 (°C).

Element		1	2	3	4	5	6	7	8	9	10
Case 4	Final	49.02	50.57	50.97	50.66	50.29	50.29	50.66	50.97	50.56	49.02
	95% CI	0.98	0.57	0.97	0.66	0.29	0.29	0.66	0.97	0.56	0.98
Case 5	Final	49.02	50.57	50.97	50.66	50.29	50.29	50.66	50.97	50.56	49.02
	95% CI	0.98	0.57	0.97	0.66	0.29	0.29	0.66	0.97	0.56	0.98
Case 6	Final	48.52	50.47	51.36	51.46	51.31	51.31	51.46	51.36	50.47	48.52
	95% CI	1.48	0.47	1.36	1.46	1.31	1.31	1.46	1.36	0.47	1.48
Case 7	Final	49.01	50.57	50.98	50.68	50.31	50.31	50.68	50.98	50.57	49.01
	95% CI	0.99	0.57	0.98	0.68	0.31	0.31	0.68	0.98	0.57	0.99
Case 8	Final	49.04	50.56	50.94	50.61	50.24	50.24	50.63	50.95	50.56	49.04
	95% CI	0.96	0.56	0.94	0.61	0.24	0.24	0.63	0.95	0.56	0.96
Case 9	Final	49.08	50.56	50.90	50.53	50.12	50.13	50.54	50.90	50.56	49.07
	95% CI	0.92	0.56	0.90	0.53	0.12	0.13	0.54	0.90	0.56	0.93

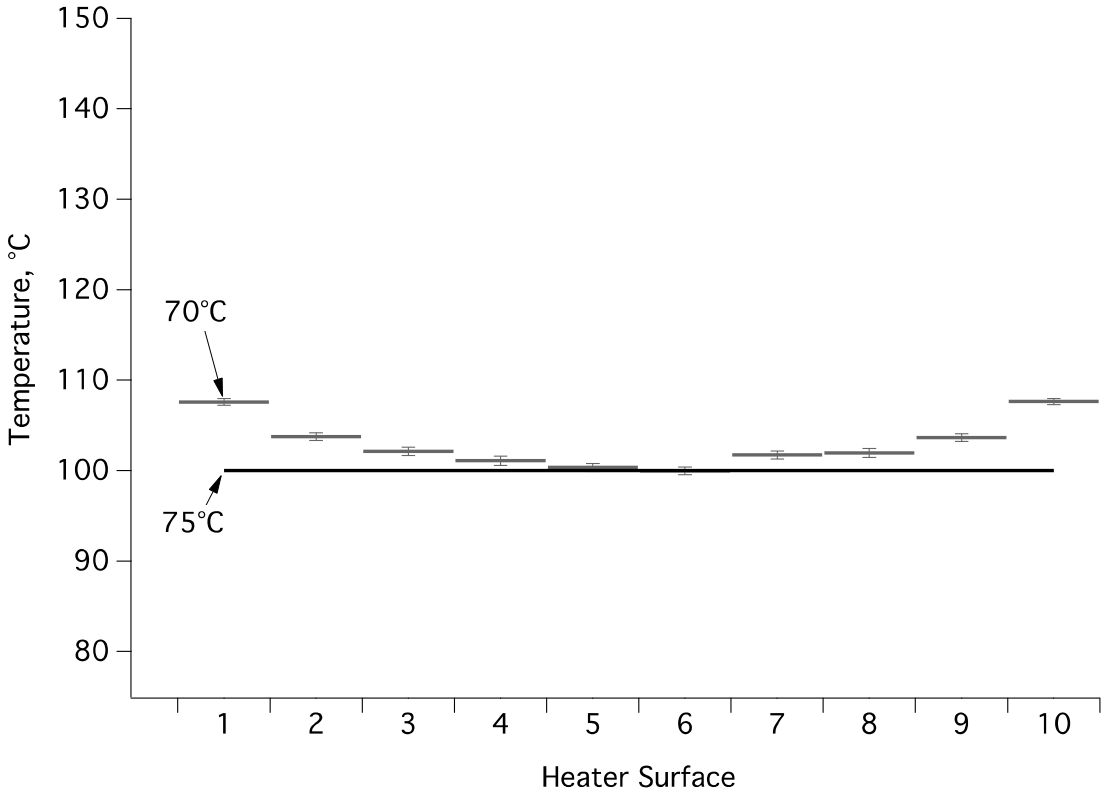


Figure 2.8. Case 1 - Heater surface temperature profile with surrounding environment at 70°C.

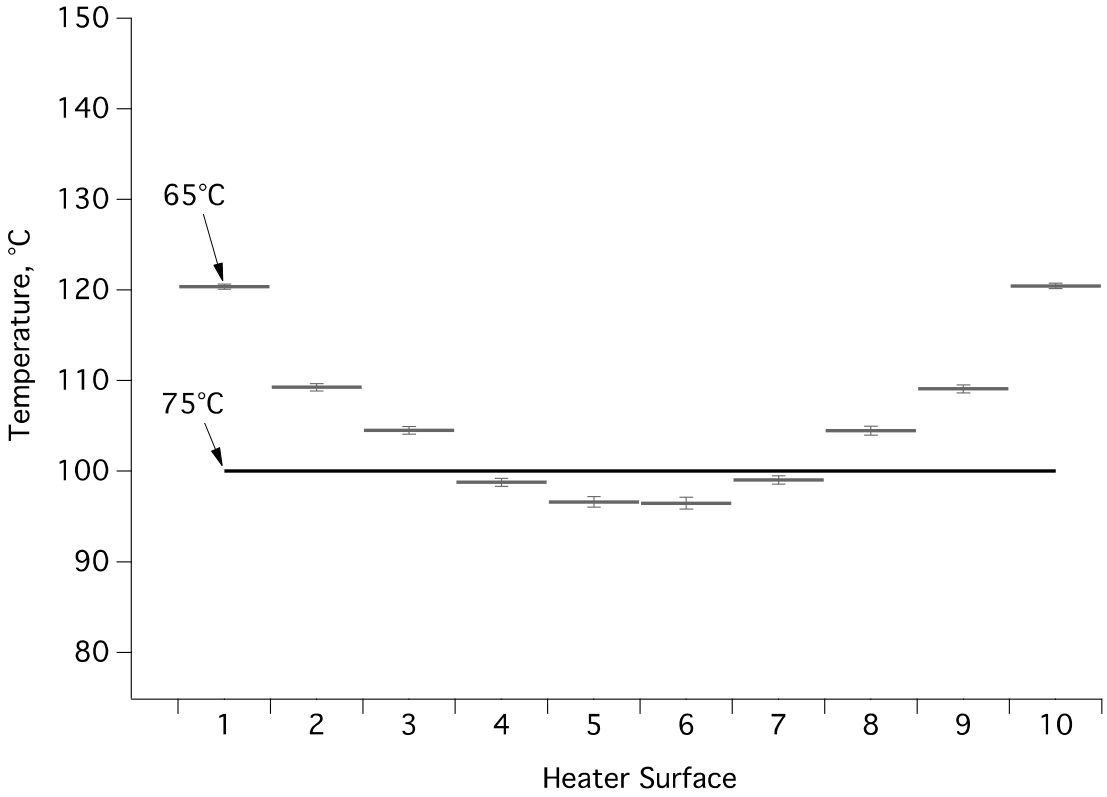


Figure 2.9. Case 2 - Heater surface temperature profile with surrounding environment at 65°C.

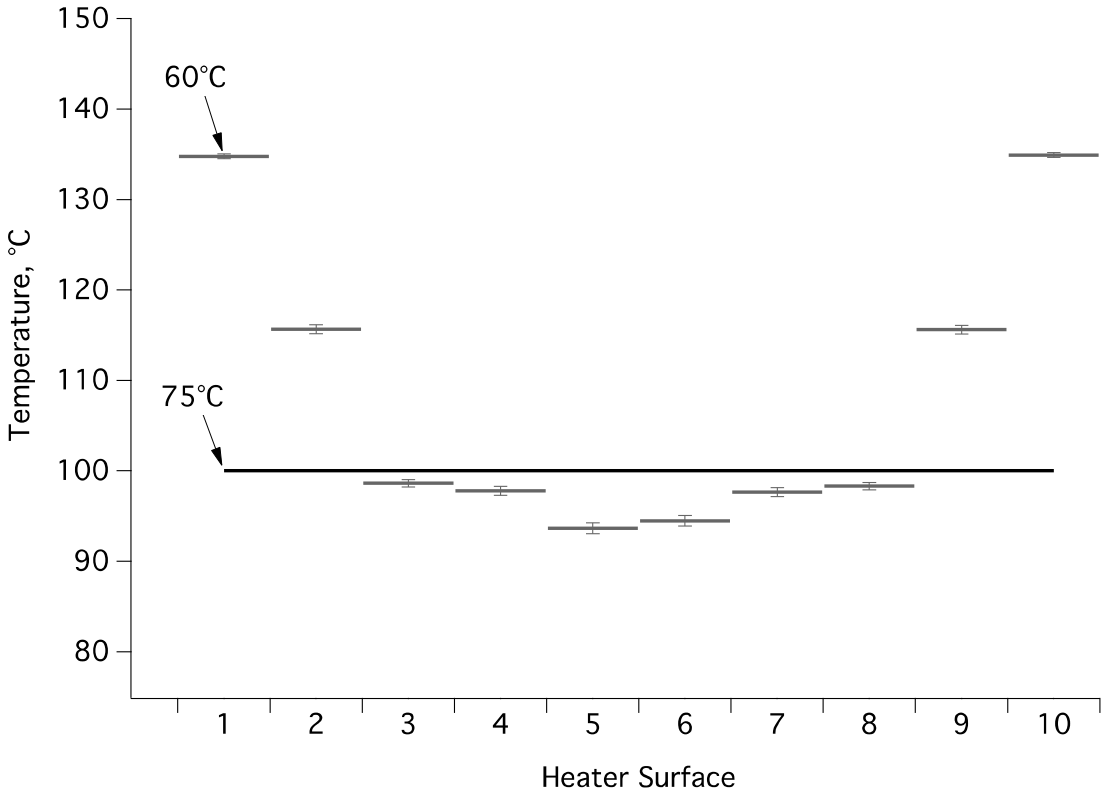


Figure 2.10. Case 3 - Heater surface temperature profile with surrounding environment at 60°C.

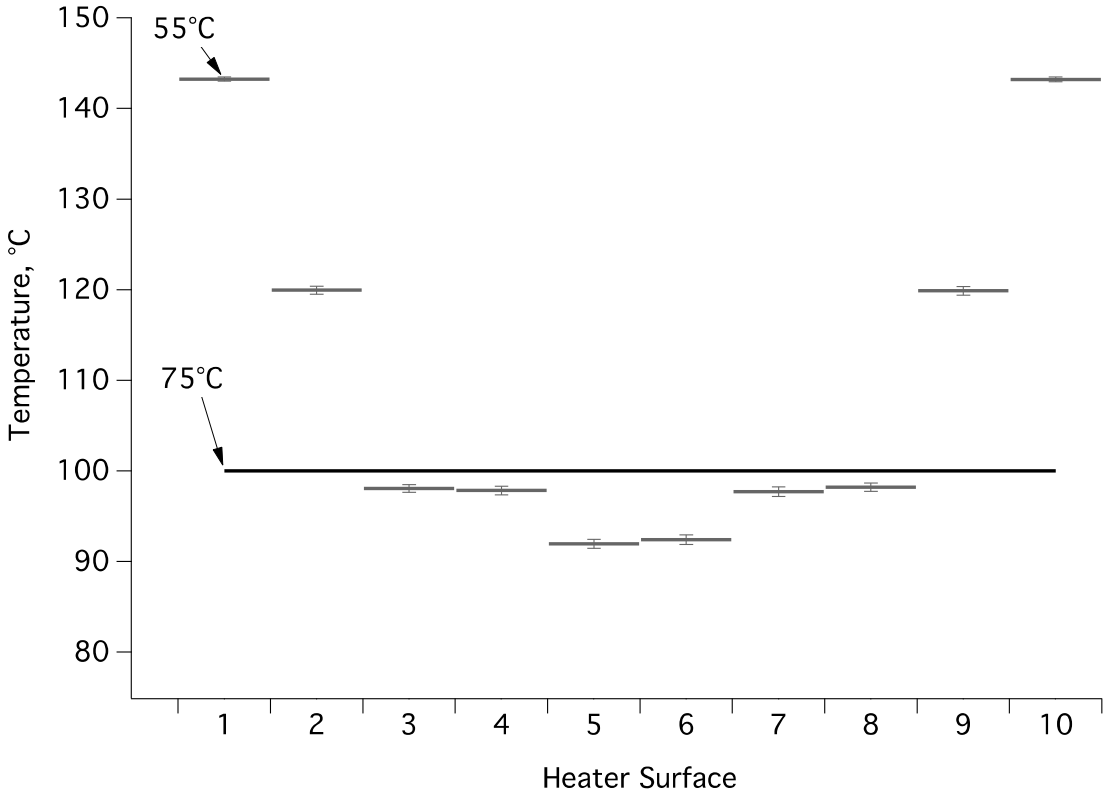


Figure 2.11. Case 4 - Heater surface temperature profile with surrounding environment at 55°C.

tolerance bands are minimized but the accuracy of the desired solution decreases. An ideal profile for the heater surfaces is one that is smooth and parabolic, while still keeping the temperature profile along the design surfaces as close to the setpoint temperature as possible.

Test case 4 was used as the base case for comparison and used a tolerance of ±1°C and a block size of 0.2°C. In this case, as shown in Fig. 2.12, all of the agents achieved their setpoint temperatures within the tolerance band of ±1°C in approximately 23,000 iterations. Once they reached their setpoint temperatures, all of the agents stayed within their designated tolerance band and did not move between their tolerance bands. The resulting heater surface profile is smooth and parabolic with minimal error in the design surface temperatures and small confidence intervals. Agents 1 and 10 require hotter heater surface temperatures than the rest of the agents in order to reach their desired design surface temperatures and as a result, design surfaces 1 and 10 take longer to reach the desired temperatures. Since the inside agents receive the most radiant heat from the all of the heater surfaces, agents 5 and 6 slightly overshoot after they initially reach their setpoint temperature.

In test case 5, shown Fig. 2.13, the temperature response of the design surfaces is given when the tolerance was set to ±0.5°C. With the exception of the outside agents (agents 1 and 10), all of the agents attained their setpoint temperatures within a tolerance of ±0.5°C in under 25,000 iterations. It was determined that approximately 30,000 iterations were needed for agents 1 and 10 to reach their setpoint temperatures within a tolerance of ±0.5°C. Once the agents reached their setpoint temperatures however, the design surfaces temperatures bounced between the upper and lower tolerance bands. Additionally, the final heater surface profile is non-uniform and did not have a smooth parabolic distribution, with heater surfaces 4-7 being hotter than the surfaces 3 and 8, which is shown in Fig. 2.14. The confidence intervals for the heater surfaces

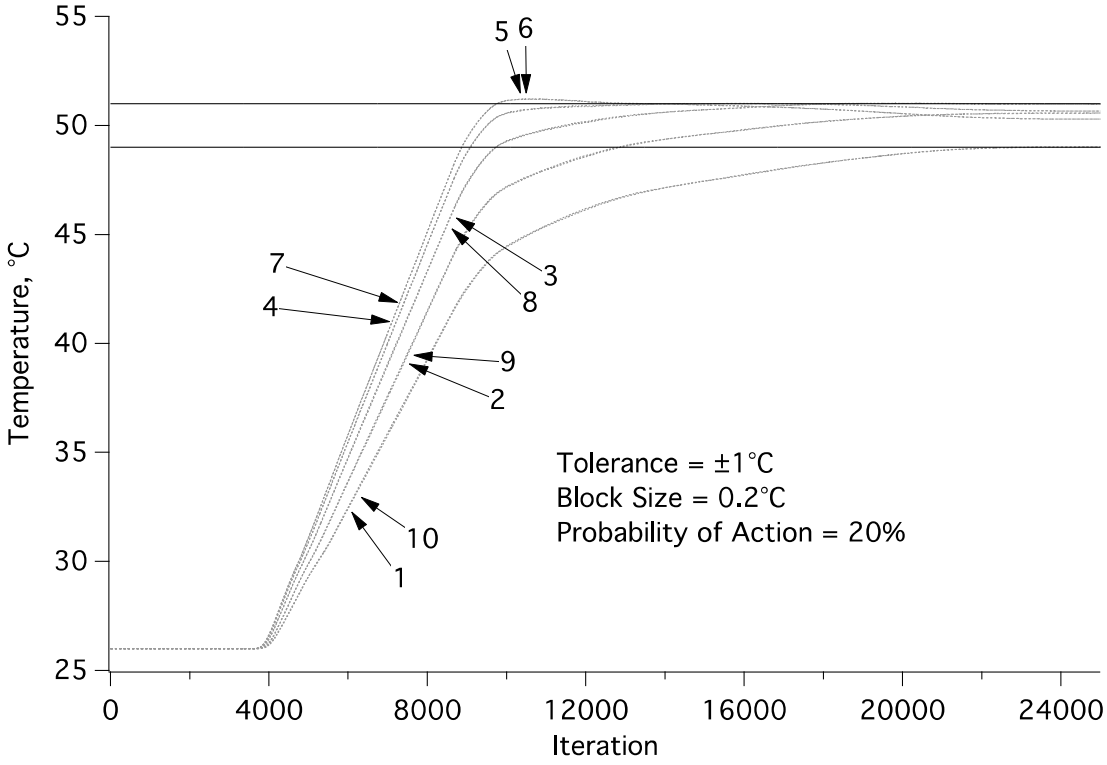


Figure 2.12. Case 4 - Design surface temperatures as a function of iteration.

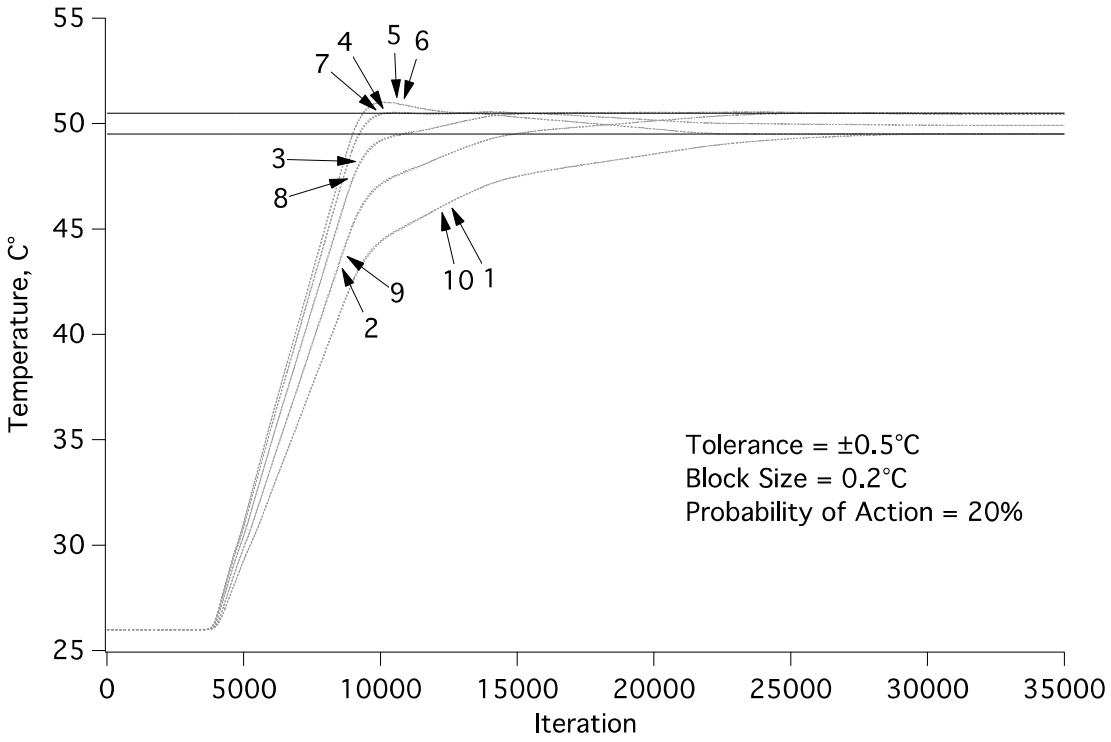


Figure 2.13. Case 5 – Design surface temperatures as a function of iteration.

and errors for the design surfaces are also greater than the previous test cases.

When the tolerance is set to ±1.5°C, as shown in Fig. 2.15, all of the agents reached their setpoint temperatures within a tolerance of ±1.5°C in approximately 5000 iterations. In test case 6, there is no movement of the design surfaces temperatures between the tolerance bands once all the agents reached their setpoint temperatures. While the final heater surface temperature profile is smooth and parabolic, the solution isn't as accurate as the previous test cases because the design surfaces are further away from their setpoint temperatures due to the increased tolerance.

2.4.2 Block Size

This algorithm is an iterative solution method and each solution will require a certain number of iterations before it is said to have converged or reached a solution. A solution is said to have converged when all of the design surface temperatures are within their tolerances and do not wander between the tolerance bands. One of the factors that influence the speed at which a solution is found is the block size, which determines the change in temperature of a heater surface with addition or subtraction of a single block. The block size also affects the fidelity of the solution i.e. the accuracy of the final design surface temperatures. A small block size correlates to a more accurate solution, but will require more iterations to reach that solution. A larger block size will produce a less accurate solution but will require fewer iterations to achieve that solution. In test cases 7-9, the affects of block size on the system response were examined and in every test case a tolerance of ±1°C was used.

When the block size is reduced to 0.1°C, as shown in Fig. 2.16, all of the agents with the exception of agents 1 and 10 were able to reach their setpoint temperature in under 25,000 iterations. Agents 1 and 10 were able to reach their setpoint temperatures in 46,000 iterations,

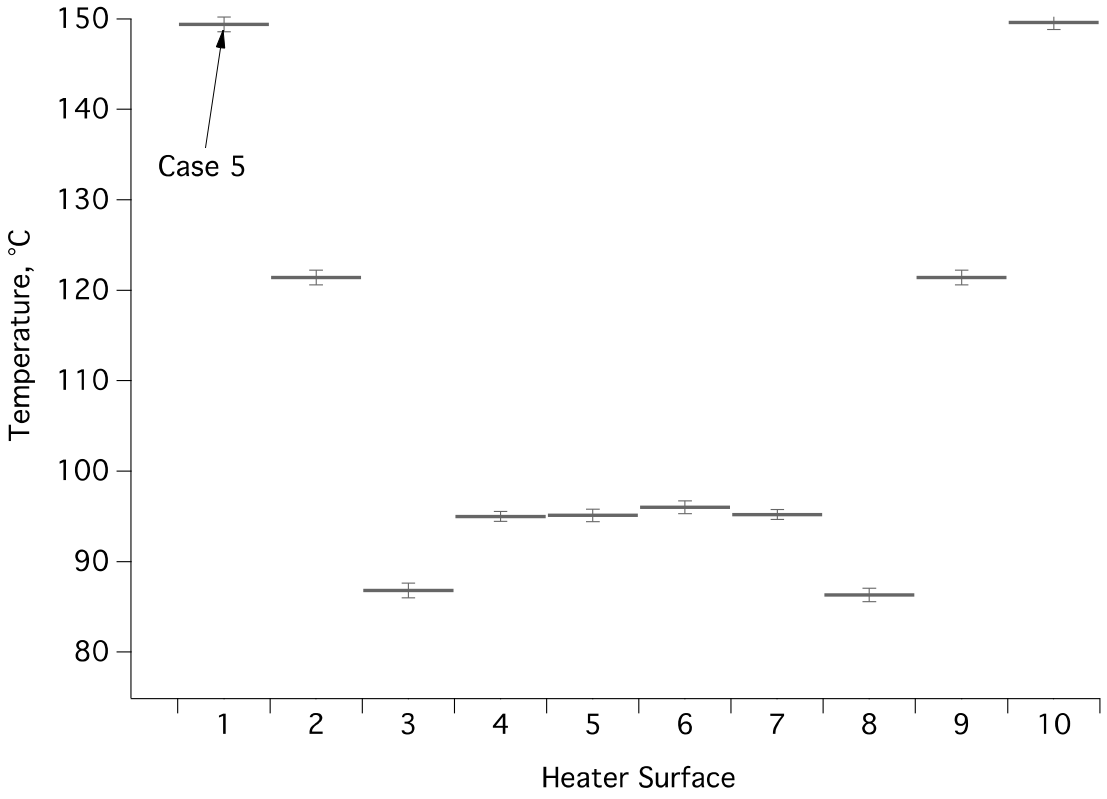


Figure 2.14. Case 5 - Heater surface temperature profile.

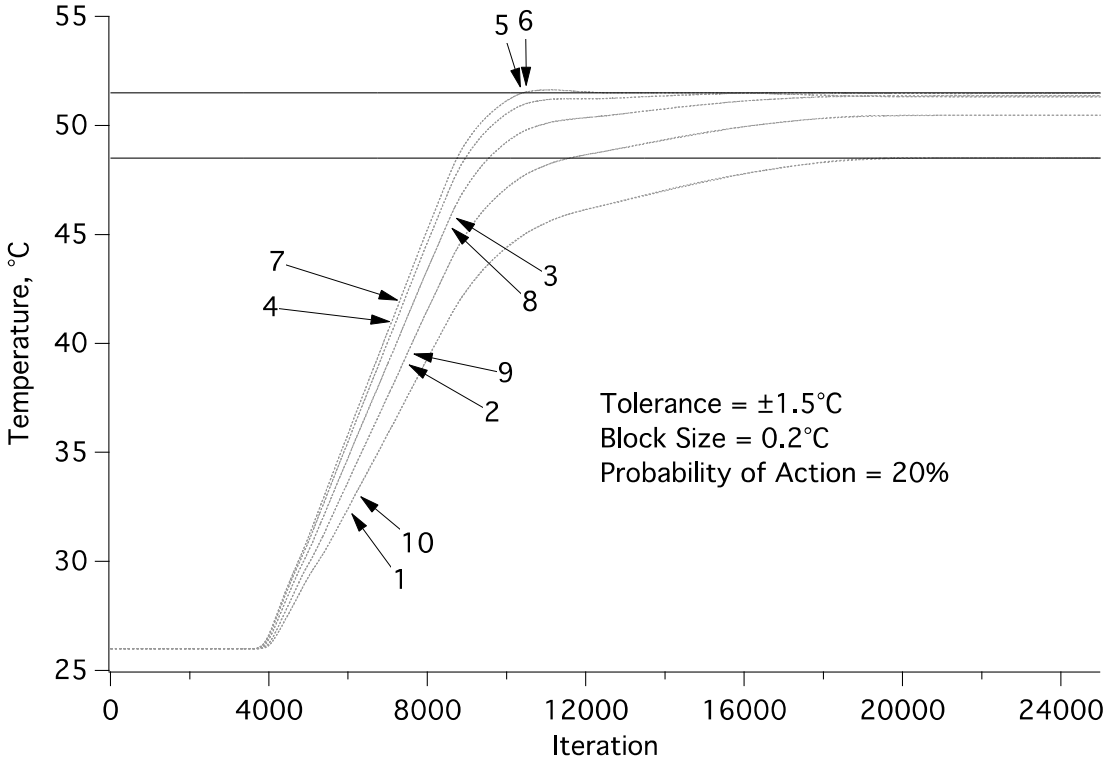


Figure 2.15. Case 6 - Design surface temperatures as a function of iteration.

which was double the iterations needed when the block size was 0.2°C. The final heater surface profile was smooth and uniform and the confidence interavals associated with the heater surfaces and the errors on the design surfaces were small. When the block size is increase to 0.5°C, as shown in Fig. 2.17, all of the agents reached their setpoint temperatures in fewer iterations (approximately 10,000 iterations) with slightly more overshoot. In test case 8, the 95% confidence interval for the heater surfaces shows slightly more variation than the previous tests, but comparable error between the desired design surface profile and the actual profile achieved. A block size of 0.5°C could also be used to achieve most of the same results as a block size of 0.2°C, as long as a slight amount of overshoot was an acceptable trade-off for converging in fewer iterations.

In test case 9, the block size was increased further to 1°C, which increased the overshoot slightly but all of the agents were able to reach their setpoint temperatures in approximately 5000 iterations, as shown in Fig. 2.18. While the error for the design surface temperature profile is similar to previous tests, the 95% confidence intervals for this test are much higher than any previous tests. This means that when the block size is increased to 1°C there is more variability in the mean values for the final design surface temperatures. Figure 2.19 shows the final heater surface temperature profile and error bars for case 9 and test case 4 for comparison.

2.5 Conclusions

In this paper a novel resource sharing algorithm was developed and demonstrated on an inverse radiation enclosure problem, with two-dimensional radiant heat transfer between ten design surfaces and ten heater surfaces. The resource sharing algorithm was able to achieve desired temperature profiles within 25,000 iterations for the design surfaces in every test case

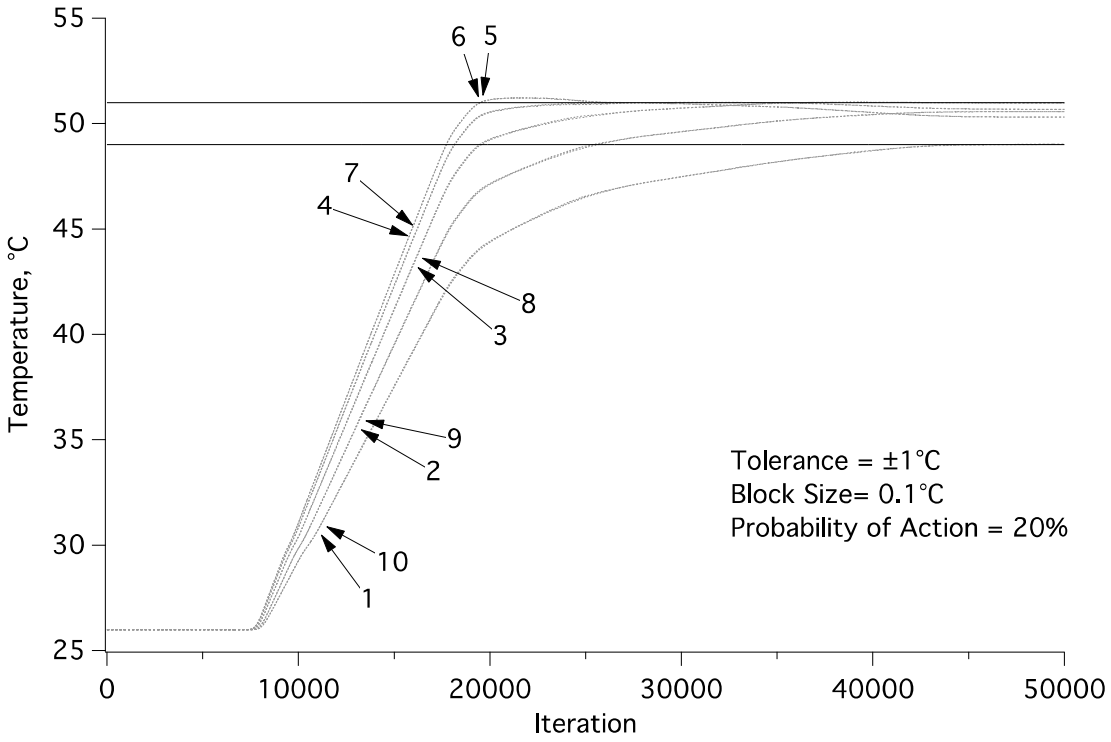


Figure 2.16. Case 7 - Design surface temperatures as a function of iteration.

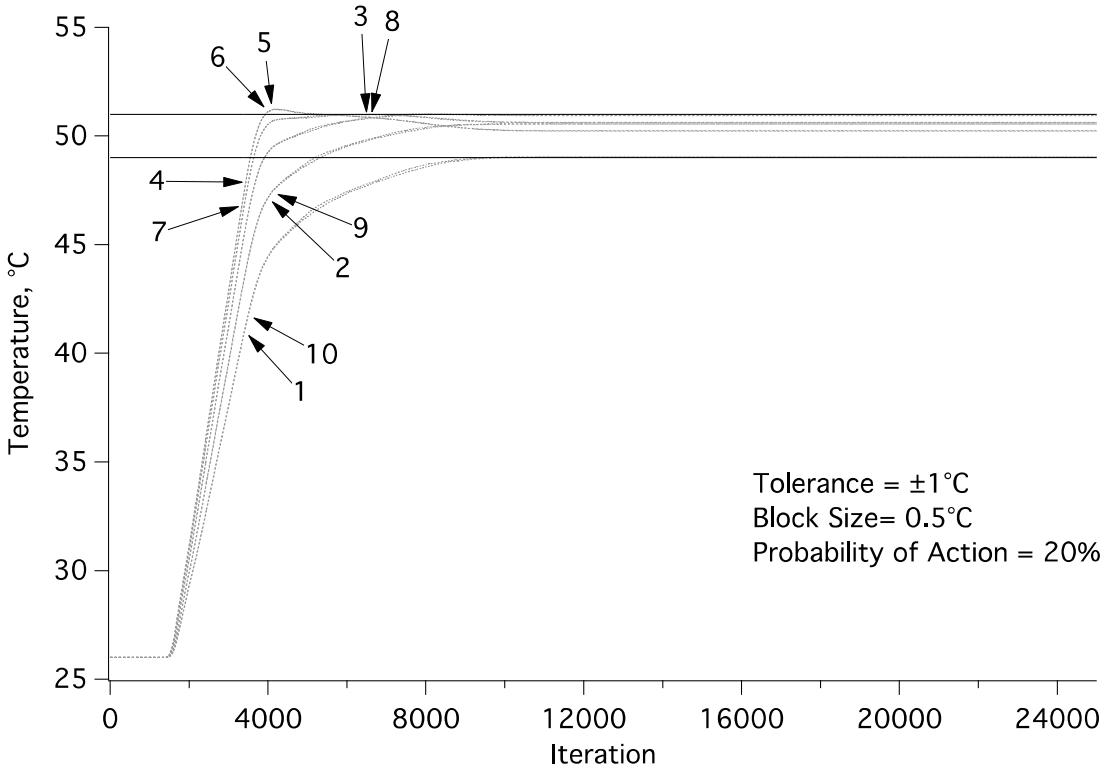


Figure 2.17. Case 8 - Design surface temperatures as a function of iteration.

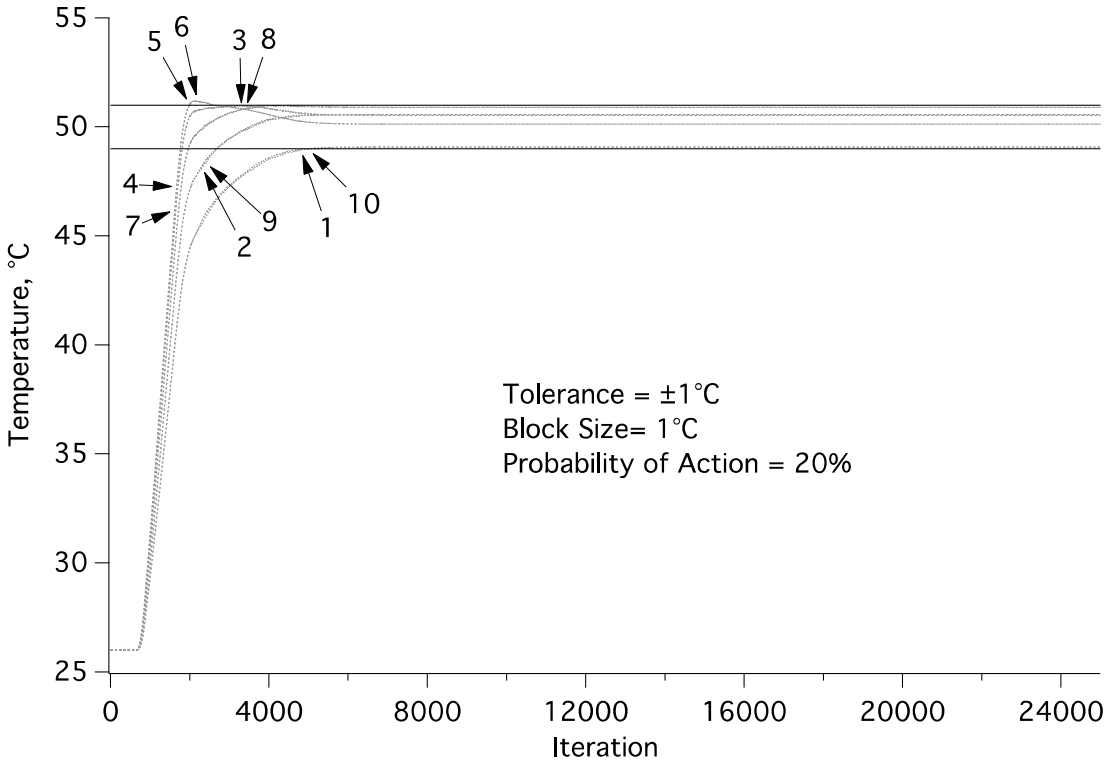


Figure 2.18. Case 9 - Design surface temperatures as a function of iteration.

