Wasp-like Agents for Distributed Factory Coordination

Vincent A. Cicirello Stephen F. Smith

CMU-RI-TR-01-39

December 2001

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
{cicirello, sfs}@cs.cmu.edu

Abstract

Agent-based approaches to manufacturing scheduling and control have gained increasing attention in recent years. Such approaches are attractive because they offer increased robustness against the unpredictability of factory operations. But the specification of local coordination policies that give rise to efficient global performance and effectively adapt to changing circumstances remains an interesting challenge. In this paper, we present a new approach to this coordination problem, drawing on various aspects of a computational model of how wasp colonies coordinate individual activities and allocate tasks to meet the collective needs of the nest. We focus specifically on the problem of configuring machines in a factory to best satisfy (potentially changing) product demands over time. Wasp-like computational agents that we call routing wasps act as overall machine proxies. These agents use a model of wasp task allocation behavior, coupled with a model of wasp dominance hierarchy formation, to determine which new jobs should be accepted into the machine's queue. We show for simple factories that our multi-agent system achieves the desired effect. For a given job mix, the system converges to a factory configuration that maximizes overall performance, and as the job mix changes, the system adapts to a new, more appropriate configuration. We also show that our system is competitive to that of an agent-based system for the problem that has been successfully demonstrated in real-world practice, outperforming this prior system in its intended domain in several important respects.

Contents

1 Introduction

Effective coordination of multiple agents interacting in dynamic environments is an important part of many real-world problems. For example, teams of robots exploring alien terrain need to coordinate such activities as task assignment and scheduling among the team members; groups of agents comprising web-based information tools need to coordinate such activities as information gathering, sharing, and so forth. Our particular interest in this paper is the domain of factory operations, which is also an important example of a dynamic scheduling / coordination problem.

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 schedules in advance. 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 & 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 can be 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 also motivated research into agent-based approaches to manufacturing scheduling and control (e.g., (Lin & Solberg, 1992; Liu, 1996; Ow, Smith, & Howie, 1988; Parunak, Baker, & Clark, 1998; Sycara, Roth, Sadeh, & Fox, 1991)) and there have been a few interesting successes. For example, Morley uses a simple bidding mechanism for assigning trucks to paint booths at a General Motors factory (Morley & Schelberg, 1993; Morley, 1996). They have shown that their decentralized approach, which imparts decision-making ability upon the paint booths themselves, outperformed the previous centralized scheduling system in terms of increased throughput and lower paint costs.

However, decentralized approaches can sometimes also be susceptible to sub-optimal and even chaotic global behavior. Kempf and Beaumariage show how a distributed manufacturing system, utilizing extremely simple dispatch policies, can exhibit formally chaotic behavior (Kempf & Beaumariage, 1994; Beaumariage & Kempf, 1995). Their simple ex-

ample system has a number of attractors with drastically different global performance. They show that small changes in policies or in system state can force the system to move between these attractors leading to large changes on a global scale. In general, 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 (i.e., policies that adapt to dynamically changing circumstances). There are many examples of effective, adaptive behavior in natural multi-agent systems (e.g., (Fitzgerald & Peterson, 1988; Gordon, 1996; Kirchner & Towne, 1994; Theraulaz, Goss, Gervet, & Deneubourg, 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, Holland, & Deneubourg, 1994; Bonabeau, Dorigo, & Theraulaz, 2000; Dorigo & Di Caro, 1999; Schoonderwoerd, Holland, & Bruten, 1997a)). Bonabeau et al. (Bonabeau, Dorigo, & Theraulaz, 1999) provide a comprehensive survey of adaptive multi-agent systems that have been inspired by social insect behavior (for a survey focused on manufacturing applications of social insect behavior see Cicirello and Smith (Cicirello & Smith, 2001c)).

In this paper, we present a system for dynamic shop floor routing based on the natural multi-agent system of the wasp colony. A computational model of real wasp behavior of Theraulaz et al. (Theraulaz et al., 1991; Bonabeau, Theraulaz, & Deneubourg, 1998; Theraulaz, Bonabeau, & Deneubourg, 1998, 1995) lies at the foundation of our approach. In this model, interactions between individual wasps and the local environment (wasp-to-environment interactions) in the form of a stimulus-response mechanism govern distributed task allocation. Further, interactions between pairs of individual wasps (wasp-to-wasp interactions) result in the self-organization of dominance hierarchies. We combine these two aspects of wasp behavior into an effective, decentralized basis for coordinating the assignment of jobs to machines in a factory. Our approach is shown to be competitive to the "real-world proven" market-based truck painting system of Morley (Morley & Schelberg, 1993; Morley, 1996) in which machines (specifically, vehicle paint booths) bid for jobs (trucks) as they arrive. Our wasp-based system is in a sense an adaptive bidding mechanism which adapts its decision policy of whether to bid or to not bid based on job type. Morley's bid strategy required a machine to bid if there was space in its queue; whereas our adaptive policy allows non-bids to occur in anticipation of near-future jobs of a type for which the machine is better-suited.

The remainder of the paper will proceed as follows. Section 2 provides an overview of the model of wasp behavior that underpins our approach. Section 3 and Section 4 detail our multi-agent system for the problem of dynamic shop floor routing. Section 5 provides an empirical analysis of the behavior of our wasp-like agents in simple simulated factory

environments. Section 6 extends the analysis to provide a performance comparison to Morley's system. More specifically, Section 6.1 makes the comparison on relatively simple problems and Section 6.2 carries the experimentation to the next level and compares our system to that of Morley's on the real-world problem for which his system was originally designed. Section 7 discusses limitations of our system. We provide a summary of related work in Section 8 and conclude in Section 9.

2 Wasp Behavioral Model

In (Theraulaz et al., 1991), Theraulaz et al. present a model for the self-organization that takes place within a colony of wasps. Interactions between members of the colony and the local environment result in dynamic distribution of tasks such as foraging and brood care. In addition, a hierarchical social order among the wasps of the colony is formed through interactions among individual wasps of the colony. This emergent social order is a succession of wasps from the most dominant to the least dominant.

The model of (Theraulaz et al., 1991) describes the nature of interactions between an individual wasp and its local environment with respect to task allocation. They model the colony's self-organized allocation of tasks using what they refer to as response thresholds. An individual wasp has a response threshold for each zone of the nest. Based on a wasp's threshold for a given zone and the amount of stimulus from brood located in that zone, a wasp may or may not become engaged in the task of foraging for that zone. A lower threshold for a given zone amounts to a higher likelihood of engaging in activity given a stimulus. Bonabeau, Theraulaz, and Deneubourg discuss in (Bonabeau et al., 1998) a model in which these thresholds remain fixed over time. But in (Theraulaz et al., 1998), a threshold for a given task decreases during time periods when that task is performed and increases otherwise. Bonabeau et al. (Bonabeau, Sobkowski, Theraulaz, & Deneubourg, 1997) demonstrate how this model leads to a distributed system for allocating mail retrieval tasks to a group of mail carriers. In this paper, we similarly adopt this task allocation model for our routing wasps to assign (or route) jobs to machines in a distributed factory setting[1].

