

Poaching and Distraction in Asynchronous Agent Activities

Mike H. Chia, Daniel E. Neiman, Victor R. Lesser
University of Massachusetts
Computer Science Department
Amherst, MA 01003
{chia,dann,lesser}@cs.umass.edu

Abstract

We investigate coordination issues in a distributed job-shop scheduling system in which agents schedule potentially contentious activities asynchronously in parallel. Agents in such a system will in general have a limited view of the global state of resources and must exchange appropriate state information with other agents in order to schedule effectively. However, even given perfect instantaneous knowledge of other agents' resource requirements, agents still may not be able to schedule effectively if they do not also model the possible future actions of other agents and the effects of their own actions. We formally describe two types of agent behaviors, poaching and distraction, arising from the asynchronous nature of distributed systems that decrease scheduling effectiveness, and we present experimental results from a distributed airport resource management system demonstrating a significant improvement in scheduling performance when coordination mechanisms are used to prevent such behaviors.

1. Introduction

The scheduling of resources and activities is known to be an extremely difficult problem [5, 4, 12, 13]. In a distributed scheduling system, the complexity increases due to the possibility of interactions between scheduling agents such as when agents are allowed to borrow resources from each other to satisfy local goals. Examples of distributed scheduling applications include scheduling machines in manufacturing facilities, scheduling airline ground services in an airport, and scheduling transportation of goods.

In this paper, we explore issues of agent coordination in distributed scheduling. In particular, we study the behavior of agents in an environment in which agents develop schedules concurrently in batch mode, are cooperative, and agree to share resources. In previous work, Neiman, *et. al.* [9] have investigated the questions of what types of meta-

level information to communicate and how that information is to be used to schedule more effectively. We extend that work by investigating the lack of coordination among agents that can exist even when agents are able to model the global state of resources. We show that when agents are not able to model the state and possible future activities of other agents, they cannot schedule effectively [3]. We investigate two types of behavior resulting from this lack of coordination: *poaching* and *distraction*.

In the following section, we give a brief overview of our experimental system, the Distributed Dynamic Scheduling System (Dis-DSS) and our problem domain, airport ground service scheduling. In section 3, we discuss in detail the agent coordination issues and agent behaviors on which we have focused in our most recent experiments with the Dis-DSS. In section 4, we present coordination mechanisms that we have added to our system and the results of experiments indicating that the quality of scheduling can be significantly improved with no appreciable increase in run time by judicious synchronization of agent activities. Finally, in section 5, we conclude and present directions for further study.

2. Experimental system

Our experimental system for investigating resource-constrained scheduling problems is a distributed version of the Dynamic Scheduling System (DSS) [6]. The DSS is a micro-opportunistic scheduler based on a blackboard architecture. A problem instance consists of a set of orders and a set of resources. Each order is represented as a set of tasks and subtasks, and each subtask requiring a resource is assigned a *service goal*. Service goals are continually rated based on the tightness of constraints on the particular task and required resource. At each time step, the system attempts to reserve resources for those service goals that are deemed to be most constrained using a multiple-perspective heuristic [11].

Service goals are satisfied by attempting three progressively costly methods: (1) Assignment: The system sim-

ply reserves an available resource to satisfy a service goal. (2) Preemption: The system cancels an existing reservation so that the resource can be reserved for a more constrained service goal. This is a limited form of backtracking. (3) Right-shift: The system forcibly relaxes the latest-finish-time constraint in order to satisfy a service goal that would otherwise go unsatisfied. In other words, the time interval of the reservation is shifted later than the service goal's latest finish time. This usually causes decreased solution quality in the form of late orders and is a method of last resort.

The Distributed Dynamic Scheduling System (Dis-DSS) [9] extends the DSS by partitioning the order and resource sets and distributing the partitions among several agents. Dis-DSS is a cooperative environment. When unable to satisfy a service goal locally, agents can generate requests in which they ask another agent to satisfy the goal. Agents operate asynchronously, simultaneously developing schedules for their respective order sets.

Each agent is essentially a DSS scheduling agent but uses additional control heuristics to account explicitly for the fact that it is operating in a multi-agent system where other agents are reserving resources at the same time. Heuristics from the single agent system cannot be applied unchanged to a multi-agent system, an observation also made by Decker, *et al.* in their work on developing control paradigms for a parallel blackboard system [1]. For example, control heuristics in a distributed system need to explicitly account for the possible outcomes of activities that are currently executing in order to avoid scheduling overlapping tasks.

