A Game-Theoretic Analysis of Multi-Agent Systems for Shop Floor Routing

Vincent A. Cicirello

CMU-RI-TR-01-28

September 2001

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

© 2001 Carnegie Mellon University

Abstract

In recent years, the desire for a more robust basis for coordination in manufacturing environments has motivated research into agent-based approaches to manufacturing scheduling and control. We have been developing a series of such multi-agent systems inspired by models of social insect behavior for the problem of dynamic shop floor routing. To date, all of our results have been empirical in terms of global system performance and we have been without any satisfying explanation for the behavior of the interacting agents of these systems. This paper is an attempt to model these interactions in game-theoretic terms as normal form games. In the analysis, we show that the behavior of the agents in a repeated game scenario appears to converge upon the repeated play of Nash Equilibria (NE). This analysis consists of the examination of games corresponding to specific problem instances. Some of the systems in this study tend to converge upon deficient NE; while the others successfully coordinate upon "better" equilibria on a consistent basis.

Contents

1 Introduction

The factory is a complex dynamic environment and manufacturing organizations are constantly faced with the need to rearrange production. New and evolving market opportunities lead to changing product demands and manufacturing priorities. Changes in resource availability affect production capacity and force reassessment of current production goals. Such changing circumstances are quite frequently at odds with attempts to build advance schedules. Though advance scheduling can provide a basis for configuring factory resources to optimize performance relative to (currently) known requirements and constraints, these prescriptive solutions also tend to be quite brittle and they can quickly become invalidated by unexpected events.

In practice, manufacturing operations are often coordinated in a decentralized manner. The use of local dispatch scheduling policies [Morton and Pentico, 1993], for example, is commonplace in many manufacturing environments. By making decisions only when needed to keep execution going and by basing them on aspects of the current dynamic state, dispatch-based strategies are quite insensitive to unexpected events and yield very robust behavior. This advantage can also be a disadvantage, however, as decisions are made myopically and this can lead to suboptimal factory performance.

The desire for a more robust basis for coordination has similarly motivated research into agent-based approaches to manufacturing scheduling and control (e.g., [Lin and Solberg, 1992; Liu, 1996; Ow *et al.*, 1988; Parunak *et al.*, 1998; Sycara *et al.*, 1991]) and there have been a few interesting successes. For example, Morley and Schelberg use a simple bidding mechanism for assigning trucks to paint booths at a General Motors factory [Morley and Schelberg, 1993]. They show that their decentralized approach which imparts decision-making ability upon the paint booths outperforms the previously used centralized scheduling system in terms of the global objective. The schedules produced in advance by the more traditional centralized system tend to break down at the occurrence of an unexpected event and require an expensive reschedule with often poor global performance as a consequence; whereas Morley and Schelberg's system is robust to such unexpected events in that all decision-making takes place only when needed.

However, decentralized approaches can sometimes be susceptible to sub-optimal and even chaotic global behavior. Kempf and Beaumariage show how a manufacturing system, utilizing simple dispatch policies, can exhibit formally chaotic behavior [Kempf and Beaumariage, 1994; Beaumariage and Kempf, 1995]. Their example system has a number of attractors with drastically different global performance and they show that small changes in policies or in the system state can force the system to move between these attractors leading to large changes on a global scale. The ability to orchestrate good global performance via local interaction protocols and strategies remains a significant and ill-understood challenge.

One approach to this class of problem is to view establishment of appropriate coordination policies as an adaptive process. There are many examples of effective, adaptive behavior in natural multi-agent systems (e.g., [Fitzgerald and Peterson, 1988; Gordon, 1996; Kirchner and Towne, 1994; Theraulaz *et al.*, 1991]), and computational analogies of these systems have served as inspiration for multi-agent optimization and control algorithms in a variety of domains and contexts (e.g., [Beckers *et al.*, 1994; Bonabeau *et al.*, 2000; Dorigo and Di Caro, 1999; Schoonderwoerd *et al.*, 1997]). Bonabeau et al. [Bonabeau *et al.*, 1999] provide a comprehensive survey of adaptive multi-agent systems that have been inspired by social insect behavior[1].

We have developed a series of systems for the problem of dynamic shop floor routing based on such models of social insect behavior [Cicirello and Smith, 2001a; 2001d; 2001b]. Empirically, these systems seem effective; but we have neither a theoretical basis for nor a theoretical understanding of the behavior of the individual agents of these systems nor of the global performance of these systems. In this paper, we model these systems in game-theoretic terms as normal form games and show that the behavior of the interacting agents can be explained in terms of Nash Equilibria (NE) and that global system performance correlates in some way to the payoffs of these equilibria (i.e., deficient equilibria correspond to poor global performance while Pareto-optimal equilibria correspond to better global performance). The problem can then be viewed as a coordination game; and the question becomes whether or not the adaptive agents of these systems can coordinate upon Pareto-optimal NE while avoiding deficient NE in a repeated game scenario.

The paper assumes the reader has some basic knowledge of game theory terminology (see [Luce and Raiffa, 1985] or [Kreps, 1990] for an introduction to game theory). The remainder of this paper proceeds as follows. Section 2 presents a formal definition of the shop floor routing problem. Section 3 first briefly describes two multi-agent systems for the problem that are based on ant trail following behavior and then analyzes the behavior of these systems in game-theoretic terms. Section 4 describes the next two multi-agent systems we have previously designed for the problem based on wasp task allocation behavior and then analyzes these two systems with respect to game theory. Finally, Section 5 concludes the paper and discusses potential future work.

2 Shop Floor Routing Formalized

In this Section, we present terminology and formal definitions regarding the particular form of shop floor routing under analysis.

[1]For a survey focused on manufacturing applications of social insect behavior see [Cicirello and Smith, 2001c].

The shop floor consists of a set of machines[2] $M = \{m_1, m_2, \ldots, m_n\}$. Each of the machines $m_i \in M$ is multi-purpose in that it is capable of performing some set of tasks $T(m_i) = \{t_{i,1}, t_{i,2}, \ldots, t_{i,p}\}$. Furthermore, there may or may not be alternative machines capable of performing a given task. That is, $t_{i,j} \in T(m_i) \Rightarrow (\neg\exists t_{k,l} \in T(m_k), t_{k,l} = t_{i,j} \wedge k \neq i) \vee (\exists t_{k,l} \in T(m_k), t_{k,l} = t_{i,j} \wedge k \neq i)$. Each task $t_{i,j}$ requires some amount of processing time $P(t_{i,j}) > 0$. If two or more machines are capable of performing the same task, it is not necessarily the case that the amount of processing time required by each of these machines for this task is the same. That is, $t_{k,l} = t_{i,j} \wedge k \neq i \Rightarrow (P(t_{k,l}) = P(t_{i,j})) \vee (P(t_{k,l}) \neq P(t_{i,j}))$. Each machine further requires some amount of setup time $S(t_{i,j}, t_{i,k}) \geq 0$ prior to performing task $t_{i,j}$ if the previous task performed was $t_{i,k}$ for $k \neq j$. Setup time is not necessarily symmetric in that $S(t_{i,j}, t_{i,k})$ does not necessarily equal $S(t_{i,k}, t_{i,j})$.

Arriving at the shop are jobs $J_1, J_2, \ldots$. Each of these jobs has an arrival time that may or may not be known in advance. We are more interested in dynamic problems where these arrival times (along with all other information about the jobs) are not known ahead of time. That is, until the job arrives it does not exist and we are unaware of its future existence. A job J_i requires a set of tasks $T(J_i) = \{t_{i,1}, t_{i,2}, \ldots, t_{i,q}\}$ that can be performed by the various resources. For each $t_{i,j}$ there exists at least one machine capable of performing that task. That is, $\forall t_{i,j} \in T(J_i), \exists m_k, t_{k,l} \in T(m_k) \wedge t_{k,l} = t_{i,j}$. The set of required tasks for a job must be completed in a specified order[3]. The problem can be further complicated with additional constraints such as due-dates and priorities, but we do not consider these in this paper.

Each machine m_i in the shop has a queue $Q(m_i)$ from which it draws jobs to process. A more complete formulation would allow multiple machines to share a single queue, but this is not considered in this paper. Furthermore, we are only going to consider shops in which the queues must be drawn from in a first-in-first-out (FIFO) order. Both of these constraints are realistic. For example, in Morley and Schelberg's system for assigning trucks to paint booths (see [Morley and Schelberg, 1993]), each booth had its own queue and trucks were painted according to a FIFO order. In that instance, these constraints were necessitated by characteristics of the shop floor itself.

Based on this formulation, it should be apparent that we have a coordination problem. In order to process jobs in as little time as possible, it is necessary to assign (route) jobs to queues following other jobs requiring the same task in order to limit the amount of setup

[2]Perhaps it would be more accurate to use the term "resource" in place of "machine" in that a more complete definition of the problem would include other elements (e.g., people) in this set; but "machine" is sufficient for the purposes of this paper.

[3]Actually this set can be partially ordered, but for the purposes of this paper the distinction is unimportant because all jobs considered later have a task set of size 1.

time. But at the same time, if there are many jobs requiring a particular task and only very few requiring some other task, then it may be beneficial to accrue some setup time by sequencing a few of these jobs requiring the task of high demand in the same queue as a job requiring the task of lesser demand.

In the Sections that follow, we briefly overview the multi-agent systems for this problem under analysis. We then model the interactions of the agents of these systems for a few problem instances as coordination games. Each instance considered will be formulated as a game in normal form. These games, as you will see, consist of multiple NE and the problem is one of coordinating upon more desirable NE.

3 Job Agents as Players

The first two multi-agent systems that we analyze are based on the trail following behavior of ants. These systems assign ant-like agents to jobs. These ant-like agents make all routing decisions for their respective jobs by following quantities of artificial pheromone. Larger quantities of such pheromone correspond to machines which are processing jobs in less time (i.e., not spending a lot of time setting up); while smaller quantities of pheromone correspond to machines that are taking more time to process jobs (i.e., spending much time setting up). The ant-like agents of both of these systems prefer machines with larger quantities of pheromone. In other words, these agents attempt to optimize the expected throughput time for their respective jobs. To this end, when we present the game-theoretic model of these systems in Section 3.3 below, the payoffs will be defined as a loss function in terms of expected throughput time; and the players (the ant-like job agents) in this model wish to minimize their losses (i.e., minimize expected throughput time). Section 3.1 and Section 3.2 now describe the systems under analysis.