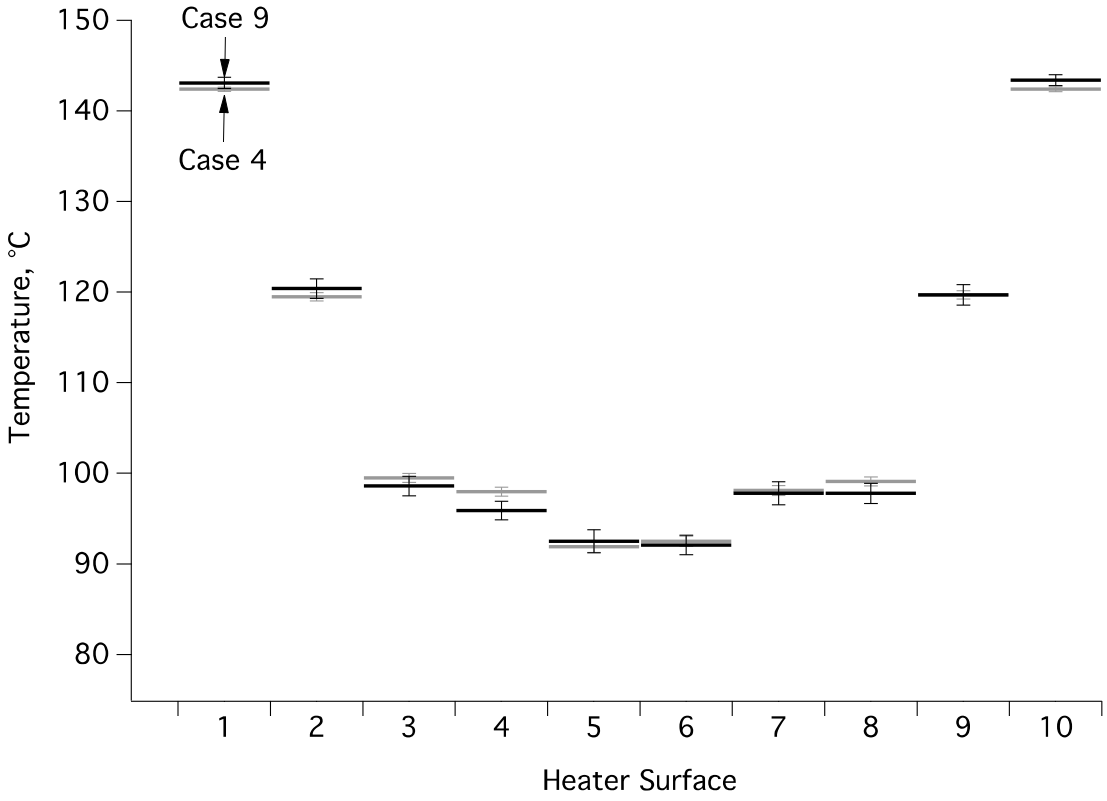


Figure 2.19. Case 9 - Heater surface temperature profile.

except cases 5 and 7, which still converged but needed more iterations. As implemented here, the proposed algorithm possesses several advantages over traditional approaches.

Finding computational solutions to radiant enclosure problems typically has been done through regularization or optimization. Regularization attempts to make the ill-posed part of the problem become well posed, which leads to a solution that is a good approximation rather than an exact answer. Optimization problems form an objective function subject to boundary conditions. Both techniques require models to accurately describe the system behaviour and require a detailed understanding of the system dynamics. With the resource sharing algorithm, the block repository only needs to know how agents are in a system and of those, how many are requesting blocks. The only information each agent needs to make a decision is whether they are at their desired state and if not, each agent will distribute and redistribute blocks until their specific state is met. Through frequent manipulation of their environment, by a group of agents acting independently, and the use of a shared resource, a global state can be achieved. Where optimization methods rely on global information to compute the value of objective function, the resource sharing algorithm uses only local information. This algorithm can be applied to any situation where a desired global state is specified, but the inputs needed to achieve that state are unclear. Agents can be added or removed seamlessly without reconfiguring the system or adjusting the rule sets of the agents or the block repository. This allows the algorithm to be more flexible and adaptable and allows for rapid implementation. This algorithm is also readily parallelizable.

Future work will include using this algorithm on other inverse problems; further exploration of the impact of block size, probability of action, and tolerance; and implementing this algorithm on small-scale physical systems. In addition we plan to test the resource sharing algorithm as a controls algorithm.

Acknowledgments

This research was supported by the US Department of Energy – Office of Fossil Energy under Contract No. DE-AC02-07CH11358 through the Ames Laboratory.

References

Amiri, H., Mansouri, S.H., Safavinejad, A., Coelho, P.J., 2011. The optimal number and location of discrete radiant heaters in enclosures with the participating media using the micro genetic algorithm. Numer. Heat Transfer, Part A 60(5), 461–483. doi:10.1080/10407782.2011.600597

Bakushinsky, A., Goncharsky, A., 1994. Ill-Posed Problems: Theory and Applications. Springer, Dordrecht.

Beckers, R., Holland, O.E., Deneubourg, J.-L., 1994. From local actions to global tasks: stigmergy and collective robotics. In: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, pp. 181–189.

Bonabeau, E., Guérin, S., Snyers, D., Kuntz, P., Theraulaz, G., 2000. Three-dimensional architectures grown by simple "stigmergic" agents. Biosystems 56(1), 13–32.

Bonabeau, E., 1998. A model for the emergence of pillars, walls and royal chambers in termite nests. Philos. Trans. R. Soc. B. Biol. Sci. 353(1375), 1561–1576. doi:10.1098/rstb.1998.0310

Bonabeau, E., Theraulaz, G., Deneubourg, J.-L., Aron, S., Camazine, S., 1997. Self- organization in social insects. Trends Econ. Evol. 12(5), 188–193.

Bonabeau, E., Dorigo, M., Theraulaz, G., 1999. Swarm intelligence: from natural to artificial systems. Oxford University Press, New York.

Bullock, S., Ladley, D., Kerby, M., 2012. Wasps, termites, and waspmites: distinguishing competence from performance in collective construction. Artif. Life 18(3), 267–290. doi:10.1162/artl_a_00065

Camazine, S., Deneubourg, J.-L., Franks, N., Theraulaz, G., Sneyd, J., Bonabeau, E., 2003. Self-organization in biological systems. Princeton University Press, Princeton.

Colaço, M.J., Orlande, H.R.B., Dulikravich, G.S., 2006. Inverse and optimization problems in heat transfer. J. Braz. Soc. Mech. Sci. & Eng. 28(1), 1–24.

Daun, K.J., Howell, J.R., 2005. Inverse design methods for radiative transfer systems. J. Quant.

Spectrosc. Ra. 93(1-3), 43–60. doi:10.1016/j.jqsrt.2004.08.012

Daun, K.J., Howell, J.R., Morton, D.P., 2004. Optimization of transient heater settings to provide spatially uniform heating in manufacturing processes involving radiant heating. Numer. Heat Transfer Part A 46(7), 651–667.

Daun, K.J., Howell, J.R., Morton, D.P., 2003a. Design of radiant enclosures using inverse and non-linear programming techniques. Inverse Probl. Eng. 11(6), 541–560.

Daun, K.J., Morton, D.P., 2003b. Geometric Optimization of Radiant Enclosures Containing Specular Surfaces. Numer. Heat Transfer Part B 43(3), 203–219.

Daun, K.J., Ertürk, H., Howell, J.R., 2002. Inverse design methods for high-temperature systems. Arab. J. Sci. Eng. 27(2C), 3–48.

Downing, H.A., Jeanne, R.L., 1988. Nest construction by the paper wasp, Polistes: a test of stigmergy theory. Anim. Behav. 36(6), 1729–1739. doi:10.1016/S0003-3472(88)80112-X

Ertürk, H., Ezekoye, O.A., Howell, J.R., 2002a. Comparison of three regularized solution techniques in a three-dimensional inverse radiation problem. J. Quant. Spectrosc. Ra. 73(2-5), 307–316.

Ertürk, H., Ezekoye, O.A., Howell, J.R., 2002b. The application of an inverse formulation in the design of boundary conditions for transient radiating enclosures. J. Heat Transfer 124(6), 1095–1102.

Fedorov, A., Lee, K., Viskanta, R., 1998. Inverse optimal design of the radiant heating in materials processing and manufacturing. J. Mater. Eng. Perform. 7(6), 719–726.

Feltell, D., Bai, L., Soar, R., 2005. Bio-inspired emergent construction. In: Proceedings of the 2003 IEEE Symposium on Swarm Intelligence, pp. 7–14. doi:10.1109/SIS.2005.1501596

França, F.H.R., Howell, J.R., Ezekoye, O.A., Morales, J.C., 2003. Inverse design of thermal systems with dominant radiative transfer. Adv. Heat Transfer 36(1), 1–110.

Hadamard, J., 1923. Lectures on Cauchy's problem in linear differential equation. Yale University Press, New Haven.

Halley, J.D., Winkler, D.A., 2008. Consistent concepts of self-organization and self-assembly. Complexity 14(2), 10–17. doi:10.1002/cplx.20235

Hansen, R.C., 1998. Rank-deficient and discrete ill-posed problems. Society for Industrial and Applied Mathematics, Philadelphia.

Holland O, Melhuish C., 1999. Stigmergy, self-organization, and sorting in collective robotics. Artif. Life 5(2) 173–202.

Howell, J.R., Ezekoye, O.A., Morales, J.C., 2000. Inverse design model for radiative heat transfer. J. Heat Trans-T ASME 122(3), 492–502. doi:10.1115/1.1288774

Kabanikhin, S. I., 2011. Inverse and ill-posed problems theory and applications. Walter de Gruyter, Berlin.

Karsai, I., Pénzes, Z., 1993. Comb building in social wasps: self-organization and stigmergic script. J. Theor. Biol. 161(4), 505–525.

Karsai, I., 1999. Decentralized control of construction behavior in paper wasps: an overview of the stigmergy approach. Artif. Life 5(2), 117–136. doi:10.1162/106454699568719

Ladley, D., Bullock, S., 2005. The role of logistic constraints in termite construction of chambers and tunnels. J. Theor. Biol. 234(4), 551–564. doi:10.1016/j.jtbi.2004.12.012

Mera, N.S., Elliott, L., Ingham, D.B., 2003. On the use of genetic algorithms for solving ill-posed problems. Inverse Probl. Eng. 11(2), 105–121. doi:10.1080/1068276021000058473

Özisik, M.N., Orlande, H.R.B., 2000. Inverse heat transfer: fundamentals and applications. Taylor and Francis, New York.

Park, H.M., Yoon, T.Y., 2000. Solution of the inverse radiation problem using a conjugate gradient method. Int. J. Heat Mass Tran. 43(10), 1767–1776.

Petersen, K., Nagpal, R., Werfel, J., 2011. TERMES: An autonomous robotic system for three-dimensional collective construction. In: Proceedings of Robotics: Science & Systems VII.

Porter, J.M., Larsen, M.E., Barnes, J.W., Howell, J.R., 2006. Metaheuristic optimization of a discrete array of radiant heaters. J. Heat Transfer. 128(10), 1031–1040

Safavinejad, A., Mansouri, S.H., Sakurai, A., Maruyama, S., 2009. Optimal number and location of heaters in 2-D radiant enclosures composed of specular and diffuse surfaces using micro-genetic algorithm. Appl. Therm. Eng. 29(5-6), 1075–1085.

Samarskii, A.A., Vabishchevich, P.N., 2007. Numerical methods for solving inverse problems of mathematical physics. Walter de Gruyter, Berlin.

Theraulaz, G., Bonabeau, E., Denebourg, J-L., 1998. The Origins of nest complexity in social insects. Complexity 3(6), 15–25.

Theraulaz, G., Bonabeau, E., 1995a. Coordination in distributed building. Science 269(5224), 686–688. doi:10.1126/science.269.5224.686

Theraulaz, G., Bonabeau, E., 1995b. Modeling the collective building of complex architectures in social insects with lattice swarms. J. Theor. Biol. 177(4), 381–400.

Tikhonov, A.N., Goncharsky, A., Stepanov, V.V., Yalola, A.G., 1995. Numerical methods for the solution of ill-posed problems. Springer, Dordrecht.

Tikhonov, A.N., Arsenin, V.I., 1977. Solutions of ill-posed problems. Winston, Washington DC.

Vanderplaats, G.N., 1984. Numerical optimization techniques for engineering design: with applications. McGraw-Hill, New York.

Yang, X-S., Cui, Z., Xiao, R., Gandomi, A.H., Karamanoglu, M., 2013. Swarm intelligence and bio-Inspired computation: theory and applications. Elsevier, Amsterdam.

CHAPTER 3. APPLICATION OF A NOVEL RESOURCE SHARING CONTROL STRATEGY ON A RADIANT ENCLOSURE TEST BED

Peter Finzell, Kenneth M. Bryden[2]
Simulation Modeling and Decision Science Program
Ames Laboratory
1620 Howe Hall, Ames, Iowa, 50011

Abstract

This paper applies a new resource sharing control strategy on the physical realization of a radiant enclosure problem. This control strategy is applied to a physical system, which is based on a tightly coupled, ill posed inverse radiation heat transfer problem. The objective will be to maintain a given two-dimensional temperature profile of flat plate by changing the temperature of a parallel plate. As implemented in this problem, two sets of parallel plates are divided into ten heater surfaces and design surfaces and the goal is to maintain a given temperature profile of 35°C along all of the design surfaces. Computational agents are created and each agent is composed of a single heater for each heater surface and an aluminum plate for each design surface. The surfaces are separated by 7.62 cm and the desired temperature is maintained along design surfaces by controlling the power flow to the heater surfaces. This paper examines and compares the response of the resource sharing control strategy with that of a traditional PI controller. The tests conducted examined the adjustable parameters of the resource sharing control strategy, which were the block size and the probability of action being taken. This control strategy was compared to a PI controller by determining the response in terms of rise time, settling time, peak overshoot. In each test, the resource sharing control strategy was able to reach the desired temperature profile within an acceptable degree of accuracy and comparable to the response of the PI controller.

[2] Corresponding author tel +15154600875
Email address: kmbryden@iastate.edu

Keywords: Stigmergy; Multi-Agent Systems; Bio-inspired; Distributed Control; Inverse Heat Transfer

3.1 Introduction

The increasing availability of small, inexpensive, and intelligent sensors, coupled with larger complex systems and a desire for a greater fidelity of control, creates an opportunity for new control strategies. Inverse problems provide an excellent framework for proving the capabilities of new algorithm or control strategy because of their inherent complexities. Inverse problems are encountered in systems that have desired output conditions but the required inputs are unknown. Inverse heat transfer problems arise in annealing, industrial process ovens, and combustion chambers. Since these problems are considered ill posed, there can frequently be multiple solutions, many of which are infeasible (Ozisik 2000). With inverse problems, small changes in input can cascade throughout the system and can have large impacts on output (Kabanikhin, 2011). These factors make it challenging to find useful and realizable solutions in inverse problems (Hansen 1998). Additional challenges to attain and maintain these desired outputs arise in physical systems because of the system changes with time and a physical system is subject to limitations that computational systems are not. Inverse problems become controls problems when there is no longer a single desired output but an operating range.

Control of dynamic systems is concerned with of determining how system inputs and outputs are related. Equations showing these relationships can be modeled using frequency response or state space (Siljak, 2013). Using frequency response, linear systems are used to construct a transfer function, which is a relationship for an output given an input over whole range of possible inputs and outputs. State space models show the relationship between every variable of the system at any point in time (Lumkes, 2002). Extensive research has been conducted into

refining existing control strategies for complex systems. While these refinements can be successful at achieving performance gains, they fail to address a central issue; how to reliably control complex systems without needing massive amounts of system information and establishing component interactions beforehand (Bakule, 2008; Siljak, 2013).

In this paper a novel resource sharing control strategy is developed for a physical radiant enclosure system based on a previously developed resource sharing algorithm (Finzell and Bryden, 2016). The problem examined in this paper is the physical realization of the original radiant enclosure problem used to develop the original algorithm. In this paper the problem of determining the temperature distribution needed on the heater surfaces to achieve a desired design surface temperature profile is recast as a distributed construction problem. This resource sharing algorithm creates simple agents which use changes made to the environment as a means of indirect communication, a shared resource, temperature and a simple set of rules to coordinates actions. Each agent is allowed to make control decisions at the local level without consulting with other agents.

3.2 Background

Control strategies are created to achieve some desired conditions or to maintain a specified set point within a dynamic system (Trentelman et al., 2001). Developing control strategies for these complex systems frequently involves extensive planning to ensure there is sufficient system information or coordination to avoid potential conflicting control decisions. There are many control strategy categories and selecting the appropriate one is very system specific (Ogata, 2010).

Two control architectures are commonly used for control of complex systems: centralized and distributed control (Khaki-Sedigh and Moaveni, 2009). Centralized control, while often

preferred because of complete system information and control, becomes increasingly difficult to coordinate as the size and complexity of system being controlled increases (Sandell et al., 1978). These controllers become more challenging to construct as the number of system variables increases or control actions can be delayed as the amount of system data that needs to be processed grows. If the system has a large amount of process variables that need simultaneous control, a MIMO or Multi-Input Multi-Output controller may be used. Multi-Input Multi-Output (MIMO) controllers obtain mathematical relationships between every relevant input and output and combine those relationships into a single matrix which allows a centralized controller to take the appropriate control action at any time step (Levine, 2010). Centralized control requires a detailed mathematical model to be obtained beforehand and sensor measurements to make accurate and appropriate decisions (Åström et al, 2012). While preferred because of accuracy in smaller systems, it becomes increasingly difficult to coordinate control actions as the size and complexity of the system increase.

The second approach is to distribute the control of the system; independent controllers using localized information to control sections of the system (Bakule, 2008). This can either be done through decentralized or distributed control. The difference between decentralized and distributed controllers how the controllers share information. Decentralized control involves the formation a hierarchy of controllers. With distributed control, each local controller is able to make many control decisions independent of a hierarchy or centralized controller, either operating autonomously or communicating directly with other controllers (Massioni and Verhaegen, 2009). PID controllers are considered distributed controllers and are an industry standard that offer many potential methods to adapt to different conditions and systems. A transfer function, which relates the controller's input to its output, is created either from first principle equations or an observed

response to a known input. PID controllers are based on a transfer function from which the proportional, integral and derivative gains can be tuned to achieve the desired response (Aström & Hagglund, 1995). The proportional gain is amount of instantaneous proportional change to the input as function of the steady state error, the integral gain is the response of the system based on previous information and the derivative gain is the response of the system based on the expected future value of the steady state error. Frequently in large complex systems, multiple PID controllers are implemented simultaneously to control multiple processes variables (Åström and Häggland 2006). This control architecture allows the system to be more scalable and flexible (Scattolini, 2009). In strongly coupled systems, using multiple controllers simultaneously means that an action taken by a single PID controller will have some effect on process variables being maintained by other PID controllers. Using feedforward techniques, one controller's decisions can be incorporated into another controller's actions (Åström and Häggland 1995).

Agent-based controls offer a significant increase in system flexibility that has the potential to effectively manage complex, highly coupled system (Weiss, 2013). Agent-based systems are composed of multiple computational entities that operate in a shared environment that are able to react to change in their environment (Wooldridge 2008). These objectives can differ based on the abilities of the agent, the environment and the composition of individual agents. The key distinction between agent based controls and traditional controls is the autonomy given to individual agents to make independent decisions while still coordinating their actions. When properly designed, these systems can be very scalable as agents have to potential to be added or removed, without redesigning the entire system. The limitation of agent-based control is the communication infrastructure required for agents to coordinate their behavior with other agents (Lewis et al. 2013). To effectively coordinate their actions, they need a means to directly

communicate with other agents to share their current actions and to ascertain not only how many other agents are in the system, but the actions taken those agents. This communication infrastructure, along with delayed or unreliable communication between agents, can be one of the major drawbacks for implementation of agent-based control in highly coupled complex systems. Typically, coordination among agents is established through bidding, auctions or negotiating (Wooldridge 2008).

Multi-agent systems have been used or proposed as method of control for manufacturing systems, network traffic flow management, sensor network coordination, distributed power grid and energy generation management and industrial control (Daneshfar and Bevrani, 2009; McArthur et al., 2007; Vinyals et al., 2011; Bussmann et al., 2013). When used in manufacturing systems or industrial control settings, these agents typically operate as supervisors and coordinators of the local controllers, which are typically traditional distributed controllers (Daneshfar and Bevrani, 2009). Most multi-agent systems are computational models, with little implementation of multi-agent systems on physical systems (Leitão, 2009).

PID control offers decentralization and creates distributed controllers which allows control decisions to be spread out through a system, but may have unintended interactions. Centralized controllers such as a MIMO controller require extensive models beforehand but can accommodate more simultaneous interactions. Most agent-based system require a communication protocol for agents to communicate their intentions and coordinate actions, which can cause delays and limit the flexibility or adaptability of a system. This paper is concerned with finding a control strategy where distributed controllers are able to coordinate their behaviors without centralized planning, control or direct communication between controllers.

Social insects (bees, ants, termites and wasps) are able to construct complex nests and maintain intricate colonies and nests involving thousands of workers, with sophisticated features and intricate structures, without centralized management or direct communication between insects (Theraulaz et al., 1998). Changes made to the environment combined with a set of simple instructions for each insect, serves as a means of coordination, with each insect making small changes to the environment using a shared resource (Karsai, 1999). Additional insects are able to understand the state of construction through observation, providing a means of indirect communication (Bonabeau et al. 1999). This enables each insect to determine the next appropriate step needed and to take action. This indirect coordination can enable the creation of very adaptable, scalable systems because any number of insects or agents can work on a task simultaneously (Sharkey, 2006). These actions reinforce each other and complex macroscopic colony level behavior is created from very simple microscopic insect actions and interactions (Camazine et al., 2003).

The stigmergic construction process has been the inspiration of a computational technique called distributed construction. Distributed construction uses computational agents, that manipulate uniform construction materials or "blocks" in a shared environment based on some very simple set of rules followed by each agent (Theraulaz and Bonabeau 1995a; Bullock et al., 2012). Agents are created to serve as the computational form of individual insects. Agents can be thought of as independent actors, which can sense changes to their environment and have the ability to change that environment (Wooldridge, 2008). Using distributed construction agents were shown to be able to construct simple or complex structures without a centralized coordinator or predefined step-by-step instructions, using only local information and uniform building blocks (Theraulaz and Bonabeau 1995b). Distributed construction has been used to examine the

applicability of stigmergic processes to the construction of complex structures (Bonabeau et al. 2000). Autonomous robots have used distributed construction to build or arrange simple blocks or discs through cooperative behavior, which an individual agent would otherwise be incapable of performing the task alone (Holland and Melhuish 1995, Sharkey 2006; Jevtic and Andina 2007). These robots are able to build walls (Melhuish, Welsby, and Edwards 1999), cluster objects (Lan, Liu, and Yang 2006; Beckers, Holland, and Deneubourg 1994) and create intricate patterns (Petersen, Nagpal, and Werfel 2011). Justin Werfel proposed the idea of "extended stigmergy" where the building blocks that were used to construct the structures could store information about their location, perform computations and communicate with neighbors (Werfel and Nagpal, 2005). Multi-agent stigmergy systems have been created for manufacturing and production control problems, but as previously discussed, the agents were given a supervisory role rather than assigned to local controllers (Valckenaers et al. 2007; Hadeli et al. 2003).

3.3 Resource Sharing Control Strategy

Distributed construction is the inspiration for the previously developed resource sharing algorithm that serves as the basis for a new resource sharing control strategy (Finzell and Bryden, 2016). The prevailing themes used to create this resource sharing control strategy are: agents are able to perform tasks using a shared resource and a shared environment, each agent is given the means to manipulate their environment and actions are taken based on some set of simple instructions. The resource sharing control strategy gradually "build up" a solution using computational agents that make changes to their environment using a shared resource, or blocks, which serve as an incremental unit of change. The value or size of these blocks can change and can be varied. This resource sharing controls strategy can be applied to any situation where a

desired global state is specified, but the inputs required to get to that state are unknown. Each agent is not given specific instructions as to how to achieve their local state but will take blocks from and give blocks back to a repository until their desired local state is achieved. It is through this frequent, independent, and random manipulation of the environment by a group of agents acting independently and the use of a shared resource that the desired global state is achieved.

In case discussed here, the agents are composed of a single actuator and sensor. These agents are manipulating their environment in a shared, but indirect way through the block repository, shown in Fig. 2.4. The block repository facilitates the creation, distribution, and redistribution of blocks but is not a controller and is provided a very limited amount of information. The agents send their current state to the block repository based on if they are above, below, or at their set point (Fig. 2.4). The block repository counts the states that are above, below, and at their set point and determines whether new blocks are needed. At each time step two random numbers are created, one to randomly select an agent out of the group and one randomly determines if that randomly selected agent can take an action based on a user defined probability. This probability can be raised or lowered based on how often the user wants action to be taken and is called the probability of action. The agent that was selected then makes a decision to take a block, give back a block, or keep all their current blocks, shown if Fig. 2.5.

3.4 Problem Setup

The resource sharing algorithm will be extended to a control strategy and applied to a physical test rig. The problem this control strategy is applied to is the physical realization of a radiation enclosure problem, which is discussed in (Finzell and Bryden, 2016). In this problem to goal is to achieve a desired temperature profile along the design surfaces by adjusting the temperature of the

heaters (Fig. 2.1). Since each heater has some effect on each design surface finding a solution requires achieving the correct balance of inputs to the heaters to achieve the desired temperature profile on the design surfaces. The resource sharing control strategy will gradually build up a solution through the distribution of blocks, which in this case are a discrete unit of temperature that are transferred to a heater surface. Temperature, like building blocks used in distributed construction, becomes the discrete unit of change use to reach a solution. These building blocks are not a conserved quantity, but can be created as needed, distributed and redistributed among agents as the temperature distribution along design surfaces approaches the desired temperature distribution. Each agent makes decisions about whether to accept more blocks, return blocks or maintain all of its blocks based on local information from a single design surface. Each agent senses the current state of their local environment, i.e. a single design surface, and takes action to change that environment, based on a simple rule set, by transferring blocks to or from a single heater surface. The two random numbers determine how often agents are able to take action.

The test rig is constructed from 80/20® aluminum with an approximate height of 39.4 cm, width of 24.1 cm and length of 105.5 cm and is shown below in Fig. 3.1. Each of the 10 heaters and design surfaces are axially aligned and evenly spaced so there is no conductive heat transfer between the heaters or the design surfaces. A separation distance of 7.62 cm was used; which enabled the heaters to affect several their neighboring design surfaces. The dimensions of the heaters and surfaces are given in Fig. 3.2. The bottom surfaces were made of anodized aluminum to maximize the radiant heat absorption and were constructed to be the same dimensions as the actual heater area (without the outer housing). The aluminum surfaces are mounted on ceramic posts to insulate them from the test rig. The thermocouples on each of the lower aluminum plates

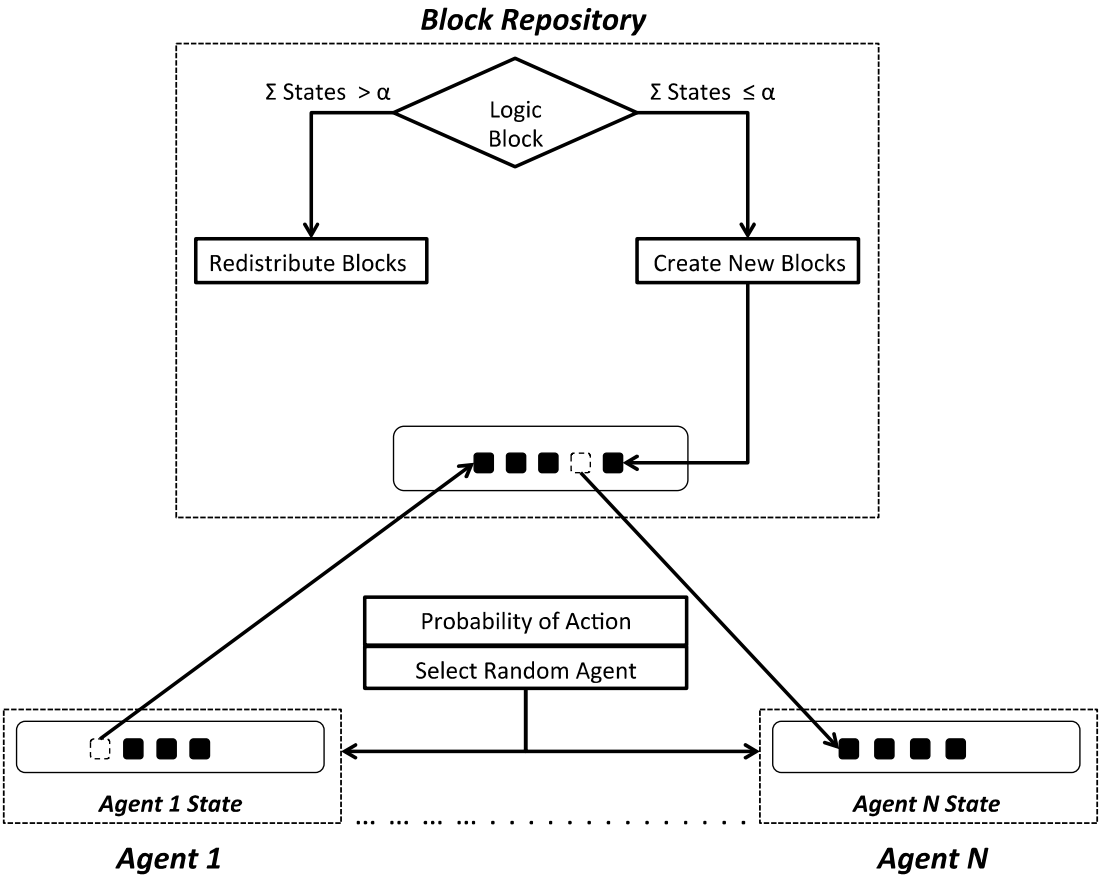


Figure 2.4. Agent and block repository interactions.

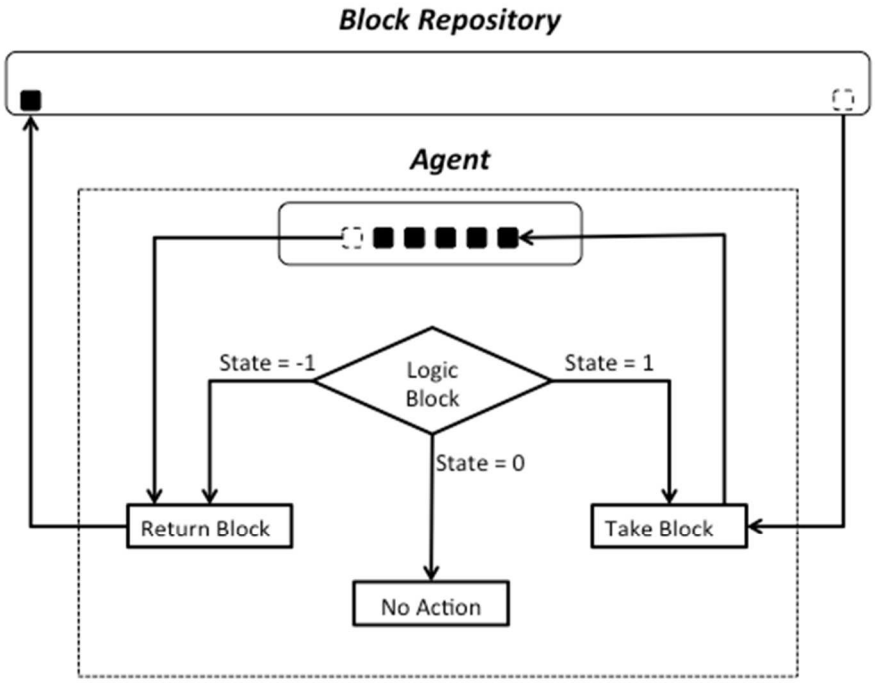


Figure 2.5. Agent rule set.

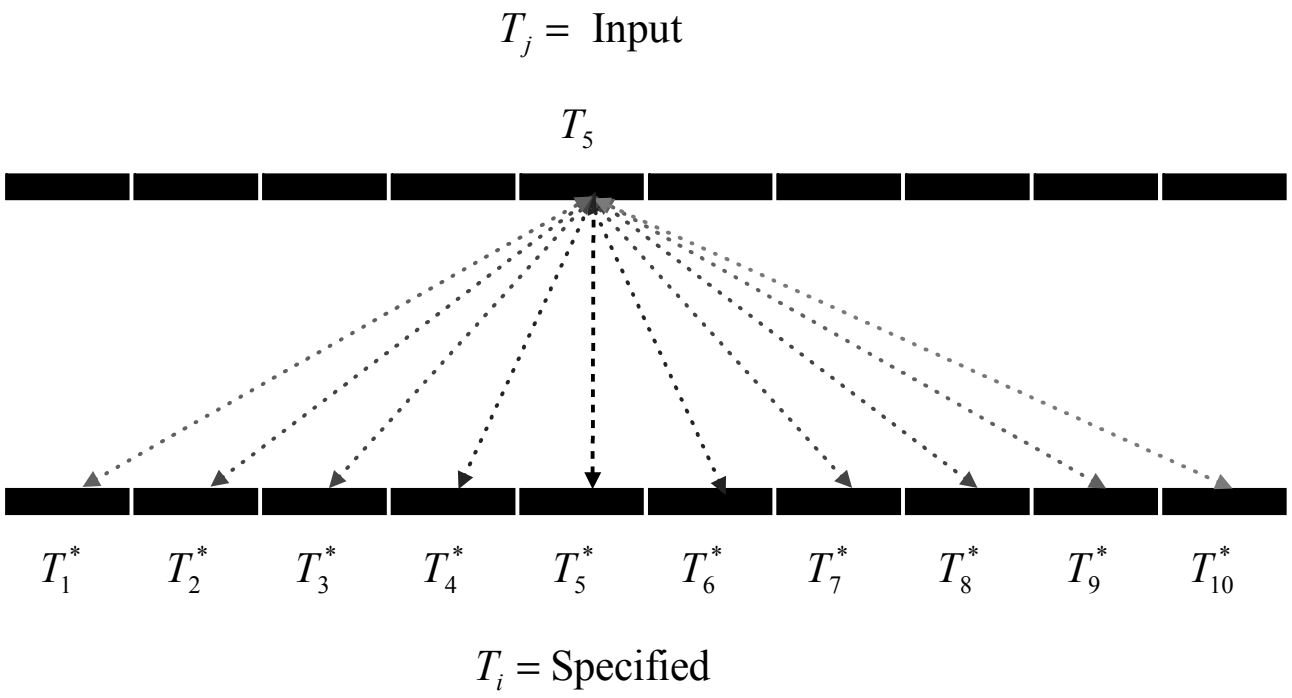


Figure 2.1. Radiation heat transfer between heater surface 5 and design surfaces 1−10.

is an Omega® flat cement on K-type thermocouple which are made from 0.0013 cm foil and 0.025 cm diameter thermocouple wire. Each of the thermocouples is centered on each of the design surfaces and held in place with high temperature Kapton® tape. To ensure good contact between surfaces, a high temperature thermal paste between the aluminum plates and the thermocouples. The heaters are 375 Watt Tempco™ ceramic emitters with embedded K-type thermocouples.

The testing rig is driven by two separate systems for data acquisition from the temperature sensors and a separate system to provide digital input signals to control the heaters. Both of these systems are brought together in a single Labview ™ VI. A SCXI data acquisition system is used to read the temperature data. The main chassis is an SCXI-1000, which housed an SCXI-1102 32-channel thermocouple/voltage input module. A panel jack system allowed for quickly disconnecting and reconnecting the thermocouples, which is connected to an SCXI-1303 isothermal terminal block. An NI-6008 box is used for output operations and provides 6 digital output channels, which can be controlled from a desktop computer running Labview ™. The voltage and current output from this device are limited, around 5 volts DC and 1 mA. To provide power the heaters, which operate at 240 volts AC, a relay must be used. A solid-state relay (SSR) acts as an on/off switch and allows a low voltage side (5V DC), to control much higher voltage side (240V AC). The SSR can either be on or off, so to adjust the heater temperature the on/off signal can be pulsed more or less frequently to adjust the equivalent voltage provided to each heater, thereby adjusting the temperature of the heater. A Labview ™ VI was created which takes in temperature data from the heaters and design surfaces and can control the frequency of these pulses send to the SSRs, which is called pulse width modulation (PWM).

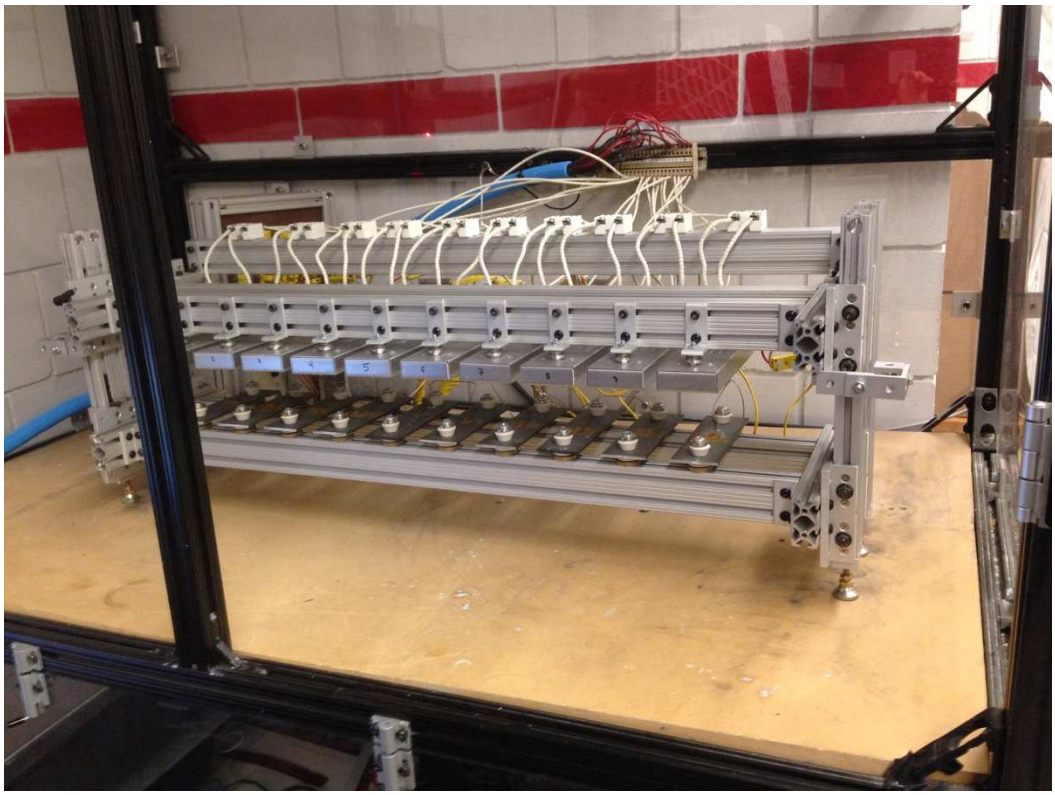
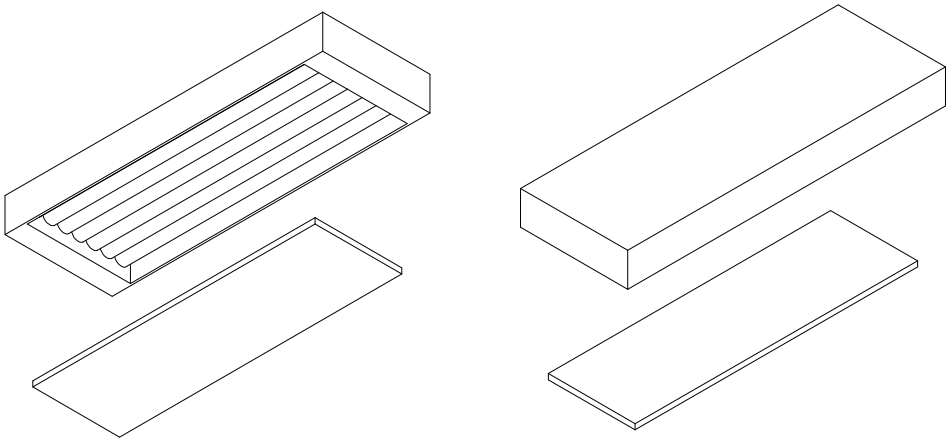
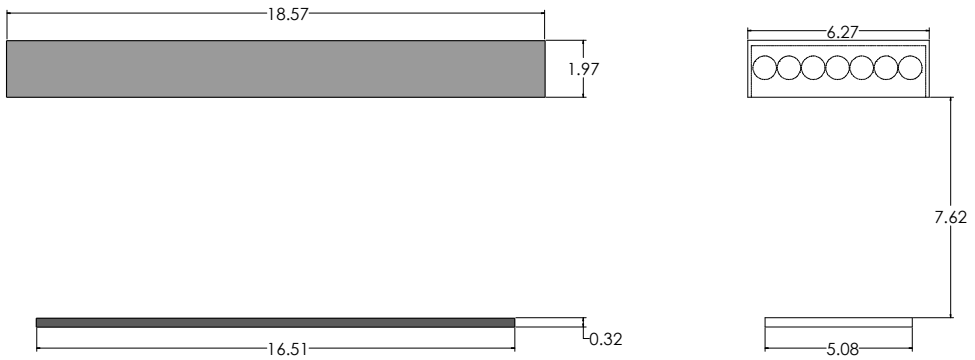


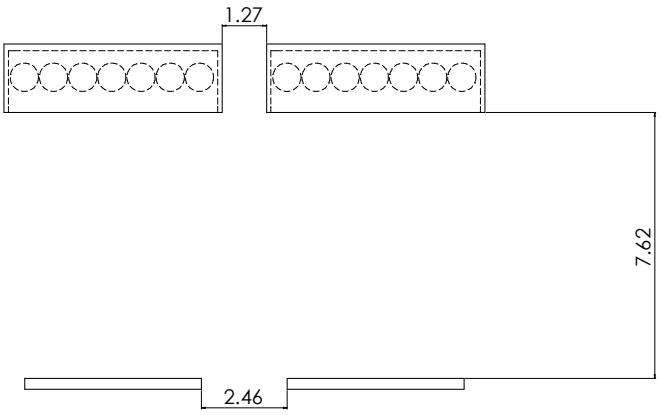
Figure 3.1. Testing rig.