Our test-bed system built on top of Dis-DSS is the Distributed Airport Resource Management (Dis-ARM) system. Dis-ARM solves airport ground service scheduling (AGSS) problems. Each agent is assigned a set of orders, which in the AGSS domain are flights requiring ground service such as loading/unloading baggage, cleaning, refueling, etc. Agents are responsible for scheduling ground service so that their flights are able to meet arrival and departure deadlines.

Each agent also owns a set of resources such as gates, baggage trucks, cleaning trucks, fuel trucks, etc. In general, the reservation of a resource for a given task must account for setup and travel times of resources as well as the actual servicing times. These times are dependent upon the type and location of individual resources. Individual resources of the same type are not considered interchangeable. Only the agent owning a particular resource will have exact information regarding the time needed for setup and travel. In addition, because of communication delays, an agent can more readily access its own individual resources than resources owned by another agent.

The search space of alternatives available to the scheduler is quite large due to the number of possible resources, the number of tasks to be scheduled, and the number of con-

straints between subtasks. Moreover, the search space is dynamic because requests for resources may arrive from other agents at any point during scheduling.

3. Coordination in distributed scheduling

In a distributed system, scheduling agents may not have a complete view of global resource availability and demand. To enable more effective scheduling, Dis-DSS agents exchange meta-level control information in the form of texture measures [14], abstract representations of an agent's resource availability, current resource usage, and estimated future demand. By obtaining texture measures from other agents, an agent can build a more global view of the status of resources. Agents use this information to make decisions regarding what service goals to schedule next (based on which are most constrained) and how to satisfy service goals (whether to use a local resource or request a resource from another agent). Agents can also use the information to determine which agents are likely to have a surplus of a given type of resource and thus are good candidates from which to request resources. However, even with complete information about the state of resource availability, agents may not be able to schedule in a globally optimal manner if they do not explicitly account for the possible actions of other agents in the system [3].

In the following sections, we describe two agent behaviors, *poaching* and *distraction*, that may decrease scheduling performance. We use the following notation for describing the conditions under which these behaviors can occur. Let $g_{a,i}^x$ denote service goal i for agent a requiring resource type x , (where a resource type is the set of resources that can satisfy the service goal). Let $R(g_{a,i}^x, t)$ denote the rating at time t of service goal $g_{a,i}^x$ and $E(g_{a,i}^x)$ denote its execution time, i.e. the time when agent a attempts to reserve a resource of type x for service goal i .

The best-first heuristic used by each scheduling agent can then be expressed as

$$E(g_{a,i}^x) = t \Leftrightarrow \forall y \forall j \neq i R(g_{a,i}^x, t) \geq R(g_{a,j}^y, t) \quad (1)$$

This simply states that agent a will execute service goal i at time t if and only if it is the most highly rated of agent a 's goals at time t .

3.1. Poaching

We define a poaching event as one in which an agent reserves a resource, thereby preventing another agent with a more constrained or critical goal from securing that resource. Poaching activity arises due to the fact that there are multiple agents scheduling asynchronously using a best-first heuristic. An agent may make a locally correct decision

Service Goal Ratings at Time 51				
Goal	Time Interval	Resource	Rating	Rank
Agent 0:				
#45	(5009 5035)	BAG-TRK	7163	0
#38	(5009 5036)	BAG-TRK	5283	1
#84	(5016 5041)	BAG-TRK	3945	2
#59	(5009 5036)	BAG-TRK	3877	3
#73	(5009 5036)	BAG-TRK	3877	4
Agent 1:				
#173	(5027 5050)	BAG-TRK	3570	5
#184	(5024 5050)	BAG-TRK	3537	8
#191	(5024 5050)	BAG-TRK	3537	7
#159	(4997 5033)	BAG-TRK	3336	9
#123	(5019 5046)	BAG-TRK	2659	12
Agent 2:				
#177	(5024 5051)	BAG-TRK	1353	22
#213	(5008 5050)	BAG-TRK	1180	25
#212	(5029 5056)	BAG-TRK	666	30
#104	(5015 5031)	CLEAN-TRK	148	43
#98	(5014 5032)	FUEL-TRK	146	45

Figure 1. An example of direct poaching. Agent 2 can potentially reserve a baggage truck for goal #177 before agent 0 can process the more constrained goal #45.

