Distributed Coordination of Resources
via Wasp-Like Agents

Vincent A. Cicirello and Stephen F. Smith

The Robotics Institute
Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213
{cicirello,sfs}@cs.cmu.edu

Abstract. Agent-based approaches to scheduling have gained increasing attention in recent years. One inherent advantage of agent-based approaches is their tendency for robust behavior; since activity is coordinated via local interaction protocols and decision policies, the system is insensitive to unpredictability in the executing environment. At the same time, such "self-scheduling" systems presume that a coherent global behavior will emerge from the local interactions of individual agents, and realizing this behavior remains a difficult problem. We draw on the adaptive behavior of the natural multi-agent system of the wasp colony as inspiration for decentralized mechanisms for coordinating factory operations. We compare the resulting systems to the state-of-the-art for the problems examined.

1   Introduction

Distributed, agent-based approaches to scheduling have gained increasing attention in recent years. One inherent advantage of agent-based approaches is their tendency for robust behavior; since activity is coordinated via local interaction protocols and decision policies, the system is insensitive to unpredictability in the executing environment. At the same time, such "self-scheduling" systems presume that a coherent global behavior will emerge from the local interactions of individual agents, and realizing this behavior remains a difficult problem.

To address this problem, we have taken a view of multi-agent resource coordination as an adaptive process, and have been investigating adaptive mechanisms inspired by natural multi-agent systems. Theraulaz et al. (see [13], [2], [12], [1], [11]) have developed a computational model of the adaptive behavior of a colony of wasps. They model two aspects of wasp behavior: 1) self-coordinated task allocation and 2) self-organized social hierarchies. Both of these have proved to provide useful bases upon which to build multi-agent coordination mechanisms.

The first aspect of wasp behavior that we have utilized is the stimulus-response mechanism which they use to perform task allocation. Wasps have response thresholds for the various tasks required of the colony such as foraging and brood care. The magnitude of a particular wasp's response thresholds to these

tasks determines the likelihood of that wasp's engagement in a task given environmental stimuli for the task. We have applied this model in a dynamic factory setting for the assignment of jobs to machines (see [6], [3]). The machines in our model are multi-purpose and we consider the constraint of sequence-dependent setup (i.e., our machines may be capable of performing more than one task but there is a cost associated with reconfiguration). As a new job arrives at the factory, it emits a stimulus and continues to do so at increasing magnitudes. Each machine is represented by a wasp-like agent which we call a *routing wasp*. Such a routing wasp maintains response thresholds for the various tasks its machine is able to perform and responds to job stimuli stochastically according to the magnitude of these thresholds as well as the magnitude of the stimuli. Response thresholds adapt to account for the current product demand.

The second aspect of wasp behavior of interest to us is their self-organizing social hierarchy. When two wasps encounter each other in the nest, they may with some probability engage in a dominance interaction. The wasp with the higher "force variable" wins the contest with a higher probability. Its force variable is then increased and the force variable of the loser is similarly decreased. A dominance hierarchy emerges from such interactions. We have shown this model of dominance contests to provide an effective basis for randomizing dispatch scheduling policies (see [6], [4]). In this work, we associate an agent which we call a *scheduling wasp* with each job in the queue of a given resource. The force variable of the scheduling wasp is defined by a dispatch heuristic (and hence can be chosen to match the characteristics of the problem at hand). Scheduling wasps engage in tournaments of dominance contests to stochastically prioritize the jobs in the queue. When the resource becomes available, the job represented by the current most dominant wasp is processed.

In this paper, we summarize initial results obtained with both the routing wasp and scheduling wasp coordination models. Section 2 details our routing wasp formulation. In Section 2.2 we benchmark our routing wasps on the real-world problem of scheduling a paintshop. Section 3 details our scheduling wasp framework. In Section 3.2 we compare our scheduling wasp framework to a state-of-the-art deterministic dispatch policy for the problem of weighted tardiness scheduling under a sequence dependent setup constraint. Finally, in Section 4 we conclude.

