Ant Colony Control for Autonomous Decentralized Shop Floor Routing

Vincent A. Cicirello and Stephen F. Smith
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
{vincent+, sfs}@cs.cmu.edu

Abstract

In this paper, we introduce a new approach to autonomous decentralized shop floor routing. Our system, which we call Ant Colony Control (AC^2), applies the analogy of a colony of ants foraging for food to the problem of dynamic shop floor routing. In this system, artificial ants use only indirect communication to make all shop routing decisions by altering and reacting to their dynamically changing common environment through the use of simulated pheromone trails. For simple factory layouts, we show that the emergent behavior of the colony is comparable to using the optimal routing strategy. Furthermore, as the complexity of the factory layout is increased, we show that the adaptive behavior of AC^2 evolves local decision making policies that lead to near-optimal solutions from the standpoint of global performance.

1. Introduction

The factory is a complex dynamical environment often plagued by unexpected events. Machines may break down. An unexpected urgent job may suddenly be released to the shop. The priority of a job may be changed. Factories must continue to function despite the uncertainty of such events. Many scheduling systems focus on optimization given known characteristics of the jobs such as their due dates, release dates, priorities, and so forth. Such a static view of scheduling often leads to schedules that break down in this dynamic and often unpredictable environment. A dynamically changing problem calls for a dynamic solution.

In practice, scheduling tasks are often addressed in a more decentralized manner. For example, the use of local dispatch scheduling rules is commonplace in many manufacturing environments (see Morton and Pentico [16] for a thorough discussion of dispatch scheduling heuristics). Reliance on such localized decision-making policies, which base decisions on such factors as the due dates or priorities of pending jobs in conjunction with the dynamic state of the shop, add robustness to the shop floor control system. As machines break down or other unexpected events occur, the system continues to function.

The challenge in specifying decentralized approaches to shop floor control is in achieving optimized global performance. Though properly matched dispatch policies can sometimes give rise to quite efficient performance, their susceptibility to myopic and globally suboptimal decision-making is well recognized. Both the machine learning [19] and the genetic programming [15] communities have attempted to address this problem. Riedmiller and Riedmiller [19] train a neural network representation of a dispatch policy using reinforcement learning techniques incorporating global information about the tardiness of jobs into their reward function. Miyashita [15] applies genetic programming to the problem, defining the fitness of a dispatch policy in terms of its global performance.

There have been a number of agent-based approaches that take a decentralized view of shop floor control [10, 12, 13, 15, 17, 19, 20, 23]. Through various forms of agent coordination these approaches attempt to attain coherent global performance by means of local decision-making. A few popular agent-based negotiation paradigms used by decentralized approaches to scheduling include the contract net protocol [10, 17, 20], auction protocols [23] and other market-based approaches [12, 13]. There is evidence that market-based approaches can produce globally optimal or near-optimal solutions. Wellman et al [23] show that for discrete resource allocation problems an equilibrium solution is an optimal solution. 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. Furthermore, these approaches are often plagued by much communications overhead as agents are constantly exchanging bids or other information needed for negotiation. Some market-based approaches incorporate a centralized auctioneer agent to attempt to alleviate this problem; but these systems are no longer completely decentralized and the auctioneer can become a bottleneck.

In this paper we consider an alternative approach to the problem of infusing local shop floor control decisions with more global performance information. The approach we have taken to this problem is that of using simulated ant colonies to decentralize and autonomize shop floor routing. We call our approach Ant Colony Control (AC^2). Ants are assigned to new jobs as they are released to the shop and these ants make all routing decisions for their respective jobs based on local information in the form of simulated pheromone trails, analogous to the chemical trails used by real ants for indirect communication and coordination. The system is completely decentralized with no central controller governing the ants. There is no direct communication among ants. All communication is accomplished indirectly through the stigmergetic process of updating pheromone quantities located at the various machines. Stigmergy is a process by which the common environment is altered by individuals of the ant colony and this dynamically changing environment is used for self-organization and coordination within the colony.

In Section 2 we give a brief overview of ant based approaches to various problems. Next in Section 3 we describe our formulation of the ant colony approach to shop floor control. Then in Section 4 we discuss a number of experiments we have conducted, followed by a discussion of the limitations of our system in Section 5. And finally in Section 6 we conclude with future work.