3.1 System 1: Ant-like Agents

System 1 is the unpublished predecessor of System 2 (see Section 3.2). It suffered from some limitations that were dealt with in System 2 and has been included in this paper because the analysis of the game-theoretic model of Section 3.3 sheds light on some of these limitations and offers a nice contrast to System 2. A detailed discussion of the background work upon which both System 1 and System 2 are based can be found in the original publication of System 2 [Cicirello and Smith, 2001a].

For each job type[4] in this system, there is a colony of ant-like agents. Each job, as it

[4]We use the term “job type” to refer to all jobs currently requiring the performance of the same task. This should not be confused with the “player type” of a Bayesian game.

arrives in the factory, is assigned to an "ant" from the appropriate colony according to the task it needs performed. Each machine on the shop floor maintains a quantity of artificial pheromone for each type of task it can handle[5]. Let $\pi_{a,b}$ be the quantity of pheromone for the colony in charge of task b located at machine a. An ant-like agent whose job requires task b will choose to enter the queue of machine a with probability:

$$P_{a,b} = \begin{cases} \frac{\pi_{a,b}}{\sum_j (\pi_{j,b})} & \text{if } a \text{ can perform task } b \\ 0 & \text{if } a \text{ cannot perform task } b \end{cases} \tag{1}$$

Upon the completion of a job, the ant-like agent in charge of that job updates the pheromone quantities according to:

$$\pi_{a,b} = \pi_{a,b} + R \quad \text{if } a \text{ performed task } b \tag{2}$$

In this equation, R is a constant. Also upon system startup, all pheromone values are initialized to π_0. At discrete time intervals, all pheromone values are decayed according to:

$$\pi_{a,b} = \alpha * \pi_{a,b} \tag{3}$$

The value of α is a constant parameter of the system between 0 and 1.

Initially, the ant-like agents of this system explore randomly; but as jobs are completed and the pheromone update rule applied, more ant-like agents will choose the machines with higher quantities of pheromone that correspond to machines for which the ant-like agents' perceived expected throughput time is minimized.

3.2 System 2: Ant Colony Control

System 2 is an extension of System 1 that we call Ant Colony Control (AC^2) [Cicirello and Smith, 2001a]. In System 1, the ant-like agents only considered the artificial pheromone of their own colony (i.e., associated with the task their job requires). In AC^2 however, the ant-like agents also consider the pheromone of other colonies. That is, in making their decision among alternative resources, they consider both their own perceived expected throughput for the machine for the task required by their job as well as the perceived expected throughput of the colonies in charge of other task types on that machine. So in other words, if an agent is faced with a decision between two machines with equal quantities of pheromone of their own type, but one of these has a much larger quantity of pheromone of some other task

[5]If put into practice, this artificial pheromone can be maintained by the computer controlling this machine and accessed over the network by the ant-like job agents.

type, then this agent will be more likely to choose the machine with the lesser amount of this other task type pheromone. The rationale behind this is that if some other type highly favors one of these machines, then choosing this highly favored machine might result in more setup time then in choosing the machine less regarded by this other task type. In order to define the new decision rule, first define:

$$\tau_{a,b} = \frac{\pi_{a,b}}{\sum_{k \neq b}(\pi_{a,k})} \tag{4}$$

In this expression, $\pi_{a,b}$ is as before. With this defined, an ant-like agent in charge of a job requiring task b will enter the queue of machine a with probability:

$$P_{a,b} = \begin{cases} \frac{\tau_{a,b}}{\sum_j(\tau_{j,b})} & \text{if } a \text{ can perform task } b \\ 0 & \text{if } a \text{ cannot perform task } b \end{cases} \tag{5}$$

The pheromone update rule employed by the ant-like agents of System 2 is also different than that of System 1. As an attempt to minimize setup time, the ants not only update the pheromone of their own task type, but also negatively reinforce the pheromone quantities of other task types which that machine can handle. It is a sort of indirect cross-colony communication. This negative reinforcement of the pheromone of other task types forewarns these other colonies that more jobs of a particular task type are likely to enter a particular queue and have an effect on their expected throughput time. In the analysis that follows in Section 3.3, it is also seen as a sort of agreement to play a particular NE. With this said, the new pheromone update rule is for the ant-like agents to update all pheromone quantities on the machine it utilized according to:

$$\pi_{a,k} = \begin{cases} \pi_{a,k} + R & \text{if } k = b \text{ (the task type of the ant)} \\ \beta * \pi_{a,k} & \text{if } k \neq b \text{ (other task types)} \end{cases} \tag{6}$$

In this expression β is a constant between 0 and 1. The value of R is as before.

3.3 Game-Theoretic Analysis of Ant Systems

To analyze the behavior of these systems, we will now look at two seemingly simple instantiations of the shop floor routing problem formalized in Section 2.

The first of these examples consists of a shop with two machines: M1 and M2. Each of these machines are identical. They can each perform two tasks: A and B. The amount of time required to process either task on either machine is 1 time unit. The amount of time required to setup either machine for task A if the previously performed task was B is two

	M1,M1	M1,M2	M2,M1	M2,M2
M1,M1	-5.5,-5.5,-5.5,-5.5	-4.3,-4.3,-4.3,-2.0	-4.3,-4.3,-2.0,-4.3	**-2.5,-2.5,-2.5,-2.5**
M1,M2	-4.3,-2.0,-4.3,-4.3	**-3.5,-3.5,-3.5,-3.5**	**-3.5,-3.5,-3.5,-3.5**	-2.0,-4.3,-4.3,-4.3
M2,M1	-2.0,-4.3,-4.3,-4.3	**-3.5,-3.5,-3.5,-3.5**	**-3.5,-3.5,-3.5,-3.5**	-4.3,-2.0,-4.3,-4.3
M2,M2	**-2.5,-2.5,-2.5,-2.5**	-4.3,-4.3,-2.0,-4.3	-4.3,-4.3,-4.3,-2.0	-5.5,-5.5,-5.5,-5.5

Figure 1: Four-player game. Players: Ant-like job agents. Row players are job type "A" and column players are job type "B". Strategies: M1 means enter queue of machine 1. M2 means enter queue of machine 2. Payoffs: expected throughput time (negative because agents want to minimize losses). The pure strategy NE are in bold.

time units (and similarly for setting up a machine to perform B if the previous task was A). The queues of both machines are initially empty. For the sake of analysis consider it equally likely that a machine is initially setup for task A or task B. There are four jobs in this example which arrive at the shop simultaneously. Jobs J1 and J2 require a single task of type A; and jobs J3 and J4 require a single task of type B (henceforth refer to the former jobs as being job type A and the latter as being job type B). These four jobs simultaneously and independently choose to enter either the queue of M1 or M2. Although they choose simultaneously, the order in which these moves are processed has an effect on the final outcome. Therefore, we consider that all 24 possible sequences are equally likely.

Figure 1 shows the normal form game corresponding to this instance. The row headings are the strategies of players J1 and J2. That is, "M1,M1" at the beginning of a row means that player J1 has chosen to enter the queue of M1 and that player J2 has also chosen to enter the queue of M1. The column headings similarly describe the strategies of players J3 and J4. Each entry in the matrix lists the payoffs for players J1, J2, J3, and J4 in that order. The payoffs correspond to the expected throughput time[6] for that job if all players play the specified strategies. These expected throughput times are calculated under the assumptions above that all possible sequences of moves are equally likely and that all initial machine setups are equally likely. All payoffs are negative because the players are minimizing losses. There are six pure strategy NE which have been listed in bold face. There is also a mixed strategy NE where all players play (0.5 M1, 0.5 M2) and the payoff to each player in this mixed strategy NE is -3.7375. There are two Pareto-optimal NE: (M1, M1, M2, M2) and (M2, M2, M1, M1). All other NE are deficient (the mixed strategy NE more so than the others). The game is one of coordination.

[6]Throughput time is used here to signify the amount of time from when the job enters the queue of a machine to when the machine is finished processing it.

To analyze how System 1 and System 2 react to such a problem, consider the more complex problem that consists of the same two machines but with hundreds of jobs rather than four. Approximately fifty percent of these jobs are of type A and fifty percent type B. The arrival times of these jobs are drawn uniformly at random from some long stretch of time and these arrival times are initially unknown. The system runs in discrete simulation under these constraints. This problem with hundreds of jobs can be viewed instead as several smaller problems with four jobs each chained together over time. It is not really an iterated version of this simpler four player game in that the players in the successive games are not the same. Also, the payoff matrices of these successive games might not be exactly the same in that it is possible that one or more of the machine queues may not be empty. However, if the machine queues are roughly the same in length then the successive payoff matrices will be equivalent only with a constant factor (the queue length) subtracted from all entries. It is reasonable to assume that this is the case. For one, if the players play a NE, then this is clearly the case since all the NE of this game will result in approximately equal queue lengths. And secondly, if in early iterations the players fail to coordinate upon any NE, then the payoffs of the following game matrix will be such that the machines with shorter queue lengths are more highly favored and the result will be that our equal queue length assumption will approximately hold for the game that succeeds this one. The players in the successive games, however, benefit from implicit knowledge of the outcomes of previous games encapsulated in the artificial pheromone quantities. In this sense, even though different players are playing, to some degree these players have the memory of the previous players and the game is much like an iterated version of the game of Figure 1. Another thing that should be noted is in regards to the assumption that all combinations of machine initial setups are equally likely. In terms of the iterated version of the game, this assumption does not always hold. On the first iteration, it is in fact the case. However, depending on the result of that game, the ant-like agents may have evidence as to the actual machine initial setups. For example, if the NE (M1, M1, M2, M2) is played, then on the next iteration of the game machine M1 will be set for type A and similarly M2 will be set for type B. Furthermore, the players will have evidence of this encapsulated in the pheromone quantities. However, although the payoffs of this successive iteration will be slightly different, the NE will remain the same[7]. Figure 2 shows this new payoff matrix and highlights the six pure strategy NE. In fact, all four possible machine initial setups if known ahead of time by the players lead to the same set of pure strategy NE although the payoffs are different under each assumption.