Service Goal Ratings at Time 868				
Goal	Time Interval	Resource	Rating	Rank
Agent 0:				
#118	(4858 4891)	SRV-TRK	232	99
#160	(4823 4891)	SRV-TRK	120	107
#36	(4851 4876)	CTR-TRK	75	111
#59	(4848 4880)	CTR-TRK	73	116
#179	(4872 4895)	CTR-TRK	61	125
Agent 1:				
#6	(4847 4868)	CLEAN-TRK	3993	4
#224	(4865 4892)	CLEAN-TRK	2537	15
#177	(4871 4896)	BAG-TRK	1272	41
#201	(4852 4890)	CLEAN-TRK	1272	40
#65	(4847 4880)	SRV-TRK	1213	44
Agent 2:				
#157	(4853 4882)	CLEAN-TRK	5994	0
#194	(4863 4887)	CLEAN-TRK	4716	1
#187	(4865 4887)	CLEAN-TRK	4689	2
#188	(4865 4887)	FUEL-TRK	4231	3
#130	(4860 4885)	FUEL-TRK	3915	5

Figure 2. An example of indirect poaching. Agent 0 can potentially reserve a service truck for goal #118 before agent 1 can process the more constrained goal #65.

by satisfying its most highly rated service goal, but this may not be the appropriate action in a global context – another agent may have a more highly rated goal that overlaps this goal and could be satisfied by the same resource. Poaching activity leads to decreased solution quality when the “poached” goal can no longer be satisfied without expensive constraint relaxation or backtracking activities.

Note that poaching can still be a problem even if two goals requiring the same resource type do not directly overlap temporally. When resources become highly constrained (they are globally scarce or need to be reserved for long periods of time), service goals that normally do not overlap directly will begin to feel the effects of poaching activity due to reduced flexibility in the schedule and the reduced ability of the scheduler to exploit slack in the schedules.

We have defined two classes of poaching activity:

(1) **Direct Poaching:** At time t , agents A and B both have as their most highly rated service goals, goals requiring the same resource type X . However, agent B’s rating is much lower than agent A’s indicating that agent B’s goal is less constrained. Because a goal requiring resource type X is agent B’s highest priority, it can reserve the required resource before agent A, despite the lower rating of agent B’s goal. Figure 1 is a Dis-ARM trace listing showing an example of such a poaching situation.

Formally, we express the situation as

$$\forall x \forall k \neq i R(g_{A,i}^X, t) > R(g_{A,k}^X, t) \quad (2)$$

$$\forall x \forall l \neq j R(g_{B,j}^X, t) > R(g_{B,l}^X, t) \quad (3)$$

$$R(g_{A,i}^X, t) \gg R(g_{B,j}^X, t) \quad (4)$$

From equations 1, 2, and 3 we obtain

$$E(g_{A,i}^X) = E(g_{B,j}^X) = t \quad (5)$$

and combining equation 5 with equation 4 we have the fol-

lowing condition:

$$R(g_{A,i}^X, t) \gg R(g_{B,j}^X, t) \wedge E(g_{B,j}^X) = E(g_{A,i}^X) \quad (6)$$

In this situation, poaching is possible because both agents execute their service goals at the same time step, and therefore it is possible (due to varying response times to remote requests for resources) that agent B can obtain a resource reservation before agent A despite that fact that agent B’s service goal has a much lower rating.

(2) **Indirect Poaching:** At time t , agent A and agent B both have service goals requiring the same resource type X . The ratings for agent A are higher or the same as those for agent B. For agent B, these are its most highly rated service goals, but agent A also has service goals requiring resource type Y which are its most highly rated goals (and thus more highly rated than those requiring resource type X). In this case, agent B can reserve resources of type X while agent A is working on scheduling resources of type Y . Figure 2 shows an example of such a situation.