The model of (Theraulaz et al., 1991) also describes the nature of wasp-to-wasp interactions that take place within the nest. When two individuals of the colony encounter each other, they may with some probability interact with each other. If this interaction takes place, then the wasp with the higher social rank will have a higher probability of dominating in the interaction. Through such interactions as these, wasps within the colony

[1] A preliminary version of our wasp-based system for factory coordination was presented in (Cicirello & Smith, 2001e, 2001b). In this paper, we further extend our initial model and provide a more detailed experimental analysis.

self-organize themselves into a dominance hierarchy. Self-organization of dominance hierarchies among wasps is discussed in greater detail in (Theraulaz et al., 1995). For example, Theraulaz, Bonabeau, and Deneubourg discuss a number of ways of modeling the probability of interaction during an encounter which range from always interacting to interacting based upon certain tendencies of the individuals. In our scheduling wasp definition of (Cicirello & Smith, 2001e, 2001d), we used this concept to model job priority and to prioritize jobs in a given machine queue. In this paper we do not consider these scheduling wasps, but instead use the concept of social dominance to determine which from among a group of routing wasps competing for the same job is assigned the job.

3 Routing Wasps

Our model is concerned most generally with the configuration of product flows. In the simplest case the problem involves a set of multi-purpose machines, each capable of processing multiple types of jobs but with a setup cost for reconfiguring from one type to another. Each machine in the system has an associated routing wasp (see Figure 1 for illustration). Each routing wasp is in charge of assigning jobs to the queue of its associated machine. Each routing wasp has a set of response thresholds:

$$\Theta_w = \{\theta_{w,0}, \ldots, \theta_{w,J}\} \tag{1}$$

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 that its associated machine can process.

Jobs in the system that have not yet been assigned to 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 in the system. This stimulus is "typed" according to job type. So the longer the job remains unrouted, 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:

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

This is the rule used for task allocation in the wasp behavioral model as described in (Theraulaz et al., 1998). The exponent of "2" can be seen as a system parameter. This is the value used in the original wasp model and it appears to work well experimentally in our domain. In this way, wasps will tend to pick up jobs of the type for which its response threshold is lowest. But it will pick up jobs of other types if a high enough stimulus is emitted.

The threshold values $\theta_{w,j}$ may vary in the range $[\theta_{min}, \theta_{max}]$. Each routing wasp, at all times, knows what its machine is doing, including: the status of the queue, whether or not

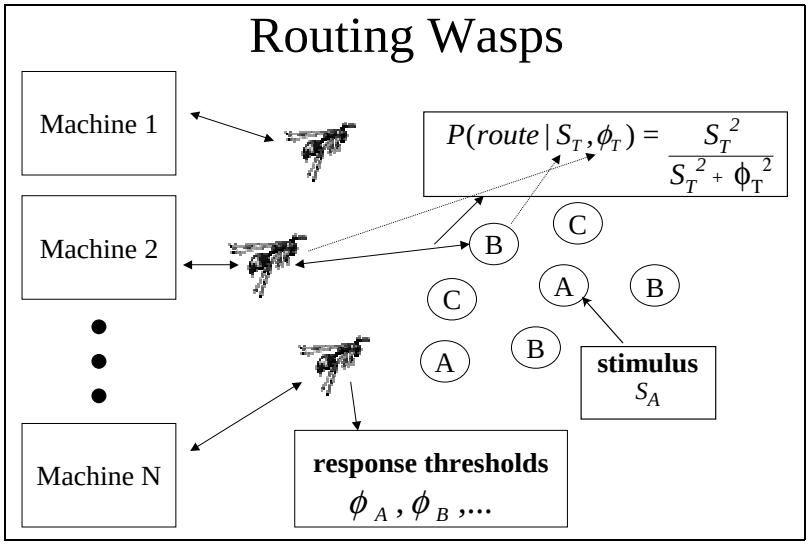

Figure 1: Routing wasps: Each multi-purpose machine is represented by a wasp-like agent called a routing wasp. This routing wasp maintains response thresholds $\theta_{w,i}$ for each job type i that the machine associated with wasp w can process. Given a stimulus for a job of type i, the routing wasp stochastically decides whether or not to bid for the job according to the type and the response threshold for that type. If more than one wasp responds with a bid, the winner is chosen with a tournament of dominance contests.

the machine is performing a setup, the type of job being processed, and whether or not the machine is idle. This knowledge is used to adjust the response thresholds for the various job types. This updating of the response thresholds occurs at each time step. 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:

$$\theta_{w,j} = \theta_{w,j} - \delta_1 \tag{3}$$

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:

$$\theta_{w,j} = \theta_{w,j} + \delta_2 \tag{4}$$

And 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 (t is the length of time the machine has been idle and is an exponent):

$$\theta_{w,j} = \theta_{w,j} - \delta_3^t \tag{5}$$

In this way, the response thresholds for the job type currently being processed are reinforced as to encourage the routing wasp to pick up jobs of the same type; while the response thresholds of other types not currently being worked on are adapted to discourage the routing wasp from taking these jobs. This specialization of routing wasps (i.e., machines) helps to minimize setup time. The first two ways in which the response thresholds are updated (equations 3 and 4) are analogous to that of the model described in (Bonabeau et al., 1997; Theraulaz et al., 1998). The third (equation 5) is included to encourage a wasp associated with an idle machine to take whatever jobs it can get. This last update rule acknowledges that although specialization can reduce setup time, over-specialization to a job type with low demand may result in lower system throughput.

4 Dominance Contests

The routing wasp formulation of Section 3 does not state what happens if two or more routing wasps respond positively to the same job stimulus. One simple approach is to make the decision randomly (Cicirello & Smith, 2001e). But there is a problem with this. Consider the case where one machine has been sitting idle and has an empty job queue. Perhaps this machine has specialized to a job type whose demand has diminished. Now consider a second machine with a long queue of jobs. Suppose a new job arrives at the factory of the type for which this second machine has developed a preference. The idle machine by this point is willing to take any job. Both machines respond to the stimulus from this new job. Previously, both machines would have an equal chance of taking on this new job. But perhaps the idle machine with the empty queue should have a higher probability of getting the job even though it is not currently configured for this job type and will accrue some setup time.

To this end we augment the basic routing wasp model with a method for deciding which routing wasp from a group of competing wasps gets the job. This method is based on the self-organized social hierarchies of real wasps. First define the force F_w of a routing wasp w as:

$$F_w = 1.0 + T_p + T_s \tag{6}$$

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[2]. The values of T_p and T_s are easily

[2]In this definition of force, the "stronger" wasp is the wasp with the smaller force. This may seem counter-intuitive with the usual connotation of the word "force", but defining force in this way is cleaner mathematically. Perhaps "weakness" may have been a more accurate term to use rather than "force", but we chose the latter to correspond more closely to the terminology of the model of real wasp behavior.

computed given the queue of jobs. 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:

$$P(F_1, F_2) = \frac{F_2^2}{F_1^2 + F_2^2} \tag{7}$$

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. Seedings in this tournament are according to the values of the force variables of the competing wasps. To deal with odd numbers of competing wasps, according to this seeding, the top $2^{\lceil \log_2 (C) \rceil} - C$ wasps, where C is the number of competing wasps, receive "buys" to the second round (i.e., they do not have to compete in the first round).

5 System Analysis

In this Section we examine the behavior of the routing wasps on a few simple factory configurations. The purpose of this analysis is to illustrate that the behavior adapted by the routing wasps corresponds to what intuitively is the "best" routing policy. In Section 5.1 we describe the design of the problems used in this analysis. Section 5.2 shows that the dominance contests lead to improved performance over the random tie-breaking rule. The behavior of the routing wasps is analyzed in Section 5.3 using plots of the response thresholds over time.