Viewing this more complex problem as an iterated version of the game of Figure 1, we

[7]This is likewise true if the NE (M2, M2, M1, M1) is played. If any of the other NE are played, the original equally likely assumption holds.

	M1,M1	M1,M2	M2,M1	M2,M2
M1,M1	-5.5,-5.5,-5.5,-5.5	-4.0,-4.0,-4.0,-1.0	-4.0,-4.0,-1.0,-4.0	**-1.5,-1.5,-1.5,-1.5**
M1,M2	-4.7,-3.0,-4.7,-4.7	**-3.5,-3.5,-3.5,-3.5**	**-3.5,-3.5,-3.5,-3.5**	-1.0,-4.0,-4.0,-4.0
M2,M1	-3.0,-4.7,-4.7,-4.7	**-3.5,-3.5,-3.5,-3.5**	**-3.5,-3.5,-3.5,-3.5**	-4.0,-1.0,-4.0,-4.0
M2,M2	**-3.5,-3.5,-3.5,-3.5**	-4.7,-4.7,-3.0,-4.7	-4.7,-4.7,-4.7,-3.0	-5.5,-5.5,-5.5,-5.5

Figure 2: Four-player game. Same format as before. Payoffs assume, however, that the initial machine setup is M1 set for A and M2 set for B. The NE have not changed.

can now analyze the behavior of the two systems. System 1 as defined in Section 3.1 will immediately begin playing the severely deficient mixed strategy NE and will continue to do so. The reason for this is that the initial quantities of pheromone at system startup are all equal so the initial exploration is a 50/50 random choice which is precisely the mixed strategy NE. If all jobs of one type are playing this mixed strategy NE, then the best the other jobs can do is to play it as well. One partial fix to this problem is to begin with initially random pheromone quantities rather than all equal quantities. However, when you do this, approximately fifty percent of starting configurations still converge upon playing the deficient mixed NE after some number of iterations of the game. To understand why this is true, consider that both job types are initially slightly more likely to choose strategy M1. After a few iterations, since there are more of both job types in the queue of machine M1, setup time becomes a big issue and the few jobs of both types that have gone to machine M2 discover that they are getting through quicker. Thus, the pheromone levels for both types begin tipping in the other direction until the system is again in the mixed strategy NE of all jobs play (0.5 M1, 0.5 M2). So System 1 appears to suffer from a severely deficient NE as a very strong focal point. Attempting to ensure that the initial conditions of the learning rules are such that this is not the case would be a pointless gesture in that it is easy to devise another example where initial conditions that are good for this problem are instead bad.

The story is different for System 2. In System 2, the ant-like agents' cross-colony communication can be likened to agreeing to play a particular NE. This agreement does not occur instantly; but when it finally does, it is a self-binding agreement. After some number of iterations, the agents begin playing one of the two NE that sends all jobs of one type to one machine and all jobs of the opposing type to the other machine. The experiments of [Cicirello and Smith, 2001a] show that this is the case. System 2 shifts the focal point to better NE (in this case to the Pareto-optimal NE). In fact, it is even better than it appears at first considering that once the agents have converged upon repeatedly playing one of these two Pareto-optimal NE, the actual payoffs of this NE are as in Figure 2 rather than Figure 1. This is due as stated earlier to the fact that our equally likely assumption regarding machine setups no longer holds.

Let us now turn our attention to a second example. In this next example, there is a shop with again two identical machines: M1 and M2. As before, they can each perform two tasks: A and B. The processing time of either task on either machine is the same as the previous example (i.e., 1 time unit). The setup time is also left unchanged from the previous example (i.e., 2 time units). The difference between this game and the previous lies in the players (as well as the payoffs). We now have five players instead of four. There are five jobs. Jobs J1, J2, J3, and J4 are all of type A. There is a single job of type B: J5. As before, the ant-like agents in charge of these five jobs simultaneously and independently choose to enter either the queue of M1 or M2. All 120 possible move sequences (as well as machine initial setups) are considered as equally likely in the computation of payoffs.

Figure 3 shows the normal form game for this instance. It consists of two payoff matrices. The first corresponds to when player J5 chooses as its strategy to enter the queue of machine M1. Similarly, the second matrix is the case when player J5 chooses M2 as its strategy. The strategies of players J1, J2, J3, and J4 are designated as before. That is, the strategies of J1 and J2 are the row headings and the strategies of J3 and J4 are the column headings. Each entry in the matrix lists the payoffs of the five players in the order: J1, J2, J3, J4, J5. The payoff for player J5 in each entry actually appears on the second line of that entry. Again, the payoff of a player in an entry of the matrix is the expected throughput time for that job if all players play the specified strategies. The players are again minimizing losses in this game. There are ten pure strategy NE that have been listed in boldface. There is also a mixed strategy NE in which all five players play: (0.5 M1, 0.5 M2). In this mixed strategy NE, the payoff to player J5 is -4.2125 and the payoff to players J1, J2, J3, and J4 is -3.6875 to each. This mixed strategy NE is clearly deficient as compared to the ten pure strategy NE. All of the pure strategy NE appear equally desirable in the sense that the sum of the payoffs of these NE are equivalent. A case can be made, however, perhaps to favor the two NE (M1, M1, M1, M1, M2) and (M2, M2, M2, M2, M1). The reason is that for each of these two NE, the expected completion time of all jobs on the machines is 5 time units for the machine with the four type A jobs and 2 time units for the machine with the one type B job; while the other eight NE have expected completion times of all jobs per machine of 5 time units and 4 time units. In the sense that we only care about expected completion time of all jobs then all ten NE are still equally desirable since that expected time is 5 time units in all ten cases. However, the two that send all type A jobs to one machine and the single type B job to the other machine have a bit more slack since one of the machines will be finished with its tasks much sooner than will the other machine in these instances. This might be beneficial if another player (job) is thrown into the game (shop floor) unexpectedly sometime after these five players have made their moves, especially if this additional job is of type B.

	Player 5’s strategy: M1			
	M1,M1	M1,M2	M2,M1	M2,M2
M1,M1	-5.6,-5.6,-5.6,-5.6, -5.6	-5.0,-5.0,-5.0,-2.0, -5.0	-5.0,-5.0,-2.0,-5.0, -5.0	-4.3,-4.3,-2.5,-2.5, -4.3
M1,M2	-5.0,-2.0,-5.0,-5.0, -5.0	-4.3,-2.5,-4.3,-2.5, -4.3	-4.3,-2.5,-2.5,-4.3, -4.3	**-3.5,-3.0,-3.0,-3.0, -3.5**
M2,M1	-2.0,-5.0,-5.0,-5.0, -5.0	-2.5,-4.3,-4.3,-2.5, -4.3	-2.5,-4.3,-2.5,-4.3, -4.3	**-3.0,-3.5,-3.0,-3.0, -3.5**
M2,M2	-2.5,-2.5,-4.3,-4.3, -4.3	**-3.0,-3.0,-3.5,-3.0, -3.5**	**-3.0,-3.0,-3.0,-3.5, -3.5**	**-3.5,-3.5,-3.5,-3.5, -2.0**

	Player 5’s strategy: M2			
	M1,M1	M1,M2	M2,M1	M2,M2
M1,M1	**-3.5,-3.5,-3.5,-3.5, -2.0**	**-3.0,-3.0,-3.0,-3.5, -3.5**	**-3.0,-3.0,-3.5,-3.0, -3.5**	-2.5,-2.5,-4.3,-4.3, -4.3
M1,M2	**-3.0,-3.5,-3.0,-3.0, -3.5**	-2.5,-4.3,-2.5,-4.3, -4.3	-2.5,-4.3,-4.3,-2.5, -4.3	-2.0,-5.0,-5.0,-5.0, -5.0
M2,M1	**-3.5,-3.0,-3.0,-3.0, -3.5**	-4.3,-2.5,-2.5,-4.3, -4.3	-4.3,-2.5,-4.3,-2.5, -4.3	-5.0,-2.0,-5.0,-5.0, -5.0
M2,M2	-4.3,-4.3,-2.5,-2.5, -4.3	-5.0,-5.0,-2.0,-5.0, -5.0	-5.0,-5.0,-5.0,-2.0, -5.0	-5.6,-5.6,-5.6,-5.6, -5.6

Figure 3: Five-player game. Players: Ant-like job agents. Row and column players are job type “A”. Player 5 is job type “B”. Strategies: M1 means enter queue of machine 1. M2 means enter queue of machine 2. Payoffs: expected throughput time (negative because agents want to minimize losses). The pure strategy NE are in bold.