(a)



(b)



(c)

Fig 3.2. Heaters and surfaces, (a) Single heater and surface, (b) Single heater and surface dimensions, (c) Heater and surface spacing.

Setting the PWM to different duty cycles adjusts the width of a pulse, which is the period that the power is turned on and off, shown in Fig. 3.3.

As shown in a previous paper, the resource sharing controls strategy was successfully tested on a computational simulation controls problem for two-dimensional radiant heat transfer between two set of heater and design surfaces, which was created to mimic the physical test bed created for this research. Mirroring the computational simulation, the heater and design surfaces are initially at the 26°C and the maximum temperature of the heater surfaces is 300°C. The goal was to achieve and maintain a uniform spatial temperature profile of 35°C on all the design surfaces. The slow response of the heaters and subsequently the design surfaces means that the testing time required was fairly large (3 hours). As shown in Fig. 2.3, each agent is composed of the temperature sensor for a design surface, an actuator or heater that sets the temperature of the associated heater surface, and uses a decision network to decide when to request more blocks or give back blocks (discrete temperature units). The problem as posed here is then composed of ten agents and one block repository. The block size can be adjusted and the frequency of block given actually being given off each time can also be adjusted, which is call the probability of action. The block size in this instance corresponded to an incremental change in the current PWM duty cycle value associated with each heater.

3.5 Experimental Results

Traditional PI control was compared to the new resource sharing control strategy through several tests. Individual PI controllers for each agent (sensor-actuator pair) were selected for a direct comparison. The gains for the PI controller were tuned using the proportional gain (i.e., the amount of instantaneous proportional change to the input as function of the steady state error)

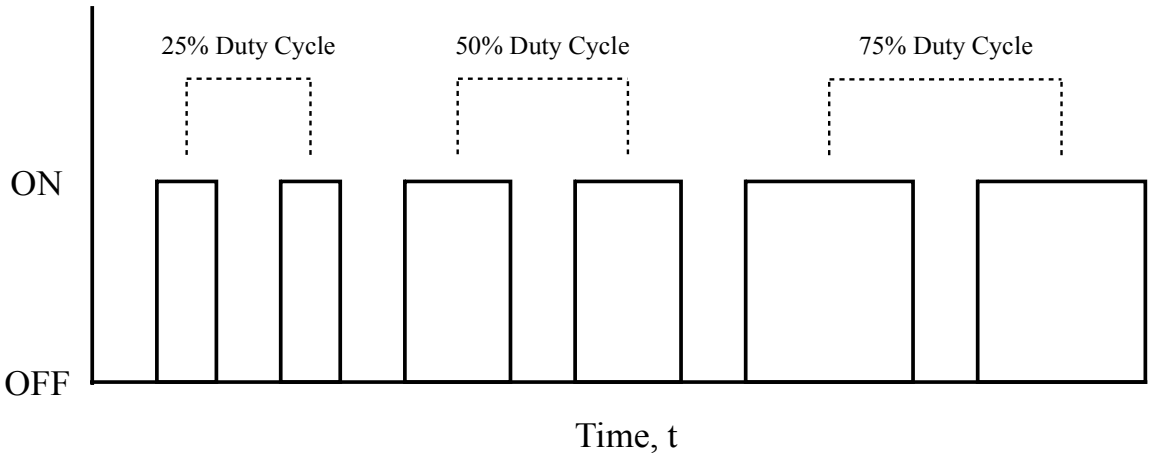


Figure 3.3. PWM diagram.

and integral gain (i.e., the response of the system based on previous information) and where found using the open loop response. The adjustment parameters in the resource sharing controller are

- *blocks size*—how much incremental change to the PWM duty cycle occurs with the addition or subtraction of a block, and
- *probability of action*—how often individual agents are allowed to take action when they are randomly selected.

The tolerance is the bounds that the setpoint can be within to be considered at the desired value. An appropriate tolerance is one that achieves as close as possible to the desire temperature on the design surface while balancing the inherent system noise. The tolerance was determine empirically and was selected to be 0.2°C.

The tests on the resource sharing and PI controllers relate the effects of adjusting tunable parameters for both controllers and compared in terms of traditional control metrics. To compare how each adjustment parameter performed three dynamic system metrics were used

- *rise time*—the time for the first surface reach the vicinity of the setpoint
- *peak overshoot*—the single highest temperature observed during the test, and
- *settling time*—the time for all of the surfaces to be within 2% of the set point temperature and stay within those bounds for the remainder of the test.

Preliminary testing conducted using shorter time periods (30 minutes to 1 hour) resulted in good initial values for the tuning parameters for both the resource sharing controller and the PI controller. The final tests were performed in the span of three hours or 180 minutes.

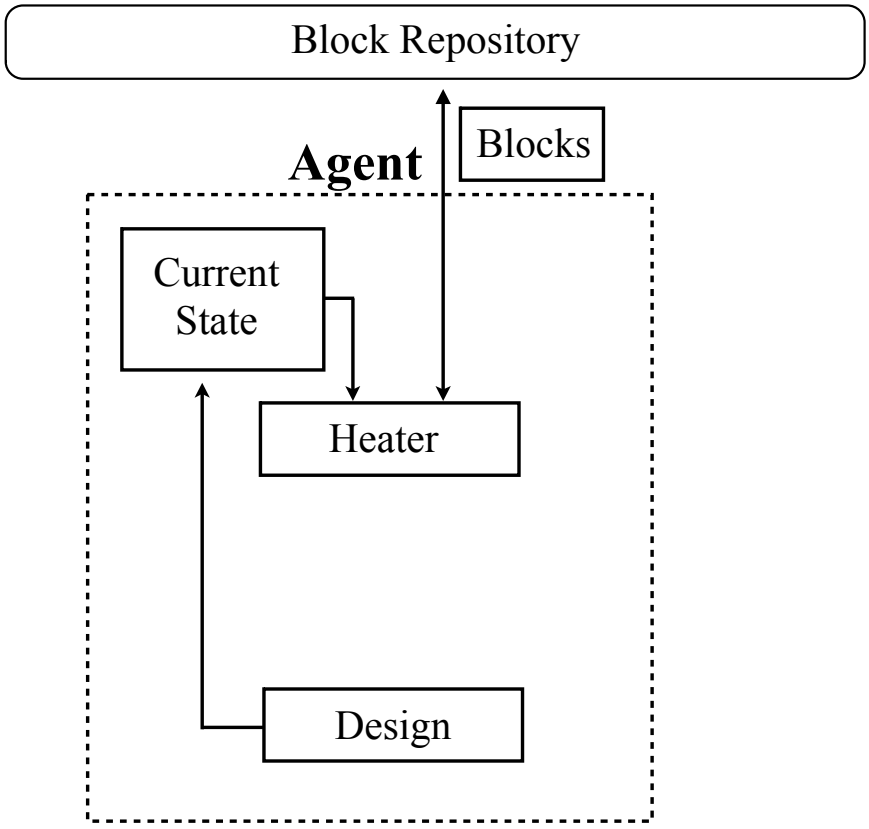


Figure 2.3. Agent composition.

The proportional initial gain for the first underdamped case examined is 2.5 and the integral gain is 0.5. The proportional and integral gains for the first overdamped case examined are 0.2 and 0.8 and 0.04 and 0.8 for first critically damped case, respectively. In the second overdamped and critically damped cases, the gains were slightly altered by altering the proportional gain by 0.1. In the second underdamped case examined a very different set of gains were used but produced similar results. The tests for the PI controller were abbreviated U1 & U2, O1 & O2 and C1 & C2 for underdamped, overdamped and critically damped respectively. The five resource sharing tests were abbreviated RS1 through RS5, with RS1 being the base case for comparison.

The two initial values that produced a good response in the resource sharing controller were a block size of 0.00125 and a 20% probability of action being taken. Besides using initial values, two additional tests were run using values higher and lower than the initial values. The block size values tested were 0.0025, 0.00125 and 0.0005. The probabilities of taking action used were 10%, 20% and 30%. Table 3.1 gives the results from the five tests. In all the tests the resource sharing control strategy performed very well at keeping the overshoot low. Decreasing the probability of action made the distribution and returning of blocks slow down, and therefor the rise time was much slower for test RS3. Test RS3 did settle much faster than tests RS1 and RS2. Similarly, in test RS4 decreasing the block size slowed the rate of change of the system down such that the rise time increased, but the settling time was reached much more quickly. The baseline test along with the two best responses (Tests RS3 and RS4) are given in Figs. 3.4(a)-(c). In Fig. 3.4(a), the base case (RS1) demonstrates that the resource sharing control strategy is able to reach the set point temperature and adjust the control action such that the design surface temperatures are closer to the setpoint over time. Fig. 3.4(b) shows test RS3 where the probability of taking an action was less than the base case, so there were fewer control actions taking place over the same

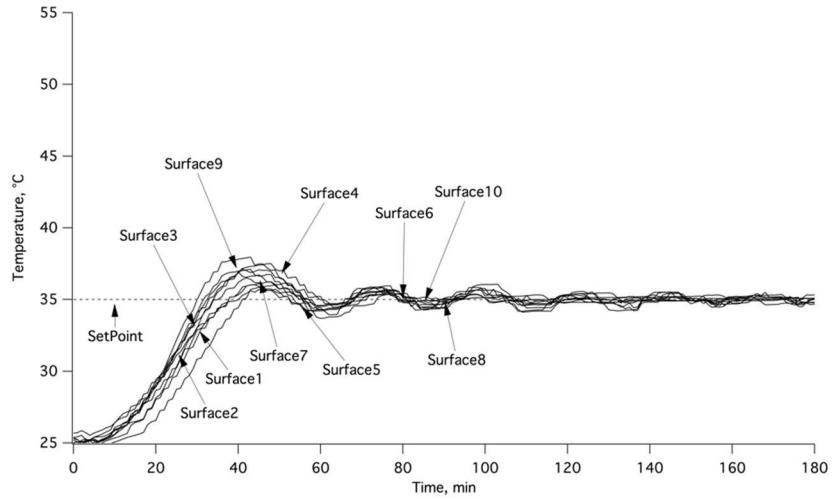
amount of time. While this test took longer for all the design surfaces to achieve their setpoint temperatures, once they achieved the setpoint they stayed closer to the setpoint and had much less overshoot than in test RS1. Test RS4, shown in Fig. 3.4(c), used a smaller block size, so the changes in input were smaller than test RS1. Like test RS3, the design surfaces took longer to reach their setpoint temperatures but were able to maintain a much tighter tolerance than the base case. As shown test RS4 had the smallest overshoot.

In the case of the PI controller the proportional and integral gains for the underdamped (Tests U1 and U2), overdamped (Tests O1 and O2), and critically damped (Tests C1 and C2) tests were found by first using the open loop response to find a transfer function and then fined tuned using the pole placement approach and compared with gains recommended by the Matlab® Controls Toolbox. The proportional and integral gains and the resulting performance metrics results are given in Table 3.2. Tests U1 and U2 overshot very quickly and had a very fast rise time because they turned the input all the way on very quickly and then off very quickly. As a result the one of the underdamped PI controllers (U1) was able reach the steady state conditions needed for a settling time, while the other take almost the entire test to settle. With the exception of the two underdamped cases, the peak overshoot was kept at a minimum. The overdamped cases (Tests O1 and O2) settled to within 2% of the setpoint but did not settle nearly as quickly as the critically damped cases (Tests C1 and C2), both of which had best performance of all the tests in this paper. Tests O1 and C1 are shown in Figs. 3.5(a) and 3.5(b). Fig. 3.5(a) shows the response for

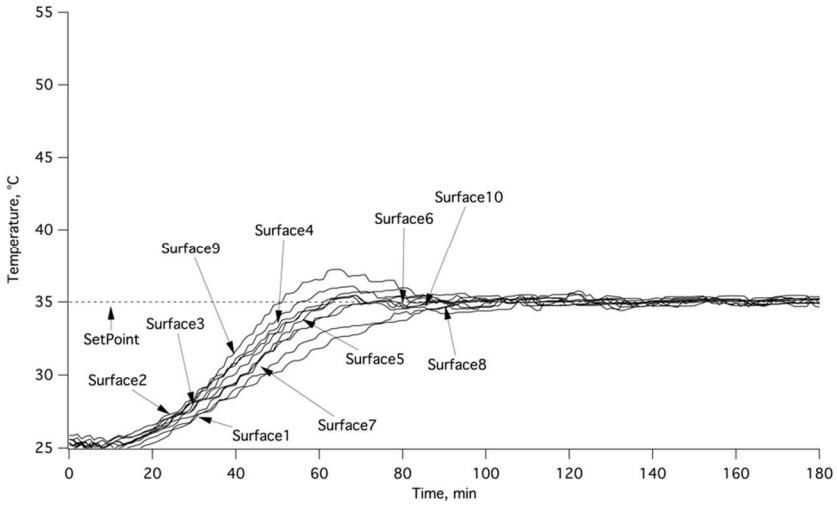
Test O1, where there is almost no overshoot and the design surfaces achieve their setpoint temperatures quickly with very little additional oscillation. Test C1 is shown in Fig. 3.5(b), where there was more overshoot than Test O1, but the design surfaces got around their setpoint temperatures quicker and had slightly larger oscillations around those setpoints. A successful test is one where the controllers actually achieve and maintain their set point temperatures with a reasonable degree of accuracy on all of the design surfaces, but that they actually find a solution that is maintained. Oscillating of the control action is deemed undesirable in that it does not find a solution. Tests RS2 and RS5 along with test U2 were shown to oscillate above and below the setpoint, never achieving the settling time criteria. Their inputs would drop to zero, making it difficult to maintain a solution once it was found.

Table 3.1. Resource sharing controller performance metrics.

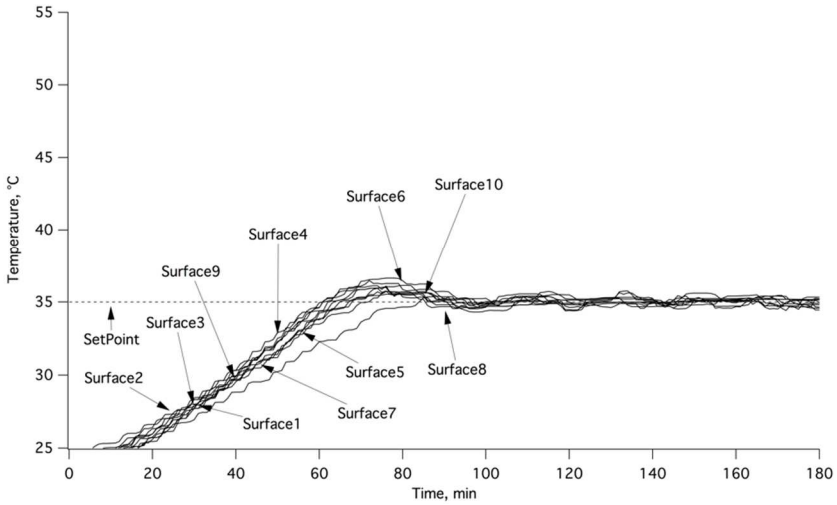
Test	Prob	Block Size	Rise Time (Min)	Peak Overshoot (°C)	Settling Time (Min)
RS1 (Base Case)	20%	0.00125	31	37.96	117
RS2	30%	0.00125	20	38.37	180
RS3	10%	0.00125	52	37.23	83
RS4	20%	0.0005	61	36.64	88
RS5	20%	0.0025	20	39.62	180



(a)



(b)



(c)

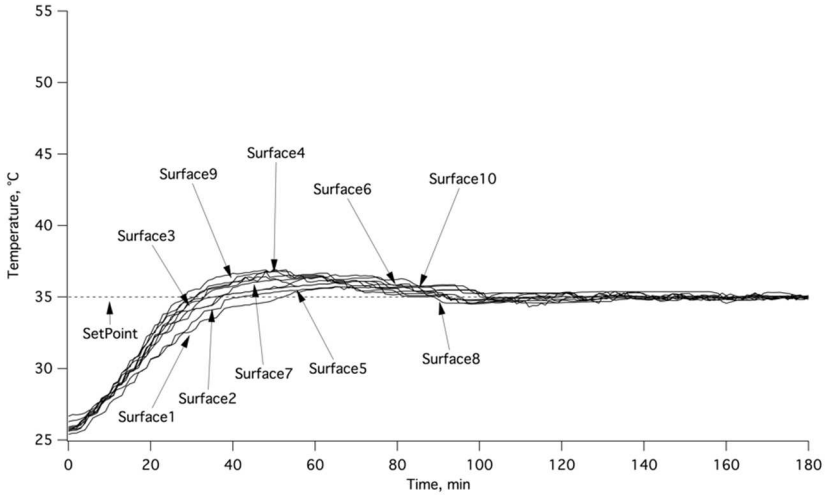
Figure 3.4. Temperature as a function of time using the resource sharing algorithm (a) Test RS1, (b) Test RS3, and (c) Test RS4.

3.6 Conclusion

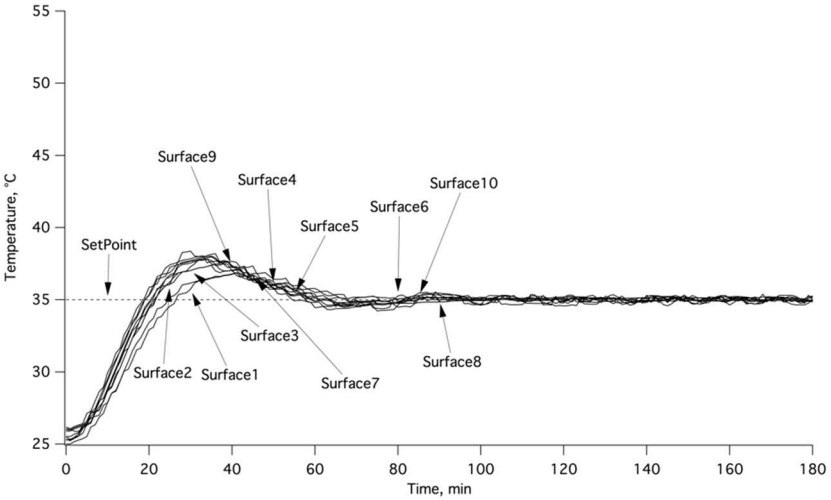
A new controls strategy based on stigmergy was successfully demonstrated on a complex, highly coupled controls problem based on two-dimensional radiant heat transfer between two set of design and heater surfaces and compared with traditional PI controls. Both the resource sharing strategy and the PI control strategy were able to maintain the settling time criteria and maintain steady state control of the design surface temperatures for all the tests except U2, RS2 and RS5. Based on settling time, the two best performing tests for the resource sharing control strategy (Tests RS3 and RS4) had faster settling times than both of the overdamped PI controller tests. Both of the critically damped cases had faster settling times than any of the resource sharing tests. Similar to the PI control strategy, when the block size was too large or if the probability of taking action was too large, the resource sharing control strategy was unable to settle the system to within 2% of the set point temperature for all surfaces.

Table 3.2. PI controller performance metrics.

Test	Proporional	Integral	Rise Time (Min)	Peak Overshoot (°C)	Settling Time (Min)
U1	2.5	0.5	2	52.19	84
U2	0.55	0.005	2	43.88	171
O1	0.02	0.8	29	36.95	95
O2	0.01	0.8	58	36.37	128
C1	0.04	0.8	19	38.67	58
C2	0.05	0.8	16	38.67	59



(a)



(b)

Figure 3.5. PI control temperature responses (a) Test O1, (b) Test C1.

In all of the tests using the PI controller, all of the controllers took identical actions at the beginning of each test, but those control actions gradually changed as the tests continued. The emergent, random behavior of the resource sharing control strategy means that not all the agents behave the same way at any point during a test. While this has the potential to be more time consuming, this behavior can lead to more robust and diverse solutions, while being adaptable to potential system changes.

The resource sharing control strategy offered comparable performance to a traditional PI controller without significant control equation tuning or using tuning information from the system response. Complex systems frequently require complex system models to describe their behavior accurately. Constructing the control equations requires a detailed understanding of the system dynamics, and the design of a sequence of experiments can be expensive both in time and cost. This resource sharing control strategy was able to control the system without constructing any control equations. The probability of action and block size were found through minimal empirical testing, which provided a method to find the appropriate values to achieve a fast but accurate response. Because the resource sharing control strategy does not require a predefined method for agents to coordinate their actions, additional agents may be added without changing the control strategy parameters.

Acknowledgments

This research was supported by the US Department of Energy – Office of Fossil Energy under Contract No. DE-AC02-07CH11358 through the Ames Laboratory.

References

Åström, K.J., Häggland, T., 1995. PID Controllers: Theory, Design and Tuning. ISA-The Instrumentation, Systems, and Automation Society, Research Triangle Park

Åström, K.J., Hagglund, T., 2006. Advanced PID control. ISA-The Instrumentation, Systems, and Automation Society, Research Triangle Park.

Åström, K.J., Albertos, P., Blanke, M., Isidori, A., Schaufelberger, W., Sanz, R., 2012. Control of Complex Systems. Springer Verlag, London.

Bakule, L., 2008. Decentralized control: An overview. Annu. Rev. Control 32(1), 87-98.

Beckers, R., Holland, O.E., Deneubourg, J.-L., 1994. From local actions to global tasks: stigmergy and collective robotics. In: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, pp. 181–189.

Bonabeau, E., Guérin, S., Snyers, D., Kuntz, P., Theraulaz, G., 2000. Three-dimensional architectures grown by simple "stigmergic" agents. Biosystems 56(1), 13–32.

Bonabeau, E., Dorigo, M., Theraulaz, G., 1999. Swarm intelligence: from natural to artificial systems. Oxford University Press, New York.

Bullock, S., Ladley, D., Kerby, M., 2012. Wasps, termites, and waspmites: distinguishing competence from performance in collective construction. Artif. Life 18(3), 267–290. doi:10.1162/artl_a_00065

Bussmann, S., Jennings, N.R., Wooldridge, M.J., 2013. Multiagent Systems for Manufacturing Control. Springer Verlag, Berlin.

Camazine, S., Deneubourg, J.-L., Franks, N., Theraulaz, G., Sneyd, J., Bonabeau, E., 2003. Self-organization in biological systems. Princeton University Press, Princeton.

Daneshfar, F., Bevrani, H., 2009. Multi-Agent Systems in Control Engineering: A Survey. J. Control Sci. and Eng. 2009(5), 1–12. doi:10.1016/S0921-8890(98)00085-2

Daun, K.J., Howell, J.R., Morton, D.P., 2003a. Design of radiant enclosures using inverse and non-linear programming techniques. Inverse Probl. Eng. 11(6), 541–560.

Finzell, P., Bryden, K., 2016. A Novel Resource Sharing Algorithm Based on Distributed Construction for Radiant Enclosure Problems. Adv. Eng. Softw., Submitted.

França, F.H.R., Howell, J.R., Ezekoye, O.A., Morales, J.C., 2003. Inverse design of thermal systems with dominant radiative transfer. Adv. Heat Transfer 36(1), 1–110.

Hadeli, Valckenaers, P., Kollingbaum, M., Van Brussel, H., 2003. Multi-agent coordination and control using stigmergy. Comput. Ind. 53(1), 75–96.

Holland O, Melhuish C., 1999. Stigmergy, self-organization, and sorting in collective robotics. Artif. Life 5(2) 173–202.

Howell, J.R., Ezekoye, O.A., Morales, J.C., 2000. Inverse design model for radiative heat transfer. J. Heat Trans-T ASME 122(3), 492–502. doi:10.1115/1.1288774

Kabanikhin, S. I., 2011. Inverse and ill-posed problems theory and applications. Walter de Gruyter, Berlin.

Karsai, I., Pénzes, Z., 1993. Comb building in social wasps: self-organization and stigmergic script. J. Theor. Biol. 161(4), 505–525.

Khaki-Sedigh, A., Moaveni, B., 2009. Control configuration selection for multivariable plants. Springer Verlag, Berlin.

Jennings, N.R., Bussmann. S., 2003. Agent-Based Control Systems - Why Are They Suited to Engineering Complex Systems? IEEE Control Systems Magazine, 23(3), 61–73.

Jevtic, A., Andina, D., 2007. Swarm intelligence and its applications in swarm robotics. In: Proceedings of 6th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics, pp. 41–46.

Lan, T., Liu, S., Yang, S.X., 2006. Collective cooperation inspired by stigmergy strategy. In: Proceedings of the Sixth World Congress on Intelligent Control and Automation, pp.441–445.

Leitão, P., 2009. Agent-based distributed manufacturing control: A state-of-the-art survey. Eng Appl Artif. Intel. 22(7), 979–991. doi:10.1016/j.engappai.2008.09.005

Levine, W.S., 2010. The control handbook. CRC Press, Boca Raton.

Lewis, F. L., Zhang, H., Hengster-Movric, K., Das, A., 2013. Cooperative control of multi-agent systems. Springer Verlag, London.

Lumkes, J.H., 2002. Control strategies for dynamic systems. CRC Press, Boca Raton.

Massioni, P., Verhaegen, M., 2009. Distributed control for identical dynamically coupled systems: a decomposition approach. IEEE T. Automat. Contr. 54(1), 124–135. doi:10.1109/TAC.2008.2009574

McArthur, S.D.J., Davidson, E.M., Catterson, V.M., Dimeas, A.L., Hatziargyriou, N.D., Ponci, F., Funabashi, T., 2007. Multi-agent systems for power engineering applications–part I: concepts, approaches, and technical challenges. IEEE T. Power Syst. 22(4), 1743–1752.

doi:10.1109/TPWRS.2007.908471

Melhuish, C., Welsby, J., Edwards, C., 1999. Using templates for defensive wall building with autonomous mobile ant-like robots. In: Proceedings of Towards Intelligent Mobile Robots.

Ogata, K., 2010. Modern control engineering. Prentice Hall, Upper Saddle River.

Petersen, K., Nagpal, R., Werfel, J., 2011. TERMES: An autonomous robotic system for three-dimensional collective construction. In: Proceedings of Robotics: Science & Systems VII.

Sandell, N.R., Jr, Varaiya, P., Athans, M., 1978. Survey of decentralized control methods for large scale systems. IEEE T. Automat. Contr., 23(2) 108–128.

Scattolini, R., 2009. Architectures for distributed and hierarchical model predictive control – a review. J. Process Contr. 19(5), 723–731. doi:10.1016/j.jprocont.2009.02.003

Sharkey A., 2006. Robots, insects and swarm intelligence. Artif. Intell. Rev., 26(4) 255–268.

Siljak, D.D., 2013. Decentralized control of complex systems. Dover, Mineola.

Theraulaz, G., Bonabeau, E., 1995a. Coordination in distributed building. Science 269(5224), 686–688. doi:10.1126/science.269.5224.686

Theraulaz, G., Bonabeau, E., 1995b. Modeling the collective building of complex architectures in social insects with lattice swarms. J. Theor. Biol. 177(4), 381–400.

Theraulaz, G., Bonabeau, E., Denebourg, J-L., 1998. The Origins of nest complexity in social insects. Complexity 3(6), 15–25.

Trentelman, H. L., Stoorvogel, A. A., Hautus, M., 2001. Control theory for linear systems. Springer Verlag, London.

Valckenaers, P., Hadeli., Germain, B., Verstraete, P., Van Brussel, H., 2007. MAS coordination and control based on stigmergy. Comput. Ind., 58(7) 621–629.

Vinyals, M., Rodriguez-Aguilar, J.A., Cerquides, J., 2011. A survey on sensor networks from a multiagent perspective. Comput J. 54(3), 455–470. doi:10.1093/comjnl/bxq018

Weiss, G., 2013. Multiagent systems. MIT Press, Cambridge.

Werfel, J Nagpal, R., 2006. Extended stigmergy in collective construction. IEEE Intell. Syst., 21(2) 20–28.

Wooldridge, M.J., 2008. An introduction to multiagent systems. Wiley & Sons, Hoboken.

CHAPTER 4. DYNAMIC SCALING OF A RESOURCE SHARING ALGORITHM

Peter Finzell, Kenneth M. Bryden[3]
Simulation Modeling and Decision Science Program
Ames Laboratory
1620 Howe Hall, Ames, Iowa, 50011

Abstract

Development of new control strategies for complex and highly coupled systems over a broad range of outputs frequently requires some adaptability in a control strategy. This research examines the response when two of the parameters of the resource sharing algorithm are scaled dynamically as a function of the error between the current value and the desired set point. The parameters that are dynamically scaled are the block size and probability of action being taken. The setpoint response of the resource sharing algorithm will be determined on an established physical test rig, based on an inverse radiant enclosure problem, where two parallel plates, separated by some distance are discretizing into ten surfaces. The temperature profile of the design surfaces is set indirectly by adjusting the spatial temperature profile of the heater surfaces. To evaluate the response of dynamics scaling, eight tests were conducted which determine the algorithm's effectiveness at achieving a desired uniform temperature profile along the design surfaces. Fours tests were conducted using probability scaling and four tests were conducted using block scaling. The resource sharing algorithm was able to reach the desired temperature profile in all of the tests examined. This new implementation of the resource sharing algorithm was also compared to a standard implementation of the algorithm. It was shown that dynamics scaling performed as well as the standard implementation and required less system knowledge beforehand.

Keywords: Stigmergy; Multi-Agent Systems; Bio-inspired; Distributed Control; Inverse Heat Transfer

[3] Corresponding author. tel +15154600875
Email address: kmbryden@iastate.edu

4.1 Introduction

Energy systems are becoming increasingly complex and require better control strategies to specifically address highly coupled, complex systems (Bakule, 2008). Highly coupled systems are not easily broken into subsystems, while complex systems are those where all of the system interactions are not easily characterized. In highly coupled systems multiple controller can have some effect of the same process variable, which means that there is a potential for conflicting control actions (Siljak, 2013). Control strategies must be able to maintain desired conditions, adjust to setpoint changes, and react to unpredictable disturbances. This requires a control strategy that is well tuned for a set of predetermined conditions but also one that is able to react to new conditions. The challenge is to create a control strategy that optimizes performance under normal conditions but can also adapt as the system and design variables change (Aström et al. 2012). Adaptive, intelligent or agent based control offer unique methods for controlling complex systems under changing conditions (Flynn, 2003). These control strategies frequently require extensive model development, offline learning and tuning, or explicit coordination between controllers.

This resource sharing algorithm was developed for highly coupled, complex systems and requires minimal system information beforehand to control a system. This algorithm was created using computational agents and designed to be flexible and scalable (Finzell and Bryden, 2016). Simple agents use changes made to the environment as a means of indirect communication, a shared resource, and a simple set of rules to coordinates actions. Agents are independent computational entities capable of sensing their environment and taken actions to alter that environment (Jennings and Bussmann, 2003; Wooldridge, 2009). As agents are created, each agent is allowed to make control decisions

at the local level without consulting with other agents. A shared virtual resource (blocks) serve as incremental units of change, which are distributed and redistributed among the agents and enables them to change their local environment.

In this paper, we expand upon the previously developed resource sharing algorithm by using dynamic scaling and test this implementation by applying the algorithm to an inverse radiant enclosure problem. Using dynamic scaling, either the probability of action or block size will be scaled as the current value approaches the desired value. While the resource sharing algorithm can be implement very quickly and without detailed control equations, dynamic scaling offers an even faster implementation than the conventional approach. It also may be utilized as a tuning mechanism to determine the optimal values for the standard resource sharing algorithm.

4.2 Background

The resource sharing algorithm is based off of the construction behavior of social insects (distributed construction). Social insects (bees, ants, termites and wasps) are known for coordinating their actions using only their local environment without direct control or individual instructions (Camazine et al., 2003). These insects are also recognized for their adaptability in how they react to unique environments and disturbances (Bonabeau et al., 1999). While the original resource sharing algorithm was inspired by the construction behavior of social insects, it is their adaptability that inspired dynamic scaling of the resource sharing algorithm i.e. when a social insect colony is disturbed, damaged, or their environment suddenly changes, the workers immediately take action (Sendova-Franks and Franks, 1999; Anderson and Bartholdi, 2000). There is no consultation or planning about how to adapt to the sudden change, yet in a very short time span the insects have adapted to the sudden change. Several models have been developed

attempting to determine the factors that go into decision making in social insect colonies (Theraulaz et al., 1991, Deneubourg and Franks, 1995). These models compare observed insect behavior (although limited) with the results from computational models, with varying degrees of success.

Colony tasks that individuals insect choose to undertake are based on their own specific preferences and experiences, which vary from insect to insect (Wilson, 1971). Additionally, choosing to execute specific tasks will vary based on each insect's preferences, physical attributes and experience level (Holbrooke, 2011). This behavior is called division of labor. One such model of division of labor assumes there is a threshold which must be exceeded for an individual insect to take action. In this model the probability of an insect taking action is based on the magnitude of the task stimulus and the probability of responding to the task when an insect is confronted with it (Bonabeau et al., 1997). It was this equation that served as the inspiration for the dynamic scaling equation, which is discussed later.

4.2.1 Control of Dynamic Systems

The resource sharing algorithm was previously implemented as a control strategy on a physical system that was created to replicate an inverse heat transfer problem (**Reference**). It was compared to a distributed controller to ascertain the potential benefits of this controller over traditional control strategies. To understand any additional benefits dynamic scaling may offer, traditional control strategies are examined.

PID control (proportional, integral and derivative) is a distributed control strategy commonly used in industry. It involves creating an independent controller for each state variable. Creating a transfer function either from an observed response or from first principle equations, a

PID controller relates a system input to an output (Aström & Hagglund, 1995). The gains of a PID controller can be tuned to achieve a desired response. That can mean either minimizing settling time, rise time or overshoot, usually changing one response characteristic at the expense of another. The proportional gain is amount of instantaneous proportional change to the input as function of the steady state error, the integral gain is the response of the system based on previous information and the derivative gain is the response of the system based on the expected future value of the steady state error. Using all three in conjunction, this algorithm is able to reduce the overshoot and settling time considerably (Aström & Hagglund, 2006). PID controllers use a single sensors and a single actuator and at any time step the sensor obtains current system state, evaluates the error based on the distance from the set point and takes the appropriate control action based on the previously tuned gains to move the current state closer to the set point (Ogata, 2010). They are frequently tuned over a specific operating range to produce an optimal response under those conditions. If the tuned gains only apply to a narrow operating window, then adaptive gain scheduling can be used, where several sets of gains are used under different conditions, which allows the controller to operate in a wider operating window (Levine, 2010). Adaptive gain scheduling is a control strategy in the category of adaptive control (Ioannou, 2012). Most control strategies that are tuned for specific operating condition and work well as long as those conditions stay relatively constant. When new conditions are encountered they are frequently not flexible enough to accommodate for those changes. Adaptive controls have the ability to change some or all of the control parameters during operation (Aström and Wittenmark, 1995). Adaptive control is a control strategy that changes as the system changes. These controllers trade off optimal performance over a limited set of conditions, for adequate performance over a much broader range of conditions.

Intelligent control seeks make informed control decisions that mimic human intelligence and learning. Intelligent controls focus on an efficient representation of information. These controllers typically deal with fuzzy logic or neural networks (Bubnicki, 2005). They focus on using a subset of artificial intelligence to generate protocols and algorithms for control. The main attraction to these types of strategies is they are able to be less "crisp" and more abstract than other control strategies. This allows them a great deal of flexibility in their representation of system parameters that may be continuous rather than discrete (De Silva, 1995). With fuzzy controllers, each variable is connected to linguistic variable, which is continuous and descriptive rather than absolute. Each state variable is expressed in terms of its association to that linguistic variable. Relationships are built among linguistic variables and can then be mapped back to a real variable for control decisions. Neural Networks are based on the way neurons interact within the central nervous system. A set of artificial neurons is used for information processing and computation. They are able to make changes to their structure during a learning phase. This learning allows the control algorithm to adapt and make decisions based on a limited set of information (Levine, 2010).

Agent-based controls offer a significant increase in system flexibility that has the potential to effectively manage complex, highly coupled system (Leitão, 2009; Lewis et al., 2013). Since agents based system allows for agents to easily be added or removed, this approach potentially offers many more computational entities than in other conventional approaches. The roles or abilities can differ among agents, based on their operating environment or specific goals (Valckenaers et al., 2007). These roles can be redundant or complimentary, which allows agent based systems to be more flexible, robust and scalable (Hadeli et al., 2004). Agent based controls focuses on the autonomy given to individual agents to make independent decisions and take action, while providing some means of coordination among agents.