5.1 Experimental Design

All of the experiments that are presented here were performed in a simulated factory environment implemented in Java and executed on a Pentium III running RedHat Linux 5.2. All experiments in this Section consider factories which produce two products (henceforth, Job Type A and Job Type B) and multi-purpose machines that can process either of the two product types (only single stage jobs are considered here). Later, in Section 6, we consider a real-world problem with 14 job types. Experiments with two and four machines are studied (the real-world problem of Section 6 has seven machines)[3]. In all cases, setup time to

[3]The remainder of the details of the real-world problem instance are deferred until Section 6.

reconfigure a machine for the alternate job type is 30 time units. Each machine can only process jobs in its queue in a first-in-first-out order and the task of routing jobs to these queues is performed by the routing wasps. When a new job is generated, its process time is 15 plus a Gaussian noise factor with mean 0.0 and standard deviation 1.0. The resulting process time is taken to the next higher integer value for times greater than 15 and taken to the next lower integer value for times less than 15. The overall process time is bounded in the range of 10 to 20, inclusive.

Jobs are released to the factory floor dynamically according to four different product mixes (3 static and 1 changing). In each, arrival rates are defined by the probability a new job of each type is released during a given time unit. The arrival rates for the two machine problems are as follows:

- 50/50 mix: P(Job Type A) = 0.05, P(Job Type B) = 0.05
- 85/15 mix: P(Job Type A) = 0.0857, P(Job Type B) = 0.0143
- 100/0 mix: P(Job Type A) = 0.133, P(Job Type B) = 0.0
- Changing mix: For the first half of the simulation P(Job Type A) = 0.0857, P(Job Type B) = 0.0143, then for the second half of the simulation P(Job Type A) = 0.0143, P(Job Type B) = 0.0857

To get the rates for the four machine problems simply multiply these rates by 2. These arrival rates correspond approximately to medium-to-heavily loaded factories.

The values of the various parameters of the system are the following (unless otherwise stated):

- $\theta_{min} = 1$
- $\theta_{max} = 1000$
- $\delta_1 = 2$
- $\delta_2 = 1$
- $\delta_3 = 1.001$

No special tuning process is behind these settings. A few different values were tried and those that seemed to give adequate performance were chosen.

All results included in this paper are averages of 100 runs with different arrival sequences. The simulations are 5000 time units in duration. The time units correspond to

Table 1: Average throughput for different job mixes comparing R-Wasps and R-Wasps-D. 95% confidence intervals and two-tailed p-values from paired *t*-tests are shown.

Two Machine Problem			
	R-Wasps	R-Wasps-D	p-value
50/50 mix	**492.82±4.33**	**492.82±4.33**	–
85/15 mix	470.47±2.20	**474.00±2.24**	<0.0001
100/0 mix	605.37±2.92	**624.92±1.65**	<0.0001
Changing	444.72±2.96	**448.53±2.32**	0.0058
Four Machine Problem			
	R-Wasps	R-Wasps-D	p-value
50/50 mix	988.84±6.58	**990.46±6.64**	<0.0001
85/15 mix	981.89±5.98	**994.03±6.03**	<0.0001
100/0 mix	1234.39±3.27	**1255.39±1.95**	<0.0001
Changing	879.25±5.47	**899.62±4.02**	<0.0001

nothing in particular so consider them minutes, hours, or whatever other time unit you fancy. In the remainder of the paper, R-Wasps will refer to the original definition of the routing wasp formulation of (Cicirello & Smith, 2001e) and R-Wasps-D will refer to the addition of the dominance contests as a tie breaking rule when more than one routing wasp is competing for a single job.

5.2 Effect of Dominance Contests

In Table 1 we see average throughput (number of jobs processed) results for the various job mixes for both the two machine and four machine problems comparing R-Wasps and R-Wasps-D. For the 50/50 job mix in the two machine case, we see no difference between the behavior of R-Wasps and R-Wasps-D. Each produces the exact same results on all 100 simulations. In this case there are on average equal numbers of both job types and the shop is fairly heavily loaded, so each of the two machines is content to specialize to one job type. Due to this, the routing wasps never have to compete for a desired job and thus the system behavior is the same in both cases. The reason this is not quite the case in the four machine problem is that the dominance contest tie-breaking rule results in a bit of load-balancing.

However, for all other job mixes, R-Wasps-D outperforms R-Wasps in terms of throughput. Paired *t*-tests were performed and in all cases R-Wasps-D was seen, with statistical significance, to process more jobs on average than R-Wasps. The dominance contests of R-Wasps-D result in improved factory routing performance as compared to the random

tie-breaking strategy of R-Wasps.

5.3 Response Threshold Analysis

In Figure 2, we see plots of the response thresholds of the routing wasps for various job mixes and two machines (the results are for R-Wasps-D). These results are averages of 100 runs. The first column is the response thresholds for job type A and the second column for job type B. The rows, respectively, are for the 50/50, 85/15, 100/0, and changing job mixes. In the 50/50 job mix, we see that each machine specializes to a different job type. In the 100/0 job mix, we see that both machines quickly adapt their configurations to that associated with the single job type in the system. For the 85/15 job mix, one machine specializes to the job type for which there are more jobs, and the other machine is willing to take either job type. The first half of the changing job mix simulations corresponds to that of the 85/15 job mix; while in the second half we see the machines changing roles to handle the new 15/85 mix. If we examine similarly the four machine problems in Figure 3 we find the same sort of behavior. That is, in the 100/0 job mix all machines specialize in the single job type, in the 50/50 job mix half of the machines specialize to each job type, and in the 85/15 job mix all machines are willing to take the job type of higher demand while only one has a strong interest in the other job type. This corresponds, intuitively, to the behavior the system should exhibit for "optimal" performance.

One thing that is worth noting, particularly in regard to the 85/15 job mix and the changing job mix, is that the system requires some time to converge to a stable behavior. For example, if you examine the second row of response threshold plots in Figure 2, you can see that both machines are taking jobs of the product of highest demand only after (approximately) time unit 1000. This is similarly true in the changing job mix (see the fourth row of Figure 2). With the changing job mix, however, we have the addition of a second stage of adaptation beginning at time unit 2500 when the job mix changes drastically from 85/15 to 15/85. In Section 6, after we have compared our system to that of a fairly successful multi-agent system for the problem, we will return to this rate of adaptation issue.

6 Experiments

In this Section, we compare R-Wasps-D of this paper to the GM truck painting system of Morley (Morley, 1996; Morley & Schelberg, 1993) mentioned earlier. Morley's system was designed for a problem very much like our problem. There are some number of truck painting booths. Trucks roll off the assembly line at some rate and have to be assigned to booths to be painted. Each truck requires a specific color paint and there are more possible