Using the same argument as before, we can view the more complex problem consisting of hundreds of jobs in a ratio of four type A jobs to every one type B job as being equivalent to an iterated version of the game of Figure 3. That is, although the players in the successive iterations are different, they sort of have the memory of the players of the preceding iterations through the adaptation of pheromone quantities and the overall result is very much like five players playing a game iteratively. Also, the payoff matrices of the successive games are equivalent if the queue lengths of both machines are approximately the same and if the assumption of equally likely machine initial setups continues to hold. The former is true if any of the NE other than (M1, M1, M1, M1, M2) and (M2, M2, M2, M2, M1) are played, but the latter assumption no longer holds. Further in the case of these

	Player 5's strategy: M1			
	M1,M1	M1,M2	M2,M1	M2,M2
M1,M1	-5.0,-5.0,-5.0,-5.0, -5.0	-4.5,-4.5,-4.5,-3.0, -4.5	-4.5,-4.5,-3.0,-4.5, -4.5	**-4.0,-4.0,-3.5,-3.5, -4.0**
M1,M2	-4.5,-3.0,-4.5,-4.5, -4.5	**-4.0,-3.5,-4.0,-3.5, -4.0**	**-4.0,-3.5,-3.5,-4.0, -4.0**	**-3.5,-4.0,-4.0,-4.0, -3.5**
M2,M1	-3.0,-4.5,-4.5,-4.5, -4.5	**-3.5,-4.0,-4.0,-3.5, -4.0**	**-3.5,-4.0,-3.5,-4.0, -4.0**	**-4.0,-3.5,-4.0,-4.0, -3.5**
M2,M2	**-3.5,-3.5,-4.0,-4.0, -4.0**	**-4.0,-4.0,-3.5,-4.0, -3.5**	**-4.0,-4.0,-4.0,-3.5, -3.5**	-4.5,-4.5,-4.5,-4.5, -3.0

	Player 5's strategy: M2			
	M1,M1	M1,M2	M2,M1	M2,M2
M1,M1	**-2.5,-2.5,-2.5,-2.5, -1.0**	-2.0,-2.0,-2.0,-3.5, -3.5	-2.0,-2.0,-3.5,-2.0, -3.5	-1.5,-1.5,-4.7,-4.7, -4.7
M1,M2	-2.0,-3.5,-2.0,-2.0, -3.5	-1.5,-4.7,-1.5,-4.7, -4.7	-1.5,-4.7,-4.7,-1.5, -4.7	-1.0,-5.5,-5.5,-5.5, -5.5
M2,M1	-3.5,-2.0,-2.0,-2.0, -3.5	-4.7,-1.5,-1.5,-4.7, -4.7	-4.7,-1.5,-4.7,-1.5, -4.7	-5.5,-1.0,-5.5,-5.5, -5.5
M2,M2	-4.7,-4.7,-1.5,-1.5, -4.7	-5.5,-5.5,-1.0,-5.5, -5.5	-5.5,-5.5,-5.5,-1.0, -5.5	-6.2,-6.2,-6.2,-6.2, -6.2

Figure 4: Five player game. Same format as before. Payoffs assume, however, that the initial machine setup is M1 set for A and M2 set for B.

two NE, neither assumption holds. Let us first look at what happens with one of these two NE. If the players play (M1, M1, M1, M1, M2), then on the following iteration machine M1 will be set for job type A and M2 for type B. These setups will be evident in the pheromone quantities so it should be safe to assume that the players are playing this next game under this assumption. The normal form of this next game is seen in Figure 4. Now although the NE are not the same as the previous game, the one that had been played (M1, M1, M1, M1, M2) is still a NE (but with better payoffs than previously) and will be played again. This same argument holds similarly if (M2, M2, M2, M2, M1) had been played. Furthermore, if any of the pure strategy NE of the original game of Figure 3 in which player five plays M2 and one of players one through four also plays M2 are played, then the payoff matrix of the next iteration will be as in Figure 5[8]. The payoffs in this matrix assume that M1 is

[8]A similar game can be defined if any of the NE where player five plays M1 and one of players one

set for A and that M2 is equally likely set for either A or B. The reason this would be the case is that M1 in the previous iteration processed nothing but type A jobs and so would be set for A. Further, M2 processed one job of each type and would be equally likely set for A or B. Now if all five players assume that every player will play the same strategy as in the previous round, the only player with incentive to do otherwise is the type A job that had played M2. This player instead would now play M1. This leads directly to the NE of (M1, M1, M1, M1, M2). And the succeeding iterations will be as in Figure 4. Through this line of reasoning, the iterated game will converge to either the NE (M1, M1, M1, M1, M2) or to the NE (M2, M2, M2, M2, M1). These are the only stable pure strategy NE in the iterated game based on the line of reasoning put forth here. If the mixed strategy NE is played then the original assumptions of equal queue lengths and equally likely machine initial setups both hold and every round of this iterated game will be as in Figure 3.

In this more complex problem, System 1 again converges upon the severely deficient mixed strategy NE. The reasons are the same as in the four player game. This behavior emerges all of the time if the pheromone values are initialized to a constant and again roughly fifty percent of the time if initialized randomly. System 1 suffers from a severely deficient focal point NE in both the previous four player game as well as the five player game of Figure 3.

In System 2, however, the agents do converge upon the repeated play of either (M1, M1, M1, M1, M2) or (M2, M2, M2, M2, M1). This can be seen in the empirical system results (see [Cicirello and Smith, 2001a]). The argument regarding the stability of the other pure strategy NE seems to hold. That is, if any of the others are played, in the next round when the payoffs change, the job that had veered from the pattern of all type A jobs to a single machine now has incentive to play differently while all other jobs happily continue playing their part in this NE.

4 Machine Agents as Players

The next two multi-agent systems that we analyze are based on the adaptive task allocation behavior of wasps. These two systems associate a wasp-like agent with each machine in the factory. These wasp-like agents use a stimulus-response mechanism to choose jobs to add to the queues of their respective machines. That is, each wasp-like agent maintains response thresholds for the various tasks that its associated machine is able to process. A lower value of a response threshold corresponds to a greater willingness to take on jobs of a particular type; while a higher value of a response threshold corresponds to a lesser desire

through four play M1 while the others play M2.

	Player 5s strategy: M1			
	M1,M1	M1,M2	M2,M1	M2,M2
M1,M1	-5.0,-5.0,-5.0,-5.0,-5.0	-4.5,-4.5,-4.5,-2.0,-4.5	-4.5,-4.5,-2.0,-4.5,-4.5	-4.0,-4.0,-2.5,-2.5,-4.0
M1,M2	-4.5,-2.0,-4.5,-4.5,-4.5	-4.0,-2.5,-4.0,-2.5,-4.0	-4.0,-2.5,-2.5,-4.0,-4.0	**-3.5,-3.0,-3.0,-3.0,-3.5**
M2,M1	-2.0,-4.5,-4.5,-4.5,-4.5	-2.5,-4.0,-4.0,-2.5,-4.0	-2.5,-4.0,-2.5,-4.0,-4.0	**-3.0,-3.5,-3.0,-3.0,-3.5**
M2,M2	-2.5,-2.5,-4.0,-4.0,-4.0	**-3.0,-3.0,-3.5,-3.0,-3.5**	**-3.0,-3.0,-3.0,-3.5,-3.5**	**-3.5,-3.5,-3.5,-3.5,-3.0**

	Player 5s strategy: M2			
	M1,M1	M1,M2	M2,M1	M2,M2
M1,M1	**-2.5,-2.5,-2.5,-2.5,-2.0**	-2.0,-2.0,-2.0,-3.5,-3.5	-2.0,-2.0,-3.5,-2.0,-3.5	-1.5,-1.5,-4.3,-4.3,-4.3
M1,M2	-2.0,-3.5,-2.0,-2.0,-3.5	-1.5,-4.3,-1.5,-4.3,-4.3	-1.5,-4.3,-4.3,-1.5,-4.3	-1.0,-5.0,-5.0,-5.0,-5.0
M2,M1	-3.5,-2.0,-2.0,-2.0,-3.5	-4.3,-1.5,-1.5,-4.3,-4.3	-4.3,-1.5,-4.3,-1.5,-4.3	-5.0,-1.0,-5.0,-5.0,-5.0
M2,M2	-4.3,-4.3,-1.5,-1.5,-4.3	-5.0,-5.0,-1.0,-5.0,-5.0	-5.0,-5.0,-5.0,-1.0,-5.0	-5.6,-5.6,-5.6,-5.6,-5.6

Figure 5: Five player game. Same format as before. Payoffs assume, however, that the initial machine setup is M1 set for A and M2 is equally likely set for either A or B.

in jobs of some type. The wasp-like agents adapt these response thresholds according to the state of their respective machines. For example, a wasp in charge of a machine currently engaged in a task of type A will decrease the response threshold associated with that task increasing its willingness to take on more jobs of that type. At the same time, this wasp will increase the response thresholds for other job types. These two rules together imply that the wasp-like agents of this system prefer to specialize if possible. Also, if the machine which a wasp is in charge of is idle and has an empty queue, then this wasp will decrease the response thresholds of all task types. In other words, these wasp-like agents dislike being idle and will increase their willingness to accept any job type if they find themselves idle. These response thresholds are further bounded in some finite interval. Several sentences ago, we referred to the mechanism as a “stimulus-response mechanism” but we have yet to mention any stimulus. As a job enters the shop, it begins emitting a stimulus in the form of the length of time it has been in the shop but not yet assigned to a queue. If a wasp-like machine agent responds to this stimulus, then the job is added to its queue. If more then one wasp-like agent responds, then the winner is chosen by a system dependent rule (see Section 4.1 and Section 4.2). If no agents respond then during the next time interval the job will emit a greater stimulus. The payoffs (as well as the set of strategies of the players) of the game formulation of Section 4.3 are better explained in that Section. The payoffs will consider the wasp-like agents' preference for specialization when possible and dislike of idleness. The payoffs will also consider expected completion time of all jobs.

4.1 System 3: Wasp-like Agents

System 3 was first presented in [Cicirello and Smith, 2001d]. It is based on a computational model of real wasp behavior. Theraulaz et al. originally suggested the potential application of this model to the organization of multi-robot teams [Theraulaz *et al.*, 1991]. A detailed overview of Theraulaz et al.'s model can be found in the original publication of System 3 [Cicirello and Smith, 2001d] as well as in the paper describing System 4 [Cicirello and Smith, 2001b].

For each machine in this system, there is a wasp-like agent referred to as a *routing wasp*. This routing wasp maintains a set of response thresholds for the types of tasks its machine can perform. Let this response threshold set be defined as:

$$\Theta_w = \{\theta_{w,0}, \ldots, \theta_{w,J}\} \tag{7}$$

where $\theta_{w,j}$ is the response threshold of wasp w to jobs of type j. Each wasp only has response thresholds for job types its associated machine can process.