4.3 Resource Sharing Algorithm

The resource sharing algorithm previously developed (Finzell and Bryden, 2016) gradually builds up a solution, through the distribution and redistribution of a shared resource, using computational agents that make changes to their environment using that resource, or blocks, which serve as an incremental unit of change. The value of these blocks can change and can be varied. This resource sharing algorithm can be applied to any situation where a desired global state is specified, but the inputs required to get to that state are unclear. Each agent is not given specific instructions as to how to achieve their local state but will take blocks from and give blocks back to a repository until their desired local state is achieved. It is through this frequent, independent, and random manipulation of the environment by a group of agents acting independently and the use of a shared resource that the desired global state is achieved.

During each time step two random numbers are created, one to randomly select an agent out of the group and one randomly determines if that randomly selected agent can take an action based on a user defined probability. This probability can be raised or lowered based on how often the user wants action to be taken and is called the probability of action. The agent that was selected then makes a decision to take a block, give back a block, or keep all their current blocks, based on a simple set of rules. Blocks are the incremental unit of change made to an input, which can be increased or decreased depending on the situation.

4.4 Dynamic Scaling

Dynamic scaling changes the probability of action or block size based on the steady state error between the current value and the desired set point, where the block size or probability of action is greater when the current value is further away from the desired value and decreases as

the current value approaches the desired value (Fig. 4.1). In this case both the block and probability scaling techniques uses the difference between the current temperature and the setpoint temperature to scale each parameter respectively. Both scaling techniques use the same equation to determine a scaling factor based on the steady state error. The scaling equation, given below, is used to determine this scaling factor for both block scaling and probability scaling, was inspired by an equation used to model social insect behavior (Bonabeau et al., 1996; Bonabeau et al., 1997). X is the steady state error and alpha is a user selected parameter that determines the magnitude of the scaling factor given that the steady state error remains constant.

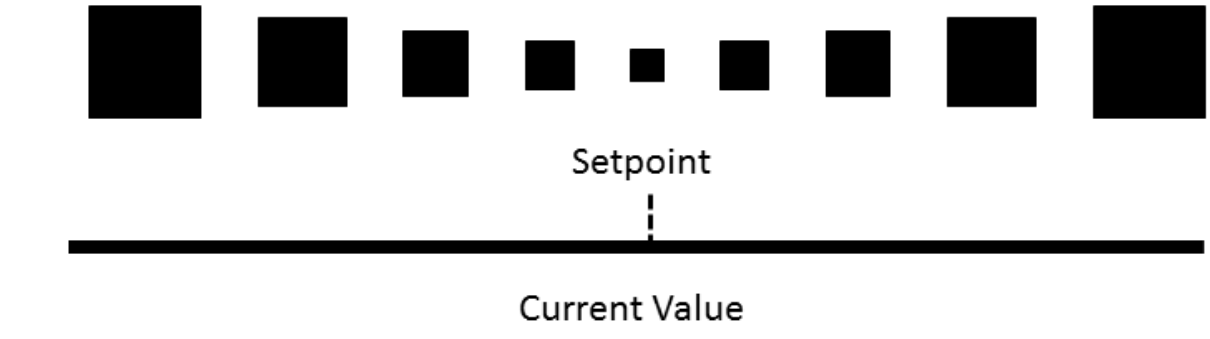
$$X = \left|\text{Current-Setpoint}\right| \tag{4.1}$$

$$\text{Scaling Factor} = \frac{X^2}{(X+\alpha)^2} \tag{4.2}$$

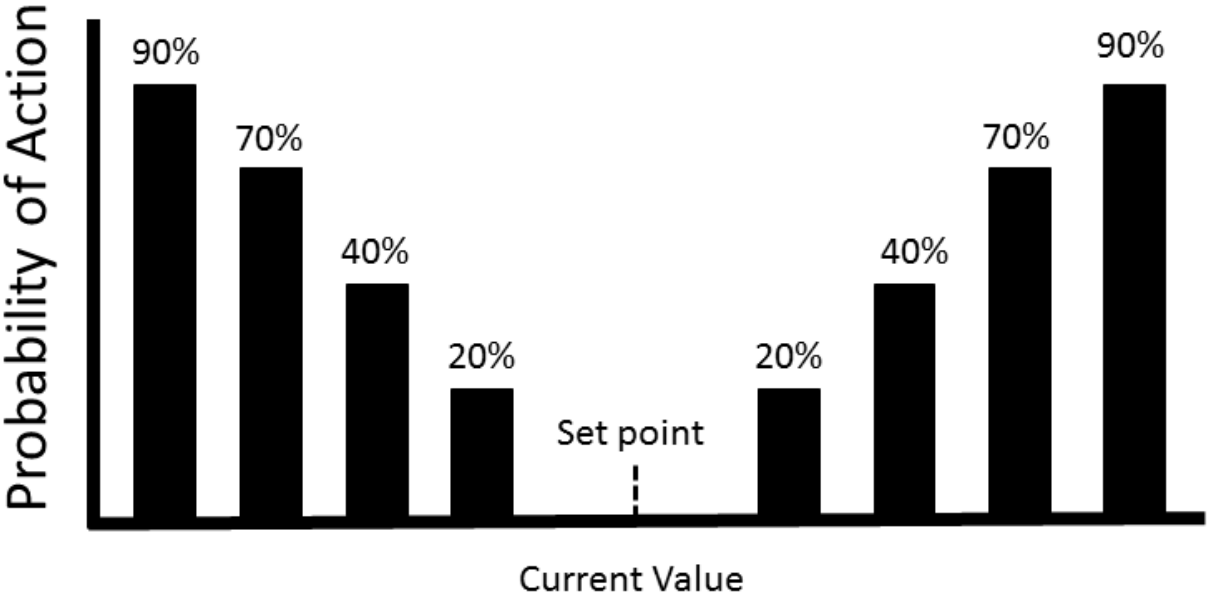
$$\text{Scaled Block} = \text{ScalingFactor} \times \text{Intial Block Size} \tag{4.3}$$

$$\text{Scaled Probability} = \text{ScalingFactor} \tag{4.4}$$

This scaling factor is a value that is either used directly as the current probability of action or is multiplied by an original block size to determine the current block size. For block scaling there is an original block size, which the current block size can never exceed. The original block size is multiplied by a scaling factor to determine the current block size. The effect is that as the current value moves further away or closer to the desired set point, the current block size is scaled accordingly. At any point during the block scaling tests, the instantaneous block size cannot exceed the original block size. If steady state error is very high, the instantaneous block size will approach the original block size. For probability scaling, the scaling equation is used directly as



(a)



(b)

Fig. 4.1a-b Block scaling and probability scaling.

the current probability of action. As the current temperature approaches the setpoint, the taking action (giving or taking a block) decreases and increases as the current value moves away from the setpoint.

4.5 Problem Setup

Dynamics scaling of the resource sharing algorithm is explored using a physical system based an inverse radiation heat transfer problem (Fig. 3.1). The problem used to evaluate the response of the dynamically scaled algorithm is an inverse radiant enclosure problem. In this problem there are ten heater surfaces, each of which are directly above ten design surfaces separated by a distance of 7.62 cm, which enabled every heater surface to have some effect on every design surface, shown in Fig. 3.2. The center heater surfaces have a greater effect on more of the design surfaces than the outer most design surfaces. The heater surfaces are composed of radiant emitters that are directly above aluminum plates of the same size, each with a thermocouple to determine their respective surface temperatures. A more detailed explanation of this physical system is given in (**Reference**). The temperature of the design surfaces is adjusted by the effective power supplied to each heater surfaces, which is controlled by pulse width modulation or PWM. The PWM has a duty cycle which determines the period over which the power is turned off and on, as shown in Fig. 3.3, changing the duty cycles adjusts the width of a pulse.

Each agent in this system is composed of a single heater surfaces and a single design surface, with each agent only able to sense the temperature of the design surface directly below it (Fig. 2.3). Each agent uses a decision network to decide when to request more blocks or give back blocks and will take action until its desired state is met.

The block size or the frequency block given actually being given can be adjusted and in this case, they are scaled as the current value changes. The block size in this instance corresponded to an incremental change to the input to each heater.

4.6 Results

The two parameters that will be dynamically scaled are the block size and probability of action. The block size determines the change in the PWM duty cycle on an individual heater surface based on the addition or subtraction of a single block. The probability of action determines how often an agent is randomly selected to make a decision about whether to give, take or keep constant its current level of blocks.

In this instance, eight tests were examined, fours probability scaling tests and four block scaling tests. Recalling that the alpha changes the magnitude of the scaling factor for identical steady state errors, the alphas for the probability scaling tests ranged from 0.5–3 and the alphas for the four block scaling tests changed from 0.25 to 2. For each test, both the design surfaces and heater surfaces were initially at room temperature (26°C) and each agent had a goal to achieve and maintain a desired temperature profile on their design surfaces, in this instance a temperature of 35°C, by adjusting the temperature of their heater surface. Achieving their desired design surface temperatures means achieving the goal temperature within a user-defined tolerance of the setpoint value. The tolerance is the upper and lower bounds that the current value can be within and still be considered at the setpoint.

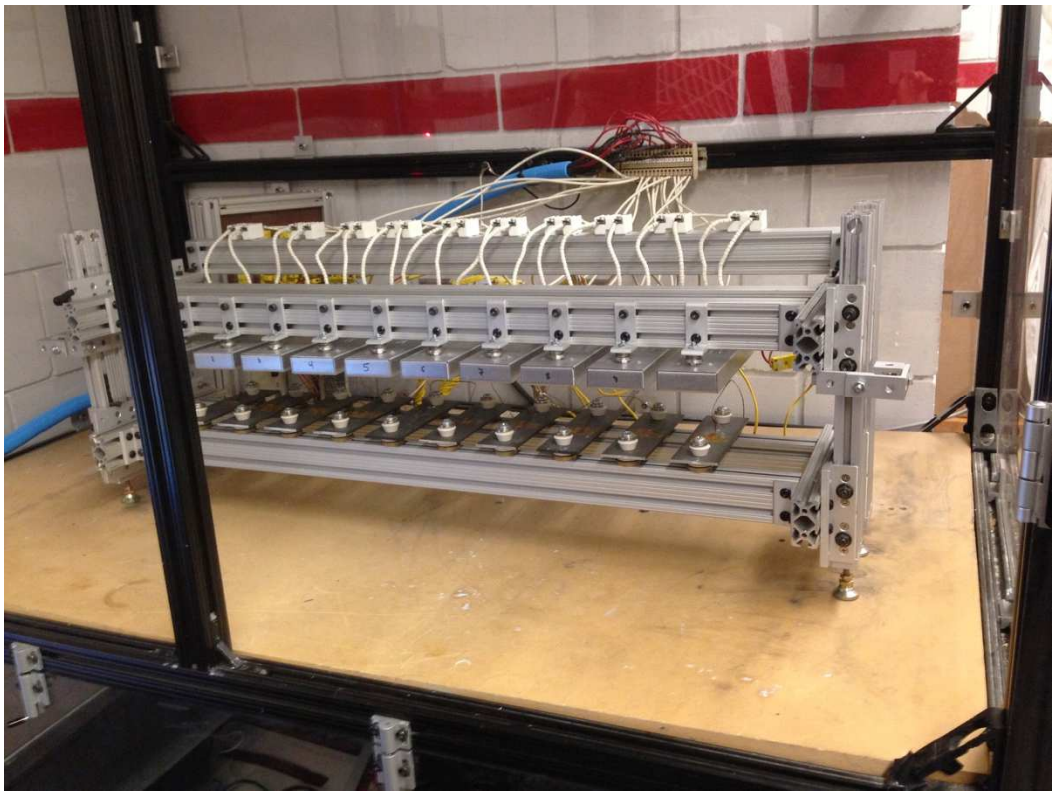
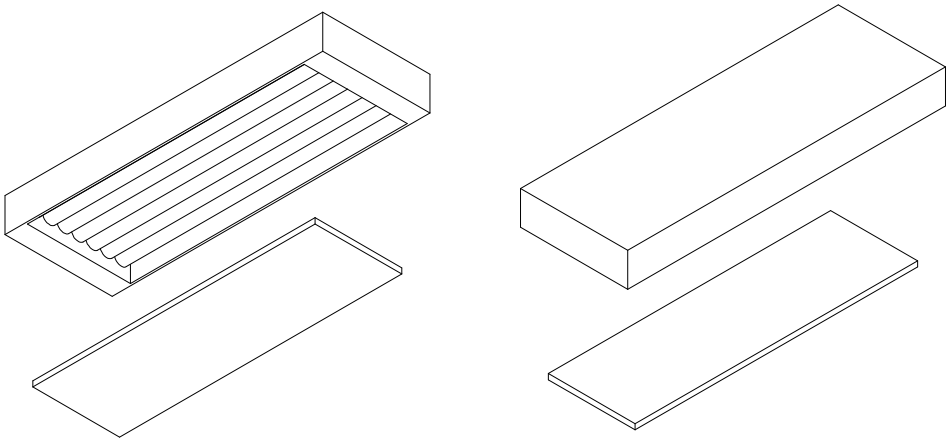
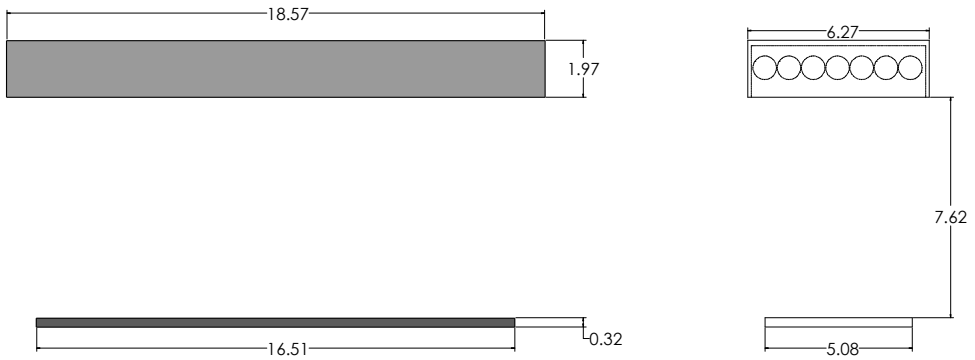


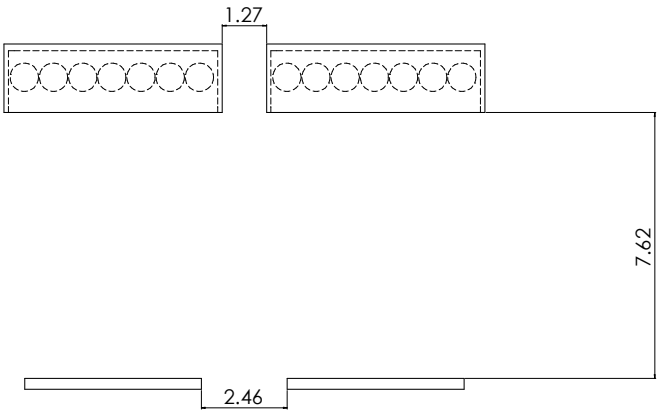
Figure 3.1. Testing rig.



(a)



(b)



(c)

Fig 3.2. Heaters and surfaces (a) Single heater and surface, (b) Single heater and surface dimensions, (c) Heater and surface spacing.

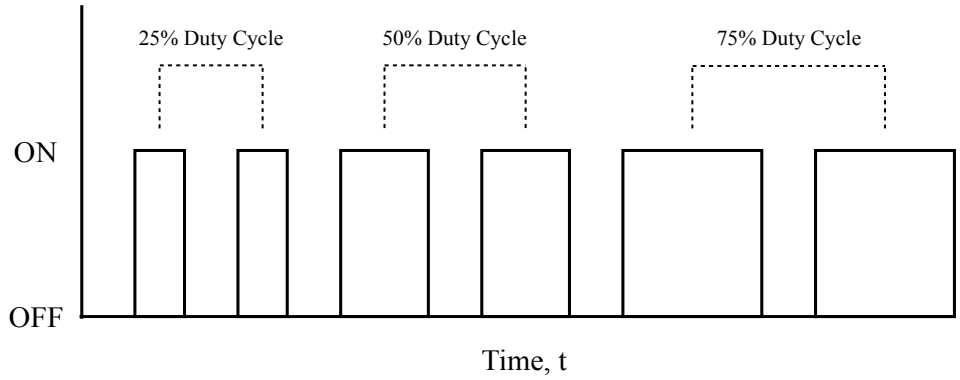


Figure 3.3. PWM diagram.

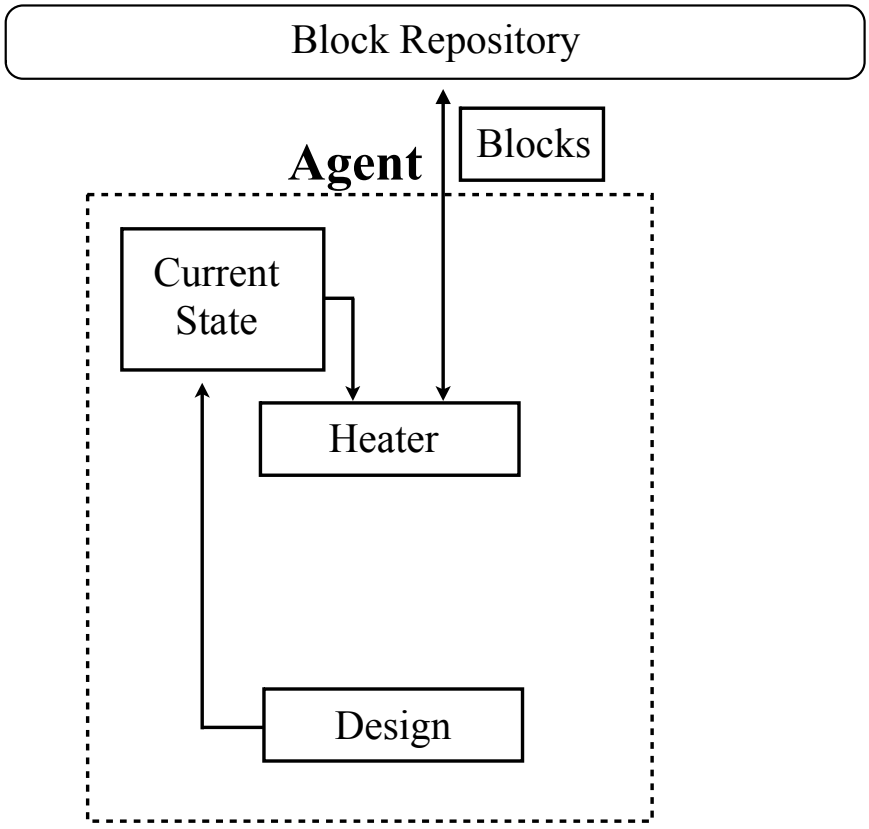


Figure 2.3. Agent composition.

Selecting an appropriate tolerance is critical as it is needs to strike a balance between the inherent system noise and the desire accuracy of the final temperature of the design surface. The tolerance was determined empirically and was selected to be 0.2°C. The slow response of the heaters and subsequently the design surfaces means the testing time required was fairly large (3 hours).

Traditional control metrics where used to compare the responses for the dynamics implementation of the resource sharing algorithm. The rise time is the time for the first design surface reach the vicinity of the setpoint. The settling time is how long it takes for all of the surfaces to be within 2% of the set point temperature and stay within those bounds for the remainder of the test. The peak overshoot is the highest single design surface temperature observed during the test. If all of the design surfaces have similar setpoint responses then the peak overshoot if sufficient to evaluate the overall overshoot behavior. If the overshoot responses between are considerably different, the overshoot trends among the design surfaces is noted.

4.6.1 Probability Scaling Tests

In the probability scaling tests, the scaled probability of action is computed directly using the scaling equation. Four difference alphas (0.5, 1, 2, 3) were used, each of which changes the magnitude of the probability of action, assuming similar conditions. For the probability scaling tests, a fixed block size needs to be specified and a block size of 0.00125 was used for all the tests. Table 4.1 show the overshoot, rise time and settling time of the four test cases examined using probability scaling. Fig. 4.2–4.5 gives the response for all ten design surfaces as the alphas increased from 0.5–3. As the alpha increases, the magnitude of the probability of action decreases, assuming similar conditions.

Test P1, with a scaling factor of 0.5, had the one of the fastest rise times and a settling time of the probability scaling tests as shown in Fig. 4.2. This fast rise time also produced some oscillations around the setpoint but most of those oscillations were within the settling time criteria and they didn't adversely affect the settling time response. This test had the most overshoot of all the probability scaling tests but was only 2°C higher than any of the other probability scaling tests. As the most radiation heat transfer goes to the center design surfaces (surface 5 and 6) and least goes to the outside design surfaces (surface 1 and 10), design surfaces 5 and 6 had the fastest settling times while surfaces 1 and 10 took the longest to settle.

An alpha of 1 was used in test P2, as shown in Fig. 4.3. This test had less overshoot than test P1, but the rise time and settling times increased slightly. Although there was less overshoot

Table 4.1. Probability scaling tests.

Test	Block Size	Alpha	Rise Time (Min)	Peak Overshoot (°C)	Settling Time (Min)
P1	0.00125	0.5	26	39.12	75
P2	0.00125	1	29	38.26	86
P3	0.00125	2	31	37.24	91
P4	0.00125	3	51	37.05	128

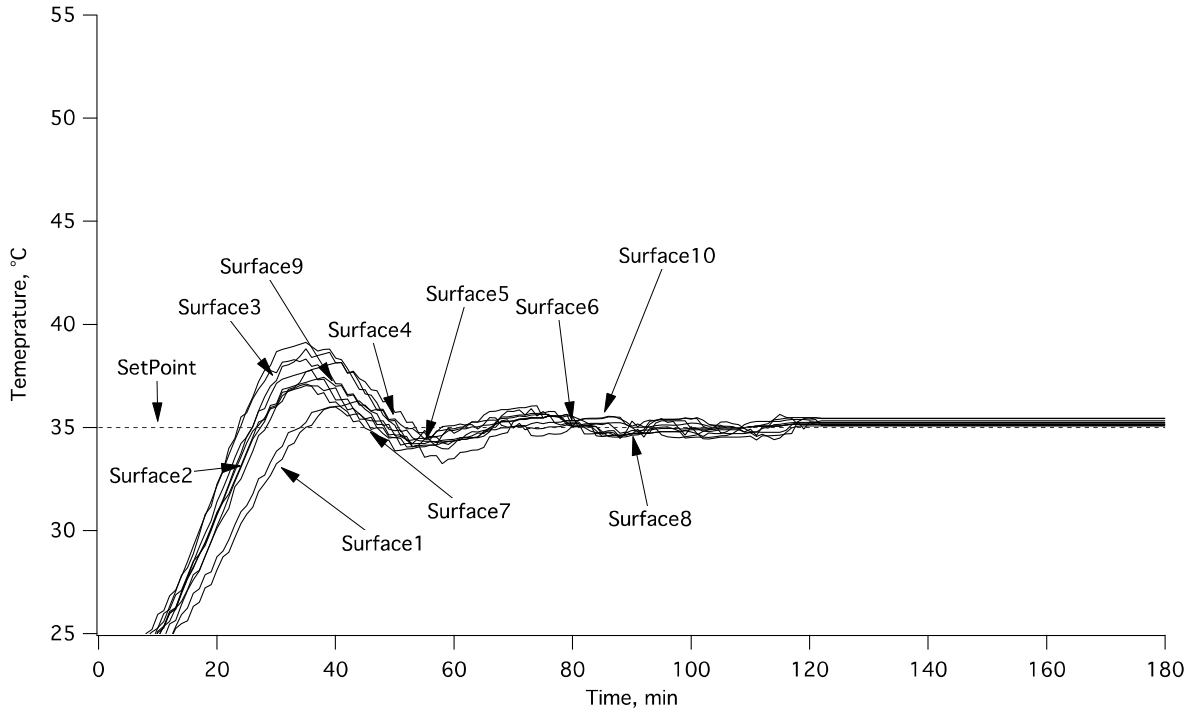


Figure. 4.2 Probability scaling test 1.

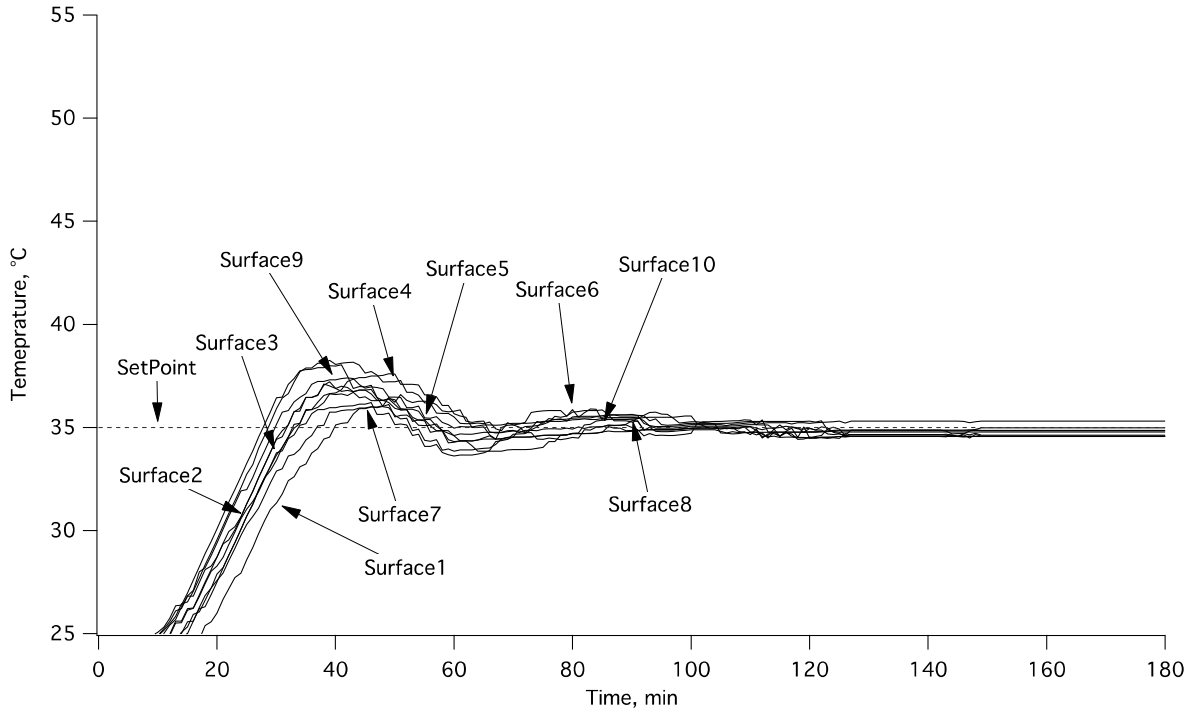


Figure. 4.3 Probability scaling test 2.

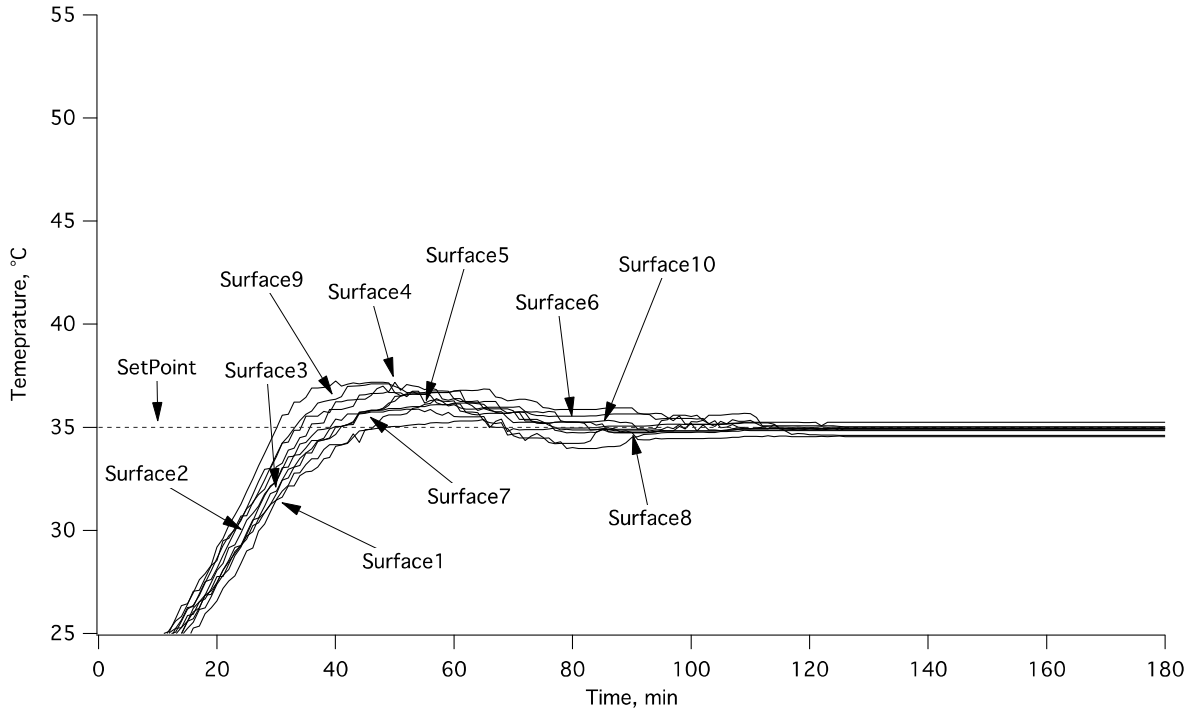


Figure. 4.4 Probability scaling test 3.

than test P1, there were still some oscillations around the setpoint, but like test P1, the oscillations were mostly within the settling time criteria and test P2 had the second fastest settling time. The overall probability of action was lower for test P2 than test P1 at any point during the test, given the same steady state error.

In test P3, there was a drop in the peak overshoot and general overshoot of all the surfaces decreased considerably (Fig. 4.4). With an alpha of 2, the magnitude of the probability of action decreased, which means fewer actions were taken overall. There were almost no oscillations around the setpoint but there was an increase in the settling time and the rise time. Because of the slower response, the difference settling times between the inside surfaces (surface 5 and 6) and the outside surfaces (surfaces 1 and 10) was much less pronounced than in test P1.

The final probability scaling test (P4) used an alpha of 3, which decreased the overall probability of action assuming similar steady state error. While the peak overshoot didn't decrease from test P3 to test P4 that was because design surface 5 was considerably hotter than the rest of the surfaces. All of the other design surfaces had lower overshoot than in test P3. Shown in Fig. 4.5, the response was overall much slower than any of the previous tests. While the peak overshoot was similar to test P3, but the rise time and settling time increased significantly. There were no oscillations which is due to an initially a very low probability of action that got smaller as the test progressed.

4.6.2 Block Scaling Tests

Block scaling requires an original block size, from which the instantaneous block size can be scaled down using the scaling equation. The original block size for all of the block scaling tests was 0.002. The block scaling tests also need a probability of action to be specified and a probability

of action of 0.8 was used. Four block scaling tests were performed using alphas of 0.25, 0.5, 1 and 2. While the instantaneous block size will always decrease as the current value approached the desired setpoint, increasing the alphas decreases the instantaneous block size, assuming similar conditions. Changing the alpha means the block size will be higher both at the start of the test (all design surfaces are at 26°C) and when they are at their setpoint temperature of 35°C. The overshoot, rise time and settling time for the block scaling tests are given in Table 4.2. Fig. 4.6–4.9 shows the response of all ten design surfaces varying the alphas and as they move towards their new setpoint of 35°C. An alpha of 0.25 was used in test B1, as shown in Fig. 4.6. This test had the most overshoot in all of the block scaling tests but had the fastest rise time and settling time.

Table 4.2. Block scaling tests.

Test	Prob	Block Size	Alpha	Rise Time (Min)	Peak Overshoot (°C)	Settling Time (Min)
B1	0.8	0.002	0.25	56	37.16	92
B2	0.8	0.002	0.5	56	36.77	101
B3	0.8	0.002	1	64	36.73	110
B4	0.8	0.002	2	76	36.43	180

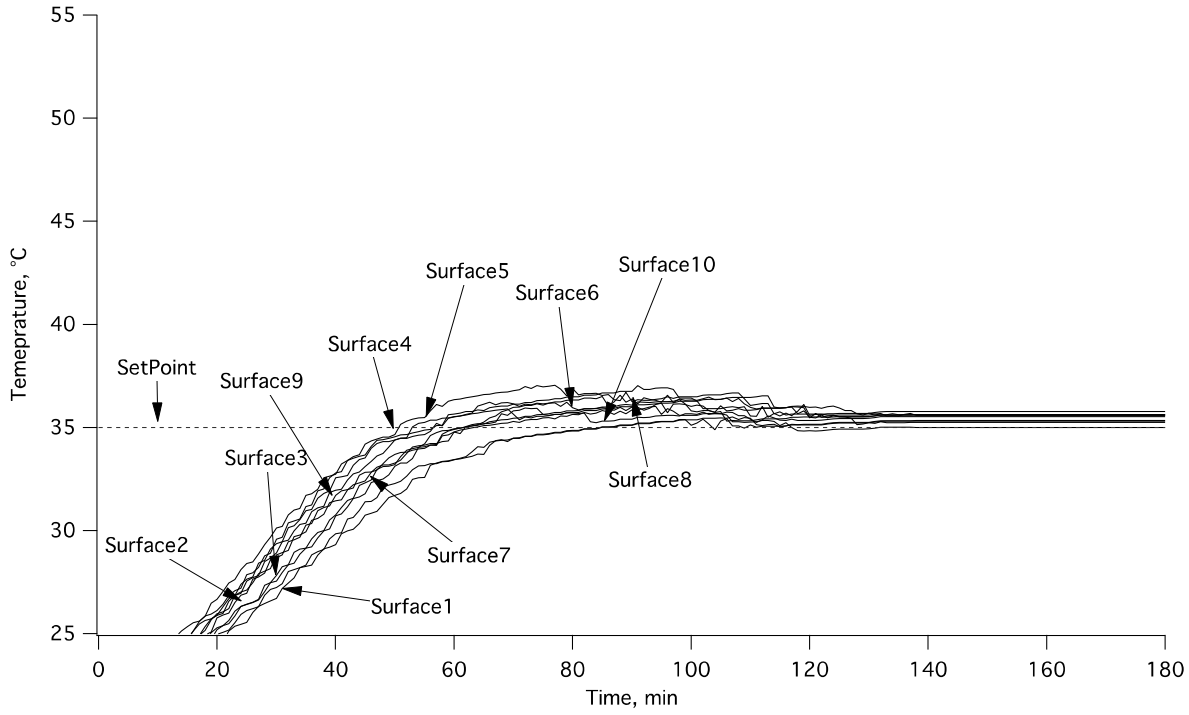


Figure. 4.5 Probability scaling test 4.

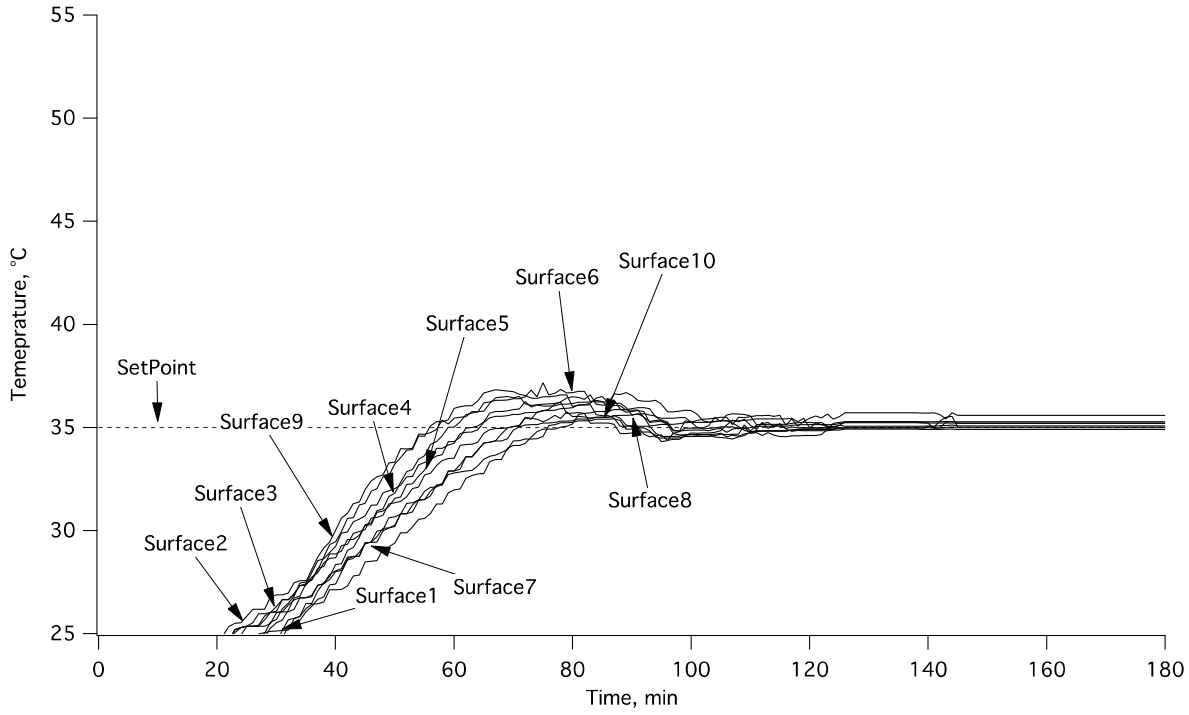


Figure. 4.6 Block scaling test 1.

