Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control. 1990. Morgan Kaufmann Pub.

# Managing Resource Allocation in Multi-Agent Time-Constrained Domains

**Katia Sycara, Steve Roth, Norman Sadeh, Mark Fox**

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

**Abstract**

We present an approach to perform asynchronous, opportunistic, constraint-directed search in multi-agent time-bound, and resource limited domains. Such domains are extremely complex because of the presence of temporal and resource constraints that give rise to tightly interacting subproblems. In a distributed environment lacking a global system view and global control, the complexity increases further. Our approach relies on a set of *textures* of the problem space being searched. Textures provide a probabilistic, graph theoretic definition of the complexity and importance of decisions in the local problem space of each agent. In other words, they provide sophisticated local control. In addition, textures provide good predictive measures of the impact of local decisions on system goals. As a result, textures can be used to make control decisions that significantly reduce the amount of search required to solve complex distributed problems. We explore the utility of the approach in the context of cooperative multi-agent job-shop scheduling.

## 1. Introduction

In this paper we present mechanisms to enable efficient distributed search for multi-agent, time-bound and resource-limited problems. Such problems are characterized by the presence of temporal precedence constraints and resource constraints. These constraints result in conflicts over the use of shared resources and make the local decisions of distributed agents highly interdependent and interacting. Our investigation is conducted in the domain of job-shop scheduling. Our work addresses concerns in three research areas: (1) managing resource allocation in multi-agent planning, (2) constraint satisfaction, and (3) job-shop scheduling. Research in multi-agent planning has primarily focused on problems where agents contend only for computational resources, such as computer time and communication bandwidth (e.g., [Cammarata 83, Durfee 87a]). In most real world situations, however, allocation of (non-computational) resources that are needed by a planner to carry out actions in a plan is of central concern. Conry [Conry 86] has investigated (non-computational) static resource allocation not involving temporal constraints. The constraint satisfaction research community has investigated the efficiency of heuristics for incrementally building a solution to a constraint satisfaction problem by instantiating one variable after another within a single agent setting [Haralick 80, Mackworth 85, Purdom 83, Dechter 88]. Job-shop scheduling has been the subject of intense investigation by both Operations Research and AI communities (e.g., [Smith 85, Ow et. al. 88, Baker 74, French 82, Rinnooy Kan 76]). With few exceptions [Parunak 86, Smith&Hynynen 87], there has been almost no research in distributed scheduling. Prosser [Prosser 89] has investigated job-shop scheduling within a hierarchical distributed architecture where the high level agent has a global view and can act as conflict arbiter. In our system, the agents form a *heterarchy*, where no agent has a global view of the problem and actions of others. We provide mechanisms both for conflict avoidance and conflict resolution.

Our model enables a set of agents to structure their individual agent problem space and focus their attention during search so as to optimize decisions in the global search space. The beneficial effects of sophisticated local control on the coordination of distributed problem solving have been recognized by prior research [Durfee 87a, Durfee 87b]. Our approach, based on problem space *textures* [Fox 89], allows agents to make rapid, intelligent local decisions without the need of excessive information exchange or the availability of detailed models of each other's problem solving activities. Our hypothesis is that these textures provide good predictive measures of the impact of local decisions on system goals and constitute abstract information summaries of expectations concerning the decision making activities of other agents. Basing local decisions on such predictive measures is very important in distributed problem solving by opportunistic scheduling agents. Since the agents operate in an asynchronous and opportunistic manner, and since each local decision

[1]This research has been supported, in part, by the Defense Advance Research Projects Agency under contract #F30602-88-C-0001, and in part by grants from McDonnell Aircraft Company and Digital Equipment Corporation.

interacts with subsequent decisions of other agents, each agent must predict and take into consideration in its local decision making the future resource needs and problem solving behavior of other agents.

## 2. The Distributed Scheduling Problem

The scheduling task can be described as assigning resources to the activities present in a plan over time in a consistent manner, i.e., so as to avoid the violation of resource and precedence constraints. In our model, a group of autonomous opportunistic schedulers build a schedule in order to synchronize their activities to avoid and resolve conflicts. The schedule is built in a cooperative fashion through local computation and communication. There is no single agent with a global system view, nor any agents whose role is coordination. In distributed job-shop scheduling, each agent has a set of orders to schedule on a given set of resources. Each order consists of a set of activities (operations) to be scheduled according to a process plan which specifies a partial ordering among these activities. Additionally, an order has a release date and a due date. Each activity also requires one or several resources, for each of which there may be one or several substitutable resources. There is a finite number of resources available in the system. Some resources are only required by one agent, and are said to be *local* to that agent. Other resources are *shared*, in the sense that they may be allocated to different agents at different times[2].

We distinguish between two types of constraints: activity precedence constraints and capacity constraints. The activity precedence constraints together with the order release dates and due dates restrict the set of acceptable start times of each activity. Capacity constraints restrict the number of activities that can be allocated to a resource at one time. Typically the limited capacity of the resources induces interactions between orders competing for the possession of the same resource at the same time. In such an environment, schedules are constructed in an *incremental fashion*. Agents make local decisions about assignments of resources to particular activities at particular time intervals and a complete schedule for an order is formed by incrementally merging partial schedules for the order. If the merging of partial schedules results in constraint violations, the resulting schedule is infeasible.