At discrete time intervals, jobs in the system that are not currently queued on a machine broadcast to all of the routing wasps a stimulus S_j which is equal to the length of time the job has been waiting to be assigned to a machine and where j is the type of job. So the longer the job remains unassigned, the stronger the stimulus it emits. Provided that its associated machine is able to process job type j, a routing wasp w will pick up a job emitting a stimulus S_j with probability:

$$P(\theta_{w,j}, S_j) = \frac{S_j^2}{S_j^2 + \theta_{w,j}^2} \tag{8}$$

In this way, routing wasps will tend to pick up jobs of the type for which its response threshold is lower. But will pick up jobs of other types if a high enough stimulus is emitted. If more than one routing wasp respond to a stimulus from a single job during the same time unit, the winner is chosen randomly.

The threshold values $\theta_{w,j}$ may vary in the range $[\theta_{min}, \theta_{max}]$. At discrete time intervals, the routing wasps update these response thresholds according to the current state of its machine. If the machine is currently processing job type j or is in the process of setting up to process job type j, then $\theta_{w,j}$ is updated according to:

$$\theta_{w,j} = \theta_{w,j} - \delta_1 \tag{9}$$

If the machine is either processing or setting up to process a job type other than j, then $\theta_{w,j}$ is updated according to:

$$\theta_{w,j} = \theta_{w,j} + \delta_2 \tag{10}$$

These two rules together imply that the wasp-like agents attempt to specialize in a particular job type. That is, by decreasing the response threshold of the type of task which it is currently engaged in, it increases the probability of taking on more tasks of that type; while by increasing the response threshold of other types, it decreases the probability of taking on tasks of other types. This specialization can be beneficial to limiting required setup time. A third way that the response thresholds are updated is the idle machine update rule. That is, if the machine is currently idle and has an empty queue, then for all job types j that the machine can process the wasp adjusts the response thresholds $\theta_{w,j}$ according to:

$$\theta_{w,j} = \theta_{w,j} - \delta_3^t \tag{11}$$

The exponent t is the length of time the machine has been sitting idle so successive idle time intervals result in greater and greater idle machine updates. To summarize these three rules together, the routing wasps prefer to specialize to a single job type (or to as few job types as possible), but at the same time they really dislike being idle. So if specializing will result in a large amount of idleness, then the machines will be willing to accept some jobs of other types.

4.2 System 4: Extension of System 3

System 4 is an extension of System 3 that is presented in [Cicirello and Smith, 2001b]. There is one major difference between these two systems. In System 3, as stated in the previous Section, when two or more routing wasps respond to the stimulus of a single job, the winning routing wasp is chosen at random. All competing routing wasps in System 3 have an equal chance of getting this job. System 4, however, incorporates a computational model of self-organized dominance hierarchies to decide the winner from among a group of competing routing wasps.

First define the force F_w of a routing wasp w as:

$$F_w = 1.0 + T_p + T_s \tag{12}$$

where T_p and T_s are the sum of the processing times and setup times of all jobs currently in the queue of the associated machine, respectively. Now consider a dominance struggle between two competing routing wasps. This contest determines which routing wasp gets the job. Let F_1 and F_2 be the force variables of routing wasps 1 and 2, respectively. Routing wasp 1 will get the job with probability:

$$P(F_1, F_2) = \frac{F_2^2}{F_1^2 + F_2^2} \tag{13}$$

In this way, routing wasps associated with machines of equivalent queue lengths will have equal probabilities of getting the job. If the queue lengths differ, then the routing wasp with the smaller queue has a better chance of taking on the new job. In the event that more than two routing wasps compete for a given job, a single elimination tournament of dominance contests is used to decide the winner.

Aside from this new tie-breaking rule, all other features of System 3 are carried over to System 4. In the game-theoretic model of the next Section, the difference between these two systems will be seen in the computation of the payoffs. So in a sense the agents of System 3 will be playing a different game than will be the agents of System 4.

4.3 Game-Theoretic Analysis of Wasp Systems

The game-theoretic analysis that appears in this Section considers again the two examples that were considered in the analysis of the ant-based systems. The players in this analysis are now the wasp-like agents of System 3 and System 4. Given the description of these two systems that appeared in Section 4.1 and Section 4.2, the design of the set of strategies for the players in this analysis is non-obvious. Recall that the mechanism employed by the wasp-like agents to assign jobs to machines is one based on the stimulus-response mechanism used by real wasps to allocate tasks among the members of the colony. Each routing wasp in each of these systems maintains response thresholds for the various tasks that its associated machine can perform. These response thresholds roughly correspond to a degree of interest in a job type – or perhaps better phrased as a degree of specialization in a job type. The actual number of strategies of the wasp-like agents of these systems is essentially infinite. That is, each of a routing wasp's response thresholds may take on any value in the allowed continuous range. A set of strategies of this magnitude would be unwieldy to deal with in the analysis of this paper. So instead, consider the following set of strategies:

- A: A player choosing this strategy always responds to jobs of type A and never responds to jobs of any other type. In terms of the response threshold mechanism, this would correspond to a response threshold for type A jobs of 0 and response thresholds for all other job types of ∞[9].

- B: A player choosing this strategy always responds to jobs of type B and never responds to jobs of any other type. In terms of response thresholds, this corresponds to a type B threshold of 0 and all other response thresholds of ∞.

- AB: A player choosing this strategy responds to all jobs of both type A and type B. Both of these response thresholds in the wasp model would be 0.

- NONE: A player choosing this strategy never responds to any job and chooses to remain idle indefinitely. This corresponds to all response thresholds having values of ∞.

This may seem rather restrictive – modeling an essentially infinite number of possible strategies with only four – but the analysis as you will see is still enlightening and the system behavior will appear to correspond to some degree to this simplified model.

Let us now turn our attention to the determination of payoffs for this game-theoretic model. The payoff computation for this model should consider the following preferences of the wasp-like agents:

- Idleness: The wasp-like agents of these two systems do not like being idle. This derives from the idle-machine response threshold update rule. If a wasp-like agent finds that its machine is idle and has an empty queue, it reduces its response thresholds to all task types it is capable of handling. By doing this, it also increases the probability of taking on any jobs requiring one of the tasks it can perform. Therefore, if a strategy results in idleness, the player is penalized in the payoff computation. Another more positive way of phrasing this preference would be a like of keeping busy rather than a dislike of idleness.

- Unprocessed jobs: The wasp-like agents of these systems further do not like when jobs go without being processed. This is a side-effect of the continuously increasing stimulus emitted by the jobs until a wasp assigns it to a machine queue. Eventually, given finite response thresholds and enough time, a job will emit a stimulus great enough to be picked up by some machine. Once this occurs, its threshold for that task type will decrease. With this in mind, if there are only two machines and they

[9]If you recall, the response thresholds were bounded in some finite range, but for ease of analysis this has here been relaxed. Also, the response thresholds are typically bounded to be strictly positive. This has also been relaxed for ease of analysis.

	A	B	AB	NONE
A	-100,-100	**-2,-2**	-6.2,-6.2	-100,-200
B	**-2,-2**	-100,-100	-6.2,-6.2	-100,-200
AB	-6.2,-6.2	-6.2,-6.2	-8.7,-8.7	-18,-18
NONE	-200,-100	-200,-100	-18,-18	-200,-200

Figure 6: System 3. Two-player game with four jobs (2 type A, 2 type B). All arrival sequences are equally likely. Players: Wasp-like machine agents. Strategies: A, B, AB, NONE. The NE are in bold.

play (A, A)[10] and if there exists at least one type B job, then eventually this type B job will force at least one of the machines to instead play either B or AB. To account for this dislike of unproductive behavior, the payoffs assign large negative penalties to all players if at least one job is not processed.

- Speciality: The wasp-like agents of these systems like to specialize to as few job types as possible (without violating the dislike of idleness principal). That is, through the response threshold update rules, the wasp-like agents decrease the thresholds for job types that their associated machines are working on and increase them for other types. This increases the probability of accepting more jobs of a single type. In light of this, if no machines have specialized (i.e., they all play AB), then they receive a penalty in the payoffs.

Both examples that are analyzed have two machines (just as before). These two machines, M1 and M2, are now the players. To describe the payoff computation, we present to you the normal form game for the example consisting of four jobs (two type A and two type B) for the wasp-like agents of System 3 (see Section 4.1) and explain each entry of the matrix one at a time. This game appears in Figure 6. It may seem somewhat tedious to walk through the payoff computation of all entries, but it is a necessary explanation since the individual objective of the agents of this system is not as clear as it was in the two ant-based systems. Also, it is necessary to relax the assumption used in the ant system analysis in regards to equally likely machine initial setups because there is implicit evidence with regards to initial setup in the choice of player strategies which will be explained.

In Figure 6 we see the normal form version of the game played between the two routing wasps of System 3 in the example consisting of two jobs of type A and two jobs of type B. In computing the payoffs, all possible arrival sequences of these four jobs are considered

[10]Similarly for the playing of (B, B).

equally likely. Initial machine setup is considered in a manner specific to the entry in the figure as will be explained. First note entries (A, A) and (B, B). In each of these strategy combinations, jobs will be left unprocessed since there is one type of job in each of these cases that are never accepted into the queue of either machine. The wasp-like agents' dislike of unprocessed jobs, as discussed earlier, results in a large penalty of -100 for both players in each of these cases[11]. Now look at entries (NONE, A), (NONE, B), (A, NONE), and (B, NONE). In each of these cases, both players receive a penalty of -100 for the unprocessed jobs and the player that played the strategy NONE receives an additional penalty of -100 (for a total of -200) for being idle while some jobs were not processed[12]. The strategy combination (NONE, NONE) results in a payoff of -200 to each of the players for this same latter reason (i.e., -100 for unprocessed jobs and -100 for being idle while jobs were left not processed). Now with regard to the remaining entries, it is first necessary to note a few assumptions made in computing these payoffs. First, if a player's strategy is A it is assumed that the initial machine setup for that machine is for task type A (and similarly for strategy B). The reason for this assumption is that if a player is playing A (similarly B) then the queue of that machine must only be accepting jobs of type A and if we assume that the game is being played iteratively (along with that strategy) then it must be set up for A at the start of each round. In computing the payoffs for entries involving the strategy AB, the assumption made regarding machine initial setup depends on the opposing player's strategy. For example, if the players play (AB, AB), it is assumed that either initial setup is equally likely. For (AB, A), the initial machine setup for the player playing AB is assumed twice as likely B than A[13]. Now, all other entries in the payoff matrix other than (AB, AB), (AB, NONE), and (NONE, AB) give to each player a loss equal to the expected length of time until all jobs have been processed. For example, the payoff of the strategy combination (B, A) is (-2, -2) because each machine expects to complete all jobs assigned to it by these strategies in 2 time units. The strategy combination (AB, A) results in a payoff of (-6.2, -6.2) because M1 (the row player) expects to be finished in 6.2 time

[11]This penalty should probably be defined in terms of θ_{max} in that the job in question will go unassigned to a queue until it gives off a stimulus sufficiently above this value in this instance. But if you recall, to simplify analysis, the strategy A is equivalent to a $\theta_B = \theta_{max} = \infty$ (and likewise for strategy B). So to deal with this we would have to allow an infinitely negative payoff if this penalty was defined in terms of θ_{max}.