2. Ant Algorithm Background

Social insects such as the ant organize themselves into highly autonomous distributed systems – namely colonies. These systems can be viewed as productive and efficient. Ants perform such activities as sorting their brood, organizing their dead into cemeteries, and foraging for food with no direct communication. They organize such activities through indirect communication – known as stigmergy – by which individuals alter the environment and this ever-changing environment is used for coordinating the colony's activities.

Analogies to the way in which ants solve problems in nature have been used to solve some difficult real-world problems. For perhaps the most complete overview to the field, see Bonabeau, Dorigo, and Theraulaz [3]. Approaches to coordinating multiple robots in the performance of tasks such as object sorting [5] and object clustering [2] have mimicked the ways in which ants sort their brood and collect their dead into cemeteries, respectively. Using local tactics based on an analogy to ant colony coordination, Drogoul [9] has developed a chess system comparable in strength to an average human. Parunak and Brueckner [18] have used an ant pheromone inspired algorithm to solve the missionaries and cannibals problem.

The ant colony problem solving analogy that we are most interested in, however, is that of foraging. As they forage for food, ants leave a trail of pheromone between the nest and the food source. This chemical trail can then be used by other ants from the colony for a more efficient search for food. Ants tend to wander randomly but generally in the direction of heavier concentrations of pheromone. Over the course of time, the trail to this food source gradually converges to the shortest path due to ants wandering slightly off the current pheromone trail and finding short cuts.

Based on this ant foraging analogy, Dorigo et al have founded a new paradigm of evolutionary computation that has come to be known as Ant Colony Optimization (ACO). Their initial Ant System (AS) [8] showed promising results with the Traveling Salesman Problem (TSP). They later refined their approach in their Ant Colony System (ACS) [7] and through the addition of some local search procedures this ACS is competitive with some of the best known heuristics for the TSP.

ACO based algorithms have been applied to a number of other optimization problems. Schoonderwoerd et al [21] have developed their Ant-Based Control (ABC) system for optimizing the routing tables in telecommunications networks. Similarly, Di Caro and Dorigo [6] incorporate ACO techniques into their AntNet system for the optimization of communications network routing tables. Varela and Sinclair [22] apply ACO to the problem of routing and wavelength allocation in multi-wavelength virtual-wavelength-path transport networks. In their approach Varela and Sinclair use multiple pheromone types for the different wavelengths in their system.

Other ACO inspired algorithms have been applied to problems such as the sequential ordering problem [11], vehicle routing [4], single machine scheduling [1], and resource-constrained project scheduling [14]. Unlike our system, each of these scheduling applications of ACO deals with a statically defined problem. But the shop floor is a dynamic environment and our approach treats it as such.

3. Our Approach: AC^2

Our system uses a colony of artificial ants as the basis for adaptive and dynamic shop floor scheduling. It is loosely based on the ACO paradigm of Dorigo and colleagues. However, our system is unlike most applications of ACO in that we are not dealing with a statically defined optimization problem. Our target environment is changing dynamically. We view scheduling in such a dynamic environment as more of a routing or a control problem and we have named our system Ant Colony Control (AC^2).

The shop floor configurations that we consider consist of multi-stage jobs, multiple machines, alternative resources, setup times, machines with variable processing speeds. All of these characteristics add to the complexity of the scheduling task. In our system, each job, as it is released onto the shop floor, is assigned to an ant to carry it through the shop. There is no direct communication with the resources or with other ants. All communication is carried out indirectly in the form of pheromone deposits that the ants leave on the trail between resources. Control is completely decentralized and is dependent upon the emergent behavior of this colony of ants.

Given resource alternatives for the current step of the job, the ant chooses randomly among these alternatives biased according to the amount of pheromone located at each resource. If the shop can process jobs of different types then each type is associated with its own type of pheromone. Ants can distinguish the pheromone of the type of job they are currently carrying through the shop from that of other job types. Their choice among alternative resources is biased in the direction of heavier concentrations of their own pheromone type and away from that of other pheromone types. This is similar to the multiple pheromone approach of Varela and Sinclair [22] for multi-wavelength routing and allocation.