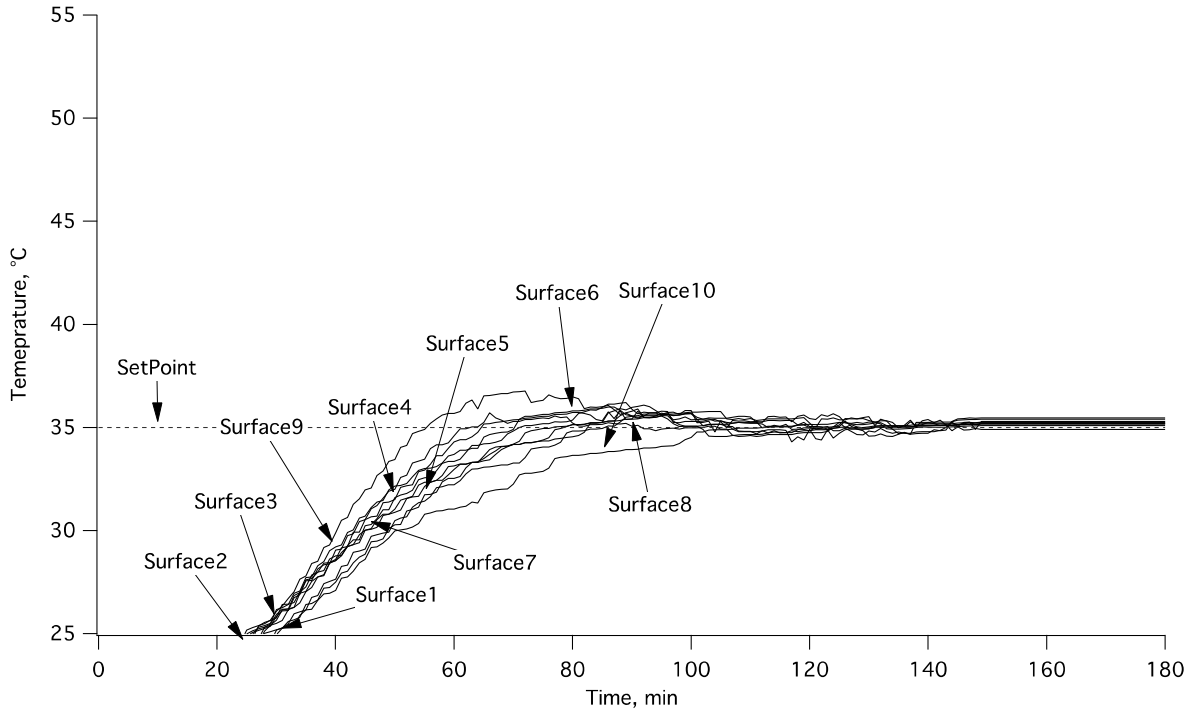


Figure. 4.7 Block scaling test 2.

There was a clear distinction into groups of design surfaces based on their relative location. Design surfaces 2 and 9, 3 and 8, 4 and 7, and 5 and 6 all had similar response rates. While the rise time and settling time were higher than any of the probability scaling tests, there was no oscillations around the setpoint. Using an alpha of 0.5 there was less peak overshoot in this test and much less overall overshoot with all the surfaces (Fig. 4.7). The rise time matched that of test B1, which was the fastest of all the block scaling tests. Design surface 9 had significantly faster rise time and peak overshoot than the rest of the surfaces, which single handedly increased the overall rise time and peak overshoot. Design surface 1 had a very slow rise time and settling time, which decreased the settling time from test B1, as the settling time is measured using all of the design surfaces. The general trend of the remaining design surfaces was there was a decrease in overshoot and they kept their settling times similar to test B1.

Most of the design surfaces in test B3 were more closely grouped than previous block scaling tests. As shown in Fig. 4.8, with the exception of surface 1, most of the design surfaces were had very similar responses for rise time and settling time to each other. The rise time increased from previous tests, as did the settling time. As the alpha increased, the initial effective block size was smaller than the previous block scaling tests, which made the effective block size around the setpoint smaller as well.

The final block scaling test, shown in Fig. 4.9, increased the rise time and the settling time criteria were never attained. As the alpha was increased to 2, the overall block size decreased so much around the setpoint, that the settling time criteria could not be achieved in the time given. There were no oscillations around the setpoint and the responses for all of the design surfaces were

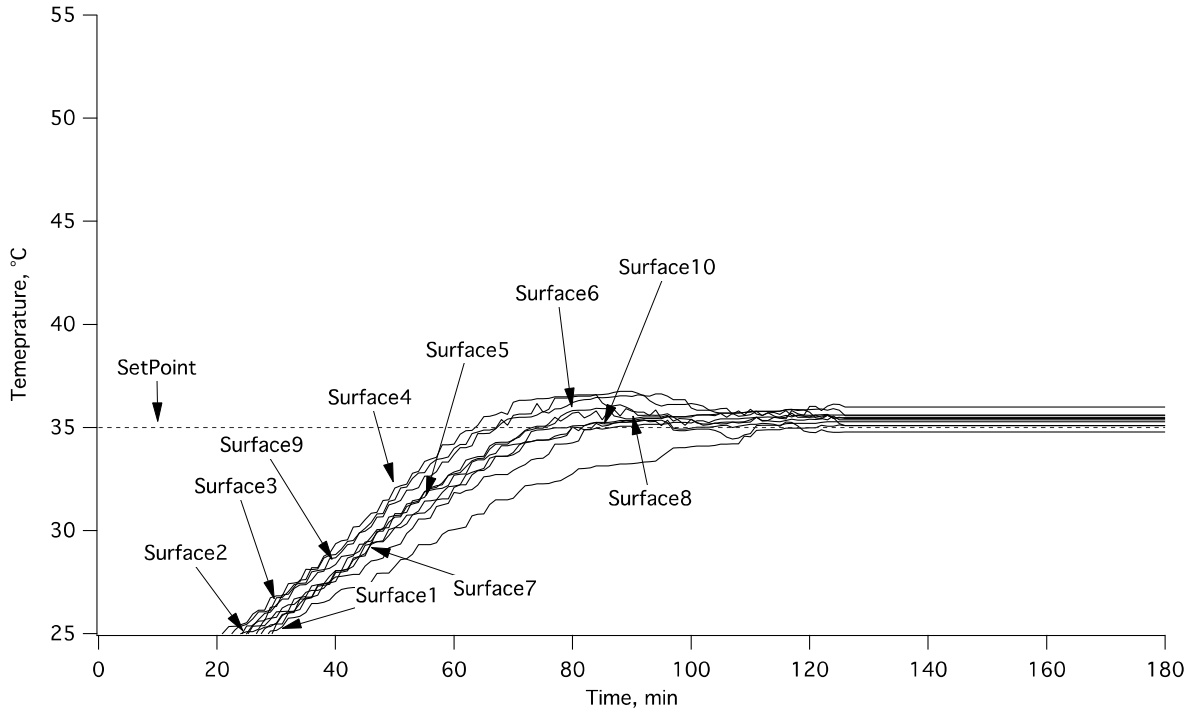


Figure. 4.8 Block scaling test 3.

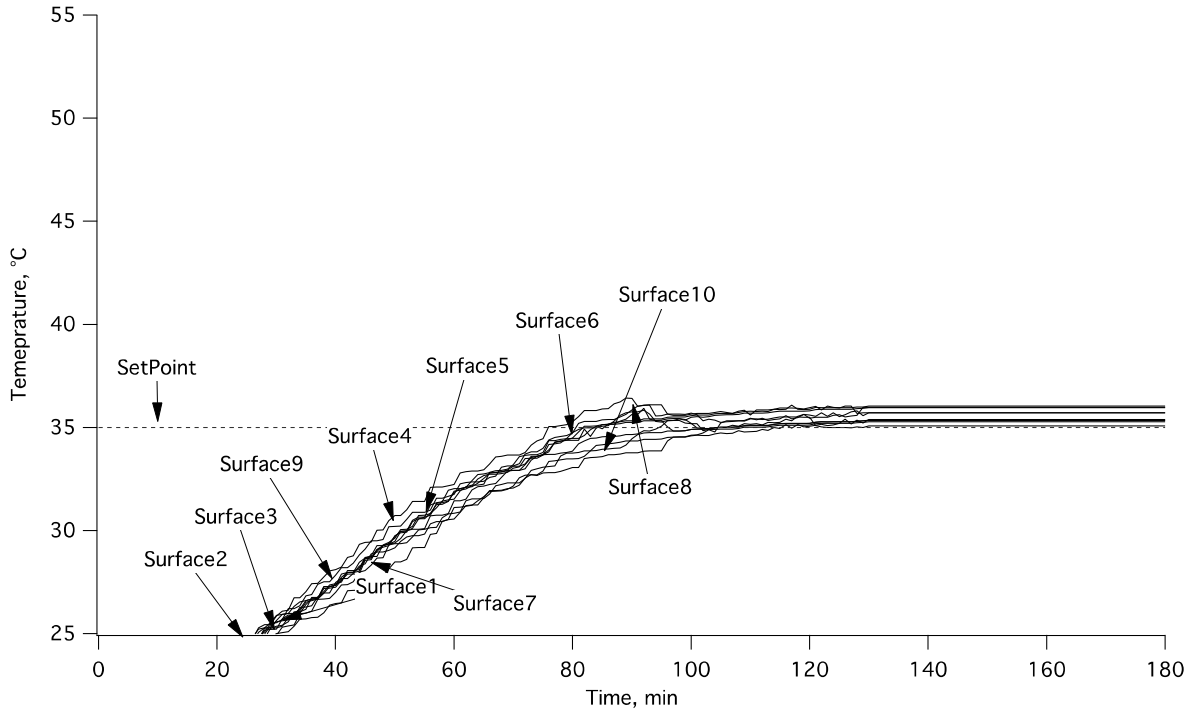


Figure. 4.9 Block scaling test 4.

very similar throughout the test. This test is an example of the maximum value of alpha, given an initial block size of 0.002, where the settling time criteria cannot be achieved.

In the block scaling tests, like the probability scaling tests, the general trend was that as the alphas increased the overshoot decreased, while the rise times and settling times increased. Some of the block scaling tests have less clear trends, where some of the design surfaces very different responses than the rest of the design surfaces. In the block scaling tests, the setpoint responses for the design surfaces was not as tightly grouped together as the probability scaling tests. One reason is that in the block scaling tests, at any given time, the block size is different both from agent to agent and from one time to another. The probability scaling tests will have different probabilities of action at any point in during a tests, but the block size remains the same in all the probability scaling tests. The rise time for all the block scaling tests was slightly higher than all the probability scaling tests and the settling time was noticeably higher than the probability scaling tests.

4.6.3 Standard Resource Sharing Algorithm and Dynamically Scaling

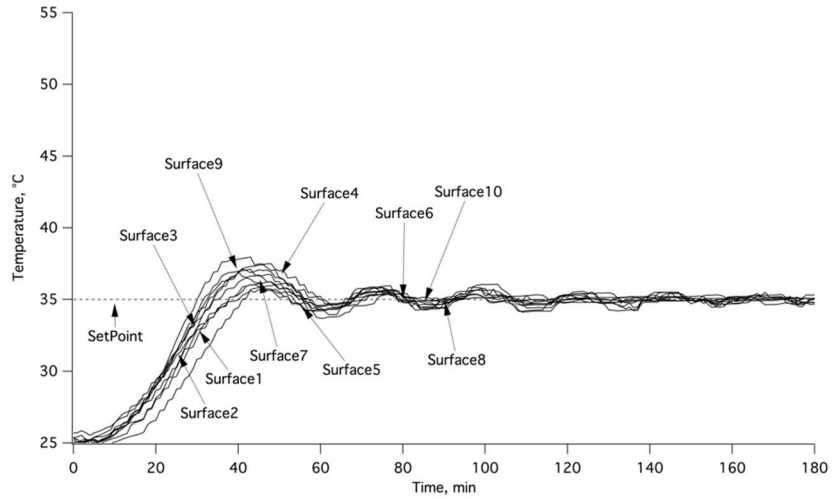
Previously, in a standard implementation of the resource sharing algorithm, the responses for five tests were examined (**Reference**). The two adjustable parameters, block size and probability of action were adjusted to determine which conditions gave the best response. In these tests the initial probability of action and block size were set and held constant through the test. Just like the tests conducted using dynamic scaling, the goal was to achieve a uniform temperature distribution of 35°C along the design surfaces in 180 min. The baseline values, which were used for comparison, were 20% probability of action and block size of 0.00125. In tests R1-R5, the block size or probability of action was changed slightly from the base case and the setpoint

responses are given in Table 3.1. The baseline test along with the two best responses (Tests R2 and R3) are given in Figs. 3.4(a)–(c).

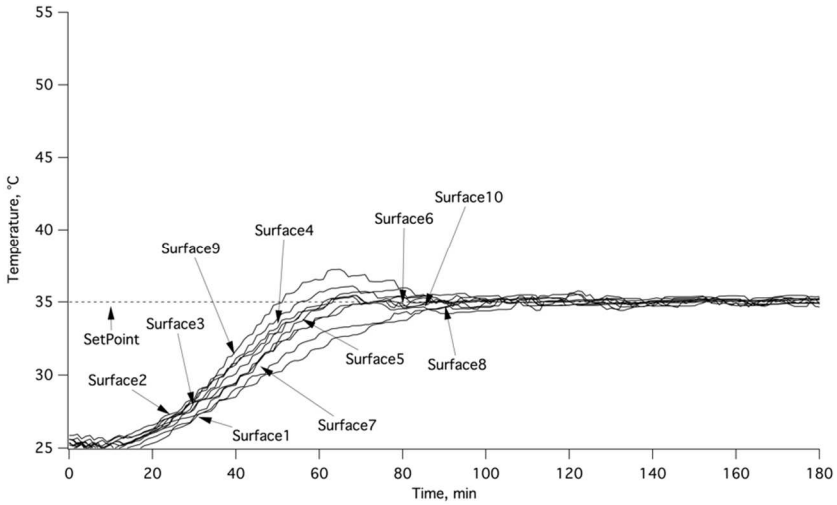
Comparing the dynamically scaled tests with the standard implementation of the resource sharing algorithm (block size and probability of action that remain static for the duration of the test) gives further insight any improvements using dynamic scaling and under what circumstances dynamics scaling might be used over a standard implementation. There were no instances where the block scaling outperformed the probability scaling, but several of the probability scaling tests gave better results than the standard implementation. The two best cases for the standard implementation of the resource sharing algorithm, R3 and R4 had low rise times and lower settling times.

Table 3.1. Resource sharing controller performance metrics.

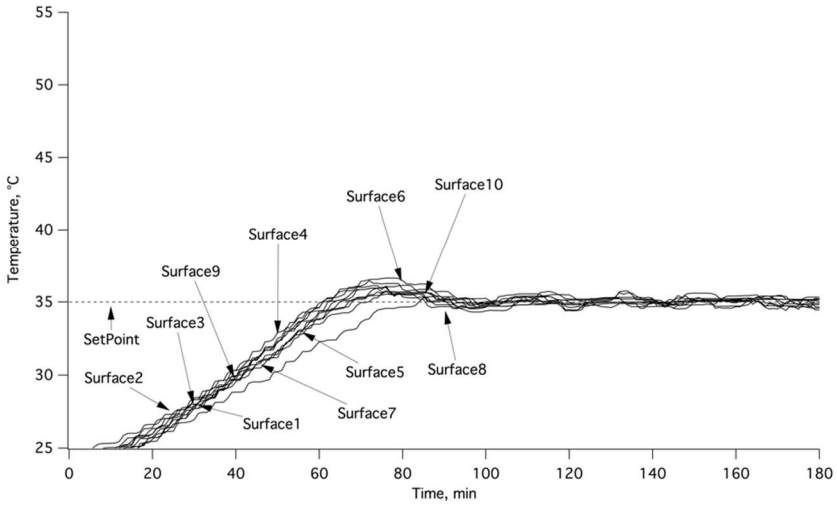
Test	Prob	Block Size	Rise Time (Min)	Peak Overshoot (°C)	Settling Time (Min)
R1 (Base Case)	20%	0.00125	31	37.96	117
R2	30%	0.00125	20	38.37	180
R3	10%	0.00125	52	37.23	83
R4	20%	0.0005	61	36.64	88
R5	20%	0.0025	20	39.62	180



(a)



(b)



(c)

Figure 3.4. Temperature as a function of time using the resource sharing algorithm (a) Test R1, (b) Test R3, and (c) Test R4.

The two best probability scaling tests, P1 and P2, had equally low rise times and even lower settling times than R3 and R4. The best instances of the block scaling did give somewhat of a better performance the worse cases of the standard implementation (R4 and R5). The best probability scaling tests (P1 and P2) can be said to outperform the best cases from the standard implementation (R2 and R3) of the algorithm because the two best cases had lower rise time and comparable settling times. While there were cases where the standard implementation had very low rise times, the tradeoff for those cases was very high settling times. All of the rise times for the probability scaling test, increased slightly but remained fairly low, while the settling times increased as the alpha increased.

4.7 Conclusions

Dynamic scaling of the resource sharing algorithm was successfully implemented on a radiant enclosure test rig. The response of eight tests was examined; fours tests using probability scaling and four tests using block scaling. These were compared with the standard implementation of this algorithm on the same test rig to determine the appropriate circumstances and potential benefits of using dynamic scaling over the standard implementation. Block scaling did not outperform the two best tests from the standard implementation but the two best cases from the probability scaling tests did although the finding in this paper may be system specific and there may be instances where block scaling will give the best response. The two best cases from the block scaling did outperform some of the tests from the standard implementation but they did not outperform any of the probability scaling tests.

Both the standard and dynamically scaled implementations of the resource sharing algorithm provide several advantages as a control strategy. The resource sharing controller can be

implemented much faster than traditional controllers because it development of detailed system models or equations beforehand. The dynamically scaled resource sharing algorithm requires specifying an additional variable, alpha, but one less variables for probability scaling (probability of action). The dynamically scaled resource sharing algorithm can be implemented even faster than the standard implementation, with less tuning. The decreased risk in picking a block size or a probability of action that is too big is minimized because any block size or probability of action will be scaled down as the current value approaches the setpoint. Dynamic scaling can be implemented in systems were there is a risk of damaging equipment since scaling offers some assurances that the response will be at least muted. Additionally, if the response from the standard implementation of the resource sharing algorithm is preferred, dynamic scaling could be used as a tuning mechanism for the probability of action or the block size.

Future implementations of dynamic scaling will be used to determine if it can offer a better performance over wider operating range for control in complex systems, in terms of both setpoint tracking and disturbance rejection. Future tests will involve testing dynamic scaling using different setpoints or comparing its ability to reject disturbances.

Acknowledgments

This research was supported by the US Department of Energy – Office of Fossil Energy under Contract No. DE-AC02-07CH11358 through the Ames Laboratory.

References

Anderson, C., Bartholdi, J.J., 2000. Centralized versus decentralized control in manufacturing: lessons from social insects. In: Proceedings Complexity and complex systems in industry pp. 92–105.

Åström, K.J., Häggland, T., 1995. PID controllers: Theory, Design and Tuning. ISA-The Instrumentation, Systems, and Automation Society, Research Triangle Park

Åström, K.J., Hagglund, T., 2006. Advanced PID control. ISA-The Instrumentation, Systems, and Automation Society, Research Triangle Park.

Aström, K.J., Wittenmark, B., 1995. Adaptive control. Addison-Wesley Longman, Boston.

Åström, K.J., Albertos, P., Blanke, M., Isidori, A., Schaufelberger, W., Sanz, R., 2012. Control of Complex Systems. Springer Verlag, London.

Bonabeau, E., Dorigo, M., Theraulaz, G., 1999. Swarm intelligence: from natural to artificial systems. Oxford University Press, New York.

Bonabeau, E., Sobkowski, A., Theraulaz, G., 1997. Adaptive task allocation inspired by a model of division of labor in social insects. In: Proceedings Biocomputing and Emergent Computation pp. 36–45.

Bonabeau, E., Theraulaz, G., Deneubourg, J.-L., 1996. Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. In: Proceedings Biological Sciences pp. 1565–1569.

Bubnicki, Z., 2005. Modern control theory. Springer-Verlag, New York.

Camazine, S., Deneubourg, J.-L., Franks, N., Theraulaz, G., Bonabeau, E., 2003. Self-organization in biological systems. Princeton University Press, New Jersey.

De Silva, C.W., 1995. Intelligent control. CRC Press, Boca Raton.

Deneubourg, J.-L., Franks, N.R., 1995. Collective control without explicit coding: the case of communal nest excavation. J. Insect Behav. 8(4), 417–432. doi:10.1007/BF01995316

Finzell, P., Bryden, K., 2016. A Novel Resource Sharing Algorithm Based on Distributed Construction for Radiant Enclosure Problems. Adv. Eng. Softw., Submitted.

Flynn, D., 2003. Thermal power plant simulation and control. Institution of Electrical Engineers, London.

Hadeli, Valckenaers, P., Kollingbaum, M., Van Brussel, H., 2003. Multi-agent coordination and control using stigmergy. Comput. Ind. 53(1), 75–96.

Holbrook, C.T., 2011.The emergence and scaling of division of labor in insect societies. PhD Dissertation. Arizona State University, Tempe, AZ.

Ioannou, P.A., Jing Sun, D., 2012. Robust Adaptive Control. Prentice-Hall, Upper Saddle River.

Jennings, N.R., Bussmann. S., 2003. Agent-Based Control Systems - Why Are They Suited to Engineering Complex Systems? IEEE Control Systems Magazine, 23(3), 61–73.

Leitão, P., 2009. Agent-based distributed manufacturing control: A state-of-the-art survey. Eng Appl Artif. Intel. 22(7), 979–991. doi:10.1016/j.engappai.2008.09.005

Levine, W.S., 2010. The control handbook. CRC Press, Boca Raton.

Lewis, F. L., Zhang, H., Hengster-Movric, K., Das, A., 2013. Cooperative control of multi-agent systems. Springer Verlag, London.

Ogata, K., 2010. Modern control engineering. Prentice Hall, Upper Saddle River.

Sendova-Franks, A.B., Franks, N.R., 1999. Self-assembly, self-organization and division of labour. Philos. T. R. Soc. B. 354(1388), 1395–1405. doi:10.2307/57032?ref=search-

Siljak, D.D., 2013. Decentralized control of complex systems. Dover, Mineola.

Theraulaz, G., Bonabeau, E., 1995b. Modeling the collective building of complex architectures in social insects with lattice swarms. J. Theor. Biol. 177(4), 381–400.

Theraulaz, G., Goss, S., Gervet, J., Deneubourg, J.-L., 1991. Task differentiation in polistes wasp colonies: a model for self-organizing groups of robots. In: Proceedings of the first international conference on simulation of adaptive behavior on From animals to animals pp. 346–355.

Trentelman, H. L., Stoorvogel, A. A., Hautus, M., 2001. Control theory for linear systems. Springer Verlag, London.

Valckenaers, P., Hadeli., Germain, B., Verstraete, P., Van Brussel, H., 2007. MAS coordination and control based on stigmergy. Comput. Ind., 58(7) 621–629.

Wilson, E.O., 1971. The insect societies. Belknap Press, Cambridge.

Wooldridge, M.J., 2008. An introduction to multiagent systems. Wiley & Sons, Hoboken.

CHAPTER 5. CONTROL OF A HYBRID POWER SYSTEM USING A RESOURCE SHARING ALGORITHM

Peter Finzell, Paolo Pezzini, David Tucker, Kenneth M. Bryden[4]
Simulation Modeling and Decision Science Program
Ames Laboratory
1620 Howe Hall, Ames, Iowa, 50011

Abstract

This paper implements a novel resource sharing control strategy on a fuel cell-gas turbine hybrid power system at the National Energy Technology Laboratory's Hyper facility. The Hyper facility combines a gas turbine with a fuel cell into a hybrid configuration. The combination of these systems into the Hyper creates a tightly coupled environment characterized by conflicting dynamics related to the simultaneous interaction of the gas turbine and fuel cell, which creates a challenging controls problem. In this paper we describe a novel approach to solving this tightly coupled controls problem by posing it as a resource sharing problem, which can be solved using a previously developed resource sharing algorithm based on distribution construction. This algorithm creates computational agents and solves the problem incrementally through the distribution and redistribution of a shared resource (blocks). Two agents were created; the first agent controls the gas turbine speed by adjusting the electric load, while the second agent controls the cathode mass flow through the fuel cell using the cold air bypass valve. The setpoint response was evaluated over the course of fifteen experimental tests for both agent 1 and 2. The algorithm was shown to have a comparable response to a previously implemented multi-input multi-output (MIMO) state space controller and was able to do so without creating equations or models used in traditional control strategies.

Keywords: Stigmergy; Multi-Agent Systems; Bio-inspired; Distributed Control; Hybrid Power

[4] Corresponding author. tel +15154600875
Email address: kmbryden@iastate.edu

5.1 Introduction

Current and future power plants that utilize fossil fuel will require higher efficiencies and lower emissions to provide for future power consumption needs while meeting higher regulatory standards. Hybrid power systems provide several advantages over standalone power generation systems and offer a unique approach to meet this challenge (DOE, 2015). Challenges arise in the implementation of hybrid power systems because of disparate components, differing time scales and tightly coupled systems. This makes controlling these advanced power systems challenging.

Recent technological advances have made individual sensors cheaper and more intelligent, providing an opportunity to involve more sensors and actuators in control strategies for advanced power systems (Maley, 2015). As the available computational power is increased at every level, control actions become more distributed, utilizing intelligent sensors that can take a more active role in controlling the system (Akyildiz et al., 2002). By utilizing as many sensors as possible, controllers can make more informed control decisions based on a more detailed perspective of the state of the system. Thus innovative control strategies are needed to effectively make use of a large number of intelligent sensors to make faster and more flexible decisions. In this paper, the previously developed resource sharing control strategy will be applied to a hybrid power system (Finzell and Bryden, 2016). The resource sharing control strategy will be compared to a state-space Multi-Input Multi-Output (MIMO) control strategy using standard performance metrics to determine the circumstances which it might be preferred over conventional controllers.

5.2 Background

In many power systems, supervisory control strategies generally use a centralized framework that utilize a large number of sensors to collect data and then direct the actions of a

small number of actuators using local controllers (Khaki-Sedigh and Moaveni, 2009). Such controllers are directly tuned for a single sensor and actuator, which are responsible for controlling a single system variable. In this case, lower level control decisions are provided and these controllers are tuned for optimal performance within specific operating regions. During transients or outside this operating window the performance can decrease dramatically. Single-input single-output controllers are generally known as proportional, integral and derivative (PID) controllers (Bakule, 2008). These controllers involve the creation of a transfer function that relate a system input to an output and are created from first principle equations or by observing the output response after a known input is given. While many distributed controllers can be used to control any number of desired system variables, if one or more of these controllers are implemented simultaneously, especially in complex and tightly coupled systems, conflicts between controllers can arise (Levine, 2010). In these systems, control actions taken by one controller can cascade throughout the system and those actions can affect other controllers (Åström and Hagglund, 2006). To avoid these potential conflicts, feed-forward techniques can be used, which are used to relate the control actions of one controller to other controllers before any control action is taken. However, feed-forward techniques are very system specific, designed specifically for operating around nominal conditions and can create problems at off-design conditions.

Centralized or multi-input multi-output (MIMO) control is able to alleviate some of the challenges associated with conflicting control actions, especially in tightly coupled systems (Khaki-Sedigh and Moaveni, 2009). MIMO control creates a matrix of the mathematical relationships between all of the system inputs and outputs, which is used to determine the appropriate control action or actions at any given time. Both centralized and distributed controllers are tuned off-line, and require an off-line retuning if the actual performance does not achieve the

desired performance during the experimental validation. With distributed control, as new sensors or controllers are added, the existing control strategy may need to be reworked (Siljak, 2013). Additionally, the relationship matrix of inputs and outputs for centralized controllers can become large and computationally complex to implement in real-time.

Adaptive control varies the controller gains or parameters during run time to accommodate changing system conditions. Since this control strategy can adapt during runtime, a limited amount of system information is needed beforehand (Aström and Wittenmark, 1995; Ioannou, 2012). Robust control deals with uncertainty and works as long as a set of uncertain parameters are within some bounded set (Aström et al., 2012). This control strategy requires sufficient information beforehand to create the bounding or varying parameters (Trentelman et al., 2001). Optimal control is based on optimization techniques, and seeks to provide the optimal control parameters to obtain the ideal system output (Bubnicki, 2005). These controllers are the basis for model predictive controllers and are often used when constraints have to be included in the optimization process. They operate well under specific conditions or around specific operating points as they require a detailed model of the system, and do not operate well under changing conditions.

Intelligent control strategies focus on an efficient representation of information and attempts to utilized and emulate human knowledge, reasoning, and learning. Contrary to adaptive control, where a controller will adapt to changes in the system, intelligent control seeks to reason through why the next control action should be taken. The main attraction to these types of strategies is they are able to be less "crisp" and more abstract than other control strategies. This allows them a great deal of flexibility in their representation of system parameters (De Silva, 1995). Learning and creating patterns from previous events builds relationships between inputs and outputs. This learning allows the control strategy to adapt and make decisions based on only a

limited set of information. Being able to learn from previous experience and modify a behavior to improve performance is what makes this control strategy unique but also challenging to implement (Levine, 2010). Frequently several of these control strategies will be compared against each other on a single system, to determine which offers the best performance for the expected conditions in that system.

5.3 Hybrid Systems

By combining conventional fossil fuel power plants with a secondary form of power generation, the resulting hybrid system may increase efficiencies, while lowering costs and emissions. A number of hybrid technologies have been proposed in which two or more energy conversion devices are coupled together e.g. fossil generation systems, renewables (wind generation, solar panels), fuel cells or energy storage devices (Bajpai and Dash, 2012). Several hybrid systems have been examined using wind or solar as the primary energy generation source and a hydrogen electrolizer to produce hydrogen gas from water, which can be stored for later use in a fuel cell. (Uzunoglu et al., 2009; Onar et al., 2008). A study was conducted examining different layouts for gas turbine and solid oxide fuel cell hybrid facility's demonstrated the diversity of possible configurations (Buonomano et al., 2015). Controlling hybrid power systems is a significant challenge as they are frequently tightly coupled, operate at different time scales, and each system may have different optimal operating conditions (Burch, 2001; Bizon, 2013). To manage these different components, hybrid systems often use supervisory or hierarchical controller that coordinates actions between components (Torres-Hernandez et al., 2008; Valenciaga and Puleston, 2005; Zerkaoui, 2012). At the component level centralized (multivariable) and distributed controllers are typically used (Carmeli et al., 2012).

The hybrid performance facility (Hyper), shown in Fig. 5.1, is a research project at Department of Energy's National Energy Technology Laboratory (NETL). The Hyper facility is a hybrid power system combining a gas turbine recuperation cycle and a virtual solid oxide fuel cell (SOFCs) stack (Fig. 5.2). This combination is significant because of a theoretical overall efficiency as high as 70% when run on natural gas (Tucker, 2002).

A gas turbine recuperation cycle consists of a compressor/turbine, a combustor, and a heat exchanger. Environmental air is compressed and run through the heat exchanger and warmed using turbine exhaust gas. This warm air is combined with fuel and combusted in the combustion chamber, which is used to spin the turbine to produce electricity. A conventional solid oxide fuel cell has a cathode side and an anode side, separated by an electrolyte. Fuel is pushed through the anode side, while warm air is pushed through the anode side and electricity is produced at the electrode. In the Hyper configuration, compressed and preheated air from the heat exchanger goes through the cathode. Before spinning the turbine, a post combustion mixing volume blends the combusted flow with bypassed compressed airflow and bypassed warm air (Zaccaria et al., 2016). Combining a gas turbine and a fuel cell offers a higher theoretical efficiency for both systems than either system operating independently and makes use of otherwise unused byproducts. For instance, the unused fuel from the fuel cell can be combusted and combined with the heat generated during the fuel cell to spin the turbine and produce electricity. The fuel cell uses preheated air from the heat exchanger as the supply air for the cathode inlet to operate the fuel cell at higher temperatures. In this type of hybrid system layout, solid oxide fuel cells are generally used because they have been shown to have higher efficiencies when they operate at higher temperatures and pressure conditions (Badwal et al., 2014).

Figure 5.1. Hyper facility.

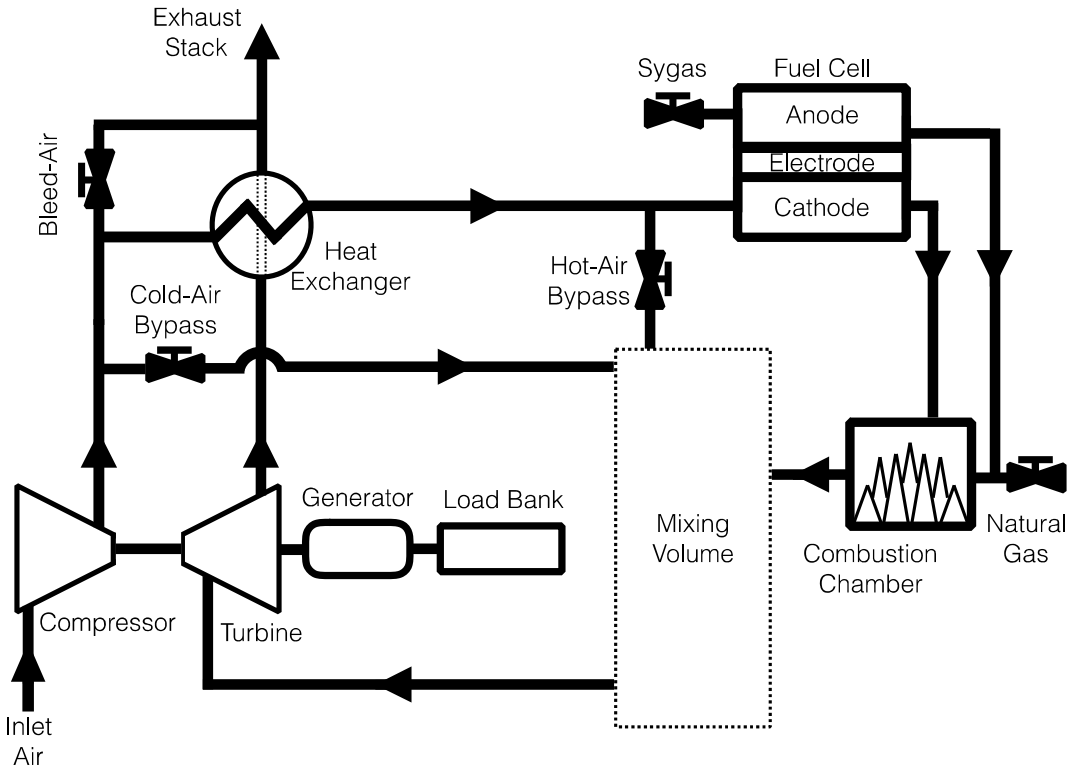


Figure 5.2. Hyper facility diagram.

Many issues still need to be resolved before commercial utilization of solid oxide fuel cell-gas turbine hybrid systems are possible. For instance, the ceramic material in the fuel cell makes them very sensitive to disturbances, while large temperature gradients may also cause significant degradation. For these reasons, a virtual fuel cell model is used to simulate the physical conditions and characteristics of a fuel cell (Tucker et al., 2002).

The tightly coupled interactions between the gas turbine and the fuel cell create a very fast dynamic response (Ferrari, 2015). When changes are made to maintain one component, those changes adversely affect the other component. Controlling the system during transients is critical and maintaining a fairly constant temperature gradient along the fuel cell is essential to not only maintain performance, but to prevent damaging or potentially destroying the fuel cell (Mueller et al., 2009). Sustaining a constant turbine speed is also important to keep a relatively constant cathode mass flow rate and to prevent compressor surge and stall (Wächter et al., 2010; Tucker et al., 2005). Given the relative fragility of these components and uncertainty during transient operations, robust control schemes are necessary to safeguard against damaging plant components, as well as to ensure the reliability and performance of the plant.

Controlling the Hyper is accomplished by adjusting one of three airflow valves (hot air, cold air, bleed air), the natural gas valve used in the combustion chamber or changing the resistive electric load from the load bank. Control strategies for the Hyper facility are still being investigated, with several studies performed over the past few years, most of which were conducted in a simulated environment. A preliminary research study on the controllability of the Hyper facility was done by creating a five-input five-output control problem, but it was demonstrated, however, that this five-by-five configuration was uncontrollable using a centralized control methodology. A three-by-three MIMO controller was used instead and transfer functions

were created for each input and output, relating each system variable to an actuator (Pezzini et al., 2014). Further testing was done in Matlab Simulink where an H_∞ controller was used to achieve optimal performance (Tsai et al., 2010). Another MIMO state-space based control strategy was implemented on a Hyper simulation model using model predictive control. Open-loop tests were performed to characterize the Hyper system and to create a simulation model to test the controller. Subsequently, the model predictive controller was used to evaluate the performance of the system during cathode mass flow and turbine speed set-point variations using all five of actuators and also when disturbances were introduced (Restrepo, 2011). Centralized, decoupled and decentralized PID control studies were also implemented on another simulation model of the Hyper. In this work, it was determined that a centralized approach, with communication between control loops provided better performance over a decentralized approach (Tsai et al., 2011). In the most recent evaluation, physical control of the Hyper facility was experimentally tested by creating a multivariable state-space controller (Pezzini et al., 2014).

Physical control of the Hyper has focused on controlling the turbine speed with the electric load and the cathode mass flow with the cold-air bypass valve. The shaft of the turbine is connected to a load bank, which can provide a resistive load, in kW, to simulate power generation demand and when the natural gas flow rate is kept constant, adjusting the resistive load can change the turbine speed. The cold-air bypass valve is used to maintain the desired flow rate through the cathode, and adjusting the percentage open determines how much of the compressed air goes through the cathode and how much goes into the mixing volume, which goes directly into the turbine. Adjustments to the cold-air bypass to change the cathode mass flow also affect the turbine inlet temperature and pressure, which subsequently change the turbine speed.

While the previously implemented simulated and physical controllers have shown varying levels of success at maintaining prescribed conditions for setpoint changes and during disturbances, each of the controllers requires system models, equations and off-line tuning, to achieve the desired control outcomes. A great deal of time and effort is needed to create these models and is due to the inherent mathematical complexity of these tightly coupled systems. Additionally, the optimal control parameters for setpoint tracking are significantly different than those for disturbance rejection.

5.4 Resource Sharing Algorithm

The resource sharing algorithm was created to control tightly coupled systems with complex interactions and many system components. It can be implemented when conventional control strategies have difficulties with conflicting control actions or creating a system model is challenging to construct accurately. The algorithm is able to find a system equilibrium without conflicting control actions because actions aren't taken at every time step, rather the frequency of control actions is based on user defined probability of action. Controllers or *agents* can be thought of as independent actors, which can sense changes to their environment and have the ability to change that environment. They establish coordination by sensing changes made to the environment and using a simple rule set to take action. Using the environment to establish coordination is called stigmergy and is derived from social insect behavior (Camazine et al., 2003). Distributed construction is where computational agents imitate social insects construction behavior (Peterson et al., 2011).

These agents manipulate uniform construction materials or *blocks* in a shared environment based on some very simple set of rules followed by each agent (Finzell and Bryden, 2016). In the resource sharing algorithm, the size of these blocks can be changed and their value is specified by the user.