[12]This "idle while some jobs not processed" penalty should also probably be defined in terms of θ_{max} and δ_3. But similarly, it would result in infinitely negative payoffs due to the simplification that allows infinite response thresholds.

[13]If (AB, A) is played in System 3, then player M1 will get both type B jobs and players M1 and M2 will on average each get 1 type A job. So on average, player M1 processes twice as many type B jobs as A. With the equally likely arrival sequence assumption, the final setup of an iteration is twice as likely to be B and thus the initial setup of the next iteration if the players continue to play this strategy combination iteratively is also twice as likely to be B.

	A	B	AB	NONE
A	-100,-100	**-2,-2**	-2.4,-2.4	-100,-200
B	**-2,-2**	-100,-100	-2.4,-2.4	-100,-200
AB	-2.4,-2.4	-2.4,-2.4	-6,-6	-18,-18
NONE	-200,-100	-200,-100	-18,-18	-200,-200

Figure 7: System 4. Two-player game with four jobs (2 type A, 2 type B). All arrival sequences are equally likely. Players: Wasp-like machine agents. Strategies: A, B, AB, NONE. The NE are in bold.

units. Although, player M2 expects to be finished in 1 time unit in this instance, it also suffers a loss of -6.2 in the payoffs. This is due to the wasp-like agents' preference for keeping busy (or dislike of idleness); therefore being idle while another machine works on jobs that it also is capable of processing is treated as a loss in the payoffs. The other entries with payoffs of (-6.2, -6.2) are computed likewise. The entries in the matrix (AB, AB), (AB, NONE), and (NONE, AB) are twice the expected completion times of all jobs. This penalty of doubling the expected completion times is due to the wasp-like agents' preference for specialization when possible[14]. In all three of these cases, neither wasp has specialized when it is clearly possible in these instances.

In Figure 7 we similarly see the payoff matrix for the game that models System 4 with four jobs (two of each type). You will notice that it looks very much like that of the System 3 game. The only difference is in the payoffs involving strategy AB. If you recall, in System 3 (see Section 4.1) when two or more routing wasps compete for the same job, the winner is chosen randomly. In the System 3 game matrix, this rule in combination with the assumption of equally likely job arrival sequences, was assumed in computing the payoffs. System 4 uses a more complex tie breaking rule as discussed in Section 4.2. A simplification of this rule[15] was used in computing the payoffs in Figure 7.

In both the case of the game of Figure 6 and the game of Figure 7, there are two pure strategy NE: (A, B) and (B, A). Each of these have payoffs of (-2, -2). To relate these NE to the game formulation of the ant-based systems, (A, B) is equivalent in global performance to the NE (M1, M1, M2, M2) of the game of Figure 1. Similarly, (B, A) is equivalent to

[14]This penalty should probably be defined in terms of δ_1 and δ_2 in some manner since these system parameters control in some way the adaptation of specialization; but it is not clear how they should be used in this penalty definition.

[15]If you recall, the machine with the shorter queue had a higher probability of getting the job in such a multi-wasp competition. The simplification made for payoff computation assumes that the machine with the shorter queue gets the job with probability 1.

the NE (M2, M2, M1, M1) of the game of Figure 1[16]. In any case, both System 3 and System 4 in the more complex problem with hundreds of jobs of equal proportions of each type converge upon behavior equivalent to playing one of these NE in the iterated version of this game (see [Cicirello and Smith, 2001d; 2001b] for the empirical evaluation of these systems). To understand why this is so, consider the following cases:

- If one of the NE are played at some point in the repeated game and if both players assume the other will continue to do so, then there is no incentive for the other to deviate from the NE. To justify the assumption that the other will continue to play its part in this NE in the repeated game, consider the NE (A, B). In this NE, the response threshold updating rules will keep player M1 interested in job type A and uninterested in job type B. The opposite will be true for player M2. Thus, they will continue to play (A, B).

- If either (A, A) or (B, B) are played, then the increasing job stimulus of the unprocessed jobs will eventually result in one or both of the players accepting those jobs. Consider the case where (A, A) is played. After some number of iterations, consider a type B job giving off a stimulus great enough to be picked up by machine M1. During the processing of this job, the response threshold updates will decrease its threshold to type B jobs (and increase the threshold to type A jobs). Depending upon system parameters this will either result in M1 playing B or AB. If B, then the players begin playing the NE (B, A) and all is well. The case of (AB, A) is considered next.

- If (AB, A) or (A, AB) are played by the System 3 wasps, then the player playing AB will end up with 2 type B jobs and 1 type A job (on average). Through this extra load of type B jobs, the response threshold updates will eventually increase the type A threshold and decrease the type B threshold to the point of specialization in type B. This leads to one of the NE (B, A) or (A, B) depending on which of (AB, A) or (A, AB) were played to begin with. This is similarly true for (AB, B) and (B, AB). If (AB, A) or (A, AB) is played by the System 4 wasps, then the player playing AB will on average get even less than 1 type A job to every 2 type B jobs so this former argument is even stronger towards convergence upon the NE.

- If (AB, AB) is played, a similar argument as in the preceding case can be made. That is, the effects of the response threshold updates strongly lead to specialization.

[16]Actually, (A, B) is more accurately equivalent to the NE (M1, M1, M2, M2) of the game of Figure 2 and (B, A) is more accurately equivalent to the NE (M2, M2, M1, M1) of the game generated in the same manner as that of Figure 2 but with the assumption of M1 set for B and M2 set for A.

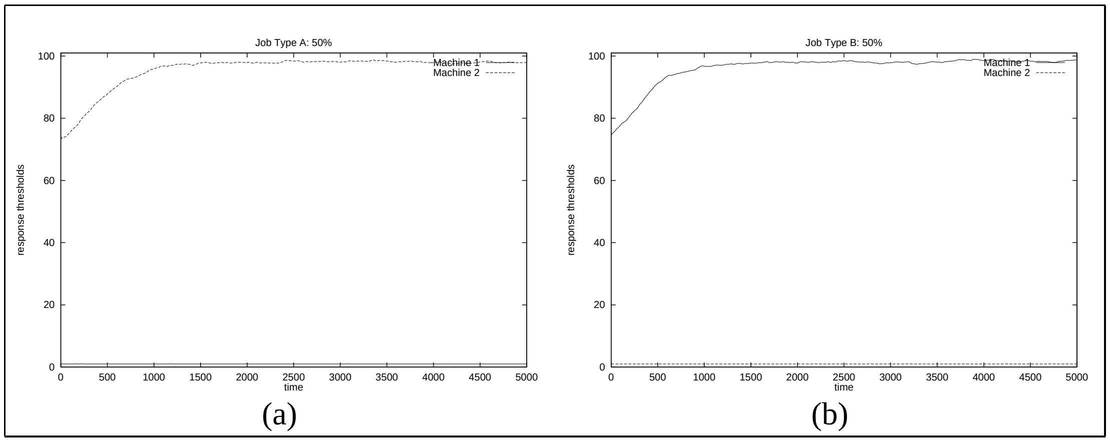

Figure 8: Plots of the average response thresholds over time of the routing wasps of System 4. These plots are averages over 100 simulation runs for a problem consisting of jobs of two types arriving in equal proportions

- If any wasp plays NONE, the idle machine response threshold update rule will force the machine to reduce its response thresholds and begin taking jobs of one or both types. The strategy NONE, as can be seen in the game matrices, is strictly dominated and never played by the routing wasps of these systems.

In Figure 8 we see plots of the average response thresholds of the wasp-like agents of System 4 for the problem consisting of jobs of two types arriving in approximately equal proportions and two machines. The simulations ran for 5000 time units and the results shown are averages of 100 runs. The system parameters are set as they were in [Cicirello and Smith, 2001b]. In Figure 8a we see the response thresholds over time of the two machines to job type A (likewise to job type B in Figure 8b). As can be seen in these plots, one machine has specialized to job type A (i.e., its response threshold for type A is low while for B it is high) and the other has specialized to job type B. Similar plots for System 3 would show likewise. This provides an empirical illustration of the wasp-like agents playing the NE (A, B) in a repeated game situation.

The five job problem (with a ratio of four type A jobs to one type B job) provides a somewhat more interesting analysis. The normal form game with System 3 wasp-like agents as players is as in Figure 9. The payoffs were computed in the same manner as they were for the four job problem. There are again two NE: (B, A) and (A, B). Both of these NE are Pareto-optimal and equally desirable. The NE (A, B) corresponds in terms of global performance to that of the NE (M1, M1, M1, M1, M2) of the game of Figure 5 in

	A	B	AB	NONE
A	-100,-100	**-4,-4**	-6.375,-6.375	-100,-200
B	**-4,-4**	-100,-100	-6.48,-6.48	-100,-200
AB	-6.375,-6.375	-6.48,-6.48	-8.69,-8.69	-18.4,-18.4
NONE	-200,-100	-200,-100	-18.4,-18.4	-200,-200

Figure 9: System 3. Two-player game with five jobs (4 type A, 1 type B). All arrival sequences are equally likely. Players: Wasp-like machine agents. Strategies: A, B, AB, NONE. The NE are in bold.