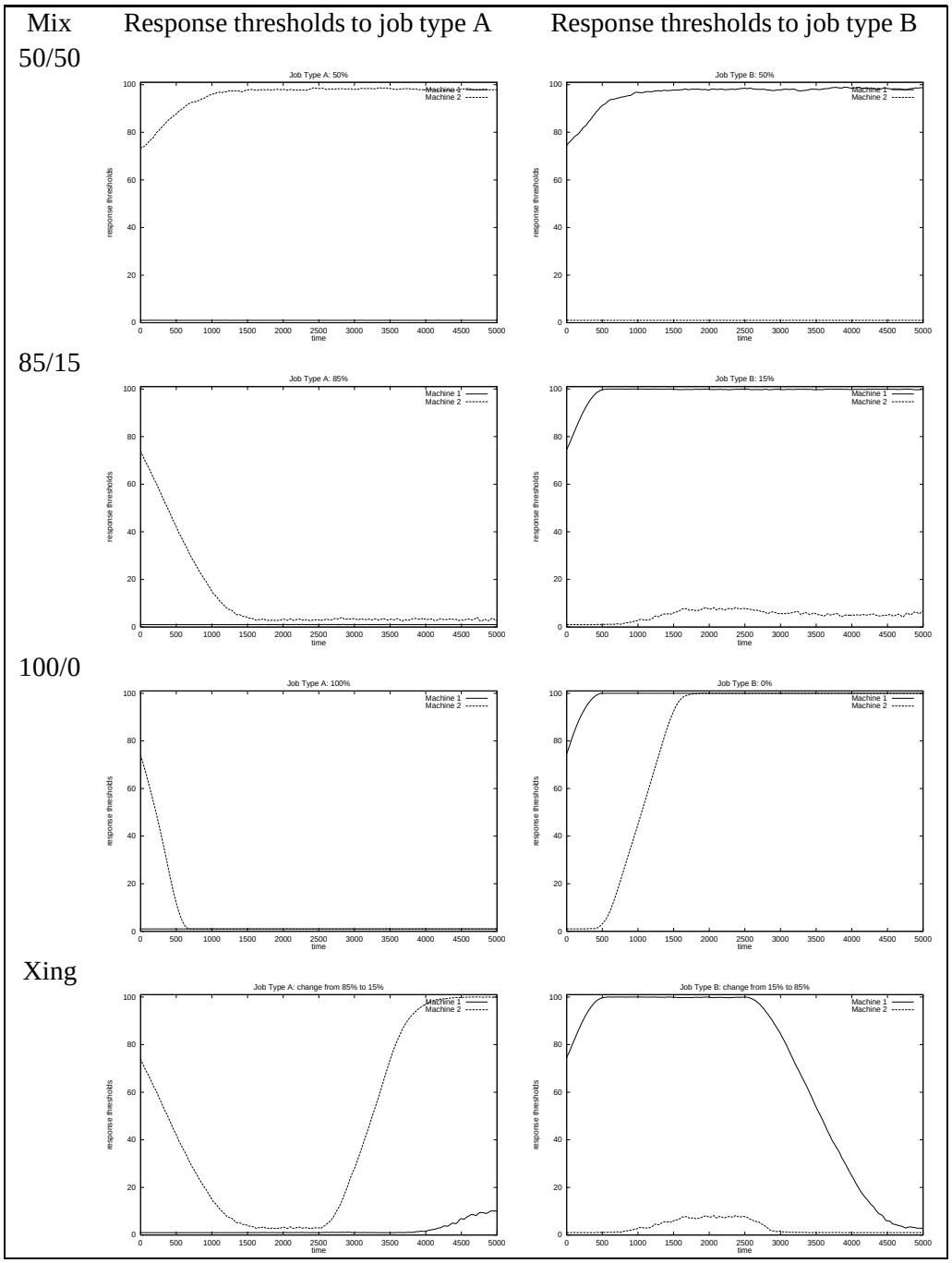

Figure 2: Plots of the average response thresholds over time of the routing wasps for different job mixes and two machines.

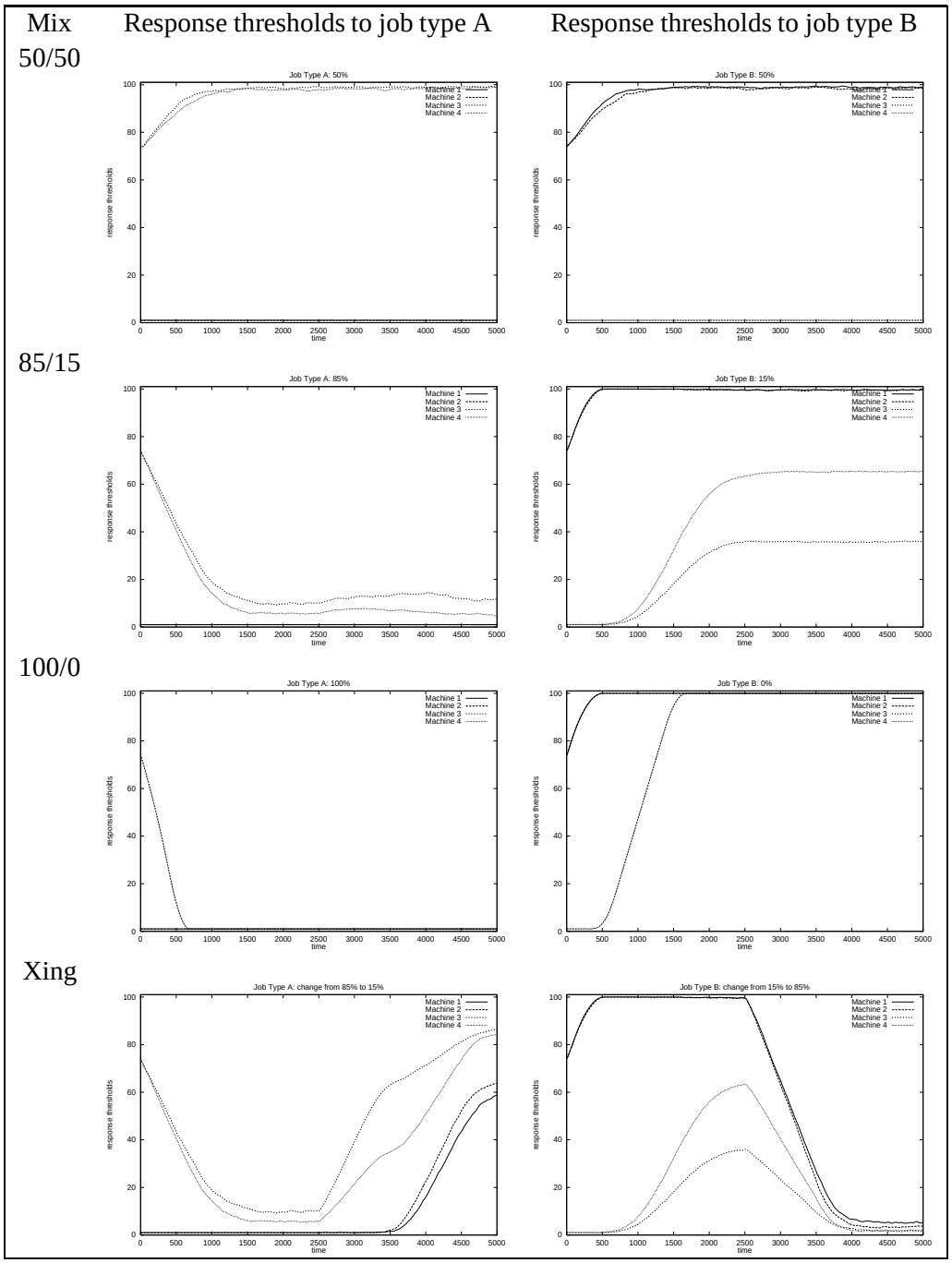

Figure 3: Plots of the average response thresholds over time of the routing wasps for different job mixes and four machines.

colors than there are paint booths. Each booth paints trucks in a FIFO order according to its personal queue. Changing the color that a booth is setup to paint is an expensive operation both in terms of time as well as paint cost so minimizing such setups is desirable.

Morley devised a simple bidding mechanism in which booth agents submit bids for trucks as they arrive according to their current queue length and the required color of the last truck in the queue. This simple multi-agent bidding system was shown in simulation to be more effective than the previously used centralized scheduler, and was subsequently put into use at a General Motors truck painting facility. When put into practice in the GM facility, Morley's system was found to be 10% more efficient (in terms of number of paint color changes) than the previously used centralized scheduler (Morley & Schelberg, 1993) and resulted in savings of nearly a million dollars in the first nine months of use (Morley, 1996). Due to its effectiveness and real-world implementation, we feel that Morley's system is an excellent choice to use as a benchmark for our system and is indicative of the state-of-the-art in agent-based systems for this class of problem.