2   Routing Wasps

2.1   Formulation

Our routing wasp 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
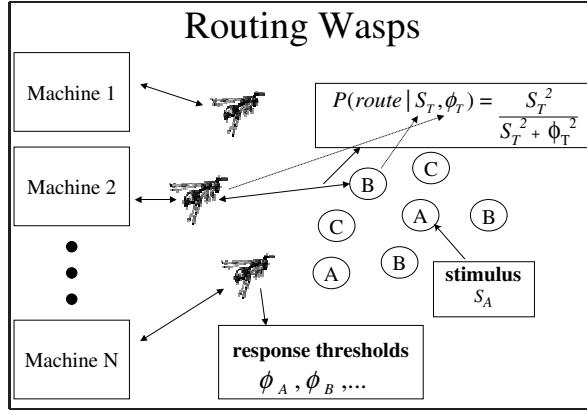
Fig. 1. Routing wasps

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 are awaiting machine assignment 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 routed and where j is the type of job. 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 [12]. 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 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):

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

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 [1, 12]. 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.

We now need a method to reconcile a competition between two or more routing wasps that are interested in routing a single job to their respective machines. The method we employ 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[1]. 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.

---

[1] 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.

Table 1. Comparison of average number of setups of R-Wasps (routing wasps) and Morley's system. Smaller numbers are better. 95% confidence intervals and two-tailed p-values from a paired T-test is shown. Result is average of 100 runs.

Morley	R-Wasps	p-value
438.22±3.70	**287.61±2.15**	<0.0001

2.2   Paintshop Problem

To benchmark the performance of our routing wasp framework, we conducted experiments comparing it to Morley's GM Paintshop system (see [8], [7]). 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 [8] and resulted in savings of nearly a million dollars in the first nine months of use [7]. 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 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.

Table 1 shows how the routing wasps compare to Morley's system on this problem[2]. Shown in the table is the average number of setups performed during the course of a simulation over 100 runs. Every setup (switching of paint color) requires some amount of time. Also, there is some chance that the system will fail to flush all of the previous paint color from the system during setup. This can

---

[2] More detailed results can be found in [5].

result in an incorrect coat of paint applied to the next truck after the setup. This increases paintcosts, requiring a rework of the truck. It also affects cycle time. To this end, minimizing the number of setups should be our objective in this problem and as can be clearly seen, the routing wasps is significantly superior to Morley's system in this regard.

3   Scheduling Wasps

3.1   Formulation

Our scheduling wasp model is the result of recognition of the fallibility of dispatch scheduling policies. If dispatch policies are viewed as good rules of thumb and if the deterministic use of such a policy leads to a schedule that lies in close proximity to many good schedules, then perhaps a better solution can be obtained by randomizing the decision-making process in some way biased by the heuristic. Our approach to randomization derives from a naturally-inspired computational model of the self-organization that takes place within a colony of wasps (see [13], [11], [1]). In nature, 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 (analogous to a prioritization of jobs on a set of machines). In the model of Theraulaz et al., the results of these interactions are determined stochastically based on the "force" variables of the wasps involved. The probability of wasp 1 winning a dominance contest against wasp 2 is defined based on the force variables, F_1 and F_2, of the wasps as:

$$P(F_1, F_2) = \frac{F_1^2}{F_1^2 + F_2^2} \qquad (8)$$

This model can be directly mapped to the problem of prioritizing jobs in a queue, and as such provides a natural basis for the randomization of dispatch policies. In our "scheduling wasp" formulation, each job is represented by a wasp and the concept of a force variable is used to define job priority (i.e., the value that the dispatch policy in use assigns). The scheduling wasps then interact with each other to prioritize the jobs in the queue. This framework for dynamic scheduling was first introduced in [6], where we considered the problem of sequencing jobs to maximize throughput under different and dynamically changing job mixes. Here, as in [4], we explore the use of this wasp model on due date problems, where dispatch-based solutions are more commonly employed.