The reason for biasing the ants choice away from pheromone of other types is to attempt to minimize setup times. For example, consider a machine capable of performing two alternative operations: Op 1 and Op 2. Now suppose that if the machine is set up for Op 1 then there will be additional setup time required in order to switch the machine over to Op 2, and vice versa. Further suppose that jobs of type A require Op 1 and jobs of type B require Op 2. By biasing ants away from pheromone of other types, an ant carrying a job of type A is less likely to follow an ant carrying a job of type B to this machine if some alternate machine exists. This potentially limits the amount of setup time required.

To define the rule used by the ants to choose among alternatives first let $\pi^c_{a,b}$ be the quantity of pheromone of job type c that lies on the trail from resource a to resource b. Now define:

$$\tau^c_{a,b} = \frac{\pi^c_{a,b}}{\sum_{i,k \neq c}(\pi^k_{i,b})} \tag{1}$$

An ant associated with a job of type c which just completed a step using resource a will choose to use resource b next with probability:

$$P^c_{a,b} = \begin{cases} \frac{\tau^c_{a,b}}{\sum_j(\tau^c_{a,j})} & \text{if } b \text{ can perform the next step} \\ 0 & \text{if } b \text{ cannot perform the next step} \end{cases} \tag{2}$$

Now that we have defined the rule that the ants use to choose where to go next, we must define the rule that the ants use to update the pheromone. As the ant carries its job through the system, it remembers the path that it has taken. Upon completing the last step of the job, the ant retraces its path in order to pick up a new job to carry through the system. This retracing is the exact reverse of the path it just took through the shop. As it backtracks through the shop it updates pheromone according to the rule:

$$\pi^c_{a,b}[t+1] = \pi^c_{a,b}[t] + R \tag{3}$$

In the previous equation, R is a constant. This update rule is similar to the pheromone update rule used in the ant-density variation of the Ant System (AS) of Colorni, Dorigo, and Maniezzo [8]. Prior to starting up the system all pheromone levels are initialized to some constant π_0.

Just like the pheromone of real ants, the pheromone left by the ants in our system decays over time. There are two ways that pheromone decays in our system. The first is strictly due to time. At each time unit all pheromone levels are decayed according to the rule:

$$\pi^c_{a,b}[t+1] = \alpha * \pi^c_{a,b}[t] \tag{4}$$

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

The second way that pheromone decays is when ants leave pheromone of the job type that they are carrying, pheromone of other job types leading to the same destination are decayed. In other words when an ant updates pheromone at $\pi^c_{a,b}$, the following pheromone decay rule is applied:

$$\pi^k_{i,b}[t+1] = \begin{cases} \beta * \pi^k_{i,b}[t] & \text{if } k \neq c \\ \pi^k_{i,b}[t] & \text{if } k = c \end{cases} \tag{5}$$

The value of β is a constant between 0 and 1. The purpose of this decay rule is to encourage ants that are carrying different job types to choose different paths through the shop, if possible, as an attempt at minimizing setup time.

4. Experiments

We consider the performance of AC^2 on a progression of increasingly complex factory configurations. In each of these experiments, we show the progress of the simulation for the first 1000 time unit interval as well as the 2000-5000 time unit interval. All of these experiments use the following values for the parameters: $R = 10$, $\pi_0 = 5$, $\alpha = 0.99$, and $\beta = 0.75$. These settings are the result of experimentation and no automatic tuning techniques have been used. They appear to work well for a number of shop floor layouts. Also, in all of the experiments, release dates of jobs are not known a priori so building schedules in advance is not feasible. In the experiments involving multiple job types, one machine is initially setup for job type A and the other machine for job type B.

The optimal solutions have been given for experiments 1 and 2. In the case of experiments 3 and 4, which deal with setup times, optimal solutions are not known as these are in the class of NP-Hard problems. In each experiment the performance of the ant colony approach is compared to the following simple local heuristics:

Heuristic 1 *Earliest estimated completion time: Estimated completion time is defined as the sum of the processing times and setup times for all jobs in the wait queue, including the job currently being processed by the machine, plus the processing time and possible setup time of the new job.*

Heuristic 2 *Random choice: Choose among alternatives randomly with uniform probability.*

4.1. Experiment 1: Two Machines / One Job Type

This first experiment considers the simple problem of allocating jobs of a single type to two alternative machines. The layout of the shop floor can be seen in Figure 1. These machines are capable of performing the same operations. However, Machine M1 only requires 10 units of processing time while Machine M2 requires 20 units. Each machine has its own queue of jobs to process. Both machines use a first-in-first-out (FIFO) processing order. Jobs are released to the shop randomly with an average of three jobs being released in any given 20 unit block of time.

