A Novel Resource Sharing Algorithm Based on Distributed Construction for Radiant
Enclosure Problems

Peter Finzell, Kenneth M. Bryden¹

Simulation Modeling and Decision Science Program
Ames Laboratory

1620 Howe Hall, Ames, Iowa, 50011

¹ Contact information

Kenneth M. Bryden
1620 Howe Hall
Iowa State University
Ames, IA 50011

Email: kmbryden@iastate.edu

Ph: 1-515-294-3891

Fax: 1-515-294-3261

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. 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 and maintain a specified temperature distribution over several *design surfaces*. A design surface could be any material in a manufacturing process that requires precise or uniform temperature control such as glass or metal. These types of problems are concerned with the geometric design of the enclosure or the correct heater surface inputs to produce a desired temperature profile along the design surface or both (Howell et al., 2000). Radiant enclosure design 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 the 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). Optimization is an iterative process that solves the problem repeatedly whereas regularization adds additional information to the problem to find a less accurate but stable solution.

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

$$\min f(x) \quad (1)$$

is subject to

$$g(x) = c \quad (2)$$

and

$$h(x) \in d \quad (3)$$

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.

$$f = \frac{1}{N} \sum_{i=1}^N [T_i - T_i^*]^2 \quad (4)$$

Where N is the number of 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 (Howell et al., 2000; Ertürk et al., 2002a). Radiant enclosure problems have also been solved using different geometric configurations and initial conditions (Daun et al., 2003a). 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. The matrix of singular values is 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 distributed 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. 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 where order is created spontaneously and without planning, using insects interactions with their environment. Dorigo's work using pheromones for routing path optimization is one of the seminal works that combines self-organization, insect behavior, and a practical implementation of stigmergy (Dorigo and Stützle, 2004). This current work differs in that it focuses on the stigmergic construction process. For example, paper 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 global goal 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.

2.1. 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 $i_1 \dots i_n$ is the desired output, and the spatial temperature profile of the heater surfaces $j_1 \dots j_n$ 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.

$$Q_i = A_i \sum_{j=1}^N F_{i-j} (T_i^4 - T_j^4) \quad (5)$$

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

$$T_i \geq 26^\circ C \quad (6)$$

$$T_j \geq 26^\circ C \quad (7)$$

$$T_j \leq 300^\circ C \quad (8)$$

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, 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 its 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 monitors the state of a single design surface and takes action to change the temperature of that design surface by transferring blocks to or from a single heater surface, shown in Fig. 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 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 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 setpoint (Fig. 4). The block repository then 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. 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 (Fig. 5). The rules governing the behavior of the agents are:

- Each agent is only capable of detecting the current state of their environment, i.e., design surface.
- There are three 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 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.
- 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. In all the cases examined there were sufficient blocks for all of the agents to reach their setpoint values because there is some overshoot or period of time where the agents give blocks back even while the block repository is still creating blocks. If there weren't enough blocks, the condition where the block repository stopped creating blocks could be changed by the user.

To better understand the algorithm, consider the example shown in Fig. 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).

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. 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 at the last iteration of a single test case for 100 cases and the design surface temperatures were averaged at each iteration for 25,000 iterations 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 $\pm 1^{\circ}\text{C}$, 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 radiative 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. 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 1. The final averaged design surface temperature profiles and their associated error from their setpoint values are given in Table 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. 12,15,17 and 18, while test cases 5 and 7 required more iterations to converge and are shown in Figs. 13 and 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.

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 the 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 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 $\pm 1^{\circ}\text{C}$ and a block size of 0.2°C . In this case, as shown in Fig. 12, all of the agents achieved their setpoint temperatures within the tolerance band of $\pm 1^{\circ}\text{C}$ 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 was smooth and parabolic with minimal error in the design surface temperatures and small confidence intervals. Agents 1 and 10 required hotter heater surface temperatures than the rest of the agents to reach their desired design surface temperatures, as a result, design surfaces 1 and 10 took longer to reach the desired temperatures. Since the inside agents received the most radiant heat from all the heater surfaces, agents 5 and 6 slightly overshot after they initially reached their setpoint temperature.

In test case 5, shown Fig. 13, the temperature response of the design surfaces is given when the tolerance was set to $\pm 0.5^{\circ}\text{C}$. With the exception of the outside agents (agents 1 and 10), all of the agents attained their setpoint temperatures within a tolerance of $\pm 0.5^{\circ}\text{C}$ 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 $\pm 0.5^{\circ}\text{C}$. 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 surfaces 3 and 8, which is shown in Fig. 14. The confidence intervals for the heater surfaces and errors for the design surfaces are also greater than the previous test cases.

When the tolerance is set to $\pm 1.5^{\circ}\text{C}$, as shown in Fig. 15, all of the agents reached their setpoint temperatures within a tolerance of $\pm 1.5^{\circ}\text{C}$ 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.

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 $\pm 1^{\circ}\text{C}$ was used.

When the block size is reduced to 0.1°C , as shown in Fig. 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, 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. 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. 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 19 shows the final heater surface temperature profile and error bars for case 9 and test case 4 for comparison.

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 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, such as energy systems, manufacturing facilities or networked systems. 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 means the algorithm is scalable and 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.
- Dorigo, M., Stützle, T., 2004. *Ant Colony Optimization*. MIT Press, Cambridge.
- 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., Péntes, 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.

Figure 1. Radiation heat transfer between heater surfaces, design surfaces and the surrounding environment.

Figure 2. Agent-based distributed construction of a grid.

Figure 3. Agent composition.

Figure 4. Agent and block repository interactions.

Figure 5. Agent rule set.

Figure 6. Radiant enclosure example.

Figure 7. Resource sharing algorithm example.

Figure 8. Case 1 - Heater surface temperature profile with surrounding environment at 70°C.

Figure 9. Case 2 - Heater surface temperature profile with surrounding environment at 65°C.

Figure 10. Case 3 - Heater surface temperature profile with surrounding environment at 60°C.

Figure 11. Case 4 - Heater surface temperature profile with surrounding environment at 55°C.

Figure 12. Case 4 - Design surface temperatures as a function of iteration.

Figure 13. Case 5 – Design surface temperatures as a function of iteration.

Figure 14. Case 5 - Heater surface temperature profile.

Figure 15. Case 6 - Design surface temperatures as a function of iteration.

Figure 16. Case 7 - Design surface temperatures as a function of iteration.

Figure 17. Case 8 - Design surface temperatures as a function of iteration.

Figure 18. Case 9 - Design surface temperatures as a function of iteration.

Figure 19. Case 9 - Heater surface temperature profile.

Table 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. 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