To fully specify our scheduling wasp model for minimizing weighted tardiness with sequence dependent setups, we need to provide a definition of force. We will define force with the dispatch policy known as R&M [9]. This dispatch policy was not originally designed with sequence-dependent setups in mind. We have here modified it to account for setup time. This modification of R&M was made originally in [10].
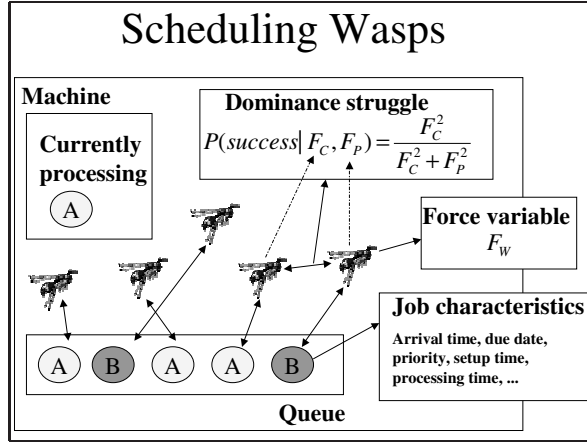
Fig. 2. Scheduling wasps

Noting this, force is defined as:

$$F_w = \frac{W_w}{T_w^p + T_w^s} \, \exp\left(\frac{-(D_w - T_w^p - T_w^s - T_{\text{now}})^+}{h\,\overline{T^p}}\right) \qquad (9)$$

where T_w^p and T_w^s are the processing time and setup time of wasp w's job, D_w is the duedate, W_w is the weight, T_{now} is the current time, $\overline{T^p}$ is the average processing time, and $(A)^+ = \max\{A, 0\}$. The winner of a dominance contest in this context is determined stochastically in the same manner as in the model of real wasp behavior.

In the typical dispatch scheduling approach, the job in the queue with the highest value of the dispatch heuristic is chosen next. Given the scheduling wasp formulation of the previous section, our system instead chooses the next job based on a tournament of dominance contests. In this tournament, the scheduling wasps are seeded based on their current position in the queue. The last two wasps in the queue engage in a dominance contest. The winner then engages in a dominance contest with the next wasp and so forth along the length of the queue. As this occurs, the jobs associated with the winning wasps move closer to the front of the queue. Whatever job is at the front of the queue when this process completes is chosen next by the machine.

3.2   Weighted Tardiness Problem

To analyze our scheduling wasp framework for randomizing dispatch policies, we have examined its performance on weighted tardiness problems under sequence dependent setup constraints. As stated earlier, there are only a few dispatch policies in the literature for this problem, all of which are modifications of policies that do not consider setups. We have taken R&M as defined earlier as the

Table 2. Average weighted tardiness comparison between S-Wasps (scheduling wasp stochastic framework) and deterministic dispatch scheduling (R&M). The problem has 1 machine and either 2 or 3 job types for various job mixes. 95% confidence intervals and two-tailed p-values from paired T-tests are shown.

Mix	S-Wasps	R&M	P-value
50/50	**2081.4±219.9**	2637.8±288.9	<0.0001
85/15	699.2±108.7	**660.7±115.1**	0.1713
100/0	214.3±31.0	**172.5±23.8**	<0.0001
33/33 /33	**2834.5±217.1**	3307.4±274.8	<0.0001
50/25 /25	**2582.4±226.1**	2912.6±299.3	0.0021

definition of the force variable for our scheduling wasps and have compared the stochastic policy that results to the deterministic policy.

Table 2 shows the results of this comparison[3]. The problem in question consists of a single machine and either two or three types of jobs with unknown arrival times. We consider 5 different job mixes as seen in the table. Processing time is chosen randomly for each job from a Gaussian centered at 15 time units. Setup time to switch a machine from one type to another is 30 time units. A job's weight is drawn uniformly from the interval $[1, 20]$, and its duedate is drawn uniformly from one of the following intervals (where P is process time, W is weight, and T is current time):

- $[T, T + 4\,P]$ if $W > 16$
- $[T, T + 6\,P]$ if $12 < W \leq 16$
- $[T, T + 6.5\,P]$ if $8 < W \leq 12$
- $[T, T + 8\,P]$ if $W \leq 8$

Simulations are 1000 time units in length and results shown in the table are averages of 100 runs.

What can be seen in Table 2 is that problems of a single job type or very nearly a single job type problem are best solved with the deterministic policy (the 100/0 job mix and the 85/15 job mix problems). A single job type problem is what the dispatch policy in question was originally designed for. However, you can also see in the table that the "harder" problems with more diverse mixes of jobs are best solved with the stochastic policy of the scheduling wasps.