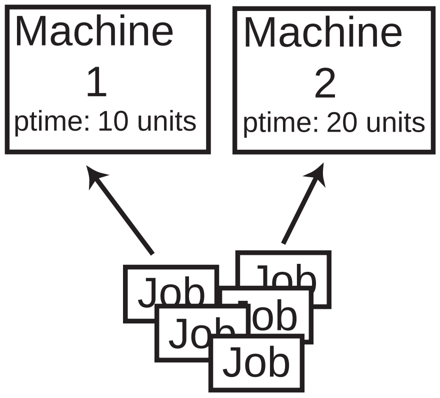

Figure 1. Two alternative resources: Machine M1 is twice as fast as Machine M2.

The shop processes a single type of job. This single job type requires a single operation that can be performed on either machine. When a new job is released to the shop it is assigned to an ant. This ant must decide which machine's wait queue to enter. Jobs are not able to switch wait queues. This is perhaps due to the location of the machines or some other factor that makes changing queues impractical. Once it is queued, a job must wait until the chosen machine processes it.

Table 1. Experiment 1 – Throughput comparison of ant colony approach to simple local heuristics for the two machine / one job type problem of Figure 1.

Time		AC^2	H1	H2	Optimal
1000	M1	89	89	79	89
	M2	41	46	46	46
	Total	130	135	125	135
2000 to 5000	M1	363	363	291	363
	M2	190	191	191	191
	Total	553	554	482	554

As can be seen in Table 1, local heuristic H1 performs better in terms of throughput as compared to AC^2. In this example, the local information is actually a global view of the shop and this simple local heuristic will lead to the optimal solution for the problem of optimizing throughput in this extremely simple factory. However, do note that as time progresses AC^2 does begin to converge toward the optimal.

4.2. Experiment 2: Two Assembly Lines / One Job Type

This next experiment is more complex than the previous. The layout of the shop floor can be seen in Figure 2. There are eight machines arranged as two assembly lines with four machines each. Machines M1 and M2 perform the same functions, machines M3 and M4 perform the same functions, and so forth. Machines M1 and M2 require the same amount of processing time: 10 units each. Machines M3, M5, and M7 of the first assembly line each require 10 units of time while machines M4, M6, and M8 of the second assembly line each require 20 units of time. Like the previous experiment, each machine has its own queue of jobs to process. And all machines use a FIFO processing order. Jobs are released to the shop randomly with an average of three jobs being released in any given 20 unit block of time.

Also like the previous experiment, the shop processes a single type of job. This single job type requires a sequence of four operations that can be performed by either assembly line. When a new job is released to the shop it is assigned to an ant. This ant must begin by deciding which machine's wait queue to enter. As before jobs are not able to switch wait queues. Jobs are also not able to move between assembly lines during later steps. Upon completing all steps of the job the ant updates the pheromone associated with the machines it used along the way.

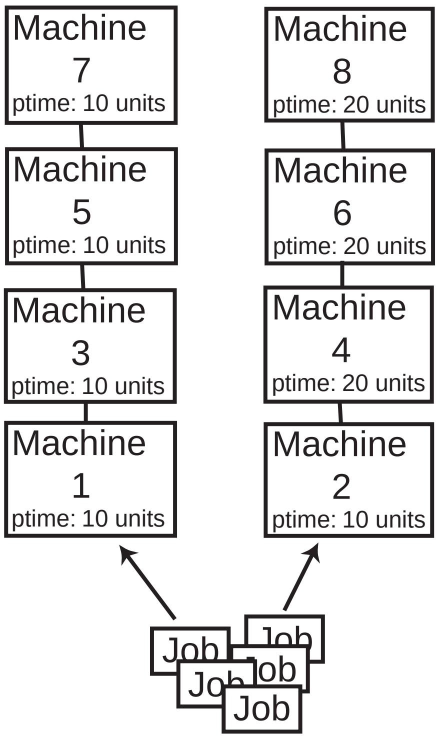

Figure 2. Two alternative assembly lines with different processing times.