At each time step two random numbers are created, one to randomly select an agent out of the group and one randomly determines if that randomly selected agent can take an action based on the user defined probability of action. If an agent is selected and allowed to make a decision, then they have the choice to take a block, give back a block or keep all their current blocks. This random probability of an action ensures that not every agent requesting a block or attempting to return a block is not allowed to do so immediately and is meant to ensure the behavior of the system is emergent. As agents are forced to wait in between taking action, conflicting control actions are avoided. The final element of this algorithm is the block repository. The block repository does not take any control action and has no explicit goal to achieve. It simply distributes new blocks as needed and stores block that have been given back so they can be redistributed to the agents. Each agent can observe a single process variable and each agent has a setpoint for that process variable. They can change make changes to that process variable to achieve their setpoint values by adjusting a single actuator. Receiving or returning a block serves as a means of incremental change to an actuator and agents will continue to give and take blocks until their own state is met, i.e. their process variable is within a given user specified tolerance of their desired value. Because the resource is shared and after a certain point the resource becomes finite, the distribution and redistribution allows the agents to achieve an equilibrium and when all of the agents have achieved their local state, the desired global state is met as well. A more detailed explanation of the resource sharing algorithm is given in (Finzell and Bryden, 2016).

5.4.1 Hyper Implementation

The resource sharing control strategy was able to control the system without the detailed system equations or models. The probability of action and block size were found through empirical testing to get the appropriate values to achieve a fast but accurate response. The algorithm was previously implemented on a computational environment that simulated an inverse heat transfer problem. This research is focused on applying the resource sharing control strategy to the Hyper facility, which has a much faster time dynamic.

The previous implementation of the resource sharing algorithm, used ten agents and the sensors and actuators were homogenous e.g. uniform heater surfaces and design surfaces. In the Hyper there are many different types of sensors and actuators, so the composition of each agents will be different. Applying the resource sharing control strategy to the Hyper facility, two agents are created: an agent to control the turbine speed and an agent to control the cathode mass flow. Agent 1 controls the turbine speed by adjusting the electric load, while agent 2 controls the cathode mass flow using the cold air bypass valve, as shown in Fig. 5.3. The electric load agent simulates changes in load demand, which requires a very fast system response to maintain turbine speed. If the turbine speed changes, this will affect the mass flow rate to the cathode. Keeping this mass flow rate fairly constant or changing the rate gradually is critical to ensure optimal fuel cell operation and prevent damaging fuel cell. Agent 1 changes the turbine speed by giving or receiving blocks each of which apply resistance (kW) to the turbine shaft. Agent 2 changes the cathode mass flow by giving or receiving blocks that adjust the cold air bypass valve.

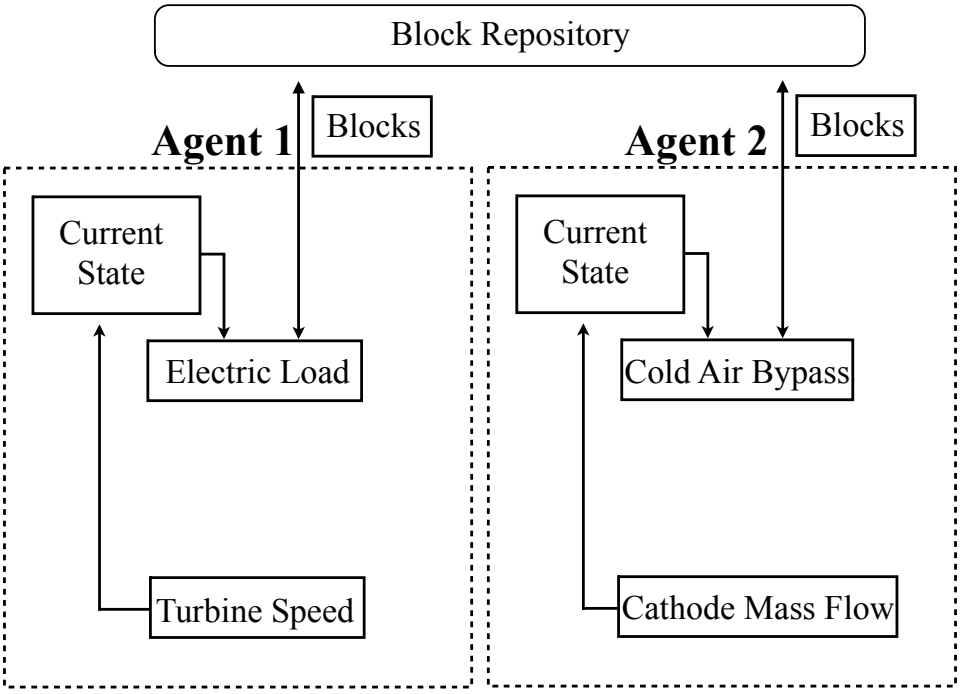


Figure 5.3. Agent 1 and 2 composition.

While ideally there would be more agents operating in this environment, these two agents control the two most critical components in the system. This controller is compared to a previously implemented MIMO state-space controller, which used the same configurations of sensors and actuators.

5.5 Results

To determine the dynamics of the algorithm on the Hyper, the system response will be evaluated as the algorithm's tunable parameters are adjusted to find the optimal values using different operating conditions. Changes in the system response are examined by holding one of the two adjustable parameters constant–probability of action or block size–and varying the other parameter. For each agent, five different probabilities of action were used, each with three different block sizes. The goal of the testing was to find which configuration of parameters gives optimal performance during a setpoint change in terms of several performance metrics. Due to the coupled nature of the system, a setpoint on one agent will affect the process variable of the other agent. Therefore the secondary agent will need to take action to maintain their desired nominal conditions during the primary agent's setpoint change. For agent 1, the setpoint change was from 40,500 rpm to 40,000 rpm and agent 2 had a setpoint change from 0.8kg/s to 0.6kg/s. A tolerance around the setpoint was chosen and that tolerance determines the bounds the current value can be within, to have effectively reached the setpoint. Selecting a reasonable tolerance means striking a balance between desired setpoint accuracy and the inherent sensor noise of the system. For both agents the tolerance was found during preliminary testing; the tolerance for agent 1 was ±75 rpm and for agent 2 was ±0.025 kg/s. The rate of each time step was set to 80 milliseconds for the entire system and determines how often decisions are made. Three block sizes and five different values for

probability of action were used in the scoping study, but a limited number of figures are shown for each agent. The results for the entire scoping study for both agent 1 and agent 2 are shown in Table 5.1 and 5.2 respectively.

The top left graphs for Figs. 5.4-5.7 shows the setpoint response when either the block size or the probability of action is held constant and corresponding parameter is varied over some range to show the differences in the responses. In the bottom set of graphs, a new constant value used in top graph is chosen (either block size or probability of action) and system response is examined as the non-constant parameter varies. The graphs below the top and bottom set of graphs show the corresponding agents response to the setpoint change. That agent's goal is to maintain their nominal conditions during the setpoint response. Fig. 5.4a-b shows setpoint response while keeping the block size constant for agent 1 and varying the probability of action. In Fig. 5.5a-b, the probability of action is held constant for agent 1 while the block size changes. The setpoint response of agent 2 while block size constant and varying the probability of action is given in Fig. 5.6a-b. Fig. 5.7a-b varies the block size for agent 2 and holds the probability of action constant.

Three common metrics that are associated with the setpoint response are rise time, settling time and peak overshoot. These can be used to compare qualitatively the effect of each parameter at achieving the new setpoint. These metrics are

- *rise time*—the time for the current value to reach the vicinity of the setpoint;
- *peak overshoot*—the single highest value observed during the test;

Table 5.1. Agent 1 system metrics.

Rise Time

Probability of Action	Block Size = 0.25kW	Block Size = 0.5kW	Block Size = 0.75kW
10%	32.32	17.12	9.44
20%	24.64	12.16	7.6
30%	15.52	10.48	7.52
60%	12.08	7.68	5.28
90%	8.16	6.48	4.72

Settling Time

Probability of Action	Block Size = 0.25kW	Block Size = 0.5kW	Block Size = 0.75kW
10%	120	78.24	104.64
20%	75.2	68.64	118.64
30%	49.44	47.52	67.04
60%	61.44	79.92	70.8
90%	50.68	91.36	53.52

Overshoot

Probability of Action	Block Size = 0.25kW	Block Size = 0.5kW	Block Size = 0.75kW
10%	95.9	138.6	168.2
20%	114.6	114.8	163.3
30%	145.6	128.6	196.5
60%	147.4	175.3	165.3
90%	112	151.1	133.2

Table 5.2. Agent 2 system metrics.

Rise Time

Probability of Action	Block Size = 1%	Block Size = 1.5%	Block Size = 2%
10%	52.64	47.2	36.4
20%	41.04	22.08	23.28
30%	22.64	15.2	13.2
60%	21.28	11.04	9.36
90%	15.12	14.96	8.48

Settling Time

Probability of Action	Block Size = 1%	Block Size = 1.5%	Block Size = 2%
10%	80.16	44.88	35.28
20%	43.92	19.36	20.64
30%	17.28	13.76	12.4
60%	13.44	8.48	8.72
90%	14.4	9.68	7.28

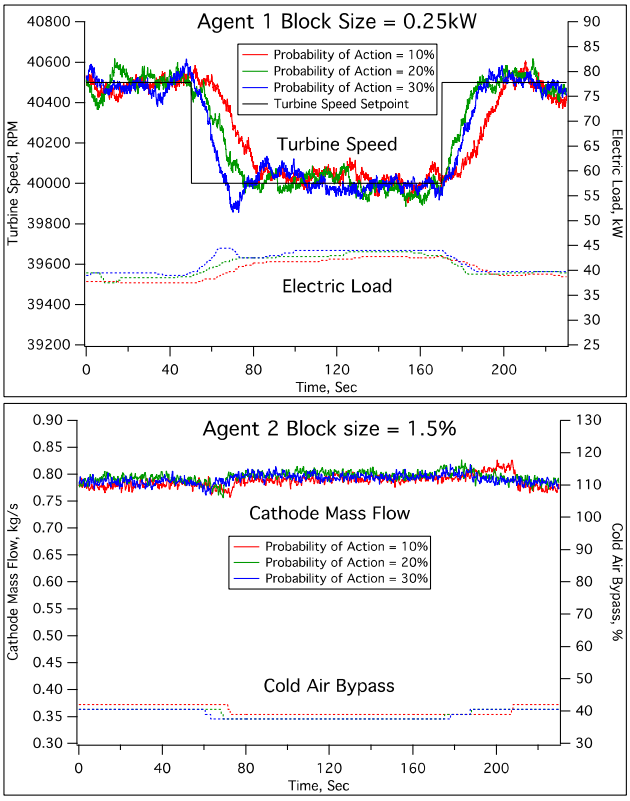
Overshoot

Probability of Action	Block Size = 1%	Block Size = 1.5%	Block Size = 2%
10%	0.005169	0.005769	0.01498
20%	0.009471	0.011912	0.00779
30%	0.014515	0.024647	0.028415
60%	0.015436	0.029429	0.028257
90%	0.020959	0.029277	0.027577

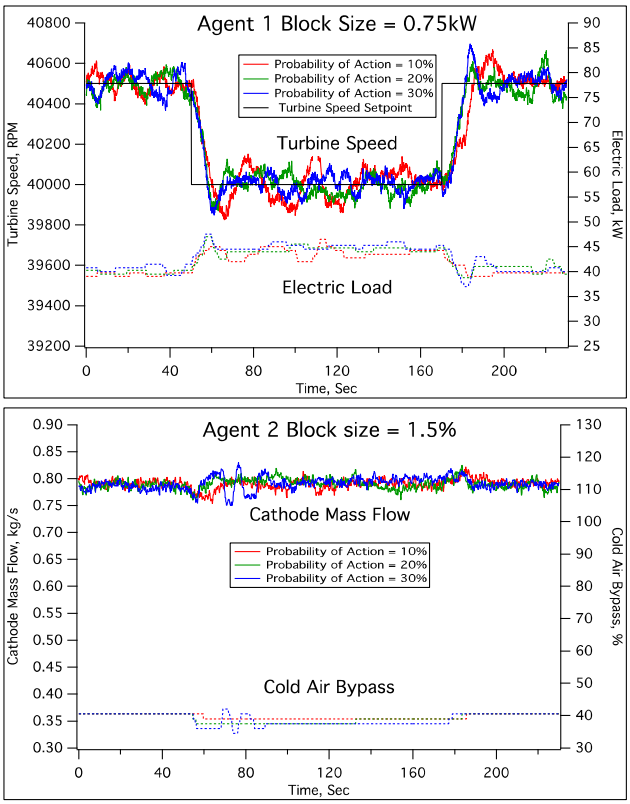
- *settling time*—the time for the controlled variable to be within 15% of the new setpoint and stay within those bounds for the remainder of the test. This settling time criteria is larger than normal because of the large amount of noise in the turbine speed.

Fig. 5.4a shows that if the block size is set to 0.25kW, changing the increasing probability of action significantly decreased the rise time. A probability of action of 10% had a slow rise time with no overshoot, while 30% probability of action had much faster rise time but more overshoot. A probability of action of 20% had a faster rise time than 10% but no overshoot. When the block size was increased and held at 0.75kW in Fig. 5.4b, there was less of a difference in the rise time or overshoot when the probability of action varied from 10% and 30%. This shows that if the block size is large enough, changing the probability of action does not affect the overall system response for agent 1. There was an unstable oscillation in the response from agent 2 when the probability of action was set to 30% and the block size for agent 1 was 0.75kW. This oscillation occurred because of a sudden reaction from agent 2, which caused a slight amount of overshoot, but this was quickly corrected.

Figure. 5.4a shows that if the block size is set to 0.25kW, changing the increasing probability of action significantly decreased the rise time. A probability of action of 10% had a slow rise time with no overshoot, while 30% probability of action had much faster rise time but more overshoot. A probability of action of 20% had a faster rise time than 10% but no overshoot. When the block size was increased and held at 0.75kW in Fig. 5.4b, there was less of a difference in the rise time or overshoot when the probability of action varied from 10% and 30%. This shows that if the block size is large enough, changing the probability of action does not affect the overall system response for agent 1. There was an unstable oscillation in the response



(a)



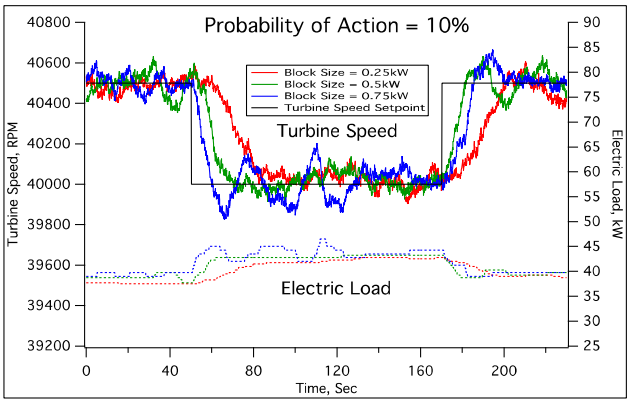
(b)

Figure 5.4a-b. Agent 1 block size changes.

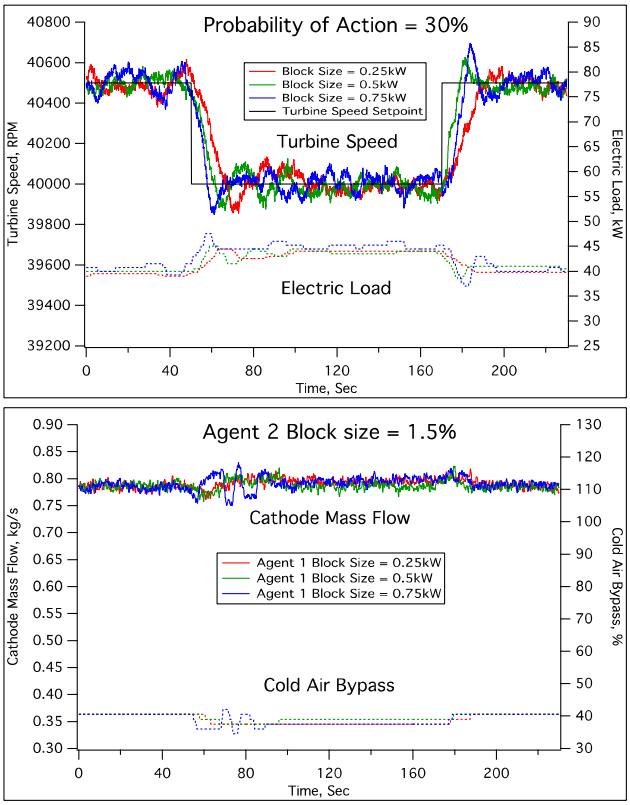
from agent 2 when the probability of action was set to 30% and the block size for agent 1 was 0.75kW. This oscillation occurred because of a sudden reaction from agent 2, which caused a slight amount of overshoot, which was quickly corrected and agent 2 maintained nominal conditions.

In Fig. 5.5a, when probability of action for agent 1 is held constant at 10%, there were significant variations in terms of rise time, overshoot and settling time as the block size was increased from 0.25kW to 0.75kW. A block size of 0.25kW had the slowest rise time and settling time with no overshoot. Increasing the block size to 0.5kW decreased the rise time and settling time, with a slight amount of overshoot. The most overshoot was seen when the block size was 0.75kW, which had the shortest rise time, but the settling time was greater than a block size of 0.5kW due to a large oscillation around the setpoint. Increasing and holding the probability of action at 30% (Fig. 5.5b) decreased the variation in the responses using different block sizes. As the block size increased, there were slight decreases in the rise times. The overshoot between the block sizes stayed relatively constant but the settling time increased as the block size increased. The best performing setting was a block size of 0.5kW, which had the least amount of overshoot, the fastest settling time and second fastest rise time. The unstable oscillation seen below Fig. 5.5b is the same oscillation seen below Fig. 4b, as there is some overlap in data.

For agent 2, when the block size is held constant at 1% and the probability of action is varied is shown Fig. 5.6a. As the probability of action changed from 10% to 30% both the rise time and the settling time decreased noticeably. When the current value approached the setpoint, there was very little overshoot for all probabilities of action. When the block size was increased and held at 2% (Fig. 5.6b), there were still large differences in the rise time and settling time as

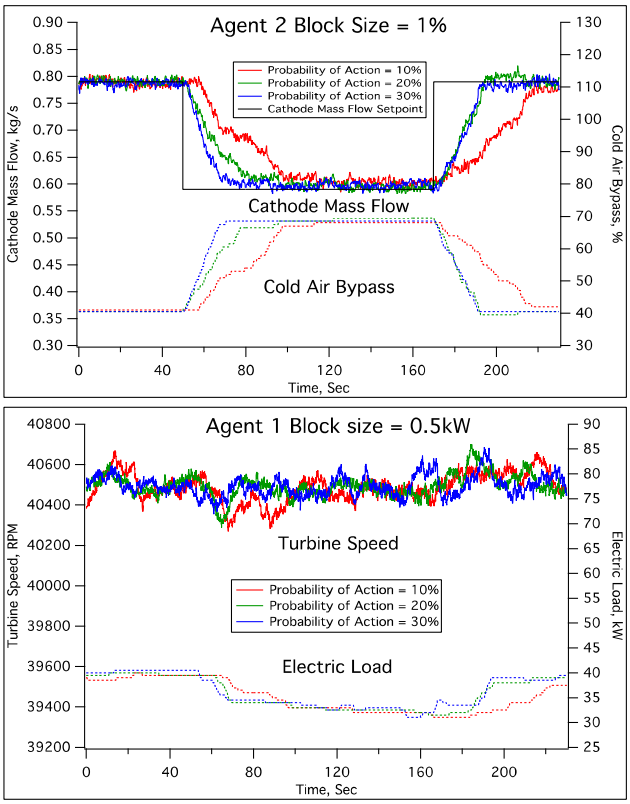


(a)

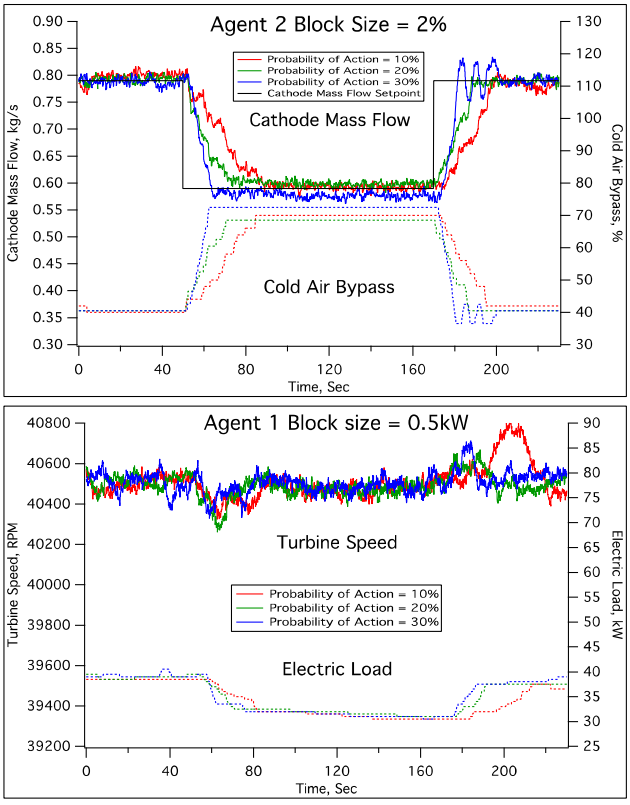


(b)

Figure 5.5a-b. Agent 1 probability of action change.



(a)



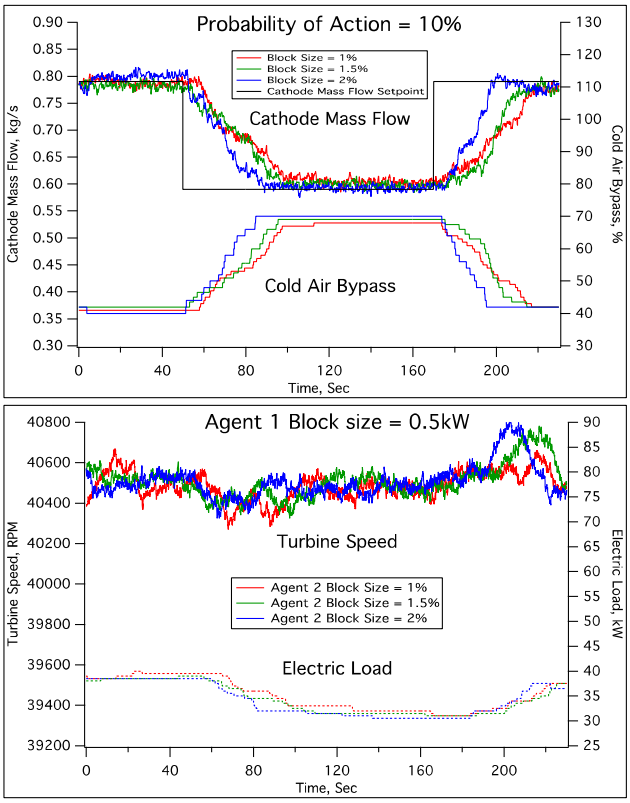
(b)

Figure. 5.6a-b Agent 2 block size.

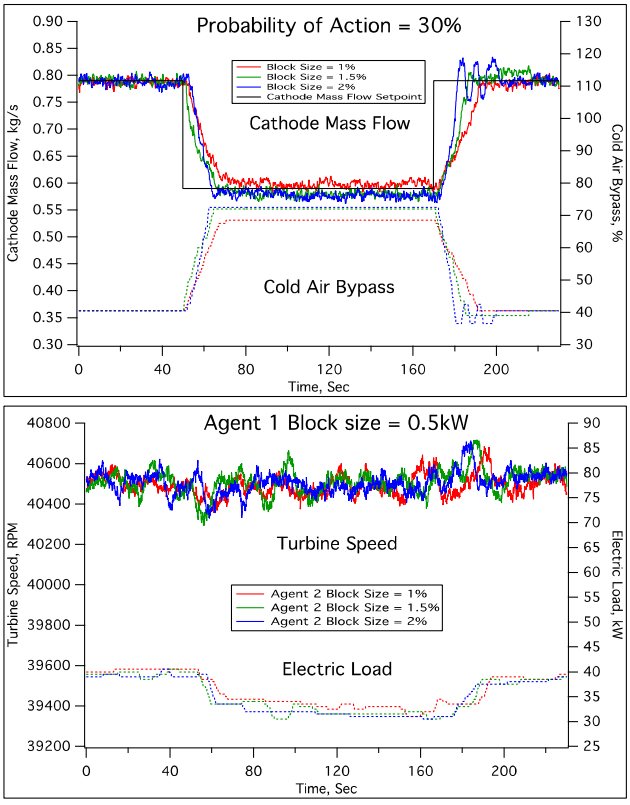
the probability of action increased from 10% to 30%. The rise time and settling time decreased as the probability of action increased. When the probability of action was increased to 30%, there was a noticeable increase in overshoot and there was an oscillation when agent 2 was changed back to nominal conditions. This was because the probability of action and block size were high enough that changes could be made very quickly and caused some overshoot in the cathode mass flow, which agent 2 was able to bring back in control. The coupled response from agent 1, shown below Fig. 5.6a and Fig. 5.6b, shows that agent 1 needed to make much larger changes to the electric load to maintain its nominal conditions, when compared to the response from agent 2 during a setpoint change for agent 1.

In Fig. 5.7a, the probability of action for agent 2 was held constant at 10% and block size varied from 1% to 2%. The rise time and settling time decreased as the block size increased, but those differences were smaller than in the previous case where the probability was varied and the block size was held constant. Increasing the block size from 1% to 1.5% lowered the rise time slightly, but did significantly lower the settling time. Increased the block size to 2% further decreased the settling time and rise time. When the probability of action was held at 30%, in Fig. 5.7b, the differences in the settling time and rise time are less pronounced as the block size is varied, with slight decreases in settling time and rise time as the block sizes increase.

Three block sizes and three probabilities of action were used to minimize the graphs used to demonstrate the trends in the system response. In the experimental tests at the Hyper, five different probabilities of action were used. The trends for all the probabilities of action for rise time and settling time for agent 1 are shown in Fig. 5.8a-b and for agent 2 are shown in Fig. 5.9a-b. For agent 1, the rise time consistently decreased as both block size and the probability of action increased. Between the two tunable parameters, the probability of action had the largest



(a)



(b)

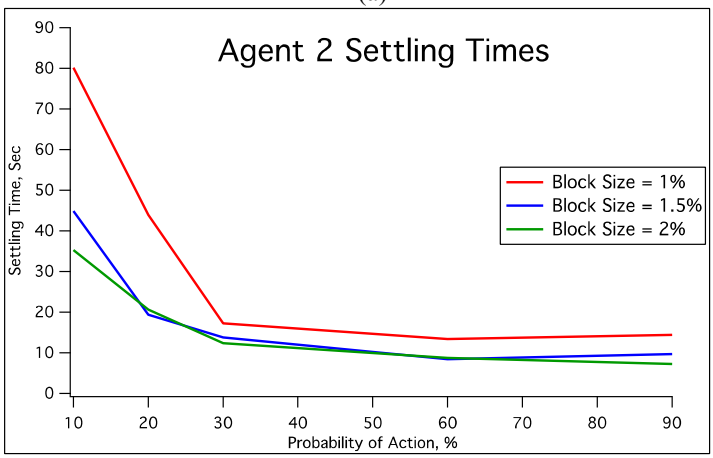
Figure. 5.7a-b Agent 2 probability of action change.

(a)

(b)

Figure. 5.8a-b Agent 1 Rise time and settling time.

(a)



(b)

Figure. 5.9a-b Agent 2 Rise time and settling time.

overall effect at reducing the rise time. Initially, increases in the block size was shown to have large reductions on the rise time specifically between probabilities of action of 10% and 30%. After 30%, however, there were diminished reductions in the rise time when the block size was increased. Examining the different block sizes, there were large differences in the rise time between a probability of action of 10% and 90%. The general trend for the settling time for agent 1 was that it decreased as the probability of action increased. The effects on the settling time as the block size changed was mixed, with block sizes of 0.25kW and 0.75kW showing generally decreasing settling times with increasing probability of action, while a block size of 0.5kW decreased settling time until about 30% probability of action at which point the settling time went back up. The rise times and settling times for agent 2 are given in Fig. 5.9a-b. With agent 2, noise was much less of an issue and it was therefore able to reach and maintain the settling time criteria, with much less overshoot than agent 1. In Fig. 5.9a, the rise times decreased as both block size and the probability of action increased. The probability of action had a much greater effect in decreasing the rise time compared to block size, especially when the probability of action was less than 30%. The overall difference in rise times between block sizes of 1% and 1.5% was apparent, but the overall trend in rise times for block sizes of 1.5% and 2% was very similar. The probability of action had the largest effect in reducing the rise time. Agent 2 showed a consistent trend for settling times. It decreased as both the block size and probability of action increased. As before there were significant difference between block sizes of 1% to 1.5% but much smaller differences between block sizes of 1.5% and 2%.

Figure 5.10 and 5.11 give the responses for a multivariable (MIMO) state-space controller. Figure 5.10 shows a setpoint change in the turbine speed using electric load and Fig. 5.11 shows a setpoint change in the cathode mass flow using the cold air bypass valve. This setup is different

from two separate PID controllers because MIMO controllers establish relationships between each process variable and each actuator. Comparing the resource sharing algorithm to a traditional multivariable state-space controller, allows for an assessment of the potential benefits and circumstance where the algorithm may offer better performance. Figure 5.10 shows the setpoint response of the turbine when a new setpoint of 41,000 rpm is set. This test increases the turbine speed setpoint by 500 rpm instead of decreasing the turbine speed by 500 rpm, as previous tests have done. However, it can give a general comparison of the response of the system using a different control approach. The setpoint response is very quick, with a rise time of 1.5 seconds and a settling time of 7.28 seconds. The best response using the resource sharing algorithm for agent 1, using a block size of 0.75 and a probability of action of 90%, had a rise time of 4.72 seconds and a settling time of 53.52 seconds. The main difference in the speed of the response is that the MIMO controller can make a change at each sample time and the response can be very rapid as it can be tuned to jump to approximately the correct inputs needed for desired outputs. The resource sharing algorithm must find those values out and will make changes incrementally. Figure 5.11 gives the MIMO response when there is a setpoint change in the cathode mass flow. With the cathode mass flow, the setpoint response was very similar to several responses using the resource sharing algorithm. The rise time for the MIMO controller was 11.12 seconds and the settling time was 13.44 seconds. The best response for the resource sharing algorithm was when the block size was 2% and the probability of action was 90%, which gave a rise time of 8.48 seconds and a settling time of 7.28 seconds, so the best response from the resource sharing algorithm outperformed the MIMO controller. One potential reason for the slow response in the cathode mass flow is the coupled response and oscillations in the electric load, which is continuing to oscillate throughout the test. Using centralized tuning for the MIMO

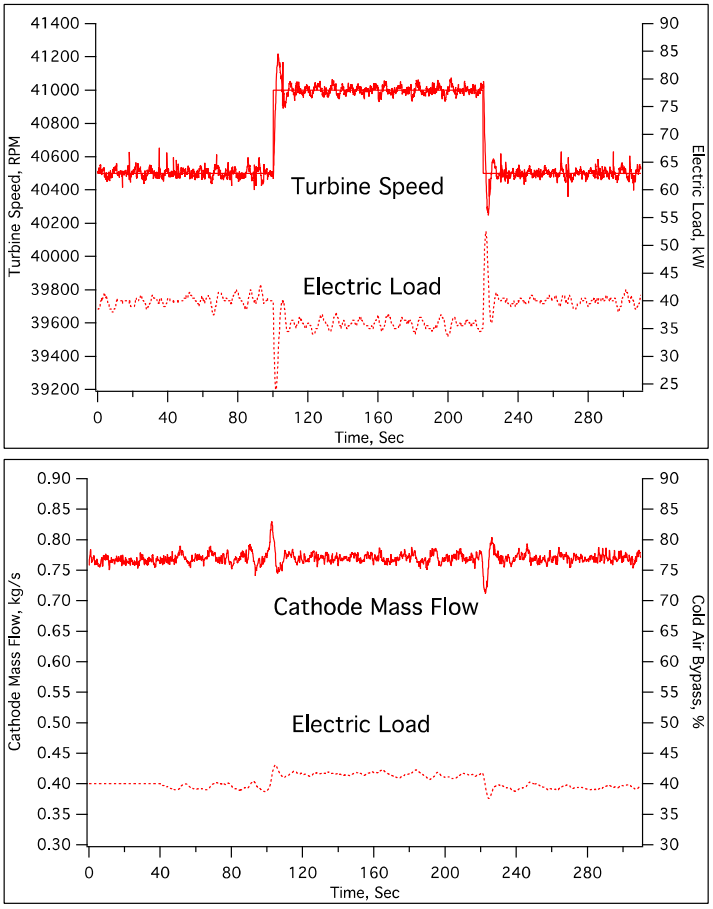


Figure. 5.10 MIMO control of turbine speed.

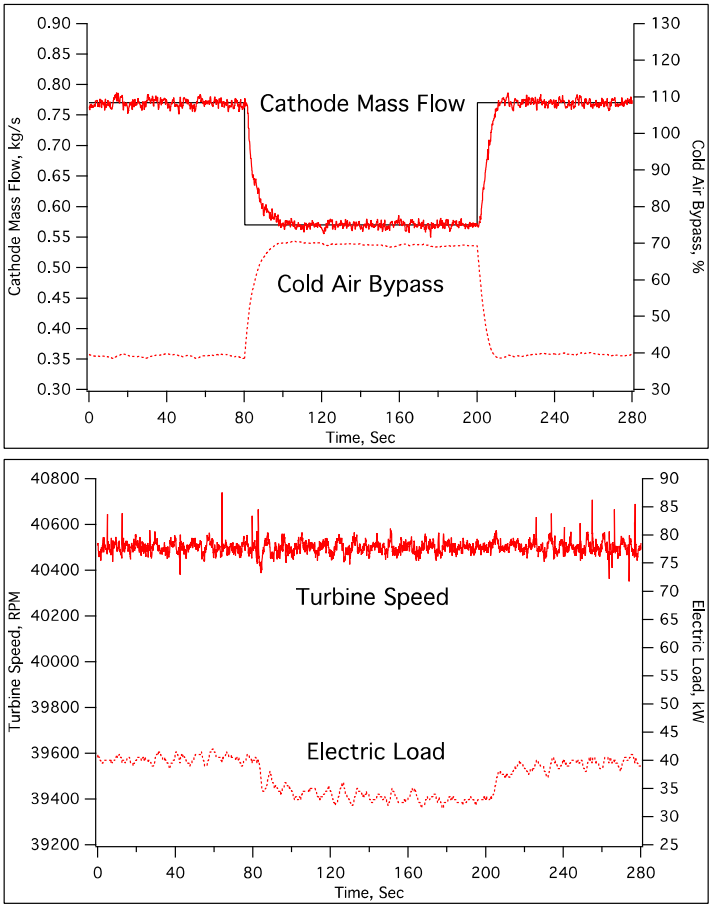


Figure. 5.11 MIMO control of cathode mass flow.

control strategy can cause issues if the time response between the control loops is very different. Increasing the controller response for the cathode airflow would also increase the turbine speed response, so a compromise needs to be reached. The tuning that allows such a fast turbine speed response in the first test also limits an increase in the cathode mass flow response in the second test.

5.6 Conclusions

The resource sharing controller was successfully implemented on the Hyper system and was able to achieve the desired setpoint in all the cases examined. The controller was able to adjust the turbine speed and cathode mass flow using the electric load and the cold air bypass valve. The system response was evaluated as probability of action and block size changed. The ideal values for maintaining steady state conditions and optimal performance during setpoint changes were found.

The difference in the setpoint responses for agent 1 and agent 2 demonstrates several important facets of the resource sharing algorithm. Varying the tunable parameters helped to determine when the block size was the critical factor and when the probability of action was the critical factor to achieve a desired setpoint response. For agent 1 there was a tradeoff between settling times and rise times, where the rise time would always go down as the block size and probability of action increased, but the settling time was not as consistent. Agent 1 was not able to control the system as well as agent 2 because there was inherently more system noise in the turbine, the turbine speed changes are very rapid and there was a limit to the control accuracy of load bank, which is less precise than the cold air bypass valve. The fidelity of the adjustment of the electric load is limited. Agent 1 can only make changes in increments of 0.5kW, but agent 2 can control

the cold air bypass valve in increments as small as 0.1%. Agent 2 had consistently decreasing rise times and settling times when both the block size and probability of action increased. In all the tests for agent 2, the probability of action played the largest effect on the settling time and rise time.

When a step change was induced for agent 2, the corresponding response from agent 1 was more pronounced than the response from agent 2 when agent 1 had setpoint change. This was shown by the magnitude of the changes to the electric load that agent 1 made to keep the turbine speed around nominal conditions and how little agent 2 had to adjust the cold air bypass to maintain the cathode mass flow. The cold-air bypass has a strong interaction on the turbine speed because changes in the cathode mass flow affect turbine inlet pressure and temperature. It is important to note that as the rise time for both agents increases, the coupling effect becomes more pronounced.

When comparing the resource sharing algorithm to a MIMO controller, the algorithm has a much slower turbine speed setpoint response. The resource sharing algorithm was able outperform the MIMO controller, however, when there was a setpoint change in the cathode mass flow. The MIMO controller will take action at each sampling time, using both controllers, when a setpoint change is made. Simultaneous control action is avoided in the resource sharing algorithm, because only a single controller can take or receive a block during any time step, even if the time in between time steps is very small. The resource sharing algorithm requires significantly less system information beforehand and less effort to tune the parameters to get a good setpoint response than the MIMO controller.

Successful implementation of the resource sharing controller requires determining the appropriate parameters to achieve the desired response as system conditions change. Implementation of the resource sharing algorithm as a control strategy showed that a sufficiently

large block size and probability of action were needed for a quick and accurate response to a setpoint change to account for the fast dynamics and tightly coupled nature of the Hyper system. A block size or probability of action that is too large, however, can create undesirable instabilities or oscillations. Future tests will determine the response of the system when both the probability of action and block size are scaled as the current value approaches the set point. Additionally, disturbances will be introduced to the system to gauge the response of the algorithm.

Acknowledgments

This research was supported by the US Department of Energy – Office of Fossil Energy under Contract No. DE-AC02-07CH11358 through the Ames Laboratory.