Formally, we express the situation as

$$\forall x \forall k \neq i R(g_{A,i}^Y, t) > R(g_{A,k}^X, t) \quad (7)$$

$$\forall x \forall l \neq j R(g_{B,j}^X, t) > R(g_{B,l}^X, t) \quad (8)$$

$$R(g_{A,m}^X, t) \geq R(g_{B,j}^X, t) \quad (9)$$

From equations 1, 7, and 8 we obtain

$$E(g_{B,j}^X) = E(g_{A,i}^Y) = t \quad (10)$$

From equation 7, we know that

$$R(g_{A,i}^Y, t) > R(g_{A,m}^X, t) \quad (11)$$

and thus

$$E(g_{A,i}^Y) < E(g_{A,m}^X) \quad (12)$$

Service Goal Ratings at Time 76					Service Goal Ratings at Time 116					Service Goal Ratings at Time 138				
Goal	Time Interval	Resource	Rating	Rank	Goal	Time Interval	Resource	Rating	Rank	Goal	Time Interval	Resource	Rating	Rank
Agent 0:					Agent 0:					Agent 0:				
#174	(4870 4899)	GATE	493	27	#12	(4853 4870)	CLEAN-TRK	93	52	#83	(4848 4875)	CLEAN-TRK	752	51
#247	(4880 4909)	GATE	185	40	#11	(4847 4870)	BAG-TRK	91	61	#11	(4847 4870)	BAG-TRK	91	85
#12	(4853 4870)	CLEAN-TRK	93	51	#84	(4848 4875)	FUEL-TRK	87	74	#84	(4848 4875)	FUEL-TRK	87	90
#11	(4847 4870)	BAG-TRK	91	60	#82	(4844 4875)	BAG-TRK	79	87	#82	(4844 4875)	BAG-TRK	79	102
#84	(4848 4875)	FUEL-TRK	87	73	#83	(4848 4875)	CLEAN-TRK	78	101	#81	(4854 4881)	BAG-TRK	70	141
#82	(4844 4875)	BAG-TRK	79	88	#81	(4854 4881)	BAG-TRK	70	132	#79	(4844 4881)	SRV-TRK	69	147
Agent 1:					Agent 1:					Agent 1:				
#43	(4845 4884)	GATE	1413	0	#43	(4845 4884)	GATE	2471	0	#48	(4855 4872)	CLEAN-TRK	3076	0
#57	(4844 4884)	GATE	1376	1	#57	(4844 4884)	GATE	2376	1	#15	(4855 4872)	CLEAN-TRK	3076	1
#13	(4845 4874)	GATE	1349	3	#19	(4848 4877)	GATE	2247	5	#62	(4854 4872)	CLEAN-TRK	3002	2
#16	(4847 4876)	GATE	1339	5	#16	(4847 4876)	GATE	2205	6	#43	(4845 4884)	GATE	2662	3
#19	(4848 4877)	GATE	1328	6	#13	(4845 4874)	GATE	2105	8	#57	(4844 4884)	GATE	2556	7
#71	(4843 4884)	GATE	1097	9	#85	(4850 4889)	GATE	1895	10	#18	(4857 4874)	CLEAN-TRK	2296	12
Agent 2:					Agent 2:					Agent 2:				
#50	(4844 4884)	GATE	1376	2	#50	(4844 4884)	GATE	2376	2	#55	(4854 4872)	CLEAN-TRK	3002	3
#64	(4843 4884)	GATE	1340	4	#99	(4850 4889)	GATE	2332	3	#69	(4853 4872)	CLEAN-TRK	2891	4
#99	(4850 4889)	GATE	1144	7	#64	(4843 4884)	GATE	2287	4	#99	(4850 4889)	GATE	2598	6
#106	(4846 4889)	GATE	1122	8	#106	(4846 4889)	GATE	2145	7	#50	(4844 4884)	GATE	2556	8
#92	(4850 4889)	GATE	929	11	#131	(4852 4894)	GATE	2039	9	#64	(4843 4884)	GATE	2458	9
#131	(4852 4894)	GATE	906	12	#92	(4850 4889)	GATE	1895	11	#106	(4846 4889)	GATE	2375	10

Figure 3. An example of distraction. At time 76, all agents are scheduling gates. At time 116, agent 0 finishes assigning gates and begins to assign cleaning trucks. By time 138, cleaning trucks have become globally constrained, causing agents 1 and 2 to begin assigning cleaning trucks in lieu of finishing their gate assignments.

Finally, combining equations 9, 10, and 12 we have the following condition

$$R(g_{A,m}^X, t) \geq R(g_{B,j}^X, t) \wedge E(g_{B,j}^X) < E(g_{A,m}^X) \quad (13)$$

In this situation, poaching is quite likely because agent B executes its service goal earlier than agent A.

If the poaching activity continues and resource type X becomes constrained, then eventually agent A's ratings for service goals requiring resource type X will rise to be the most highly rated, and agent A will begin competing with agent B for the same resources. Depending on the global availability of that resource type and on the communication delay for exchanging texture measures, the damage can be done before agent A is able to respond. In fact, in some situations we do not want agent A to respond at all because shifting agent A's attention away from resource type Y to compete for resources of type X can have a detrimental effect on overall scheduling performance as we will see in the next section.