In Section 6.1, we compare our wasp-based system (referred to as R-Wasps-D) to Morley's agent-based truck painting system (referred to as Morley) on the simple two and four machine problems described in the system analysis of Section 5. This comparison shows that our system is competitive with Morley's on simple problem configurations. Then in Section 6.2, we compare our system, R-Wasps-D, to Morley's system on Morley's real-world problem instance with seven machines and fourteen job types. The details of that problem instance will be described in that Section. The comparison shows that our system greatly outperforms the benchmark system of Morley as the problem is scaled to real-world complexity.

6.1 Simple Problems

Table 2 show a throughput comparison between our R-Wasps-D and Morley's truck painting system for the simple two and four machine problems described in Section 5. In all of the static job mixes (i.e., 50/50, 85/15, and 100/0) and for both two and four machines, our system is superior to Morley's in terms of overall throughput (i.e., total number of jobs processed) attained. All of these results are statistically significant according to paired *t*-tests except for the 100/0 job mix with two machines.

In the changing job mix, Morley's system performs better than R-Wasps-D (see Table 2). It is interesting to explore the explanation for this. First recall that in the changing job mix problem, the first half of the simulation consists of an 85/15 mix and the second half consists of a 15/85 mix. From the point of view of Morley's simple bidding mechanism, both halves of the simulation are more or less the same but with the proportions of the two jobs reversed. This leads to the idea that Morley's system sees the 85/15 job mix

Table 2: Average throughput for different job mixes comparing R-Wasps-D and Morley's GM truck painting system. 95% confidence intervals and two-tailed p-values from paired *t*-tests are shown.

Two Machine Problem			
	Morley	R-Wasps-D	p-value
50/50 mix	491.62±4.37	**492.82±4.33**	<0.0001
85/15 mix	464.50±4.18	**474.00±2.24**	<0.0001
100/0 mix	622.98±3.37	**624.92±1.65**	0.1707
Changing	**465.79±3.63**	448.53±2.32	<0.0001
Four Machine Problem			
	Morley	R-Wasps-D	p-value
50/50 mix	973.67±6.84	**990.46±6.64**	<0.0001
85/15 mix	959.03±7.52	**994.03±6.03**	<0.0001
100/0 mix	1234.13±3.93	**1255.39±1.95**	<0.0001
Changing	**962.95±6.00**	899.62±4.02	<0.0001

problem as essentially equivalent to the changing job mix problem. If you examine Table 2, the results for Morley's system of both the 85/15 and changing job mixes are roughly the same. This is not true of R-Wasps-D. So why not? The answer lies in the analysis of the response thresholds that appeared earlier in Section 5.3. For the first 1000 or so time units of the 85/15 job mix problem, R-Wasps-D is adapting to the mix. During this time, it is performing less than desirably. But once it converges to a stable behavior it begins outperforming Morley's system and by the end of the 5000 time unit simulation overtakes Morley's in terms of overall results. With the changing job mix, the first 1000 time units or so are again spent adapting to the 85/15 job mix. In addition, however, the 1000 time units or so beginning at time unit 2500 are then spent adapting to the very drastic job mix change to a mix of 15/85. So in the changing job mix experiment, approximately 40 percent of the time is spent adapting to the mix in this 5000 time unit simulation. The remainder of the simulation is not enough time to overtake Morley's system in overall performance.

To confirm this conclusion, consider the results given in Table 3. Table 3 shows the changing job mix problems again for both Morley's system and R-Wasps-D. This time the simulation length is 10,000 time units with the job mix change occurring at time unit 5000. For the two machine problem, Morley still outperforms R-Wasps-D. But this time, R-Wasps-D is seen as superior to Morley's system for the four machine problem.

A question that can be raised is whether or not such a drastic job mix change from 85/15 to 15/85 occurring during a single time unit is realistic. Consider what such a change may

Table 3: Average throughput for changing job mix with simulations 10,000 time units in length comparing R-Wasps-D and Morley's GM truck painting system. 95% confidence intervals and two-tailed p-values from paired *t*-tests are shown.

# machines	Morley	R-Wasps-D	p-value
2	**965.41±5.22**	948.54±3.03	<0.0001
4	1959.51±8.48	**1986.18±7.33**	<0.0001

represent. One possibility is that a factory which produces two similar products but with very different market demands is suddenly faced with a complete reversal of the product demands. From this perspective, this changing job mix problem is certainly an extreme case. Perhaps a better problem to examine would be a job mix that changes gradually from 85/15 to 15/85 rather than such a sharp change all at once. We chose the drastic change problem instead for a couple of reasons: 1) more of a challenge; and 2) if we solve this problem, we solve the gradual change problem as well. With that, we are working on improving the rate of adaptation of R-Wasps-D.

6.2 Real-World Benchmark Problem

In this Section, we detail a comparison between R-Wasps-D and Morley's system on the real-world problem instance for which Morley's system was originally designed and implemented. The problem is described in (Morley & Schelberg, 1993; Morley, 1996).

In Morley's problem, trucks rolled off the assembly line at a rate of one per minute. The system was faced with the problem of assigning each truck to a paint booth as it emerged from the end of the assembly line. There were seven paint booths in Morley's problem and it took three minutes to paint a truck. Each truck could possibly require any of fourteen paint colors and the trucks arrived in no particular order. Approximately 50% of the trucks required a single color. The other 50% required colors drawn uniformly at random from among the other 13 colors. A paint booth could only be set for one color at a time and there was a cost to reconfigure the booth for another color in terms of both the time it required to perform this color change as well as a monetary cost associated with paint usage. They gave no details regarding the monetary cost of such a change so we do not consider that objective here. There was a further constraint that the queue of a paint booth could have at most 3 trucks. Presumably, this constraint was due to physical space limitations in the factory. In any case, it is a very realistic constraint and characteristic of real-world problems.

Our implementation of this problem has seven machines (one for each paint booth) and fourteen job types (one for each paint color). Five simulation time units is equal to one simulated minute. Jobs arrive at a rate of one every five time units (one simulated minute).

With probability 0.5, a job is of type T1 (requires color C1). A job has a probability of $\frac{1}{26}$ of being type T2 (similarly for types T3, . . ., T14). The processing time of a job is equal to fifteen time units (three simulated minutes). Setup time to reconfigure a machine from one type to another was left somewhat vague in Morley's original definition of the problem. We here consider two alternatives for setup time:

1. Hard Problem: very significant setup time – setup time of 50 time units (ten simulated minutes)

2. Easier Problem: less significant setup time – setup time of 5 time units (one simulated minute)

We consider each of these problems separately below. The entire simulation runs for 5000 time units (1000 simulated minutes). Morley didn't specify the length of a simulation.

To handle the queue length limit of three jobs, we add a new condition to the rule used by the routing wasps in determining whether or not to respond to a stimulus from a job. If a queue is full (contains three jobs), the routing wasp in charge of that queue does not respond to any job stimulus. Otherwise, it uses the stochastic rule described in Section 3. The system parameters for R-Wasps-D for this problem have been set as follows[4]:

- $\theta_{min} = 1$
- $\theta_{max} = 10000$
- $\delta_1 = 100$
- $\delta_2 = 10$
- $\delta_3 = 1.05$

For Morley's system, the queue length limit has an interesting effect. Although Morley's system is formulated and implemented as a simple bidding mechanism, the short queue length simplifies that bidding mechanism to the following three rules considered in this order (as described in (Morley & Schelberg, 1993)):

1. Assign the job to the machine with the shortest queue (with space) whose last job is of the same type as this next job (if such a machine exists).

2. Assign the job to a machine with an empty queue if such a machine exists.

[4]Our experience is that these system parameters do require tuning according to problem structure (i.e., number of machines and job types). This is discussed further in Section 7 and is a topic of ongoing study.