References

Akyildiz, I.F., Weilian Su, Sankarasubramaniam, Y., Cayirci, E., 2002. A survey on sensor networks. IEEE Commun. Mag. 40(8), 102–114. doi:10.1109/MCOM.2002.1024422

Åström, K.J., Hagglund, T., 2006. Advanced PID control. ISA-The Instrumentation, Systems, and Automation Society, Research Triangle Park.

Aström, K.J., Wittenmark, B., 1995. Adaptive control. Addison-Wesley Longman, Boston.

Åström, K.J., Albertos, P., Blanke, M., Isidori, A., Schaufelberger, W., Sanz, R., 2012. Control of Complex Systems. Springer Verlag, London.

Badwal, S., Giddey, S., Munnings, C., Kulkarni, A., 2014. Review of progress in high temperature solid oxide fuel cells. J. Aust. Cer. Soc. 50(1), 23–37.

Bajpai, P., Dash, V., 2012. Hybrid renewable energy systems for power generation in stand-alone applications: A review. Renewable and Sustainable Energy Reviews 16, 2926–2939. doi:10.1016/j.rser.2012.02.009

Bakule, L., 2008. Decentralized control: An overview. Annu. Rev. Control 32(1), 87–98.

Bizon, N., Shayeghi, H., Mahdavi Tabatabaei, N., 2013. Analysis, control and optimal operations in hybrid power systems. Springer Verlag, London. doi:10.1007/978-1-4471-5538-6

Bubnicki, Z., 2005. Modern control theory. Springer-Verlag, New York.

Burch, G.D., 2001. Hybrid Renewable Energy Systems. Presented at: U.S. DOE Natural Gas/Renewable Energy Workshops.

Camazine, S., Deneubourg, J.-L., Franks, N., Theraulaz, G., Sneyd, J., Bonabeau, E., 2003. Self-organization in biological systems. Princeton University Press, Princeton.

Carmeli, M.S., Castelli-Dezza, F., Mauri, M., Marchegiani, G., Rosati, D., 2012. Control strategies and configurations of hybrid distributed generation systems. Renew. Energ. 41, 294–305. doi:10.1016/j.renene.2011.11.010

DOE, 2015. Quadrennial Energy Review: An assessment of energy technologies and research opportunities. Available at: http://energy.gov/sites/prod/files/2015/09/f26/Quadrennial-Technology-Review-2015_0.pdf

Finzell, P., Bryden, K., 2016. A Novel Resource Sharing Algorithm Based on Distributed Construction for Radiant Enclosure Problems. Adv. Eng. Softw., Submitted.

Ioannou, P.A., Jing Sun, D., 2012. Robust Adaptive Control. Prentice-Hall, Upper Saddle River.

Khaki-Sedigh, A., Moaveni, B., 2009. Control configuration selection for multivariable plants. Springer Verlag, Berlin.

Levine, W.S., 2010. The control handbook. CRC Press, Boca Raton.

Maley, S., Rawls, P., 2014. Crosscutting Research Sensors and Controls Project Portfolio.

Mueller, F., Jabbari, F., Brouwer, J., 2009. Linear quadratic regulator for a bottoming solid oxide fuel cell gas turbine hybrid system. J. Dyn. Sys. Meas., Control 131(5), 23–37.

Onar, O.C., Uzunoglu, M., Alam, M.S., 2008. Modeling, control and simulation of an autonomous wind turbine/photovoltaic/fuel cell/ultra-capacitor hybrid power system. J Power Sources 185(2), 1273–1283.

Petersen, K., Nagpal, R., Werfel, J., 2011. TERMES: An autonomous robotic system for three-dimensional collective construction. In: Proceedings of Robotics: Science & Systems VII.

Pezzini, P.N., 2014. Control Strategy for a Direct-Fired Fuel Cell Turbine Hybrid Power System and Decentralized/Centralized MIMO Control Approach. PhD Dissertation, University of Genoa, Genoa, Italy.

Restrepo, B., 2011. Experimental characterization of fuel cell gas turbine power system and determination of optimal trajectories of operation using a model predictive controller. PhD Dissertation, West Virginia University, Morgantown, WV.

Siljak, D.D., 2013. Decentralized control of complex systems. Dover, Mineola.

Torres-Hernandez, M.E., Velez-Reyes, M., 2008. Hierarchical control of Hybrid Power Systems. In: Proceedings of the 11th IEEE International Power Electronics Congress. pp. 169–176. doi:10.1109/CIEP.2008.4653837

Tsai, A., Banta, L., Tucker, D., 2010. Multivariable robust control of a simulated hybrid solid oxide fuel cell gas turbine plant. J. Fuel Cell Sci. Tech. 7(4), 0410081– 0410089.

Tsai, A., Tucker, D., Groves, C., 2011. Improved Controller Performance of Selected Hybrid SOFC-GT Plant Signals Based on Practical Control Schemes. In: Proceedings ASME Turbo Expo 2010.

Tucker, D., Lawson, L., Gemmen, R., 2005. Characterization of air flow management and control in a fuel cell turbine hybrid power system using hardware simulation. In: Proceedings of ASME 2005 Power Conference.

Tucker, D., Liese, E., VanOsdol, J., Lawson, L., Gemmen, R.S., 2002. Fuel cell gas turbine hybrid simulation facility design. In: Proceedings of ASME 2002 International Mechanical Engineering Congress and Exposition

Uzunoglu, M., Onar, O.C., Alam, M.S., 2009. Modeling, control and simulation of a PV/FC/UC based hybrid power generation system for stand-alone applications. Renew. Energ. 34(3), 509–520. doi:10.1016/j.renene.2008.06.009

Valenciaga, F., Puleston, P.F., 2005. Supervisor control for a stand-alone hybrid generation system using wind and photovoltaic energy. IEEE T. Energy Conver. 20(2), 398–405. doi:10.1109/TEC.2005.845524

Wächter, C., Lunderstädt, R., Joos, F., 2010. Using linear control theory for parameterization of a controller for a SOFC/GT hybrid power plant. J. Fuel Cell Sci. Technol. 7(3), 0310031–0310039. doi:10.1115/1.3206972

Zaccaria, V., Tucker, D., Traverso, A., 2016. Transfer function development for SOFC/GT hybrid systems control using cold air bypass. Appl. Energ., 165(2016), 695–706. doi:10.1016/j.apenergy.2015.12.094

Zerkaoui, S., 2012. Online Hierarchical Controller for Hybrid Power System. ISRN Renewable Energy 2012, 1–13. doi:10.1016/j.apenergy.2005.11.004

CHAPTER 6. APPLYING DYNAMIC SCALING TO THE RESOURCE SHARING ALGORITHM ON A HYBRID POWER SYSTEM

Peter Finzell, Paolo Pezzini, David Tucker, Kenneth M. Bryden[5]
Simulation Modeling and Decision Science Program
Ames Laboratory
1620 Howe Hall, Ames, Iowa, 50011

Abstract

This paper demonstrates the system response when the dynamic scaling is applied to the resource sharing algorithm at the Hyper facility. Dynamically scaling means the two adjustable parameters of the resource sharing algorithm (block size and probability of action) are scaled as function of the error between the current value and the desired setpoint. The algorithm is implemented on the Hyper facility by creating two separate agents, controlling two independent system variables. The first agent controls the turbine speed by adjusting the electric load, while the second adjusts controls the cathode mass flow using the cold air bypass valve. For agents 1 and 2, nine block scaling tests and three probability scaling tests were performed. In each test, the goal is reach a desired setpoint from nominal conditions, while attempting to optimize performance metrics. In all the tests examined, the controllers were able to reach their desired setpoint values in a reasonable amount of time, with limited overshoot and oscillations around the setpoint. The results from this implementation of the resource sharing algorithm will be compared to a previous implementation of the algorithm and a multi-variable (MIMO) controller.

Keywords: Stigmergy; Multi-Agent Systems; Bio-inspired; Distributed Construction; Hybrid Power Systems

[5] Corresponding author. tel +15154600875
Email address: kmbryden@iastate.edu

6.1 Introduction

A power plant is an energy system that uses a network of sensors and actuators to maintain a particular configuration (e.g., a given power and emissions level). A controller directs the operation of the plant by providing controls signals to actuators throughout the plant. Hybrid technologies are currently being evaluated to work in conjunction with, combine with or replace existing power generation facilities. These hybrid technologies frequently involve two or more energy generation devices including: fossil generation system, renewable power generation (wind generation, solar panels), fuel cells or energy storage such as batteries (Bizon, 2013). By combining these technologies, the objective is to lower costs and emissions while simultaneously increasing the efficiencies (Burch, 2001**)**. These hybrid systems are frequently highly coupled, complex systems, which creates a unique set of challenges when selecting an appropriate control strategy.

Critical power plant parameters are determined by the individual sensors, which are monitoring system outputs, emissions levels or other performance criteria (Astrom et al., 2012). The increasing computational power of sensors, along with the relative reduction in price and a desire to retrieve more granular system information means that these power generation facilities may use thousands of sensors (Malley and Rawls, 2014). To make effective use of the large increase in system information, control strategies must be able to utilize this information in a meaningful way. Control strategies can encompass one or more control algorithms and are created to achieve some desired conditions or to maintain a specified setpoint within a dynamic system. There are many control strategy categories and selecting the appropriate one is very system specific. Typical power plants are controlled by a small subset of critical sensors that provide data to either a centralized controller or are a single piece of data to one or more distributed or

proportional-integral-derivative (PID) controllers. Adaptive control can be implemented using distributed or centralized control and means that the gains or certain parameters will change as the operating conditions change. Adaptive controllers change with the system and allow for a system to be managed and controlled over a larger range. Control architectures describe the flow of information and the various scales that control actions are taken. Three of the most common control architectures are centralized, distributed and hierarchical or decentralized. In a centralized architecture, like a centralized control strategy, information flows to one central location, which manages and coordinates all of the control actions. Distributed architectures, like distributed control strategies, create independent controllers that do not coordinate their actions with each other. With a hierarchal architecture information is pass from low levels to high levels, with each level making higher level decisions.

This paper demonstrates dynamic scaling resource sharing algorithm of the resource sharing algorithm on a Hybrid power system. Dynamic scaling allows two of the parameters in the resource sharing algorithm to change as the system changes. Dynamic scaling offers a very fast implementation, even over the standard resource sharing algorithm because there is less risk of selecting values that may overshoot or cause large oscillations because the response is dampened around the setpoint.

6.2 Background

Previous methods for generating energy have focused on single source production that is scaled up as demand increases. Control strategies for these systems are tuned specifically for their configurations and work well within a defined operating envelope. Future energy systems will focus more on hybrid facilities. Hybrid systems will form some combination of renewables, non-

renewables and energy storage. These systems combine different components in such a way that they work together and provide advantages over operating each component independently (Jun et al., 2011). Many hybrid systems use wind or solar power which can be intermitted and may need conventional generation facilities to even out changes in demand or limitations in supply of wind or sun.

These systems face a unique set of control challenges as there is frequently strongly coupled interactions between components and their ideal operating conditions and times scaled can be very different. Control strategies for single source energy production operate well under well-defined and sometimes limited operating conditions, but with hybrid systems adaptive and intelligent control methods offer benefits (adaptability, flexibility) and potentially widen the operating conditions. Neural networks and fuzzy logic two methods of intelligent control that offer methods to characterize complex system interactions without fully understanding their relationships to each other (Natsheh and Albarbar, 2013; De Silva, 1995; Lin et al., 2011). These control strategies provide some adaptability but do not provide the scalability or re-configurability needed for these hybrid systems.

Agent based control provides for some scalability by creating independent controllers or agents that sense their environment, take action to modify their environment and coordinate their actions. This coordination is through direct communication, which means that while the system is scalable to an extent, as more agents are added to the system, all agents need to be updated to communicate with the new agents (Chakraborty et al., 2013). When implimentated for control of hybrid systems, agents based controllers are typically used in a supervisory role or hierarchal configuration (Dou et al., 2015) .There is an opportunity to develop a novel control strategy for hybrid power systems that is adaptable, scalable and reconfigurable.

The Hybrid Performance (Hyper) facility is a 2500 square foot research project that combines a gas turbine recuperated cycle and a virtual solid oxide fuel cells (SOFCs) stack, Fig. 5.1. It is currently being undertaken at Department of Energy's National Energy Technology Laboratory (NETL) and has an 800kW capacity using a 120 kW gas turbine and a simulated fuel cell that can generate between 200-700kW (Tucker, 2003). Fig. 5.2 shows the layout of the Hyper facility, which is composed of a gas turbine recuperation cycle (a compressor, turbine, combustor, heat exchanger and mixing volume) and a simulated solid oxide fuel cell. A conventional gas turbine recuperation cycle, the inlet air is compressed and goes through a heat exchanger before entering the mixing volume where it mixes with the post combustion gases. These gases spin the turbine to generate power. The exhaust goes through the opposite side of the heat exchanger before venting out. Opening and closing the cold air bypass valve determines how much compressed air goes through the cathode and how much goes to the post combustor mixing volume. The turbine and compressor shaft is connected to a load bank, which can set a resistive load that simulates a load demand on the system. Increasing the load will decrease the turbine speed, as long as the fuel valve is set to a constant value. The benefits of combining a gas turbine and a fuel cell are the fuel cell can operate at higher temperatures using preheated air from the heat exchanger and fuel from the anode can be used as extra fuel in the combustions chamber.

When the turbine speed or the mass flow rate changes, it will have an adverse effect on the other component, which is why this system is so highly coupled. The speed at which these changes occurs is very fast, on the order of 5 milliseconds, which is why any controller must also

Figure 5.1. Hyper Facility.

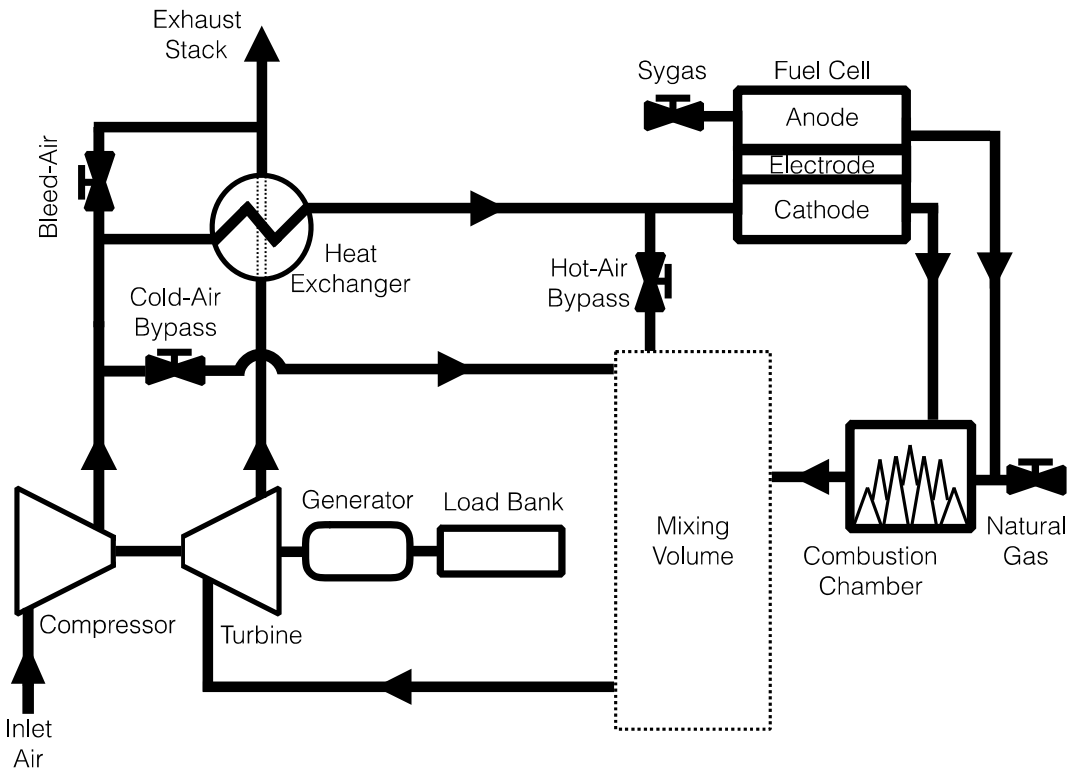


Figure 5.2. Hyper layout.

have a very fast time response. Keeping a fairly constant turbine speed is essential to prevent compressor surge and stall (Pezzini, 2014). As the compressor and turbine are on the same shaft, a disruption in the airflow to the compressor can cause either surge or stall. Surge is a brief disruption in the airflow, where stall is a complete disruption in the flow. Maintaining a fairly constant temperature gradient along the fuel cell by either keeping the mass flow rate fairly constant or changing the rate gradually, is critical to prevent damage to the fuel cell and ensure optimal fuel cell operation. (Mueller et al., 2009; Wächter et al., 2010). Since there is a high cost associated with using a physical fuel cell and experimental testing may damage or destroy that fuel cell, so a virtual fuel cell model is used. This virtual fuel cell combines a computational model of the fuel cell in DSpace® and an air plenum that simulates the physical volume and flow characteristics of a fuel cell.

Previous control methods using computational models of the Hyper include MPC, PID and multi-variable control (Tsai et al., 2010; Restrepo, 2011; Tsai et al. 2011). Physical testing involving centralized and distributed control have been implemented on the Hyper facility and have shown varying levels of success at maintaining prescribed conditions for setpoint changes and during disturbances (Pezzini, 2014). It was shown that decoupling the system into separate controllers by creating several decentralized PID controllers was less successful than more centralized approaches. Using decentralized PID controllers, steps must be taken to avoid potential conflicts between controllers taking simultaneous and conflicting actions (Siljak, 2013). Feedforward techniques can be used to alleviate some of these potential conflicts, but this requires additional planning. Centralized or multi-variable approaches require a significant amount of time to create, tune and validate a model to achieve the desired control outcomes, due to the complexity of the mathematical models involved (Khaki-Sedigh and Moaveni, 2009). Both of these controllers

may encounter limitations in their range of operating conditions and while one set of parameters may be ideal for setpoint tracking, another set of parameters may be needed for disturbance rejection.

6.3 Resource Sharing Controller

One of the main concerns when implementing any control strategy is how long it will take to develop a model or to create a controller that will accurately and safely control the system. The resource sharing algorithm was developed specifically so that there was limited development time and computational agents could be added or removed without reconfiguring the whole algorithm. The algorithm can be implemented without creating these complex models and can be implemented much quicker and offers a new approach that has the potential to increase flexibility, scalability and robustness.

This algorithm is based off of distributed construction, which is the when computational agents imitate the construction behavior of social insects. The resource sharing algorithm creates simple agents that use a shared resource and a simple set of rules to make changes to their environment which serve as a means of indirect communication and to coordinates actions. By evaluating the current state of their environment, each agent uses a simple set of rules to determine the appropriate action to take. As agents are created, each agent is allowed to make control decisions at the local level independently, without consulting with other agents. A shared virtual resource (blocks) serve as incremental units of change, which are distributed and redistributed among the agents and enables them to change their local environment. It is through the sharing and distribution of a common resource that the solution can be “built up” using only the state of the environment and very simple rule sets. A block repository serves as the environment in which

they interact. It creates and stores blocks which are distributed among the agents according to a very simple rule set. The block repository does not take any control action and is has no explicit goal to achieve. It simply creates blocks as needed, stores them temporarily so they can be distributed and redistributed to agents.

Previous implementations of the resource sharing algorithm have been on a computational system, modeled after an inverse heat transfer problem (Finzell and Bryden, 2016). This computational system was built into a physical system, where the algorithm was again applied as a control strategy (**Reference**). The standard implementation of the algorithm was compared to dynamic scaling on this system to determine under what conditions, dynamic scaling might improve the system response **(Reference)**. An initial implementation of the resource sharing algorithm was conducted on the Hyper facility and the tunable parameters were varied to find which values gave an optimal setpoint response **(Reference)**. In past implementations of the algorithm, the probability of action and block size were found through empirical testing to get a fast but accurate response. These values may give a desired response for setpoint tracking or disturbance rejection but not both. It may be possible that using dynamic scaling, the resource sharing algorithm may be able to give desired responses for both setpoint tracking and disturbance rejection.

6.4 Dynamic Scaling

Dynamic scaling changes the probability of action or block size based on the steady state error between the current value and the desired setpoint, where the block size or probability of action is greater when the current value is further away from the desired value and decreases as the current value approaches the desired value (Fig 4.1). Both the block and probability scaling

techniques uses the difference between the current temperature and the setpoint temperature to scale each parameter respectively. Both scaling techniques use the same equation to determine a scaling factor based on the steady state error. The scaling equation, given below, used to determine this scaling factor for both block scaling and probability scaling, was inspired by an equation used to model social insect behavior (Bonabeau et al., 1996; Bonabeau et al., 1997). X is the steady state error and the alpha value is a user selected parameter that determines the magnitude of the scaling factor given that the steady state error remains constant.

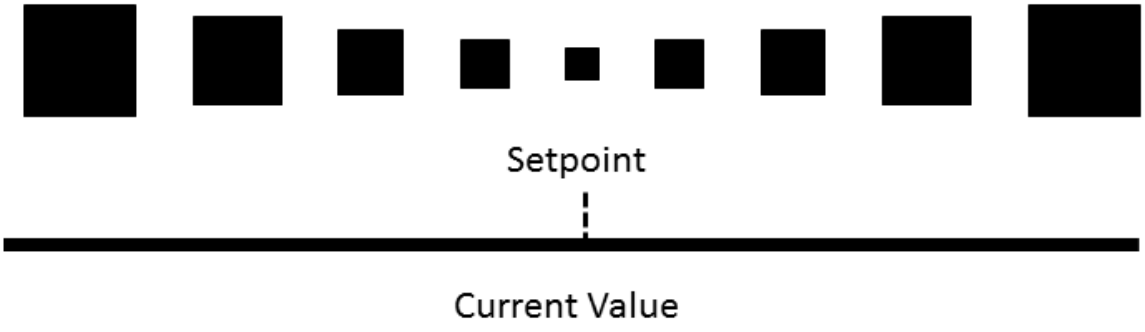
$$X = \left|\text{Current-Setpoint}\right| \tag{4.1}$$

$$\text{Scaling Factor} = \frac{X^2}{(X+\alpha)^2} \tag{4.2}$$

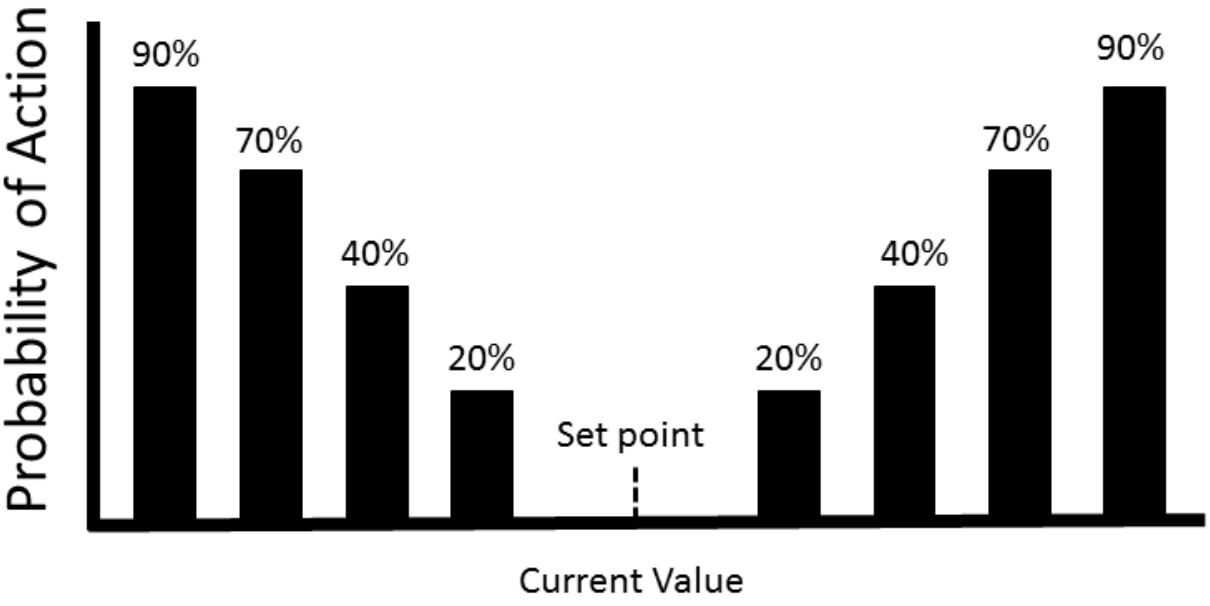
$$\text{Scaled Block} = \text{ScalingFactor} \times \text{Intial Block Size} \tag{4.3}$$

$$\text{Scaled Probability} = \text{ScalingFactor} \tag{4.4}$$

This scaling factor is a dimensionless parameter that creates a value that is either used directly as the current probability of action or is multiplied by an original block size to determine the current block size. The scaling equation gives a value between 0 and 1, which is the same scale as the probability of action. The block size is determined by the specific actuator is frequently not between 0 and 1. For block scaling there is an original block size, which the instantaneous block size can never exceed. The original block size is multiplied by a scaling factor to determine the instantaneous block size. The effect is that as the current value moves further away or closer to the desired setpoint, the current block size or probability of action will increase or decrease accordingly. At any point during the block scaling tests, the instantaneous block size cannot exceed the original block size. If steady state error is very high, the instantaneous block size will approach the original block size. For probability scaling, the value



(a)



(b)

Fig. 4.1a-b Block scaling and probability scaling.

from the scaling equation is used directly as the current probability of action. As the current value approaches the setpoint, the probability of that agent taking action (giving or taking a block) decreases and increases as the current value moves away from the setpoint.

6.5 Results

In a previous implementation of the standard resource sharing control strategy on the Hyper facility, two agents were developed, each composed of unique sensors and actuators. As shown in Fig. 5.3, agent 1 adjusts the electric load to control the turbine speed and agent 2 controls the cathode mass flow using the cold air bypass valve (**Reference**). Since each agent uses a different actuator (electric load and cold air bypass) to control their respective process variables, they will have different block sizes.

For each agent, three different alpha values were used with three probabilities of action. Changing these alpha values determines the magnitude of the initial probability of action or block size under similar conditions, i.e. at the instant of a setpoint change or when current value has reached the new setpoint. For each test, the setpoint is changed only after the system has reached steady state and is maintaining nominal conditions. For agent 1 that means a change from nominal conditions 40,500 RPM to 40,000 RPM and for agent 2 that means a change from 0.8kg/s to 0.6kg/s. Setpoint changes are used to evaluate and compare changes in system performance based on different configurations of the tunable parameters. A setpoint change was held for two minutes, after which the setpoint was changed back to nominal conditions. Due to the highly coupled nature and very rapid time dynamics of the system, both agents will be actively maintaining nominal conditions during the primary agent's setpoint change, to gauge to response of the secondary agent.

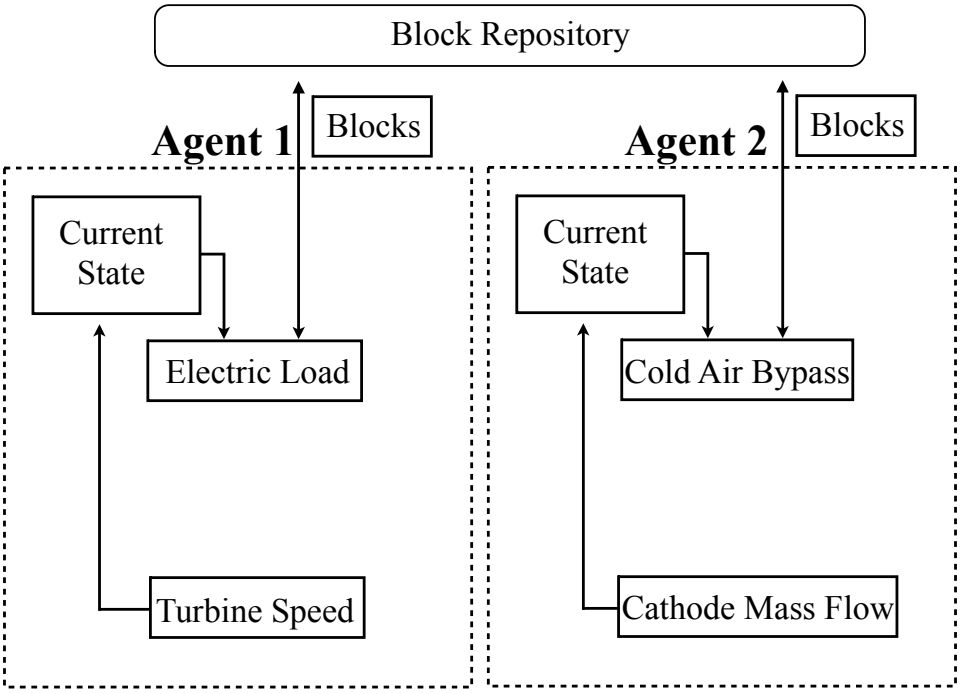


Figure 5.3. Agent 1 and 2 composition.

The tolerance determines the bounds around the setpoint in which the current value has to lie to have effectively reached a new setpoint. The tolerance for agent 1 was 75 RPM and for agent 2 was 0.025kg/s. The time step is set to 80 milliseconds, which determines how often decisions are made. For block scaling tests, probabilities of action of 30% and 90% are used in Fig. 6.1a-b and Fig. 6.3a-b for agent 1 and agent 2, respectively. The probability scaling test results are shown in Fig. 6.5 for agent 1 and Fig. 6.6 for agent 2. Below each graph of the setpoint response is the reaction by other agent as it maintains the nominal conditions. The results for both block scaling for agent 1 and 2 are shown in Table 6.1 and 6.2 and the probability scaling results are given in Table 6.3 and 6.4.

6.5.1 Block Scaling Tests

Since each agent has a unique block size associated with a specific actuator, block scaling requires an original block size from which the instantaneous block size can be scaled down using the scaling equation. For agent 1, the original block size was 1 kW and for agent 2 it was 3%. The instantaneous block size cannot exceed the original block size. If steady state error is very high, the instantaneous block size will approach the original block size. The block tests scaling used three difference probabilities of action (30%, 60% and 90%) and each probability of action was evaluated using three separate alpha values. Using the scaling equation the block size will always decrease as the current value approached the desired setpoint, so a smaller alpha value means that instantaneous block size will be higher at the instant of a setpoint change and when the current value approaches the new setpoint.

In Fig. 6.1a and Fig. 6.1b, shows the setpoint response for agent 1 when the alpha values were varied from 120-400. With a probability of action of 30% (Fig. 6.1a), as the alpha value

decreases, the rise time decreases but the overshoot increases. Values of 120 and 150 had very similar responses, with very fast rise times and a noticeable overshoot, while a value of 400 had a much slower rise time but the least amount of overshoot. Values of 150 and 400 had nearly the same settling times, while a value of 120 had the fastest settling time. In Fig. 6.1b, as the probability of action increased to 90%, the rise times and settling times were faster than when a probability of action of 30% was used. Like the previous test, the rise times for alpha values of 120 and 150 both fairly quick, but still had more overshoot than a value of 400. The trends for the settling times gave mixed results, where the shortest settling times were using alpha values of 120 and 400 and longest settling time was when the value was 150. Below each of the graphs for the turbine response (Fig. 6.1a-b) is the coupled response for the cathode mass flow and the control action taken using the cold air bypass valve (agent 2). For a probability of action of 90%, using an alpha value of 120 caused a spike in the cathode mass flow that created an unstable oscillation. This occurred because the probability of action is set very high (90%) and allowing control action be taken very quickly. If the control action causes the current value to overshoot, the rapid correcting action causes some oscillations around the setpoint. For this reason the probability of action is normal set lower than 90%. Having the probability of action set very high may have the potential to cause nearly simultaneous controller interactions, which is what the resource sharing algorithm is trying to avoid. Fig. 6.2a-b shows the overall trends for the rise times and settling times for agent 1 for probabilities of action of 30%, 60% and 90%. For all the block scaling tests, as the alpha value decreased and the probability of action increased, the rise times all decreased. There is there is less of an impact on settling time when the probability of action or values changed, but two fastest settling times are when the alpha value is at 120. Fig. 6.3a and Fig. 6.3b shows the setpoint response for agent 2 and when the alpha values change from 0.01-0.16. In Fig. 6.3a, as

the values decreased, so did the rise times and settling times. There very little overshoot or oscillations around the setpoint in all of the block scaling tests for agent 2. One potential reason for this is there is more fidelity in the cold air bypass valve than the electric load and there is less noise in the cathode mass flow sensor than the turbine speed sensor. As the probability of action of action increased to 90%, shown in Fig. 6.3b, the rise times and settling times decreased consistently for all alpha values.

Table 6.1. Agent 1 System Metrics (Block Scaling).

Rise Time

Alpha	POA = 30%	POA = 60%	POA = 90%
400	20.08	12.4	10.9
150	11.48	9.2	5.36
120	9.04	6.24	6.32

Settling Time

Alpha	POA = 30%	POA = 60%	POA = 90%
400	92.8	87.1	70.9
150	91.6	91.2	119.4
120	63.6	81.2	56

Overshoot

Alpha	POA = 30%	POA = 60%	POA = 90%
400	75.2	144.7	129.3
150	201.2	197.7	147.3
120	184	280	189.4

There was an unstable oscillation when the probability of action was 90% and the value was 0.01. Just like in the previous case when the probability of action was set to 90%, the corrective action to compensate for overshooting caused oscillations around the setpoint. Agent 2 was able to correct this oscillation fairly quickly by gradually reducing the overshoot. When there is a setpoint change to the cathode mass flow, the turbine speed also changes, which elicits a response from agent 1, which is shown below both Fig. 6.3a and Fig. 6.3b. The response from agent 1 was more pronounced and rapid than the response from agent 2 to a setpoint change from agent 1 and also increased in magnitude as probability of action increased.

Table 6.2. Agent 2 System Metrics (Block Scaling).

Rise Time

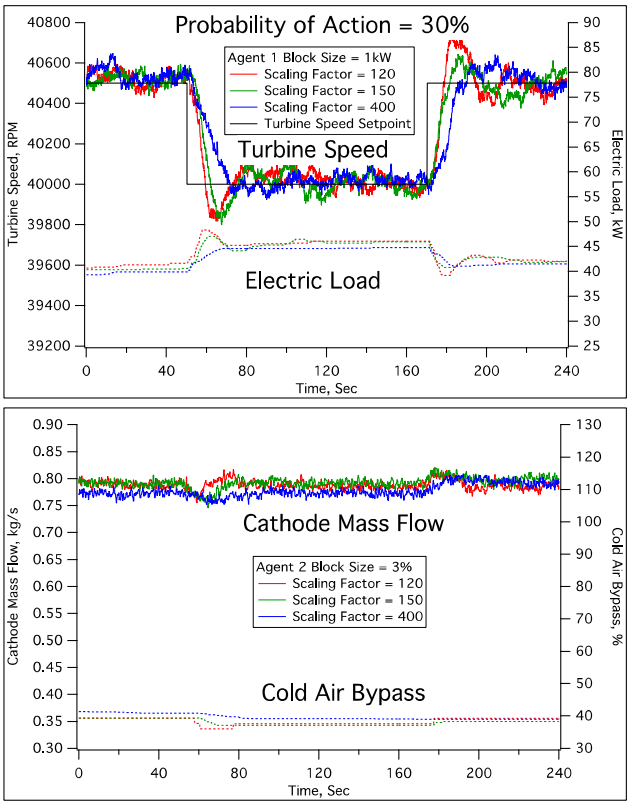
Alpha	POA = 30%	POA = 60%	POA = 90%
0.16	60.1	55.9	44.8
0.06	31.76	15.04	19.2
0.01	12.8	7.28	4.88

Settling Time

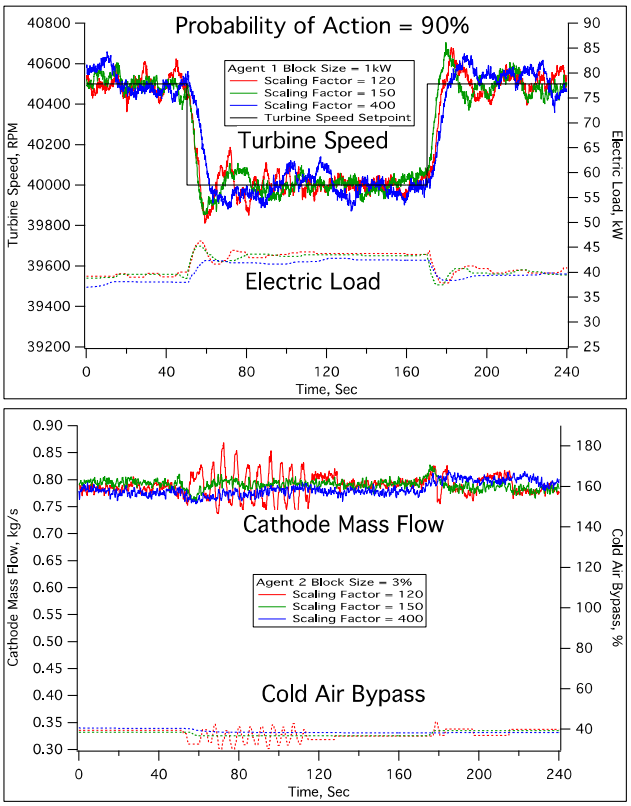
Alpha	POA = 30%	POA = 60%	POA = 90%
0.16	79.84	48.08	39.3
0.06	28.9	17.12	15.63
0.01	9.3	6.96	4.32

Overshoot

Alpha	POA = 30%	POA = 60%	POA = 90%
0.16	0.003	0.0002	0.0009
0.06	0.005	0.006	0.014
0.01	0.012	0.027	0.012



(a)



(b)

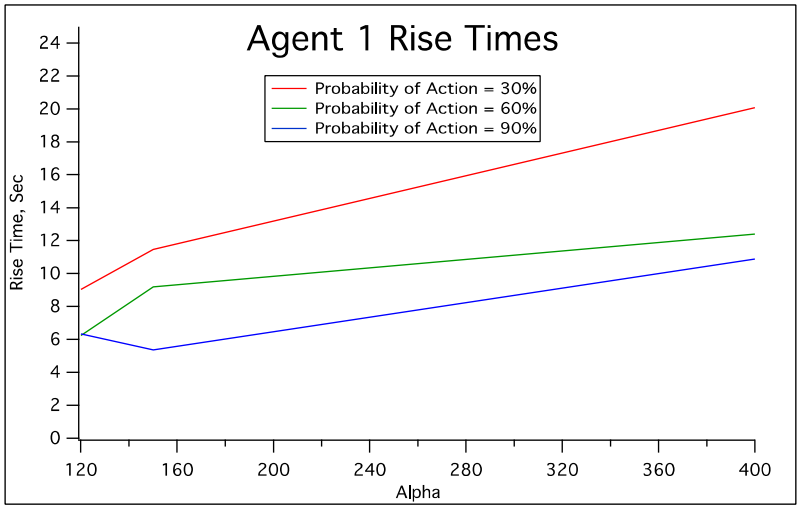
Figure 6.1a-b. Agent 1 block scaling (probability change).