4   Conclusion

In this paper, we have presented mechanisms for coordinating factory operations in a decentralized manner inspired by the natural self-organization that takes place within a colony of wasps. The routing wasp framework provides superior

---

[3] More detailed results can be found in [4].

performance to a real-world proven system for sequence-dependent setup problems. The scheduling wasp framework proves an effective stochastic framework for randomizing dispatch scheduling policies in instances where they are less informed.

We believe the computational mechanisms underlying these wasp behavioral models have broader applicability to task allocation and resource coordination in other multi-agent domains. We are currently pursuing application of these models to two distinct types of problems. First, we are exploring generalization from the dynamic factory routing and scheduling problem to the broader-scoped problem of supply chain management. Just as factories must reconfigure product flows as the mix of jobs changes over time, supply chains must realign material flow to capitalize on new market opportunities, and current trends toward specialization and increased partnering increase the need for good distributed solutions. Second, we are also considering applicability of our "self-scheduling" models to the somewhat different problem of multi-robot coordination. We are interested in domains such as space exploration and hazardous waste cleanup, where multiple robots with differential capabilities must dynamically configure themselves into teams and cooperate in the performance of complex tasks over time.

Acknowledgments

References

[1] E. Bonabeau, A. Sobkowski, G. Theraulaz, and J. L. Deneubourg. Adaptive task allocation inspired by a model of division of labor in social insects. In D. Lundh and B. Olsson, editors, *Bio Computation and Emergent Computing*, pages 36–45. World Scientific, 1997. 71, 74, 76

[2] E. Bonabeau, G. Theraulaz, and J. L. Deneubourg. Fixed response thresholds and the regulation of division of labor in insect societies. *Bulletin of Mathematical Biology*, 60:753–807, 1998. 71

[3] Vincent A. Cicirello and Stephen F. Smith. Improved routing wasps for distributed factory control. In *The IJCAI-01 Workshop on Artificial Intelligence and Manufacturing, Working Notes*, pages 26–32. AAAI SIGMAN, 4-9 August 2001. Seattle, WA. 72

[4] Vincent A. Cicirello and Stephen F. Smith. Randomizing dispatch scheduling poli-
cies. In *Using Uncertainty Within Computation: Papers from the 2001 AAAI Fall
Symposium, Technical Report FS-01-04*, pages 30–37. AAAI Press, 2-4 November
2001. North Falmouth, Massachusetts.   72, 76, 78

[5] Vincent A. Cicirello and Stephen F. Smith. Wasp-like agents for distributed
factory coordination. Technical Report CMU-RI-TR-01-39, Robotics Institute,
Carnegie Mellon University, Pittsburgh, PA, December 2001.   75

[6] Vincent A. Cicirello and Stephen F. Smith. Wasp nests for self-configurable fac-
tories. In J. P. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of
the Fifth International Conference on Autonomous Agents*, pages 473–480. ACM
SIGART, ACM/SIGGRAPH, ACM/SIGCHI, ACM Press, May-June 2001. Mon-
treal, Quebec, Canada.   72, 76

[7] D. Morley. Painting trucks at general motors: The effectiveness of a complexity-
based approach. In *Embracing Complexity: Exploring the Application of Complex
Adaptive Systems to Business*, pages 53–58. The Ernst and Young Center for
Business Innovation, 1996.   75

[8] 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.   75

[9] R. V. Rachamadugu and T. E. Morton. Myopic heuristics for the single machine
weighted tardiness problem. Working Paper 30-82-83, GSIA, Carnegie Mellon
University, Pittsburgh, PA, 1982.   76

[10] N. Raman, R. V. Rachamadugu, and F. B. Talbot. Real time scheduling of an
automated manufacturing center. *European Journal of Operational Research*,
40:222–242, 1989.   76

[11] G. Theraulaz, E. Bonabeau, and J. L. Deneubourg. Self-organization of hierarchies
in animal societies: The case of the primitively eusocial wasp polistes dominulus
christ. *Journal of Theoretical Biology*, 174:313–323, 1995.   71, 76

[12] G. Theraulaz, E. Bonabeau, and J. L. Deneubourg. Response threshold reinforce-
ment and division of labour in insect societies. *Proceedings of the Royal Society
of London B*, 265(1393):327–335, February 1998.   71, 73, 74

[13] 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 Ani-
mals to Animats: Proceedings of the First International Conference on Simulation
of Adaptive Behavior*, pages 346–355. MIT Press, 1991.   71, 76