Table 4: Hard Problem: Throughput comparison of R-Wasps-D and Morley's system on Morley's problem. 95% confidence intervals and two-tailed p-values from a paired *t*-test are shown.

	Morley	R-Wasps-D	p-value
Average	873.69±20.07	**972.22±2.11**	<0.0001
Median	876	974	–
Low	678	934	–
High	993	991	–

Table 5: Hard Problem: Comparison of average cycle time of R-Wasps-D and Morley's system on Morley's problem. The numbers are in simulated minutes. 95% confidence intervals and two-tailed p-values from a paired *t*-test are shown.

Morley	R-Wasps-D	p-value
42.37±6.48	**26.72±1.08**	<0.0001

3. Assign the job to a machine with the shortest queue (with space) if such a machine exists.

Rule 2 is somewhat redundant and is actually encapsulated in rule 3 but is included here to remain consistent to Morley (Morley & Schelberg, 1993).

Let us first examine the "hard problem" (i.e., very significant setup time). Table 4 shows the throughput results of an experimental comparison of R-Wasps-D and Morley's system. Table 5 shows the average cycle time for the problem for both R-Wasps-D and Morley's system. The data is based on 100 runs of a simulation of the problem as described above. You will first notice that R-Wasps-D produces far greater throughput than does Morley's system. And this result is statistically significant according to a *t*-test. In addition, the results of R-Wasps-D range in the interval $[934, 991]$; whereas the results of Morley's system range in the interval $[678, 993]$. Morley's system seems to perform rather inconsistently from one run to the next; whereas R-Wasps-D exhibits rather consistent performance. Also, you should note that the average cycle time in R-Wasps-D is significantly shorter than that or Morley's system. So jobs take less time to get through the system. This performance difference is likely due to the routing wasps ability to specialize in a particular job type or to a few job types. When a job of one of the thirteen relatively infrequent job types arrives in Morley's system, there is a good chance that none of the queues have a job of this type at the end. The result will be that this job is assigned to the shortest queue with space. Our system, R-Wasps-D, on the other hand, allows one or more of the machines to specialize in the job type of high demand. So for example, let's consider that two or three of the ma-

Table 6: Hard Problem: Comparison of average number of setups of R-Wasps-D and Morley's system on Morley's problem. Smaller numbers are better. 95% confidence intervals and two-tailed p-values from a paired *t*-test are shown.

Morley	R-Wasps-D	p-value
406.13±9.54	**265.33±6.08**	<0.0001

chines have specialized in this single job type of high demand and are unwilling to accept any of the other job types. Now when this job of one of the less frequent types arrives, even if the shortest queue is one of these two or three specialized machines, the system will route it elsewhere. After all, if a setup is going to be accrued anyway and if there is such a high demand for the job type these machines have specialized to and there is a good chance another job of this highly demanded type will arrive soon, then it makes perfect sense to route this less demanded job away from the specialized machines. Otherwise you are sure to accrue two setups rather than just a single setup. The fundamental difference between our system and Morley's (aside from the stochastic nature of our routing wasps) is that Morley's paint booth agents are required to bid as long as there is queue space; whereas any of our routing wasps can choose to ignore a job stimulus and remain specialized even if there is space in its queue.

In terms of Morley's problem, an equally important (or perhaps more important) objective than maximizing throughput and minimizing cycle time is that of minimizing the number of setups. During a color change, there is a chance that the paint system might not be fully flushed of the previous color paint. Such a failure may result in a truck receiving a bad coat of paint. Failed paint jobs such as these need to re-enter the system and receive an additional paint coat to fix the problem (Bickford, 2001). These additional coats add up to a monetary cost. This monetary cost was of interest to Morley. He supplied no details as to the specifics of this objective. But minimizing the number of setups encapsulates the goal of this objective. In Table 6, we see a comparison of the average number of setups over the 100 runs of the simulation. R-Wasps-D requires significantly less machine setups than does Morley's system and therefore results in a smaller number of expected single job failures.

Now turn to the results of the "easier problem" (i.e., less significant setup time) in Table 7. You will first notice that in terms of throughput and cycle time, it appears that Morley's system performs better than does R-Wasps-D. But, in terms of the number of setups performed on average, R-Wasps-D performs significantly fewer setups. As stated above, the more important objective in this problem is minimizing the number of setups. In an actual paintshop, every time a machine is reconfigured for a different paint color, there is a chance that the next job will fail in some manner and require rerouting back into the

Table 7: Easier Problem: Setup time is less significant in this problem. 95% confidence intervals and two-tailed p-values from a paired *t*-test are shown.

	Morley	R-Wasps-D	p-value
Throughput	**997.09±0.21**	994.03±0.36	<0.0001
Cycle Time	**3.87±0.03**	7.16±0.10	<0.0001
Number of Setups	438.22±3.70	**287.61±2.15**	<0.0001

incoming buffer of jobs. These failures have not been modeled in the simulations presented in this paper, but if they had been we suspect the throughput and cycle time numbers would be affected. And perhaps more importantly, under the assumption that the number of setups is indicative of the extra cost associated with additional paint usage due to such single job failures, it is clear from the comparison of the number of setups that R-Wasps-D accrues significantly less paint costs. In fact, assuming a linear relationship between number of setups and such paint costs, R-Wasps-D shows a savings of 35% over Morley's system on this problem.

A natural question to ask at this point is "why with less setups we are not seeing smaller average cycle times?" There are a couple of reasons for this behavior. The first is that R-Wasps-D through its response threshold adaptation is able to specialize a couple of machines to the more highly demanded job type, so jobs of lesser demand will tend to be assigned to the queues of other machines not specialized to this high demand type even if these queues are longer. A second reason is that through the ability of machines to "not bid," R-Wasps-D is able to put off assignment of arriving jobs for a few time units to allow for better sequencing. In Morley's system, such delayed job assignment is not allowed (i.e., all machines bid provided there is space in their queues); while R-Wasps-D use a stochastic rule to choose whether or not to bid at all even in cases where there is space in the queue. In this "easier" problem with only one minute of setup time, the system in the absolute worst case of every job requiring setup can process seven jobs every four minutes on average while only four jobs are arriving during that four minutes. With approximately 43.9% of its jobs requiring setup, Morley's system is somewhat near this worst case. But R-Wasps-D, utilizes this difference in production capacity and job arrival rates; and through machine specialization and delayed job assignments, R-Wasps-D is better able to sequence the jobs in queues as to minimize number of setups with only approximately 28.9% of jobs requiring setup.

7 Limitations

Rate of Adaptation. Our routing wasps require some amount of time to adapt to the product mix as well as to re-adapt to changing product demands. In the experiments of Section 6, this adaptation time appeared to be somewhat serious with respect to the changing job mix problem. It is not quite as serious as it at first appears. If you recall, the routing wasps did not appear to suffer during adaptation time in the real-world problem of Section 6.2. The reason we did not observe poor performance during adaptation time in this problem was the queue limit constraint. In Section 6, there was no queue limit and during the initial adaptation to the 85/15 mix and the re-adaptation in the changing mix one of the machines ends up over-specializing and building up a large queue size until the other machine adapts its response thresholds appropriately. The queue limit constraint prevents such drastic over-specialization. This constraint is also realistic and characteristic of real-world problems. However, although this constraint is realistic, we wish to overcome the rate of adaptation limitation in the event of dealing with a system with no such queue length limit constraint (or a very large limit). There are three approaches we are exploring for dealing with this problem: 1) maintaining some statistics on the job mix and boosting the values of the learning parameters δ_i if a large enough job mix change has been detected; 2) incorporating an additional response threshold update rule that is triggered when a machine is idle and has an empty queue and its routing wasp receives a stimulus from a job but does not respond; and 3) incorporating queue length into the stochastic rule used by the routing wasps in determining whether or not to accept a job. The first of these potential solutions is domain specific; whereas the second and third solutions appear to be more generally applicable to other problems as well as a more cohesive fit with the underlying model. For these reasons, our preferred solution will be to follow one of these latter approaches.