The rise time and settling time trends for all of the probabilities of action for agent 2 are given in Fig. 6.4a-b, which were both shown to decrease steadily as the alpha values decreased and probabilities of action increased.

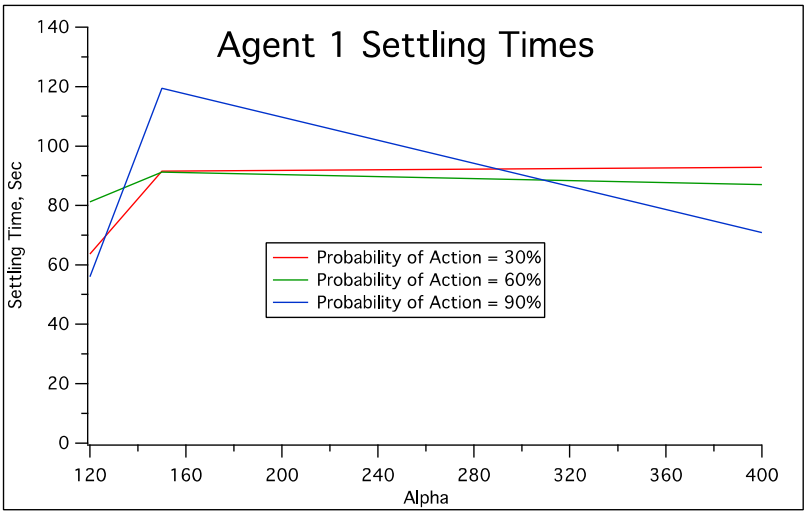
6.5.2 Probability Scaling Tests

The probability scaling tests used the results from the scaling factor equation directly as the current probability of action. These tests required setting block sizes, which were held constant for agent 1 and 2, for all the probability scaling tests. The block sizes were 0.5kW for agent 1 and 1.5% for agent 2. Fewer tests were needed to evaluate the setpoint response using probability scaling, because a constant block size was used and no probability of action needed to be specified. For both agents, three alpha values were used to evaluate the different setpoint responses. Tables 6.3 and 6.4 give the setpoint responses for the probability scaling test for agents 1 and 2 respectively.

Fig. 6.5 shows setpoint response for agent 1 using probability scaling. There is very little difference in the rise times as the alpha values increase, with the rise time slightly decreasing from values of 50 to 75 and an alpha value of 50 had noticeably more overshoot. For alpha values of 75 and 50, the settling times were roughly the same but using an alpha value of 25 gave the shortest settling time of any other test performed, including probability scaling. In all of the probability scaling tests for agent 1, the rise time was fairly quick so agent 2 needed to react quickly, which it was able to do with minimal deviation from the nominal cathode mass flow conditions. With agent 2 there was very little difference in both the rise time and settling

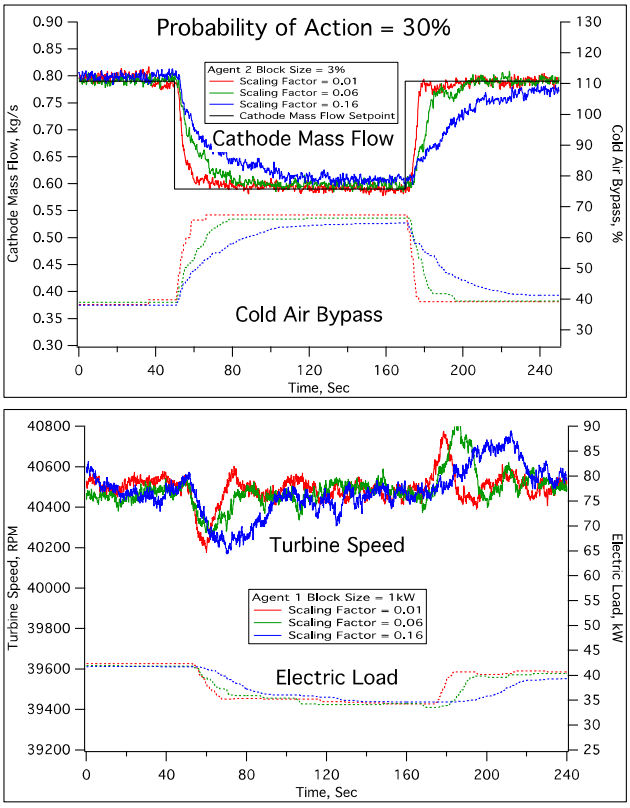


(a)

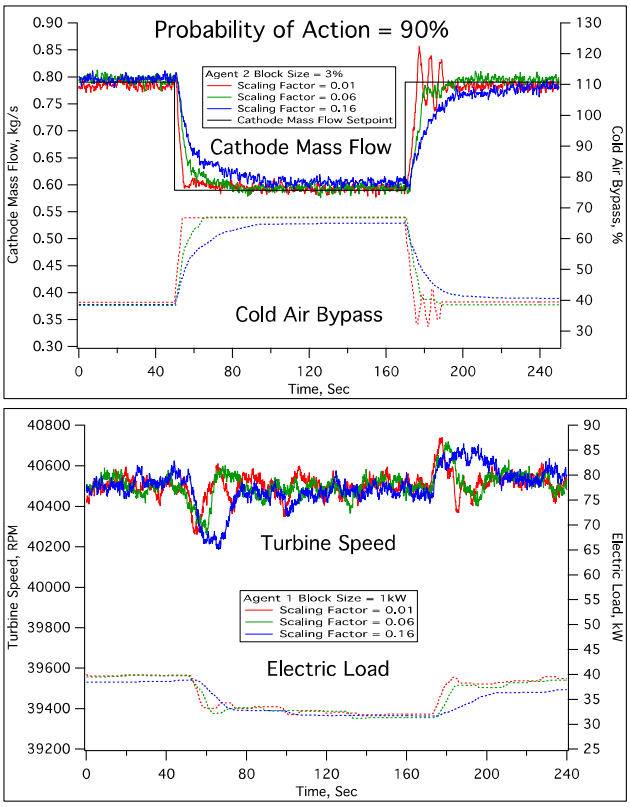


(b)

Figure. 6.2a-b Agent 1 Rise time and Settling Time (Block Scaling).

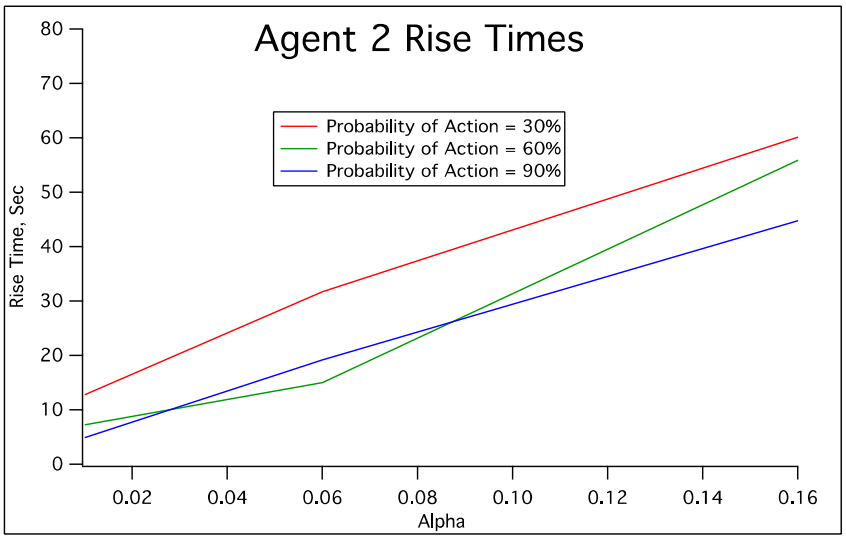


(a)

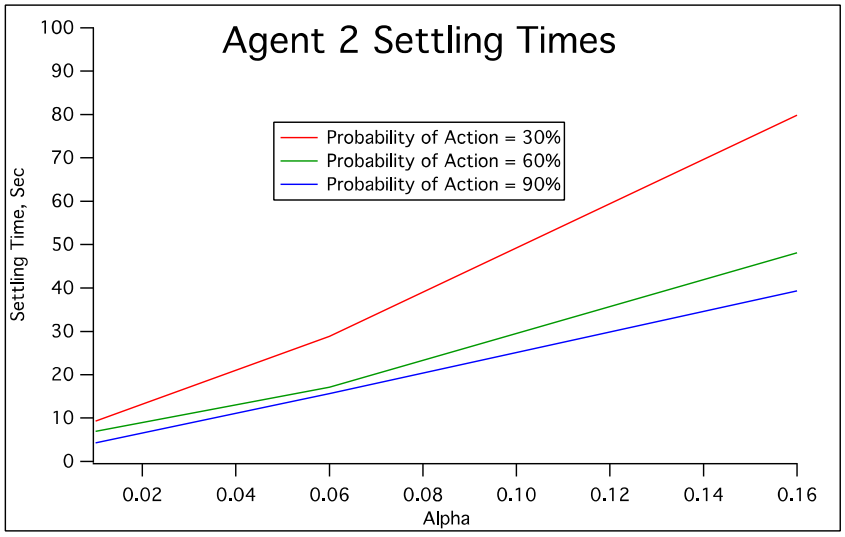


(b)

Figure. 6.3a-b Block Scaling Agent 2 Probability Change.



(a)



(b)

Figure. 6.4a-b Agent 2 Rise time and Settling Time (Block Scaling).

Table 6.3. Agent 1 System Metrics (Probability Scaling).

Rise Time

Alpha	BS = 0.5kW
75	5.2
50	7.28
25	7.28

Settling Time

Alpha	BS = 0.5kW
75	105.2
50	95.6
25	19.6

Overshoot

Alpha	BS = 0.5kW
75	121.5
50	245.1
25	127.9

Table 6.4. Agent 2 System Metrics (Probability Scaling).

Rise Time

Alpha	BS =1.5%
0.03	9.84
0.02	10.96
0.01	9.68

Settling Time

Alpha	BS =1.5%
0.03	8.32
0.02	8.24
0.01	9.1

Overshoot

Alpha	BS =1.5%
0.03	0.008
0.02	0.035
0.01	0.027

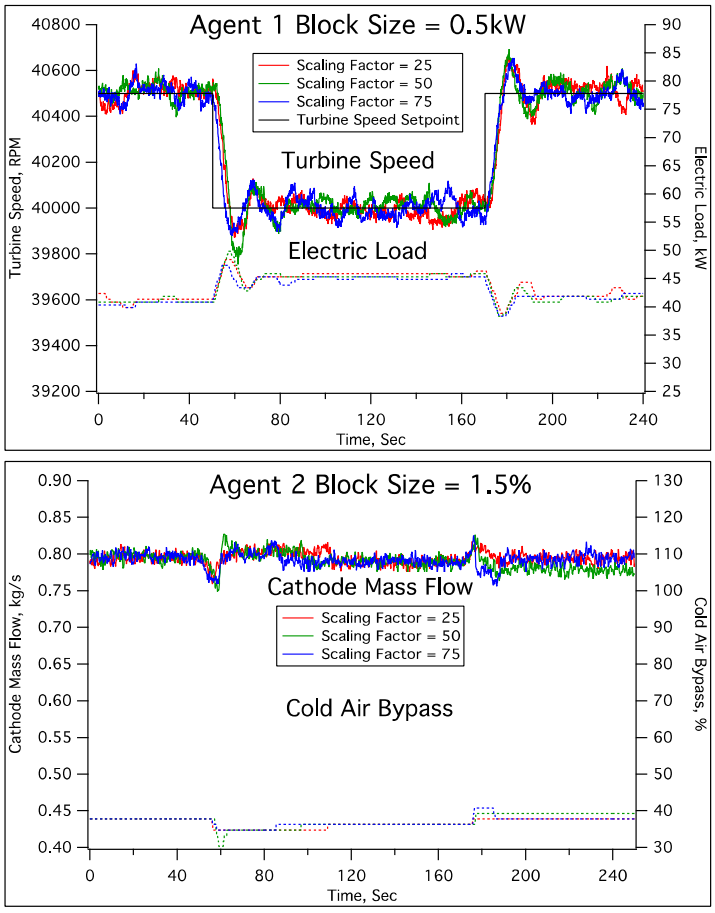


Figure. 6.5. Agent 1 Probability Scaling.

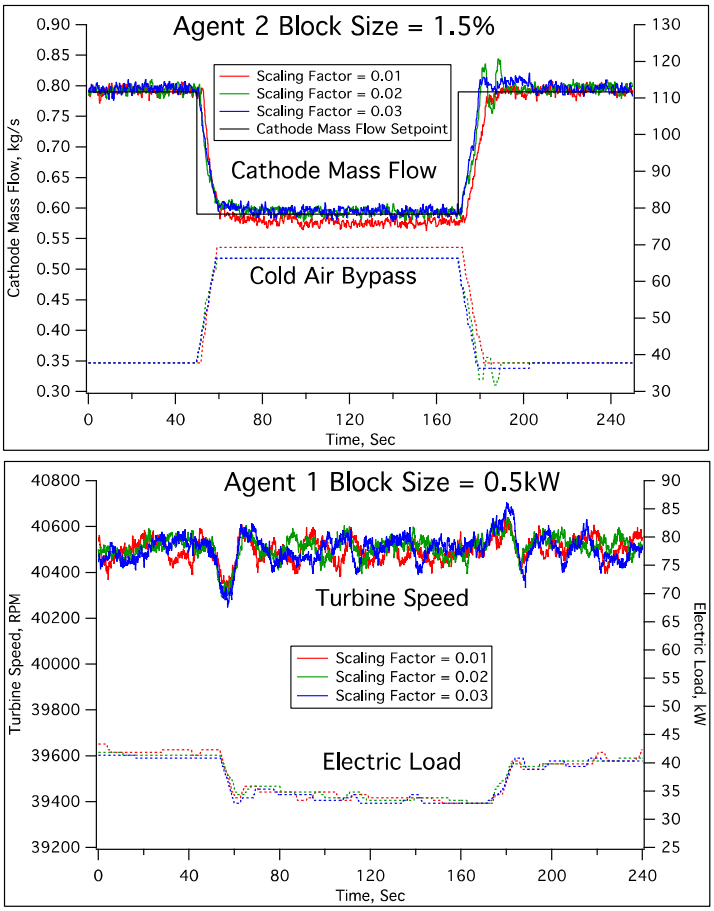


Figure. 6.6 Agent 2 Probability Scaling.

time between alpha values, with a slight decrease in the rise time when the value changed from 0.02 to 0.03 (Fig. 6.6). The response from agent 1 to the setpoint change of agent 2 was fairly quick and although the deviations from nominal conditions were more pronounced than the response from agent 2 when there was a setpoint change for agent 1, the turbine speed deviations were less than 200 RPMs for all three alpha values.

6.5.3 Standard Implementation, Dynamic Scaling and MIMO Controller

This section presents a comparison between a dynamic and standard implementation of the resource sharing algorithm and a MIMO controller. For more detailed discussion on the standard implementation and the MIMO controller see (**Reference**). Tables 5.1 and 5.2 have the rise times, settling times and peak overshoot values from the standard implementation for agent 1 and 2, respectfully.

Fig. 5.10 and 5.11 give the response for a multi-variable (MIMO) controller controlling turbine speed using electric load and cathode mass flow using the cold air bypass valve. Fig. 5.10 shows the setpoint response of the turbine when a new setpoint of 41,000 RPMs is set. This test increases the turbine speed setpoint by 500 RPMs instead of decreasing the turbine speed by 500 RPMs. The rise time for the turbine speed setpoint is 1.5 sec and the settling time is 7.28 seconds. Fig. 5.11 gives the MIMO controller response when there is a setpoint change in the cathode mass flow, where the rise time is 11.12 seconds and the settling time is 13.44 seconds.

For agent 1 the results were mixed when comparing all of the different controllers. The rise times for probability scaling were shorter than nearly all of the other resource sharing tests, with the exception of the fastest standard implementation test and the MIMO controller response. Probability scaling also produced the shortest settling time (with the exception of the MIMO

Table 5.1. Agent 1 System Metrics (Standard Implementation).

Rise Time

Probability of Action	Block Size = 0.25kW	Block Size = 0.5kW	Block Size = 0.75kW
10%	32.32	17.12	9.44
20%	24.64	12.16	7.6
30%	15.52	10.48	7.52
60%	12.08	7.68	5.28
90%	8.16	6.48	4.72

Settling Time

Probability of Action	Block Size = 0.25kW	Block Size = 0.5kW	Block Size = 0.75kW
10%	120	78.24	104.64
20%	75.2	68.64	118.64
30%	49.44	47.52	67.04
60%	61.44	79.92	70.8
90%	50.68	91.36	53.52

Overshoot

Probability of Action	Block Size = 0.25kW	Block Size = 0.5kW	Block Size = 0.75kW
10%	95.9	138.6	168.2
20%	114.6	114.8	163.3
30%	145.6	128.6	196.5
60%	147.4	175.3	165.3
90%	112	151.1	133.2

Table 5.2. Agent 2 System Metrics (Standard Implementation).

Rise Time

Probability of Action	Block Size = 1%	Block Size = 1.5%	Block Size = 2%
10%	52.64	47.2	36.4
20%	41.04	22.08	23.28
30%	22.64	15.2	13.2
60%	21.28	11.04	9.36
90%	15.12	14.96	8.48

Settling Time

Probability of Action	Block Size = 1%	Block Size = 1.5%	Block Size = 2%
10%	80.16	44.88	35.28
20%	43.92	19.36	20.64
30%	17.28	13.76	12.4
60%	13.44	8.48	8.72
90%	14.4	9.68	7.28

Overshoot

Probability of Action	Block Size = 1%	Block Size = 1.5%	Block Size = 2%
10%	0.005169	0.005769	0.01498
20%	0.009471	0.011912	0.00779
30%	0.014515	0.024647	0.028415
60%	0.015436	0.029429	0.028257
90%	0.020959	0.029277	0.027577

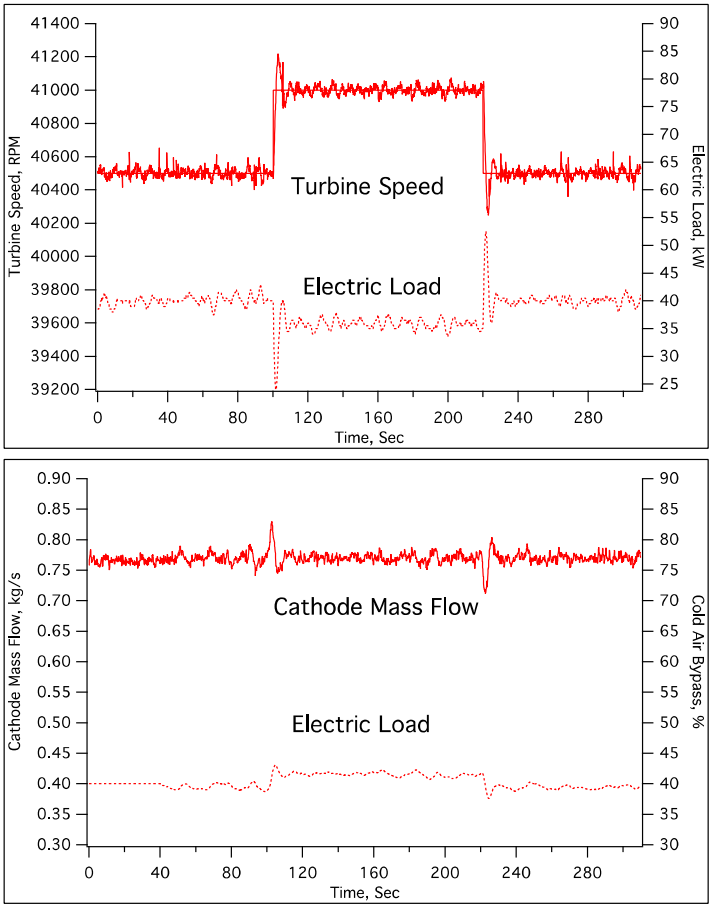


Figure. 5.10 MIMO control of turbine speed.

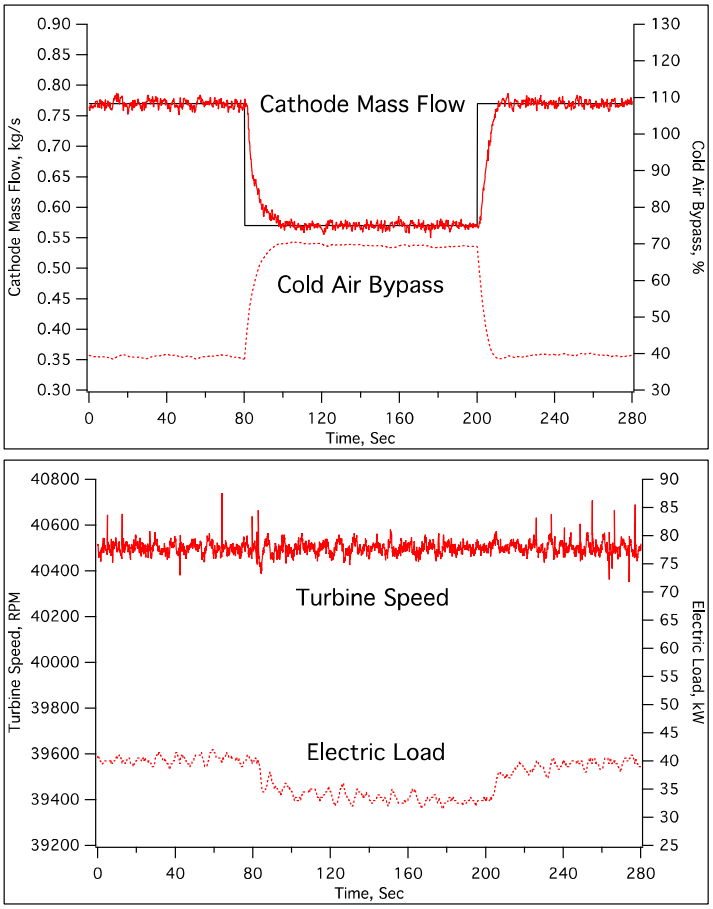


Figure. 5.11 MIMO control of cathode mass flow.

controller) with an alpha value of 25. The settling times for the block scaling generally fell in the middle range of all the tests using the resource sharing algorithm, while the rise times were on the faster end of all the tests. When comparing the same three block sizes and the same probabilities of action (30%, 60% and 90%), the standard implementation of the algorithm outperformed the block scaling in both rise time and settling time. One reason that the MIMO controller outperformed the resource sharing algorithm in terms of rise time and settling time is that it can make sudden input changes, jumping to approximately the correct inputs needed for desired outputs. Even with a large block size and large probability of action, the resource sharing algorithm will still make changes incrementally. For agent 2 the results were much clearer when comparing dynamic scaling, the standard implementation, and the MIMO controller. While the response from the probability scaling tests was comparable to the standard implementation of the algorithm, the block scaling tests had the fastest settling time and rise time, outperforming block scaling, the standard implementation of the algorithm and even the MIMO controller. The best response from the standard implementation of resource sharing algorithm, as well as the best responses from both the block scaling and probability scaling tests outperformed the MIMO controller for cathode mass flow response. The coupled response in the electric load may have contributed to the slower response in the setpoint change in the cathode mass flow. The quick control actions of the MIMO controller when adjusting the electric load, which allowed for an optimal response in the first test, may have caused delays in the setpoint response of the cathode mass flow. For agent 2 the best response came from block scaling, which had the shortest rise time and settling time of all the tests.

There is a tradeoff between the speed of the rise time for one agent during a setpoint change and the subsequent reaction of the other agent. The faster an agent is able to achieve a new setpoint,

the larger the spike in the corresponding agent's response. This is more pronounced when there is a very quick setpoint change as in the cathode mass flow. The deviations from the cathode mass flow nominal conditions during a turbine speed setpoint change were more pronounced using probability scaling than during block scaling. The turbine speed changed very quickly using probability scaling, which caused a greater instability in the cathode mass flow. Agent 1 was also able to bring the turbine speed back to nominal conditions during a cathode mass flow setpoint change much faster using probability scaling than any of the block scaling tests. There were also no unstable oscillations in the cathode mass flow using probability scaling, which happened twice in the block scaling implementation. These oscillations were caused by the large probability of action using block scaling. Using probability scaling, when the current value is around the setpoint, the probability of action will be very small.

6.6 Conclusions

The implementation of the dynamically scaled resource sharing algorithm on the Hyper has shown that it was able to control both the turbine speed and the cathode mass flow simultaneously under nominal conditions and during setpoint changes. The Hyper system response was evaluated for nine different setpoint changes using block scaling and three setpoint changes using probability scaling.

Finding the optimal values of the standard resource sharing controller to give the best setpoint response required balancing the size of the tunable parameters to reach a new setpoint quickly but not overshoot or oscillate around the setpoint. Dynamic scaling can be implemented faster than the standard implementation because the block size or probability of action will be smaller around the new setpoint and larger further away from it. The setpoint response for dynamic

scaling is comparable to the standard implementation and in some instances outperforms the standard implementation.

For agent 1, probability scaling showed significant performance increases over block scaling and the standard implementation of the algorithm in regards to settling time. It had faster settling times than both the standard implementation of the algorithm and the block scaling, but did not outperform the MIMO controller. All of the rise times for the probability scaling tests were faster than all but three of the rise times for the standard implementation. For agent 2, the block scaling test gave the best performance, in terms of rise time and settling time over all the tests, including the MIMO controller. This shows there are instances when the probability scaling is preferred, when block scaling is preferred, and when the standard implementation is preferred.

Dynamic scaling means that the algorithm can be implemented very quickly, but if the response characteristics of the normal implementation are preferred, it may also be possible to use information from dynamic scaling to determine the optimal values for the standard implementation. Future tests will attempt to use valued from dynamic scaling in the standard implementation. Combining dynamic scaling and the standard implementation, could be more beneficial than using a single methodology for both agents. Using the best performing implementations for each agent, Agent 1 could use a standard implementation or block scaling while agent 2 used probability scaling. Future tests will include mixing the dynamic scaling and the standard implementation.

The standard implementation and the dynamically scaled resource sharing algorithm possesses several advantages over traditional control approaches. Hybrid energy systems, especially when composed of many different sensors and actuators, frequently require complex models to describe their behavior accurately. The resource sharing control strategy can offer a

greater level of control for these complex, coupled hybrid systems using only local information, simple rule sets and a shared resource distributed among agents. The creation of these computational agents not only allows for more flexibility and adaptability where agents can be added or removed as needed or reaction to new conditions. These computational agents can also make use of many sensors, all without a detailed system model and avoiding undesirable simultaneous interactions.

Acknowledgments

This research was supported by the US Department of Energy – Office of Fossil Energy under Contract No. DE-AC02-07CH11358 through the Ames Laboratory.

References

Akyildiz, I.F., Weilian Su, Sankarasubramaniam, Y., Cayirci, E., 2002. A survey on sensor networks. IEEE Commun. Mag. 40(8), 102–114. doi:10.1109/MCOM.2002.1024422

Åström, K.J., Albertos, P., Blanke, M., Isidori, A., Schaufelberger, W., Sanz, R., 2012. Control of Complex Systems. Springer Verlag, London.

Åström, K.J., Hagglund, T., 2006. Advanced PID control. ISA-The Instrumentation, Systems, and Automation Society, Research Triangle Park.

Bajpai, P., Dash, V., 2012. Hybrid renewable energy systems for power generation in stand-alone applications: A review. Renew. Sustaina. Energy Reviews 16(5), 2926–2939. doi:10.1016/j.rser.2012.02.009

Bakule, L., 2008. Decentralized control: An overview. Annu. Rev. Control 32(1), 87–98.

Chakraborty, S., Simoes, M.G., Kramer, W.E., 2013. Power Electronics for Renewable and Distributed Energy Systems. Springer-Verlag, London.

Bizon, N., Shayeghi, H., Mahdavi Tabatabaei, N., 2013. Analysis, control and optimal operations in hybrid power systems. Springer Verlag, London. doi:10.1007/978-1-4471-5538-6

Burch, G.D., 2001. Hybrid Renewable Energy Systems. Presented at: U.S. DOE Natural Gas/Renewable Energy Workshops.

De Silva, C.W., 1995. Intelligent control. CRC Press, Boca Raton.

DOE, 2015. Quadrennial Energy Review: An assessment of energy technologies and research opportunities. Available at: http://energy.gov/sites/prod/files/2015/09/f26/Quadrennial-Technology-Review-2015_0.pdf

Dou, C.-X., Wang, W.-Q., Hao, D.-W., Li, X.-B., 2015. MAS-based solution to energy management strategy of distributed generation system. Int. J. Elec. Power 69, 354–366.

Khaki-Sedigh, A., Moaveni, B., 2009. Control configuration selection for multivariable plants. Springer Verlag, Berlin.

Jun, Z., Junfeng, L., Jie, W., Ngan, H.W., 2011. A multi-agent solution to energy management in hybrid renewable energy generation system. Renew. Energ. 36(5), 1352-1363.

Lin, W.-M., Hong, C.-M., Chen, C.-H., 2011. Neural-Network-Based MPPT Control of a Stand-Alone Hybrid Power Generation System. IEEE T. Power Electr. 26(12), 3571–3581. doi:10.1109/TPEL.2011.2161775

Maley, S., Rawls, P., 2014. Crosscutting Research Sensors and Controls Project Portfolio.

Mueller, F., Jabbari, F., Brouwer, J., 2009. Linear quadratic regulator for a bottoming solid oxide fuel cell gas turbine hybrid system. J. Dyn. Sys. Meas. Control 131(5), 23–37.

Natsheh, E.M., Albarbar, A., 2013. Hybrid power systems energy controller based on neural network and fuzzy logic. Smart grid renew. energ. 4(2) 187–197. doi:10.4236/sgre.2013.42023

Pezzini, P.N., 2014. Control Strategy for a Direct-Fired Fuel Cell Turbine Hybrid Power System and Decentralized/Centralized MIMO Control Approach. PhD Dissertation, University of Genoa, Genoa, Italy.

Restrepo, B., 2011. Experimental characterization of fuel cell gas turbine power system and determination of optimal trajectories of operation using a model predictive controller. PhD Dissertation, West Virginia University, Morgantown, WV.

Shelton, M., Celik, I., Liese, E., Tucker, D., 2010. A study in the process modeling of the startup of fuel cell/gas turbine hybrid systems. J. Eng. Gas Turb. Power 132(1), 012301–012308. doi:10.1115/1.2830551

Siljak, D.D., 2013. Decentralized control of complex systems. Dover, Mineola.

Tsai, A., Banta, L., Tucker, D., 2010. Multivariable robust control of a simulated hybrid solid oxide fuel cell gas turbine plant. J. Fuel Cell Sci. Tech. 7(4), 0410081– 0410089.

Tsai, A., Tucker, D., Groves, C., 2011. Improved Controller Performance of Selected Hybrid SOFC-GT Plant Signals Based on Practical Control Schemes. In: Proceedings ASME Turbo Expo 2010.

Tucker, D., 2003. U.S. Department of Energy, “NETL HYPER Test Plan”.

Wächter, C., Lunderstädt, R., Joos, F., 2010. Using linear control theory for parameterization of a controller for a SOFC/GT hybrid power plant. J. Fuel Cell Sci. Technol. 7(3), 0310031–0310039. doi:10.1115/1.3206972

CHAPTER 7. CONCLUSION

A new control strategy based on distributed construction was developed for advanced energy systems to manage complex and highly coupled components and interactions, while making effective use of advances in sensor technology. This resource sharing control strategy was able to control two different systems without developing controlling equations, offline learning and tuning, or explicit coordination between controllers.

Validation of the algorithm began on a computational environment that simulated an inverse radiation enclosure problem for radiant heat transfer between ten design surfaces and ten heater surfaces. The resource sharing algorithm was able to maintain the desired temperature profiles for design surfaces for all the cases examined. The algorithm was adapted into a control strategy, which was applied to a radiant heater test rig that simulated the same radiant heater enclosure problem examined in the computational simulation. The setpoint response was compared to a PI controller and the research sharing algorithm demonstrated comparable performance to the PI controller without significant a priori knowledge of the system response. This research culminated by applying the resource sharing control strategy to the Hyper facility. The Hybrid Performance (Hyper) is a 2500 square foot research project that combines a gas turbine recuperated cycle and a virtual solid oxide fuel cells (SOFCs) stack, Fig. 5.1. It is currently being undertaken at Department of Energy's National Energy Technology Laboratory (NETL) and has an 800kW capacity.

The resource sharing controller was successfully implemented on the Hyper system and was able to achieve the desired setpoint in all the cases examined. The resource sharing algorithm was compared to a MIMO controller on the Hyper facility. The MIMO controller has a very fast response and takes action using both controllers simultaneously, which causes a very fast setpoint response for the turbine setpoint change but creates oscillating control actions when there is a cathode mass flow setpoint change. Simultaneous control action is avoided in the resource sharing algorithm, because only a single controller can take or receive a block during any time step, even if the time in between time steps is very small. The resource sharing algorithm requires significantly less system information beforehand compared to the MIMO controller to get a good setpoint response.

Implementation of the resource sharing control strategy on the Hyper facility demonstrated that a sufficiently large block size and probability of action were needed for a quick and accurate response to a setpoint change to account for the fast dynamics and tightly coupled of the system. Smaller block sizes and slower probabilities of action were needed for the slower time dynamic of the radiant heater test rig. This algorithm was able to control the several different systems, each without detailed system models. Although, the probability of action and block size were found through minimal empirical testing, future implementations will use dynamic scaling to determine the appropriate initial values for the block size and probability of action.

Dynamic scaling uses the steady state error to scale the block size or probability of action as the current value approaches the setpoint. The effect is that as the current value moves further away or closer to the desired setpoint, the current block size or probability of action will increase or decrease accordingly. The decreased risk in picking a block size or a probability of action that is too big is minimized because any block size or probability of action will be scaled down as the

current value approaches the setpoint. Dynamic scaling can be implemented in systems were there is a risk of damaging equipment since scaling offers some assurances that the response will be at least muted. Additionally, if the response from the standard implementation of the resource sharing algorithm is preferred, dynamic scaling can be used as a tuning mechanism for the probability of action or the block size.

Complex energy systems frequently require detailed system models to describe their behavior accurately. Constructing equations relating inputs and outputs serves as the basis for conventional control strategies. The resource sharing algorithm uses only local information, a shared resource and simple rule sets to achieve the desired conditions. The block repository only needs to know how agents are in a system and of those, how many are requesting blocks. The only information each agent needs to make a decision is whether they are at their desired state and if not, each agent will distribute and redistribute blocks until their specific state is met. Using local environmental information as a means of coordination and communication, computational agents, using simple rule sets, are able to manipulate their environment to indirectly achieve a global goal. This algorithm can be applied to any situation where a desired global state is specified, but the inputs required to get to that state are unclear. Agents can be added or removed seamlessly without reconfiguring the system or adjusting the rule sets of the agents or the block repository. This allows the algorithm to be more flexible and adaptable and allows for rapid implementation. The emergent and random behavior of the resource sharing algorithm means that not all the agents behave the same way at any point during a test. While this has the potential to be more time consuming, this behavior can lead to more robust and diverse solutions, while being adaptable to potential system changes.

7.1 Future Areas of Research

Continuing research will include testing of the resource sharing control strategy on the two existing systems with different control objectives. The control strategy will also be compared against additional control strategies to evaluate its effectiveness and performance characteristics. Future research would include developing a systematic tuning method for obtaining the optimal block size and probability of action under specific circumstances. One potential method could be using dynamic scaling as a tuning mechanism for the resource sharing control strategy. Once a tuning methodology is developed on existing systems, either the radiant heater test rig or the Hyper facility, this control strategy could be applied to new systems, where more agents could be used.

More agents could be added to the current test systems, using the same sensors and actuators, by introducing the concept of *co-worker agents*. Individual insects within social insect colonies have different abilities and different preferences but there is still a robust division of labor without a centralized coordinator. Choosing which task to perform is also based on the abilities and preferences of an individual insect. Several insects may decide to work on the same task simultaneously. In all of the previous implementation of the research sharing algorithm and research sharing control strategy, there has only been a single agent for each sensor and actuator. Co-worker agents perform the same task with different abilities and different preferences about when to take action. If co-worker agents were created, many agents could work on tasks simultaneously and there could be overlap and specialization. Agents can be created and tuned for a specific operating range or set of circumstances but they would still have a random probability of taking action at any given time. They could, however, be more or less likely to take action in certain circumstances and if they were selected, they would have the most appropriate response.

Setting the range at which these agents could take action would be set using the tolerance and probability of action. By controlling the tolerance, the range that an agent will decide to take action can be manipulated. Having different tolerances for each co-worker agent creates overlapping region where multiple agent can take action but limits the range that some agents will take action, especially around the setpoint. Varying the probability of action between co-worker agents creates agents that are more likely to take action and less likely to take action. This can be accompanied by varying the block size as well, i.e. agents with a large block size might have a low probability of action while agents with a small block size might have a higher probability of action. These overlapping regions, along with multiple agents working on the same task with different skill sets and preferences, creates a distribution of labor that mimics social insects and will hopefully create much better system response under varied conditions.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Prof. Kenneth Mark Bryden for the continuous support during my Ph.D study, for his patience, motivation, and immense knowledge. His guidance helped me in throughout my research, academic studies and writing of this thesis.

I would like to thank my thesis committee: Dr. Xinwei Wang, Dr. Atul Kelkar, Dr. Halil Ceylan, and Dr. Umesh Vaidya for their valuable insight, tough questions and new perspectives which challenged me to expand my thinking. I would like to thank Dr. Paolo Pezzini, who's guidance and wisdom have been invaluable during this research. Thanks to everyone at the National Energy Technology Laboratory who funded and supported this research and to fellow labmates for the stimulating discussions and invaluable feedback. I would like to thank my friends and family for their support and reassurance during stressful and uncertain times. And lastly to my parents, whose love, support and encouragement made all of this possible. Thank you.

Acknowledgments

This research was supported by the US Department of Energy – Office of Fossil Energy under Contract No. DE-AC02-07CH11358 through the Ames Laboratory.