An interesting phenomenon in a distributed system is that an agent can actually "poach" on its own goals (that is, satisfy goals out of order) due to the communication delays in dealing with remote requests. An agent may execute service goals in order with respect to their ratings, but the times at which the resources are actually reserved (which depend on the actions of other agents) may not be in order.

3.2. Distraction

One effect of the use of texture measures combined with a most-constrained-variable heuristic in a micro-opportunistic system is that agents tend to synchronize their scheduling activities. As agents begin to reserve resources, a particular type of resource becomes constrained and service goals requiring that resource type become more highly

rated causing agents to increasingly focus on scheduling those service goals. This causes more reservations of the given resource type causing the resource to become even more constrained. This feedback process continues until eventually all agents are focused on scheduling the same type of resource, the globally most constrained type.

This type of synchronization can be an undesirable feature in some situations. Not only does it increase the likelihood of poaching, but its affect on the focus of an agent's activity can cause an agent to be distracted. *Distraction* occurs whenever an agent receives information from another agent and based on this information pursues an undesirable course of action [7]. In our system, we have encountered an analog to distraction in which an agent is prevented from executing important precursor activities by the increasing pressure of external events. Note that in Lesser and Erman [7], distracting information has the connotation of being noisy or misleading, and the receiving agent has the responsibility of interpreting it as such and pursuing more critical computations. Here, we present a situation in which the communicated information is good, and the receiving agent is placed on the horns of a dilemma; either it reacts to the incoming information, leaving time critical processing unperformed, or it ignores the new information, thereby failing to compete for critical resources. Unlike the original distraction scenario encountered in HEARSAY-II, it is the *distracting* agent that should act responsibly, either by working in a less globally sensitive area or by simply idling until other agents are capable of synchronizing planning activities. Such behavior is possible only if the distracting agent has enough information to gauge the effects of its actions.

Distraction is particularly evident in our system in cases where agents possess dramatically different loads. An agent who has relatively few orders to schedule can be-

gin scheduling certain resources well before other agents. This may cause a resource to become constrained, which in turn causes other agents to begin scheduling that resource instead of what they were scheduling. This is a problem when the order in which resources are scheduled is important. For example, in the AGSS domain, scheduling a gate location for a flight allows the scheduler to better estimate travel times for other resources such as baggage trucks and fuel trucks needed to service that flight. Without a fixed gate assignment, the scheduler must assume the maximum travel time when scheduling these other resources; this can cause a task to appear more constrained than it really is. It is clearly beneficial for agents to schedule gates early on in a scheduling episode. In the situation with an unbalanced load among agents, one agent can distract other agents from scheduling gates early. Figure 3 is an example trace of such a situation.

For the Dis-ARM system, we would like for the following to be true

$$\forall a, i, j \forall y \neq G \ E(g_{a,j}^G) < E(g_{a,i}^y) \quad (14)$$

where G represents the gate resource type. That is, we would like each agent to finish executing their service goals requiring gates before starting to execute other service goals.

If the order distribution is not balanced evenly among agents we have the possibility that an agent (say B) may begin to schedule at time t_1 a non-gate resource type $Y \neq G$ before another agent A has finished all of its gate assignments:

$$E(g_{B,i}^Y) = E(g_{A,j}^G) = t_1 \quad (15)$$

This creates an increase in agent A's rating of goals requiring resource type Y , and at some time $t_2 > t_1$ we may have

$$R(g_{A,k}^Y, t_2) > R(g_{A,l}^G, t_2) \quad (16)$$

and thus

$$E(g_{A,k}^Y) < E(g_{A,l}^G) \quad (17)$$

which violates our desired condition that all agents schedule gates before any other resource type.

In the next section we present coordination mechanisms for addressing the problems of poaching and distraction and the results of experiments for testing their effectiveness.

4. Discussion of experimental results

In the following sections, we present coordination mechanisms for addressing the problems of poaching and distraction, and we present the results of experiments for testing their effectiveness. We show the effect of adding these mechanisms to two reference systems: (1) a system in which agents only have a local view of resource availability

Run #	Number of orders:		
	Agent 0	Agent 1	Agent 2
1-5	16	16	17
6-10	10	16	23
11-15	10	10	23
16-20	5	22	22
21	0	0	49

Table 1. Order distribution among agents for each set of experiments.