	A	B	AB	NONE
A	-100,-100	-4,-4	**-3.8,-3.8**	-100,-200
B	-4,-4	-100,-100	-4.05,-4.05	-100,-200
AB	**-3.8,-3.8**	-4.05,-4.05	-7.096,-7.096	-18.4,-18.4
NONE	-200,-100	-200,-100	-18.4,-18.4	-200,-200

Figure 10: System 4. Two-player game with five jobs (4 type A, 1 type B). All arrival sequences are equally likely. Players: Wasp-like machine agents. Strategies: A, B, AB, NONE. The NE are in bold.

the analysis of the ant-based systems[17]. The problem is one of coordinating upon one of these NE. And, empirically, System 3 does just that in the problem with hundreds of jobs in a ratio of four type A to every one type B (see [Cicirello and Smith, 2001d]). That is, when viewed as a repeated game between these two wasp-like agents, the system eventually converges upon the repeated play of one or the other of these two NE. This convergence can be explained in the exact same manner as in the four job case above.

The more interesting aspect of this five job problem is seen when you examine the normal form game that represents the payoffs of the wasp-like agents of System 4. This game is seen in Figure 10. The payoffs were computed as previously. The interesting thing is that for System 4 we have different NE. The pure strategy NE for this game are: (AB, A) and (A, AB). These are superior in terms of global system performance for the problem over that of the System 3 NE. Let's examine what happens in these strategies for both System 3 and System 4. First in System 3 (see game in Figure 9), if one wasp specializes to type A jobs (plays strategy A) and the other accepts either job type (plays strategy AB),

[17]The NE (B, A) would similarly correspond to the NE (M2, M2, M2, M2, M1) of a normal form game generated in a manner analogous to that of Figure 5 but with the opposite machine initial setup assumption.

then the player playing AB will always get the one type B job and will also have a 50/50 chance of getting each of the type A jobs. Further this player is assumed to be twice as likely set for type A initially as for type B since on average this strategy gives half of the type A jobs to this player in an iterated game (twice as many A jobs as B). So playing AB while the other player plays A (in the System 3 model) can result in this player's job queue looking something like: A,B,A. Now assume that the slightly less likely initial machine setup of B is the case. This sequence would then require 9 time units to complete; while the other machine would have processed it's two type A jobs in 2 time units. Not every arrival sequence amounts to something quite this bad, but cases like this one are why the payoffs to each player in the game of Figure 9 are as poor as they are for the entry (AB, A) in the matrix (i.e., (-6.375, -6.375)). If we contrast this to what happens if the wasp-like agents of System 4 play (AB, A) we see a much different picture. Like in System 3, the machine playing AB will definitely get the type B job. But, with respect to type A jobs, the choice is no longer unbiased at random. Whichever machine's queue appears to be shorter at the time of a type A job's arrival has a much greater chance of getting that job[18]. In this respect, the extremely deficient example result will not occur in System 4. The player playing AB in the NE (AB, A) or in the NE (A, AB) will only take type A jobs when it appears to benefit both players (i.e., when otherwise the other machine will be working while it remains idle). The global system performance of the two NE (AB, A) and (A, AB) of the game of Figure 10 is an improvement over that of the previous game. That is, if System 4 converges upon one of these, then it will outperform all of the three other systems discussed in this paper. Empirically, System 4 does appear to converge upon one of these two NE in the problem corresponding to the iterated version of this game (see [Cicirello and Smith, 2001b] for empirical results). These two NE of the System 4 game do not seem to have analogies in the analysis of the ant-based systems. They might be somewhat related to a sort of correlated equilibrium of the NE of the game associated with the ant-based systems (see Figure 3), but we do not feel that it is entirely accurate to state with certainty that this is the case. To explain why one of these two NE are converged upon, consider the following cases:

- The strategy NONE is again never played by the wasp-like agents for the reasons described previously. It is strictly dominated (very strongly so).

- If (A, A) is played, the type B jobs will eventually give off a stimulus great enough to result in some machine accepting them. This will result in either (A, B), (B, A), (A, AB), or (AB, A). This is similarly the case if (B, B) is played.

[18]The payoff calculation assumes the machine with shorter queue gets the job with probability 1 to simplify analysis.

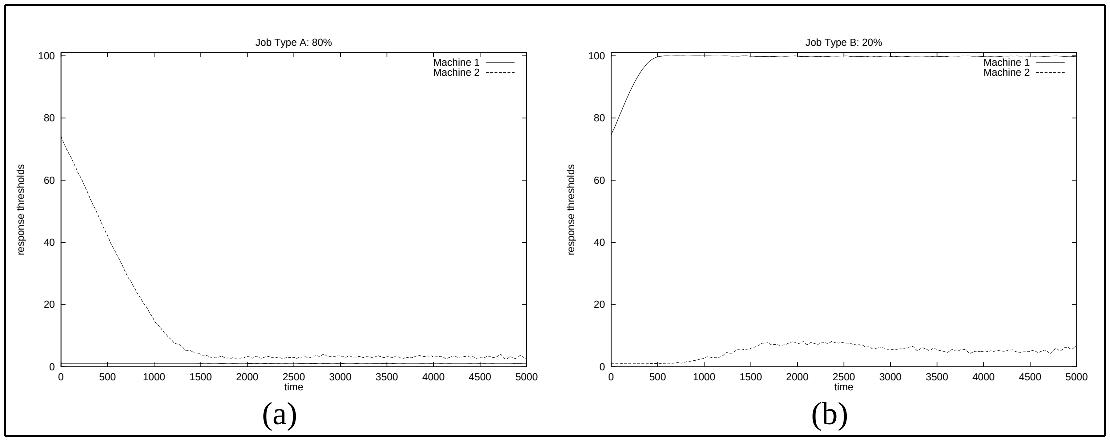

Figure 11: Plots of the average response thresholds over time of the routing wasps of System 4. These plots are averages over 100 simulation runs for a problem consisting of jobs of two types arriving in a ratio of four type A jobs to every one type B job.

- If (A, B) or (B, A) is played, the player that plays B will suffer from somewhat of an idleness penalty given that it gets one job each round while the other player gets four type A jobs. The idle machine response threshold updates will eventually cause that machine to play AB instead, resulting in either (A, AB) or (AB, A).

- If (AB, A) or (A, AB) are played, we are at a NE. Further, both machines are happy. The expected completion times of the machines separately are approximately the same. That is, the System 4 rule that breaks ties when both machines request a type A job, on average, keeps approximately equal queue lengths in this instance. There is no idleness penalty and no idle machine response threshold updates occur. The machine playing A is clearly satisfied since it is specialized to a single task type. The machine playing AB processes enough jobs of both types to keep the response thresholds low enough to accept more jobs of both types; while at the same time is not over-allocated type A jobs due to the System 4 tie breaking rule.

- If (AB, AB) is played, specialization pressures coupled with the very low demand for job type B will shift one of the machines to playing A, resulting in one of the NE.

In Figure 11 we see plots of the average response thresholds of the wasp-like agents of System 4 for the problem consisting of jobs of two types arriving in approximately a ratio of four type A jobs to every one type B job. The simulations ran for 5000 time units and the results shown are averages of 100 runs. The system parameters are set as they were in [Cicirello and Smith, 2001b]. In Figure 11a we see the response thresholds over

	A	B	AB	NONE
A	-100,-100	-4,-4	-4.05,-4.05	-100,-200
B	-4,-4	-100,-100	**-3.8,-3.8**	-100,-200
AB	-4.05,-4.05	**-3.8,-3.8**	-7.096,-7.096	-18.4,-18.4
NONE	-200,-100	-200,-100	-18.4,-18.4	-200,-200

Figure 12: System 4. Two-player game with five jobs (1 type A, 4 type B). All arrival sequences are equally likely. Players: Wasp-like machine agents. Strategies: A, B, AB, NONE. The NE are in bold.

time of the two machines to job type A (likewise to job type B in Figure 11b). As can be seen in these plots, both machines are willing to accept jobs of type A (i.e., their response thresholds for type A are low) and only one of these has a low response threshold for type B jobs. This illustrates the convergence of the System 4 wasp-like agents to the repeated play of the NE (AB, A).

Also worth examination is what happens if the problem changes (i.e., the game payoffs change) after the wasp-like agents have converged upon a NE in the repeated game. Consider the case where in the beginning there is a ratio of four type A jobs to every one type B job, but where after some amount of time has elapsed (some number of game iterations) this ratio changes to four type B jobs to every one type A job. The payoff matrix for the first set of iterations of this game will be as in Figure 10 for the System 4 wasp-like agents[19]. After the job type proportions change, the remaining iterations of the game will have payoffs as in Figure 12. There are again two pure strategy NE, but this time they are (B, AB) and (AB, B). Figure 13 illustrates how the System 4 wasp-like agents respond to such a change. Figure 13a shows plots of the response thresholds over time to job type A and Figure 13b shows the response thresholds over time to job type B. These plots are averages of 100 runs of 5000 time unit simulations. The problem change occurs at time unit 2500. In the first half of the simulations we see that the wasps have converged upon the repeated play of the NE (A, AB). When the problem change occurs at time unit 2500, (A, AB) is no longer a NE. Almost instantly, player two adapts to this change and begins playing strategy B to improve its payoffs. This is seen in Figure 13a where M2's threshold for job type A rises sharply. At the same time, player M1 begins receiving an idle time penalty due to the smaller proportion of type A jobs and eventually begins playing AB as can be seen in the sharp drop in M1's threshold for type B jobs seen in Figure 13b. The result is that the agents have converged upon the repeated play of the NE (AB, B) of Figure 12.

[19]This example would not be as interesting for the System 3 agents in that the NE are the same before and after the payoff change occurs.