Parameter Tuning. The response threshold formulation involves a number of parameters. The experiments presented in this paper used parameter values that were tuned by hand. This hand-tuning was not very systematic and settings that appeared to perform well were used. But there is no reason to believe that these parameter settings are the best settings for all problem instances and all factory configurations. In fact, it is doubtful that there exists one set of parameter values that are the "best" for all possible factory configurations. We plan on performing an extensive analysis of the effects of the various control parameters (i.e., the δ_i) with respect to various factory configurations. The goal of such an analysis is to be able to define guidelines for appropriate settings with respect to such system characteristics as number of machines, average processing time, average setup time, etc.

8 Related Work

There has been much research in recent years into distributed and agent-based approaches to scheduling problems. One agent coordination paradigm that appears to be quite popular among the researchers in this area is that of artificial market systems. In the various market-based and auction approaches that appear in the literature, agents representing resources use bidding mechanisms of one sort or another to coordinate the assignment of required resources to tasks or jobs (e.g., (Collins, Tsvetovatyy, Mobasher, & Gini, 1998; Goldsmith & Interrante, 1998; Fischer, Müller, Pischel, & Schier, 1995; Kuwabara & Ishida, 1992; Lin & Solberg, 1992; Morley & Schelberg, 1993; Morley, 1996; Rabelo & Camarinha-Matos, 1994; Walsh, Wellman, Wurman, & MacKie-Mason, 1998; Wellman, 1992)). There is evidence that market-based approaches can produce globally optimal or near-optimal solutions. For example, Walsh et al. (Walsh et al., 1998) show that for discrete resource allocation problems an equilibrium solution is an optimal solution. However, this characteristic of auction-based approaches is not a cure-all. The problem is that equilibrium solutions do not always exist; and often when they do exist, the problem of finding the equilibrium is NP-hard. In any case, bidding mechanisms appear to be fairly closely related to the stimulus-response mechanism of the wasp behavior model as pointed out in (Bonabeau et al., 1999). That is, high bids correspond to low response thresholds; and low bids correspond to high response thresholds. However, the relation of bidding mechanisms to the wasp model goes beyond the simple analogy made by Bonabeau et al. If coupled with a market-based system, the stimulus-response mechanism of wasps can impart on the agents the ability to simply not bid and to ignore a "call for bids" if the stimulus is sufficiently below the response threshold even if it is capable of performing the requested task. More importantly, it allows for a mechanism for the adaptation of a decision policy regarding when to bid and when to not bid.

Another distributed scheduling approach that has been gaining in popularity in the evolutionary computation community in recent years builds from the Ant Colony Optimization (ACO) metaheuristic of Dorigo et al. (Dorigo & Di Caro, 1999; Dorigo & Gambardella, 1997a, 1997b). ACO uses a population of ant-like agents that communicate indirectly via trail laying and following to build solutions to the scheduling problem at hand. There have been numerous applications of ACO to various scheduling problems including: the sequential ordering problem (Gambardella & Dorigo, 1997), job shop scheduling (van der Zwaan & Marques, 1999), flow shop scheduling (Stützle, 1998), vehicle routing (Bullnheimer, Hartl, & Strauss, 1999; Gambardella, Taillard, & Agazzi, 1999), bus driver scheduling (Forsyth & Wren, 1997), tardiness scheduling problems (Bauer, Bullnheimer, Hartl, & Strauss, 1999; den Besten, Stützle, & Dorigo, 2000), and resource-constrained project scheduling (Merkle, Middendorf, & Schmeck, 2000). However, al-

though all of these applications of ACO to scheduling problems use a population of agents to solve their respective problems, all of these systems build their solutions in advance. So although their solutions are computed in a distributed manner, they deal with a "static" problem rather than the dynamically changing problem that we have considered in this paper. Based on the ACO paradigm, our previous attempt at a distributed solution to this dynamic problem was that of AC^2 (Cicirello & Smith, 2001a). However, it suffered from frequent convergence to sub-optimal equilibrium in a game-theoretic sense (Cicirello, 2001). A more successful application of ACO to a dynamically changing problem has been to the problem of network routing. Schoonderwoerd et al. (Schoonderwoerd et al., 1997a; Schoonderwoerd, Holland, Bruten, & Rothkrantz, 1997b) have developed an effective system called Ant Based Control (ABC) for adapting routing tables in circuit-switched networks based on the ACO framework. Similarly, Di Caro and Dorigo (Di Caro & Dorigo, 1998) have developed a system called AntNet in which artificial ants adapt the routing tables of packet-switched networks.

Another aspect of our research not discussed in this paper but closely related is a framework for the randomization of dispatch scheduling heuristics based on the model of wasp social hierarchy formation (see (Cicirello & Smith, 2001e, 2001d)). This framework uses a population of what we call scheduling wasps that interact with each other to prioritize jobs waiting in a queue. The result is an effective stochastic mechanism for amplifying the performance of dispatch scheduling policies in cases when the deterministic policy is less informed. For hard instances of the dynamic scheduling objective of minimizing weighted tardiness under sequence-dependent setup constraints, our stochastic framework of the scheduling wasps is superior to state-of-the-art deterministic dispatch policies for the problem (Cicirello & Smith, 2001d).

9 Conclusion

In this paper we have presented an adaptive multi-agent system for dynamic factory routing based on various aspects of a computational model of the adaptive behavior observed in wasp colonies. In our system, routing activities are performed by computational agents called routing wasps in a manner analogous to task allocation among real wasps. In the event more than one routing wasp attempts to route the same job to their respective machines, our system employs a model of self-organized social hierarchies within wasp colonies to decide among the competing routing wasps via a tournament of dominance contests.

For simple factory configurations, we demonstrated that our routing wasps adapt behavior corresponding to what intuitively is the right thing to do. We have also demonstrated

that our system is robust and is capable of efficiently adapting to dynamically changing and uncertain environments. Plots of the response thresholds over time for these simple configurations aided us in the analysis. We further showed for these simple configurations that our system is competitive with (and in many cases superior to) a previously successful and industry-proven agent-based system for the problem. Furthermore, when we scaled the problem up to a problem of real-world complexity, we showed that our system far outperformed the benchmark system of Morley in a simulation of the GM paintshop for which his system was originally designed.

In the future we plan to extend the experimentation of our system to problem environments that consist of uncertain events. For example, one type of uncertain event would be that of unexpected machine failures. One possible way of modeling such machine failures would be with a "mean time to failure." Another less structured model might consist of a simple probability at each time-step for failure. We will explore our system's performance under a number of such models. Another type of uncertainty we wish to include in our future experimentation is that of single job failures. One example of such a failure may be if something goes wrong during setup and the wrong operation is applied to a job resulting in a "rework" (i.e., the job having to re-enter the system at some previous point to correct the error). For instance, this type of failure is characteristic of vehicle-painting plants. If the paint system is not fully flushed of the previous color paint during setup, a truck may receive an incorrect coat of paint and require an additional coat to correct the error. We feel that our system will be robust to such uncertainties in the problem environment.