Resource type	Number of resources:		
	Agent 0	Agent 1	Agent 2
Gates	17	16	16
Fuel trucks	6	6	6
Catering trucks	4	4	4
Service trucks	1	2	2
Baggage trucks	11	11	11
Cleaning trucks	3	3	2

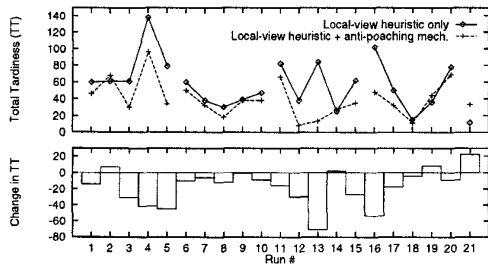
Table 2. Resource distribution among agents for all experiments.

and demand (which we refer to as the “local-view heuristic”) and (2) a system in which agents exchange texture measures for building a global view of resources (which we refer to as the “texture-based heuristic”).

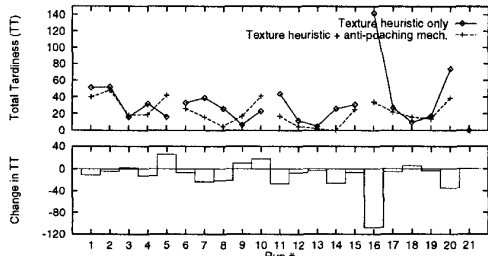
Because of the complexity of interactions among scheduling agents, the addition of our new coordination mechanisms does not guarantee an increase in scheduling performance in all cases. For some individual runs, we observe a decrease in performance, but on average, the system performance increased significantly. Performance is measured in terms of the total tardiness in the schedules of all agents (and thus, smaller numbers are better with zero tardiness being the best).

Each set of experiments consists of 21 runs, grouped into five different load distribution conditions (see table 1). Each run within a condition has a different order distribution among three agents (orders were assigned randomly). Orders are taken from a set of 49 actual Northwest Airlines flights requiring service at Detroit Metropolitan Airport during a one hour period. Resources are distributed more or less evenly among the agents for all runs. The total number of resources is set to be minimally sufficient for satisfying the requirements of the 49 orders. Table 2 gives the exact resource distribution among the three agents.

We include one order distribution that assigns all 49 orders to a single agent in order to approximate a centralized scheduler. It is an approximation in that one agent must schedule all of the orders but it does not own all the resources and will need to request resources from other agents. Because the other agents do not have any orders of their own to schedule, these requests will be handled



(a)



(b)

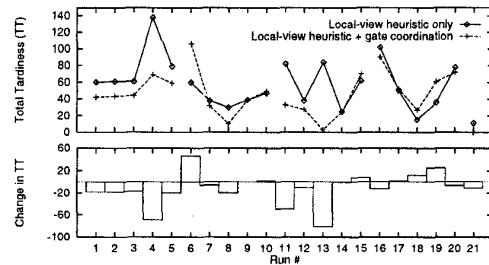
Figure 4. Change in performance due to the addition of the anti-poaching mechanism to the (a) local and (b) texture heuristics.

promptly. This “centralized” scenario produced very good results indicating that the tardiness we encounter during scheduling is, in fact, an artifact of the asynchronous and distributed nature of the system.

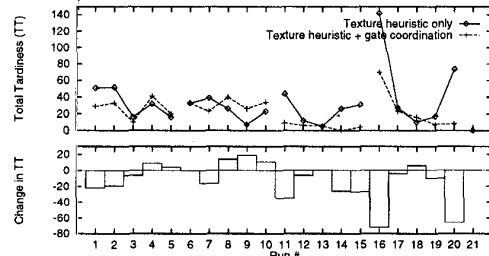
4.1. Anti-poaching mechanism

We hypothesize that in order to prevent poaching, agents need to communicate goal ratings for their top n service goals in addition to information about resource supply and usage in the texture measures. By exchanging this information agents can determine whether their top priority scheduling activities would poach on another agent’s activities. Our anti-poaching mechanism works as follows: Before scheduling a service goal, each agent checks whether the goal is rated above a given threshold and therefore accessing a highly contested resource. It then checks whether any other agent has a service goal that is rated much more highly (where “much” is a parameterized value) than its own, and if so the agent will not schedule its service goal but instead idle to avoid a possible poaching activity.

Both heuristics show a significant performance increase with the addition of the anti-poaching coordination mechanism. Figure 4 shows the change in total tardiness observed in our experiments, and table 3 shows the mean change in total tardiness along with the significance value p for the matched-pair t -test.



(a)



(b)

Figure 5. Change in performance due to the addition of the gate coordination mechanism to the (a) local and (b) texture heuristics.