If one has a global view of the system, it is clear that the first assembly line can process jobs quicker than the second. However, as you can see in the results of the simple heuristics reported in Table 2, it is not completely clear as to how much better the first assembly line is based only on the local view at the start. However, AC^2 evolves local heuristics that do lead to better global results. In the 1000 time unit interval, these results are not drastic, however there is considerable improvement on the performance of AC^2 relative to the local heuristics during the 2000-5000 time unit interval. As time goes on, AC^2 evolves a rule that on average sends approximately twice as many jobs to the faster assembly line, whereas heuristic H1 views both assembly lines as roughly equivalent given that the first machines of both are equivalent. Heuristic H1 no longer leads to the optimal solution for this problem while AC^2 evolves an optimal decision policy.

4.3. Experiment 3: Two Machines / Two Job Types

This experiment adds a level of complexity to the problem in that it deals with a shop that can process two job types. Each of the two job types require a single operation, although each job type requires a different operation.

The layout of the shop floor can be seen in Figure 3. Like the first experiment, there are only two machines. These machines are identical in terms of capability and the operations that they can perform. Each machine can perform one of two operations. The time required by either machine to perform either operation is 10 units. However, 20 units of time are required to setup the machine for an operation that it is not already setup for. Each job enters a wait queue and is processed by the corresponding machine in FIFO order. It is beneficial if jobs of the same type follow each other in the wait queues to avoid delays caused by setup times. On average at random in any 10 unit block of time one job of type A and one job of type B are released to the shop floor.

In Table 3 we see that H1 does a good deal better than the completely random choice of H2. This heuristic somewhat takes into consideration the sequencing of jobs of the same type together on the same machine. However, it does not lead to the best solution and breaks down overall. And

Table 2. Experiment 2 – Throughput comparison of ant colony approach to simple local heuristics for the two assembly lines / one job type problem of Figure 2.

Time		AC^2	H1	H2	Optimal
1000	A1	86	78	73	86
	A2	38	44	44	44
	Total	124	122	117	130
2000 to 5000	A1	363	322	290	363
	A2	190	190	190	190
	Total	553	512	480	553

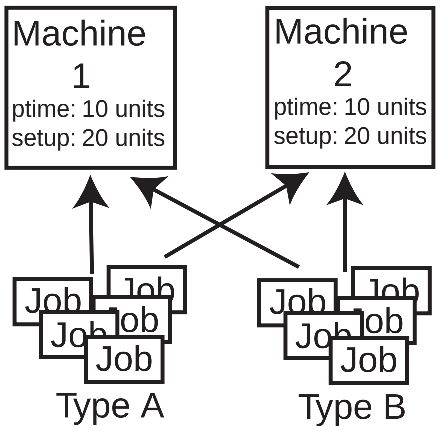

Figure 3. Two alternative machines each capable of performing two operations but with setup time to switch between the two.

Table 3. Experiment 3 – Throughput comparison of ant colony approach to simple local heuristics for the two machines / two job types problem of Figure 3.

Time	Type		AC^2	H1	H2
1000	A	M1	6	21	18
		M2	76	35	28
		Total	82	56	46
	B	M1	67	30	20
		M2	3	26	22
		Total	70	56	42
	Both	M1	73	51	38
		M2	79	61	50
		Total	152	112	88
2000 to 5000	A	M1	0	115	79
		M2	364	117	103
		Total	364	232	182
	B	M1	364	93	85
		M2	0	118	85
		Total	364	211	170
	Both	M1	364	208	164
		M2	364	235	188
		Total	728	443	352

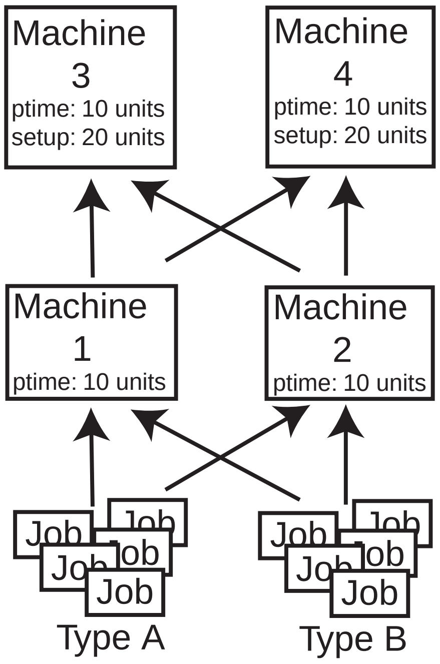

Figure 4. Machines M2 and M3 are capable of handling two operations but with setup time to switch between the two. Machines M1 and M2 are identical.