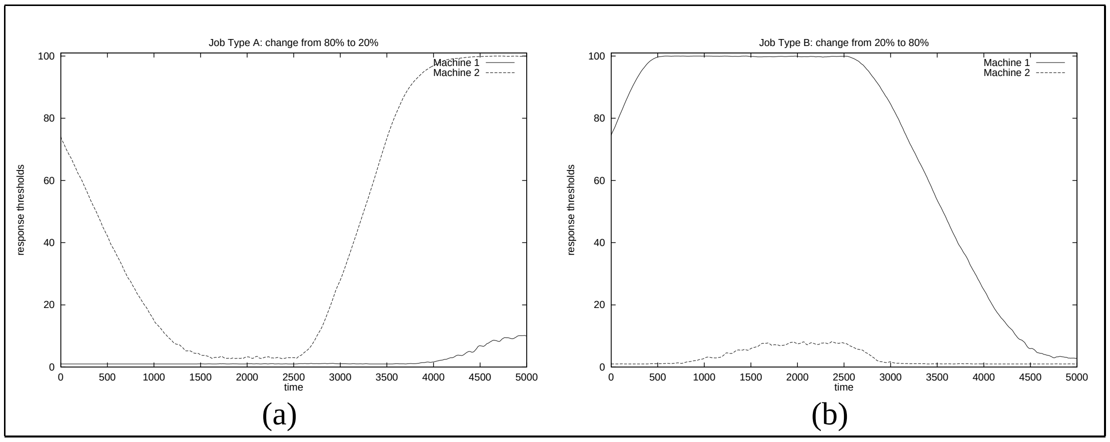

Figure 13: Plots of the average response thresholds over time of the routing wasps of System 4. These plots are averages over 100 simulation runs for a problem consisting of jobs of two types arriving in a ratio of four type A jobs to every one type B job during time units prior to 2500 and a ratio of one type A job to every four type B jobs after time unit 2500.

5 Conclusion

In this paper, a series of multi-agent systems for the dynamic shop floor routing problem have been modeled in game-theoretic terms. The games in question, corresponding to various specific problem instances, have all been games of coordination. That is, they all contain multiple NE and the problem became that of coordinating upon one of these in a repeated game scenario.

In Section 3, the coordination games under analysis are complicated by the fact that some NE are more desirable than others. System 1 suffers from extremely strong focal points on severely deficient mixed strategy NE. System 2 conquers this problem with the addition of a form of “cross-colony” communication that allows an agreement to be made in regard to playing a particular pure strategy NE with much better payoffs as compared to the deficient mixed strategy NE. We also see in the five job type problem that eight of the ten pure strategy NE are not very stable when the game is played repeatedly. That is, if one of these eight are played, they result in a change to the game structure in the next repetition that results in their non-existence. However, the other two pure strategy NE are very stable in this respect. If they are played, they too result in a change to the game structure for the next iteration; but this change is in such a way as to enforce their status as a NE. Further, when one of the eight NE that are not very stable in the repeated game are played, the game structure change strongly favors the other two NE in terms of payoffs.

In Section 4, there are far fewer NE to deal with and in the examples that are examined in this paper all NE appear equally desirable. The problem is one of coordination upon one of these. The derivations of the payoffs for the games that model the wasp-like agents' behavior are not as obvious as are those for the ant-like agents of Section 3. With the ant-like agents, it is clear that they are self-interested in shortest expected throughput time for their own jobs. The wasp-like agents of Section 4 do not have such clearly defined utilities. Despite this, as modeled in this paper, the behavior of the wasp-like agents under analysis seems to correspond to convergence upon the repeated play of NE in iterated games corresponding to specific problem instances. The interactions of the System 4 agents appear to lead to superior global system performance in these instances as compared to the other systems that were analyzed. This seems to be the case both in the payoffs of the NE in the game that models the System 4 agents as well as in empirical system results.

Returning to the payoff derivations for the wasp-based systems, potential for future work can be seen. It was mentioned in footnotes that the idleness, unprocessed jobs, and specialization factors considered in payoff computation would probably be more accurately defined in terms of the system parameters. Due to some simplification assumptions, this was not done. Considering that the model described in this paper appears to match the behavior of the agents closely, it might be argued that the payoff computation in this paper is sufficient. However, if these factors could be modeled more accurately in terms of system parameters, then it may be possible to derive "better" values for these system parameters from an analysis of the resulting games. In the problem instances seen in this paper in the wasp-based system discussion, all NE were seen as equally desirable as well as Pareto-optimal. But perhaps, for some other problem instance, an analysis that considers system parameters in the payoff calculation may lead to something more like a prisoner's dilemma rather than the coordination games seen so far for certain ranges of parameter values. If such problem instances exist, then it would be beneficial to characterize these as well as to determine system parameter values that do not lead to such games if possible. This is an area for further study.

Acknowledgments

This work has been funded in part by the Department of Defense Advanced Research Projects Agency and the U.S. Air Force Rome Research Laboratory under contracts F30602-97-2-0066 and F30602-00-2-0503 and by the CMU Robotics Institute. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force or U.S. Government.

References

[Beaumariage and Kempf, 1995] T. Beaumariage and K. Kempf. Attractors in manufacturing systems with chaotic tendencies, 1995. Presentation at INFORMS-95, New Orleans, http://www.informs.org/Conf/NewOrleans95/ TALKS/TB07.3.html.

[Beckers *et al.*, 1994] R. Beckers, O. E. Holland, and J. L. Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. In R. A. Brooks and P. Maes, editors, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 181–189, 1994.

[Bonabeau *et al.*, 1999] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 1999.

[Bonabeau *et al.*, 2000] E. Bonabeau, M. Dorigo, and G. Theraulaz. Inspiration for optimization from social insect behaviour. *Nature*, 406:39–42, July 2000.

[Cicirello and Smith, 2001a] V. A. Cicirello and S. F. Smith. Ant colony control for autonomous decentralized shop floor routing. In *ISADS-2001, Fifth International Symposium on Autonomous Decentralized Systems*. IEEE Computer Society Press, March 2001.

[Cicirello and Smith, 2001b] V. A. Cicirello and S. F. Smith. Improved routing wasps for distributed factory control. In *IJCAI-01 Workshop on Artificial Intelligence and Manufacturing: New AI Paradigms for Manufacturing*, August 2001.

[Cicirello and Smith, 2001c] V. A. Cicirello and S. F. Smith. Insect societies and manufacturing. In *IJCAI-01 Workshop on Artificial Intelligence and Manufacturing: New AI Paradigms for Manufacturing*, August 2001.

[Cicirello and Smith, 2001d] V. A. Cicirello and S. F. Smith. Wasp nests for self-configurable factories. In *Agents 2001, Proceedings of the Fifth International Conference on Autonomous Agents*. ACM Press, May-June 2001.

[Dorigo and Di Caro, 1999] M. Dorigo and G. Di Caro. The ant colony optimization metaheuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*. McGraw-Hill, 1999.

[Fitzgerald and Peterson, 1988] T. D. Fitzgerald and S. C. Peterson. Cooperative foraging and communication in caterpillars. *BioScience*, 38(1):20–25, January 1988.

[Gordon, 1996] D. M. Gordon. The organization of work in social insect colonies. *Nature*, 380:121–124, March 1996.

[Kempf and Beaumariage, 1994] K. Kempf and T. Beaumariage. Chaotic behavior in manufacturing systems. In *AAAI-94 Workshop Program, Reasoning About the Shop Floor, Workshop Notes*, 1994.

[Kirchner and Towne, 1994] W. H. Kirchner and W. F. Towne. The sensory basis of the honeybee's dance language. *Scientific American*, 270(6):74–80, June 1994.

[Kreps, 1990] D. M. Kreps. *Game Theory and Economic Modelling*. Clarendon Lectures in Economics. Clarendon Press / Oxford University Press, 1990.

[Lin and Solberg, 1992] G. Y. J. Lin and J. J. Solberg. Integrated shop floor control using autonomous agents. *IIE Transactions*, 24(3):57–71, 1992.

[Liu, 1996] J. S. Liu. *Coordination of Multiple Agents in Distributed Manufacturing Scheduling*. PhD thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 1996.

[Luce and Raiffa, 1985] R. D. Luce and H. Raiffa. *Games and Decisions: Introduction and Critical Survey*. Dover Publications, Inc., New York, 1985.

[Morley and Schelberg, 1993] D. Morley and C. Schelberg. An analysis of a plant-specific dynamic scheduler. In *Final Report, Intelligent Dynamic Scheduling for Manufacturing Systems*, pages 115–122, June 1993.

[Morton and Pentico, 1993] T. E. Morton and D. W. Pentico. *Heuristic Scheduling Systems: With Applications to Production Systems and Project Management*. John Wiley and Sons, 1993.

[Ow *et al.*, 1988] P. S. Ow, S. F. Smith, and R. Howie. A cooperative scheduling system. In M. D. Oliff, editor, *Expert Systems and Intelligent Manufacturing*, pages 43–56. Elsevier Science Publishing Co., Inc., 1988.

[Parunak *et al.*, 1998] V. Parunak, A. Baker, and S. Clark. The AARIA agent architecture: From manufacturing requirements to agent-based system design. In *Proceedings of the ICAA'98 Workshop on Agent-Based Manufacturing*, Minneapolis, MN, May 1998.

[Schoonderwoerd *et al.*, 1997] R. Schoonderwoerd, O. Holland, and J. Bruten. Ant-like agents for load balancing in telecommunications networks. In *Agents '97, Proceedings of the First International Conference on Autonomous Agents*, pages 209–216. ACM Press, 1997.

[Sycara *et al.*, 1991] K. P. Sycara, S. F. Roth, N. Sadeh, and M. S. Fox. Resource allocation in distributed factory scheduling. *IEEE Expert*, 6(1):29–40, February 1991.

[Theraulaz *et al.*, 1991] G. Theraulaz, S. Goss, J. Gervet, and J. L. Deneubourg. Task differentiation in polistes wasp colonies: A model for self-organizing groups of robots. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 346–355. MIT Press, 1991.