It is interesting to note that the anti-poaching mechanism improves performance for the local heuristic as well as the texture heuristic. Our mechanism is based on exchanging information about agents’ goal ratings and assumes that these ratings are comparable so that the decision about which agent has the more constrained goal is correct. With the texture-based heuristic, agents are able to build a view of global resource constraints and thus ratings are comparable. With the local-view heuristic, it is not immediately obvious that ratings are comparable. However, an agent with a locally highly constrained resource (and corresponding highly rated goal) will benefit from the opportunity to rapidly transmit requests regarding that resource to other agents.

4.2. Gate coordination mechanism

In our AGSS domain, distraction is primarily a problem when agents are distracted from scheduling gates early on in a scheduling episode. To prevent this distraction effect, we have added a gate coordination mechanism to Dis-ARM which simply disallows scheduling of resources other than gates until all agents have finished their gate assignments. Obviously, a similar heuristic can be developed for any sufficiently critical prerequisite operation in a scheduling or planning environment.

Both local and texture-based heuristics show a signifi-

Heur.	Coord. mech.	Δ_{TT}	p
Local	Anti-poach	-17.1	0.0011
Texture	Anti-poach	-11.3	0.033
Local	Gate	-11.6	0.038
Texture	Gate	-11.7	0.033

Table 3. Mean change in performance due to the addition of coordination mechanisms.

Coord. mech.	Δ_{TT}	p
none	-24.4	0.001
Anti-poach	-24.6	0.0001
Gate	-18.6	0.0001

Table 4. Mean difference in performance between the local and texture heuristics with and without coordination mechanisms.

cant performance increase with the addition of the gate coordination mechanism. Figure 5 shows the change in total tardiness observed in our experiments, and table 3 shows the mean change in total tardiness along with the significance value p for the matched-pair t -test.

4.3. Effectiveness of texture measures

In previous work, Neiman *et. al.* have shown the effectiveness of using the information contained in the communicated texture measures for scheduling [9, 10]. Here, we present a similar comparison of scheduling performance with and without the use of texture measures. The following data are the same as those presented in the previous two sections, but are displayed so that the texture-based heuristic and local heuristic are directly compared.

Figure 6 shows the difference in performance between the local and texture heuristics with and without additional coordination mechanisms. Even though the local heuristic does gain by using the additional mechanisms, the texture heuristic still performs significantly better when using the same coordination mechanisms. Table 4 shows the mean difference in total tardiness and significance value p for the three cases.

4.4. Effect of mechanisms on parallelism

While our coordination mechanisms decrease total tardiness in flight schedules, they also decrease parallelism which in turn may lead to increased run times. We have found that on average, run times in fact do not increase and in some cases even decrease. This is due to fewer backtracking episodes; by enforcing idleness in agents that may be about to poach or distract, the mechanisms reduce the number of mistakes that must be undone later. The amount

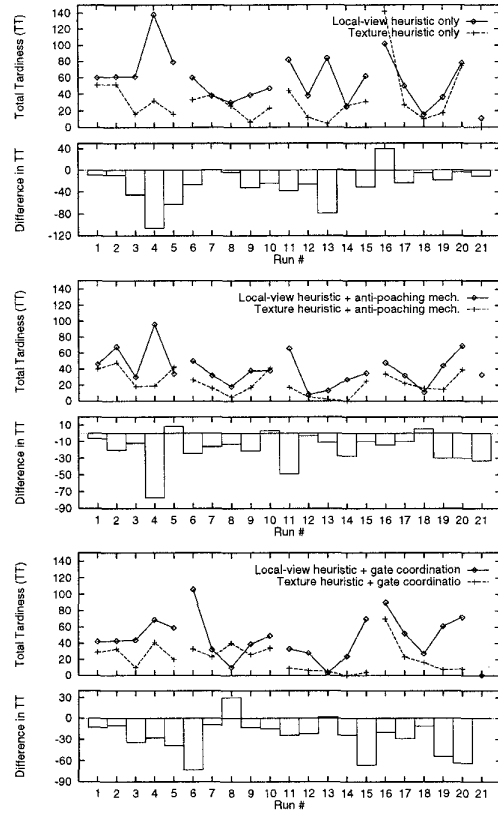


Figure 6. Difference in performance between the local and texture heuristics with and without coordination mechanisms.

of time spent idle is compensated by avoiding costly chains of backtracking.

Table 5 shows the mean change in the total number of knowledge source (KS) executions (a measure of system run time) and the mean change in number of KS executions that are preemptions (a measure of backtracking) due to the addition of each mechanism for both heuristics tested. The table also includes the p -value significance for the paired t -test for each condition.