we can see that AC^2 does extremely better for this problem compared to the simple local heuristics. Over time AC^2 evolves a local rule that sends all of type A to one machine and all of type B to the second machine eventually avoiding all setup times.

4.4. Experiment 4: Four Machines / Two Job Types

This experiment is the most complex presented in this paper. Like the previous experiment there are two job types. This time however each of these job types consist of a sequence of two operations. The first operation of both job types is identical. The second, however, defines the difference between these two job types. Consider these two job types to be from the same family of products so in one way they are similar to each other but yet different.

In Figure 4 we see the shop floor layout for this problem. Machines M1 and M2 are identical in terms of capabilities and processing times. Likewise, machines M3 and M4 are identical, each with the ability to perform two operations but with setup time to switch between these operations. One of these operations is required by the first job type and the other by the second job type. The setup time required to switch the machine between these operations is 20 time units. Each machine has its own wait queue and jobs are processed FIFO. It is beneficial for jobs of the same type to follow behind each other within the same queue in order to minimize setup time. After completing the first operation on either machine M1 or M2, the ant must decide to which of machines M3 or M4 to bring the job for the second operation. So for this experiment there are two decisions for the ant to make. On average at random in any 10 unit block of time one job of type A and one job of type B are released to the shop floor.

In Table 4 we again see the downfall of simple local heuristics for this problem. Heuristic H1 makes no distinction between the two job types when making the first decision as the first operation of both job types is the same. However, this is acceptable as both machine M1 and machine M2 are identical. It is with the second decision that heuristic H1 fails miserably. However, the local rule evolved by AC^2 leads to the jobs self-organizing themselves according to their job type.

In Table 4 for both the 1000 time unit and 2000-5000 time unit intervals we see that AC^2 sends all jobs of type A to machine M3 and all jobs of type B to machine M4. To understand why there has been a complete self-organization right from the start of the simulation with respect to machines M3 and M4 consider the following scenario. Sup-

Table 4. Experiment 4 – Throughput comparison of ant colony approach to simple local heuristics for the four machines / two job types problem of Figure 4.

Time	Type		AC^2	H1	H2
1000	A	M3	87	41	18
		M4	0	17	27
		Total	87	58	45
	B	M3	0	19	24
		M4	84	37	16
		Total	84	56	40
	Both	M3	87	60	42
		M4	84	54	43
		Total	171	114	85
2000 to 5000	A	M3	364	147	81
		M4	0	89	103
		Total	364	236	184
	B	M3	0	88	95
		M4	364	125	73
		Total	364	213	168
	Both	M3	364	235	176
		M4	364	214	176
		Total	728	449	352

pose the first job through the system chooses M3 to process its second operation. Further suppose that this job is of type A and that M3 is already setup for job type A. The job in question will finish on M3 in time to update the pheromone prior to any further jobs choosing between M3 and M4. So jobs of type A will now be more likely to choose M3 while jobs of type B will be more likely to choose M4. Further simulations with different seeds for the random number generator have shown that sometimes this occurs while at other times a few jobs will queue themselves on the alternate machine early on acquiring some setup time.

5. Limitations

One major limitation that our system suffers from is that it fails to adapt appropriately to changing job mixes once it has converged to some stable behavior. In particular, consider the simple case of two machines and two job types with a 50/50 job mix (ie. Experiment 3). In this case, our system specializes each machine to one job type (ie. each machine has a large quantity of pheromone of a single type and a very small quantity of pheromone of the alternate type). If we suddenly have a large influx of one job type and the other type eases off, the system as defined here will not know to try sending some jobs of the heavier concentration to both machines. Schoonderwoerd et al [21] solved this problem in their communications routing algorithm by making some small percent of the decisions at random. This guarantees that even paths where the pheromone levels have decayed to zero are explored. This does not seem to work very well, however, for our system. Although it does allow our system to explore alternatives once it has converged to some stable behavior, in the case when the job mix has not changed it often results in temporary "bad" performance until the system converges to its previous configuration. One possible solution which we will explore is to incorporate "virtual" jobs. That is, ants carrying "real" jobs will follow the pheromone rules defined in this paper; while ants carrying "virtual" jobs will use some combination of these same pheromone rules and random behavior. These "virtual" jobs will not have an effect on system performance, but will instead explore the current load characteristics of the shop.