More generally, we believe that our multi-agent model offers a flexible, decomposable approach to coordinating material flows to meet changing demands and other dynamic constraints. As such, it should also be naturally applicable to more global supply-chain coordination problems. With continuing trends toward specialization on core competencies, manufacturing organizations must rely increasingly on coupling their respective capabilities and partnering to capitalize on new market opportunities, and the ability to rapidly and dynamically reconfigure supply chains becomes increasingly important.

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

Bauer, A., Bullnheimer, B., Hartl, R. F., & Strauss, C. (1999). An ant colony optimization approach for the single machine total tardiness problem. In *CEC99: Proceedings of the Congress on Evolutionary Computation*, pp. 1445–1450.

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

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

Bickford, T. R. (2001). Personal communication. Agent Institute, The University of Maine.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (2000). Inspiration for optimization from social insect behaviour. *Nature*, *406*, 39–42.

Bonabeau, E., Sobkowski, A., Theraulaz, G., & Deneubourg, J. L. (1997). Adaptive task allocation inspired by a model of division of labor in social insects. In Lundh, D., & Olsson, B. (Eds.), *Bio Computation and Emergent Computing*, pp. 36–45. World Scientific.

Bonabeau, E., Theraulaz, G., & Deneubourg, J. L. (1998). Fixed response thresholds and the regulation of division of labor in insect societies. *Bulletin of Mathematical Biology*, *60*, 753–807.

Bullnheimer, B., Hartl, R. F., & Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, *89*, 319–328.

Cicirello, V. A. (2001). A game-theoretic analysis of multi-agent systems for shop floor routing. Tech. rep., The Robotics Institute, Carnegie Mellon University.

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

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

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

Cicirello, V. A., & Smith, S. F. (2001d). Randomizing dispatch scheduling heuristics. In *The AAAI Fall Symposium: Using Uncertainty within Computation*.

Cicirello, V. A., & Smith, S. F. (2001e). Wasp nests for self-configurable factories. In *Agents 2001, Proceedings of the Fifth International Conference on Autonomous Agents*. ACM Press.

Collins, J., Tsvetovatyy, M., Mobasher, B., & Gini, M. (1998). MAGNET: a multi-agent contracting system for plan execution. In Luger, G. F. (Ed.), *Proceedings of the Artificial Intelligence and Manufacturing Research Planning Workshop*, pp. 63–68. AAAI Press.

den Besten, M., Stützle, T., & Dorigo, M. (2000). Ant colony optimization for the total weighted tardiness problem. In *Proceedings of the Parallel Problem Solving from Nature Conference*.

Di Caro, G., & Dorigo, M. (1998). AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research, 9*, 317–365.

Dorigo, M., & Di Caro, G. (1999). The ant colony optimization meta-heuristic. In Corne, D., Dorigo, M., & Glover, F. (Eds.), *New Ideas in Optimization*. McGraw-Hill.

Dorigo, M., & Gambardella, L. M. (1997a). Ant colonies for the traveling salesman problem. *BioSystems, 43*, 73–89.

Dorigo, M., & Gambardella, L. M. (1997b). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*(1), 53–66.

Fischer, K., Müller, J. P., Pischel, M., & Schier, D. (1995). A model for cooperative transportation scheduling. In *ICMAS-95: Proceedings of the First International Conference on Multi-Agent Systems*, pp. 109–116. AAAI Press / The MIT Press.

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

Forsyth, P., & Wren, A. (1997). An ant system for bus driver scheduling. Technical report 97.25, University of Leeds, School of Computer Studies. Presented at the 7th International Workshop on Computer-Aided Scheduling of Public Transport, Boston, July 1997.

Gambardella, L. M., & Dorigo, M. (1997). HAS-SOP: Hybrid ant system for the sequential ordering problem. Technical report IDSIA 97-11, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Lugano, Switzerland.

Gambardella, L. M., Taillard, E., & Agazzi, G. (1999). MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. Technical report IDSIA-06-99, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Lugano, Switzerland.

Goldsmith, S. Y., & Interrante, L. D. (1998). An autonomous manufacturing collective for job shop scheduling. In Luger, G. F. (Ed.), *Proceedings of the Artificial Intelligence and Manufacturing Research Planning Workshop*, pp. 69–74. AAAI Press.

Gordon, D. M. (1996). The organization of work in social insect colonies. *Nature, 380*, 121–124.

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

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

Kuwabara, K., & Ishida, T. (1992). Equilibratory approach to distributed resource allocation: Toward coordinated balancing. In Castelfranchi, C., & Werner, E. (Eds.), *Artificial Social Systems: 4th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (Selected Papers)*, pp. 133–146. Springer-Verlag.

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

Liu, J. S. (1996). *Coordination of Multiple Agents in Distributed Manufacturing Scheduling*. Ph.D. thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Merkle, D., Middendorf, M., & Schmeck, H. (2000). Ant colony optimization for resource-constrained project scheduling. In *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 893–900. Morgan Kaufmann.

Morley, D. (1996). Painting trucks at general motors: The effectiveness of a complexity-based approach. In *Embracing Complexity: Exploring the Application of Complex Adaptive Systems to Business*, pp. 53–58. The Ernst and Young Center for Business Innovation.

Morley, D., & Schelberg, C. (1993). An analysis of a plant-specific dynamic scheduler. In *Final Report, Intelligent Dynamic Scheduling for Manufacturing Systems*, pp. 115–122.

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

Ow, P. S., Smith, S. F., & Howie, R. (1988). A cooperative scheduling system. In Oliff, M. D. (Ed.), *Expert Systems and Intelligent Manufacturing*, pp. 43–56. Elsevier Science Publishing Co., Inc.

Parunak, V., Baker, A., & Clark, S. (1998). The AARIA agent architecture: From manufacturing requirements to agent-based system design. In *Proceedings of the ICAA'98 Workshop on Agent-Based Manufacturing*.

Rabelo, R. J., & Camarinha-Matos, L. M. (1994). Negotiation in multi-agent based dynamic scheduling. *Robotics and Computer-Integrated Manufacturing*, *11*(4), 303–309.

Schoonderwoerd, R., Holland, O., & Bruten, J. (1997a). Ant-like agents for load balancing in telecommunications networks. In *Agents '97, Proceedings of the First International Conference on Autonomous Agents*, pp. 209–216. ACM Press.

Schoonderwoerd, R., Holland, O., Bruten, J., & Rothkrantz, L. (1997b). Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, *5*(2), 169–207.

Stützle, T. (1998). An ant approach to the flow shop problem. In *Proceedings of EUFIT'98*, pp. 1560–1564.

Sycara, K. P., Roth, S. F., Sadeh, N., & Fox, M. S. (1991). Resource allocation in distributed factory scheduling. *IEEE Expert*, *6*(1), 29–40.

Theraulaz, G., Bonabeau, E., & Deneubourg, J. L. (1995). Self-organization of hierarchies in animal societies: The case of the primitively eusocial wasp polistes dominulus christ. *Journal of Theoretical Biology*, *174*, 313–323.

Theraulaz, G., Bonabeau, E., & Deneubourg, J. L. (1998). Response threshold reinforcement and division of labour in insect societies. *Proceedings of the Royal Society of London B*, *265*(1393), 327–335.

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

van der Zwaan, S., & Marques, C. (1999). Ant colony optimisation for job shop scheduling. In *Proceedings of the '99 Workshop on Genetic Algorithms and Artficial Life GAAL'99*.

Walsh, W. E., Wellman, M. P., Wurman, P. R., & MacKie-Mason, J. K. (1998). Some economics of market-based distributed scheduling. In *Proceedings of the Eighteenth International Conference on Distributed Computing Systems*, pp. 612–621.

Wellman, M. (1992). A general-equilibrium approach to distributed transportation planning. In *AAAI-92: Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 282–289. AAAI Press / The MIT Press.