We see that in most cases, there is a decrease in system run time and in the amount of backtracking performed. One exception was that the addition of the anti-poaching mechanism to the local heuristic did not significantly change the run time or amount of backtracking performed. This is not unexpected because the anti-poaching mechanism is only truly effective if the underlying system makes use of non-local information about goal ratings.

Heur./Mech.	$\Delta_{KS\ total}$	p	$\Delta_{KS\ preempt}$	p
Local/Anti-poach	41.4	0.6	2.2	0.7
Local/Gate	-77.5	0.002	-5.1	0.04
Texture/Anti-poach	-99.5	0.04	-9.9	0.02
Texture/Gate	-112.1	0.003	-7.8	0.004

Table 5. Mean change in total number of KS executions and mean change in number of KS executions that are preemptions.

5. Conclusions and future work

In an attempt to understand better the issues involved in distributed scheduling we have investigated in detail the dynamic behavior of the Dis-DSS scheduler. In doing so, we have observed poaching and distraction events affecting the performance of the distributed scheduling system and have developed simple mechanisms to reduce or eliminate such events. Our mechanisms suggest that in addition to state information about resource availability, agents need to model the likely future actions of other agents.

While our mechanisms have been shown to increase schedule quality, they are also highly serializing. By enforcing idle time for agents that are about to poach or distract, the mechanisms we have described reduce the parallelism that is the *raison d'être* for distributed systems. Although we have shown that this reduction in parallelism does not increase the system run time, we hope to develop more sophisticated mechanisms that prevent agents from poaching or distracting while allowing them to pursue other scheduling activities that will not negatively affect other agents.

We also plan to develop a more general model of distributed scheduling that can describe poaching and distraction behaviors in a domain-independent way. We hope to investigate related work such as Decker and Lesser [2], which uses a commitment framework for coordinating agents, and Liu and Sycara [8], which augments standard dispatch scheduling with a look-ahead coordination mechanism.

Finally, the complexity of a real-world domain such as the AGSS has limited us to studying relatively small problem instances. A better understanding of distributed systems may be gained by looking at problem instances with many more orders and scheduling agents. In the near term, we plan to continue testing our system on additional sets of orders to give us a better understanding of the issues in distributed scheduling.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. IRI-9321324 and No. IRI-9523419. Any opinions, findings, and conclusions or recommendations expressed in this material are

those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

- [1] K. S. Decker, A. J. Garvey, M. A. Humphrey, and V. R. Lesser. Control heuristics for scheduling in a parallel blackboard system. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(2), 1993.
- [2] K. S. Decker and V. R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems*, San Francisco, July 1995.
- [3] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 36(11), 1987.
- [4] M. S. Fox. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, Dec. 1983.
- [5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York, 1979.
- [6] D. W. Hildum. *Flexibility in a Knowledge-Based System for Solving Dynamic Resource-Constrained Scheduling Problems*. PhD thesis, Computer Science Department, University of Massachusetts, Amherst, MA, May 1994.
- [7] V. R. Lesser and L. D. Erman. Distributed interpretation: A model and an experiment. *IEEE Transactions on Computers – Special Issue on Distributed Processing*, C-29(12):1144–1163, Dec. 1980.
- [8] J.-S. Liu and K. P. Sycara. Multiagent coordination in tightly coupled task scheduling. In *Proceedings of the Second International Conference on Multi-Agent Systems*, Kyoto, Japan, Dec. 1996.
- [9] D. E. Neiman, D. W. Hildum, V. R. Lesser, and T. W. Sandholm. Exploiting meta-level information in a distributed scheduling system. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA, July 1994.
- [10] D. E. Neiman and V. R. Lesser. A cooperative repair method for a distributed scheduling system. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, Edinburgh, Scotland, May 1996.
- [11] D. E. Neiman and V. R. Lesser. Combining coordination and repair strategies in a distributed scheduling system. Technical Report 97-19, Computer Science Department, University of Massachusetts, Amherst, Nov. 1997.
- [12] N. Sadeh. *Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, Mar. 1991.
- [13] S. F. Smith, M. S. Fox, and P. S. Ow. Constructing and maintaining detailed production plans: Investigations into the development of knowledge-based factory scheduling systems. *AI Magazine*, 7(4):45–61, Fall 1986.
- [14] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Distributed constrained heuristic search. *IEEE Transactions on System, Man and Cybernetics*, 21(6), Dec. 1991.