6. Conclusions and Future Work

In this paper, we have introduced a new approach to decentralized shop floor routing based on the ACO research of Dorigo and others. Our system, which we have named Ant Colony Control (AC^2), coordinates a population of simple agents, namely the ants, via indirect communication accomplished by altering and reacting to the commonly shared distributed environment of the factory. The global behavior that emerges from this completely autonomous and decentralized system performs comparably to following the optimal routing strategy on simple problems. However, for more complex problems, AC^2 evolves local decision making policies that lead to near-optimal solutions from the standpoint of global performance.

In this paper we have taken some initial steps toward the broad goal of achieving self-organizing factories. Our hypothesis is that such ant colony inspired techniques provide a basis for localized, but globally informed control policies under a wide range of operating constraints. We plan to further test the limits of AC^2 on problems of increasing complexity such as by introducing priorities and due dates of jobs as well as the addition of machines that use more sophisticated dispatching rules. We will also investigate the performance of AC^2 in complex environments consisting of such characteristics as dynamically shifting bottlenecks, job priority changes, machine failures, and dynamically changing job mixes.

7. 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

F30602-97-2-0227 and F30602-00-2-0503 and by the CMU Robotics Institute.

References

[1] A. Bauer, B. Bullnheimer, R. F. Hartl, and C. Strauss. An ant colony optimization approach for the single machine total tardiness problem. In *CEC99: Proceedings of the Congress on Evolutionary Computation*, pages 1445–1450, July 1999.

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

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

[4] B. Bullnheimer, R. F. Hartl, and C. Strauss. An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89:319–328, 1999.

[5] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien. The dynamics of collective sorting robot-like ants and ant-like robots. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 356–363. MIT Press, 1990.

[6] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

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

[8] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(1):29–41, February 1996.

[9] A. Drogoul. When ants play chess (or can strategies emerge from tactical bevahiours?). In C. Castelfranchi and J. Müller, editors, *From Reaction to Cognition: 5th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW '93*, pages 13–27. Springer, August 1993.

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

[11] L. M. Gambardella and M. Dorigo. 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, 1997.

[12] K. Kuwabara and T. Ishida. Equilibratory approach to distributed resource allocation: Toward coordinated balancing. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems: 4th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (Selected Papers)*, pages 133–146. Springer-Verlag, S. Martino al Cimino, Italy, July 1992.

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

[14] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. In *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 893–900. Morgan Kaufmann, July 2000.

[15] K. Miyashita. Job-shop scheduling with genetic programming. In *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 505–512. Morgan Kaufmann, July 2000.

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

[17] H. V. D. Parunak. Manufacturing experience with the contract net. In M. N. Huhns, editor, *Distributed Artificial Intelligence*, pages 285–310. Morgan Kaufmann, 1987.

[18] H. V. D. Parunak and S. Brueckner. Synthetic pheromones for distributed motion control. In *Advances in Enterprise Control Symposium Proceedings*, pages 151–160. DARPA-ISO, San Diego, CA, November 15-16 1999.

[19] S. Riedmiller and M. Riedmiller. A neural reinforcement learning approach to learn local dispatching policies in production scheduling. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 764–769, Stockholm, Sweden, July-August 1999.

[20] T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 256–262, Washington, DC, July 1993.

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

[22] G. N. Varela and M. C. Sinclair. Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation. In *CEC99: Proceedings of the Congress on Evolutionary Computation*, pages 1809–1816, July 1999.

[23] W. E. Walsh, M. P. Wellman, P. R. Wurman, and J. K. MacKie-Mason. Some economics of market-based distributed scheduling. In *Proceedings of the Eighteenth International Conference on Distributed Computing Systems*, pages 612–621, May 1998.