Distributed scheduling has the following characteristics:

- To achieve global solutions, agents must make consistent allocations of resources needed to perform system activities. Conflicts in the system arise due to contention over optimal allocation of limited capacity shared resources.
- Because of conflicts over shared resources it is impossible for each agent to optimize the scheduling of its assigned orders using only local information.
- Due to limited communication bandwidth, it is not possible to exchange detailed constraint information during problem solving.
- All the given orders have to be scheduled. In other words, agents cannot drop any local goals. In addition, constraints cannot be relaxed (e.g., precedence constraints among the operations of an order, resource capacity constraints, and due dates).
- Because of the tightly interacting nature of scheduling decisions, an agent's problem solving context is rapidly changing. Moreover, an agent's decisions can produce constraint violations for other agents which may lead to backtracking. Backtracking can have major ripple effects on the multi-agent system since it may invalidate resource reservations that other agents have made.

A consequence of the above characteristics is that ag need methods to deal efficiently with incom information and a rapidly changing problem sol context. In addition, agents must maintain coh behavior [Durfee 87b] in a *heterarchical* setting. address these requirements, our approach gives the ag mechanisms to enable them to accomplish the follow (1) predict and evaluate the impact of local decision global system goals, (2) develop and communicate i concise form robust expectations and predictions about resource needs and decision-making behavior of o agents, (3) avoid and resolve conflicts over resources time intervals, and (4) help focus the attention of the ag opportunistically on parts of their search space where i expected that good solutions, in terms both of sched quality and minimal interactions, will be found. Th mechanisms, based on problem textures, result in sea and communication efficiency.

## 3. Constrained Heuristic Search

Our approach to scheduling relies on the combination local constraint propagation techniques with texture-ba heuristic search. We have developed a formal model this search mechanism which we call Constrained Heuris Search (CHS) [Fox 89]. CHS provides a methodology solving Constraint Satisfaction Problems (CSPs) a

[2]This model mirrors actual factory floor situations where the factory is divided into work areas that might share resources, such as machines, fixtures and operators in order to process orders.

Constrained Optimization Problem (COPs). A CSP is defined by a set of variables, each with a predefined domain of possible values, and a set of constraints restricting the values that can simultaneously be assigned to these variables [Montanari 71, Mackworth 77, Dechter 88]. A solution to a CSP is a complete set of assignments that satisfies all the problem constraints. COPs are CSPs with an objective function to be optimized. The general CSP is a well-known NP-complete problem [Garey 79]. There are however classes of CSPs and COPs that do not belong to NP, and for which efficient algorithms exist. The CHS methodology is meant for those CSPs/COPs for which there is no efficient algorithm. A general paradigm for solving these problems consists in using Backtrack Search (BT) [Golomb 65, Bitner 75]. BT is an enumerative technique that incrementally builds a solution by instantiating one variable after another. Each time a new variable is instantiated, a new search state is created that corresponds to a more complete partial solution. If, in the process of building a solution, BT generates a partial solution that it cannot complete (because of constraint incompatibility), it has to undo one or several earlier decisions. Partial solutions that cannot be completed are often referred to as deadend states (in the search space).

Because the general CSP is NP-complete, BT may require exponential time in the *worst-case*. CHS provides a methodology to reduce the *average* complexity of BT by interleaving search with *local constraint propagation* and the computation of *texture-based heuristics*. Local constraint propagation techniques are used to prune the search space from alternatives that have become impossible due to earlier decisions made to reach the current search state. By propagating the effects of earlier commitments as soon as possible, CHS reduces the chances of making decisions that are incompatible with these earlier commitments [Mackworth 85]. Typically, pruning the search space can only be done efficiently on a local basis [Nadel 88]. Hence local constraint propagation techniques are not sufficient to guarantee backtrack-free search. In order to avoid backtracking as much as possible as well as reduce the impact of backtracking when it cannot be avoided, CHS analyzes the pruned problem space in order to determine critical variables, promising values for these variables, promising search states to backtrack to, etc. The results of this analysis are summarized in a set of textures that characterize different types of constraint interactions in the search space. These textures are operationalized by a set of heuristics to decide which variable to instantiate next (so-called variable ordering heuristics), which value to assign to a variable (so-called value ordering heuristics), which assignment to undo in order to recover from a deadend, etc.

In the factory scheduling domain, variables are activities whose values are reservations consisting of a start time and a set of resources (e.g. a human operator, a milling machine, and a set of fixtures). Local constraint propagation techniques are used to identify reservations that have become unavailable for an unscheduled activity due to the scheduling of another activity (e.g. a resource that has been allocated to an activity over some time interval, or a start time that has become infeasible due to the scheduling of an earlier activity in a process plan). Within this context, texture-based heuristics are concerned with such decisions as which activity to schedule next, which reservation to assign to an activity, which reservation assignments to undo if the current partial schedule cannot be completed.

## 4. Distributed CHS Scheduling

The model concerns a set of scheduling agents, $\Gamma=\{\alpha, \beta, ...\}$. Each agent $\alpha$ is responsible for the scheduling of a set of orders $O^{\alpha}=\{o_1^{\alpha},...,o_{N_\alpha}^{\alpha}\}$. Each order $o_l^{\alpha}$ consists of a set of activities $A^{l\alpha}=\{A_1^{l\alpha},...,A_{n_{l\alpha}}^{l\alpha}\}$ to be scheduled according to a process plan (i.e. process routing) which specifies a partial ordering among these activities (e.g. $A_p^{l\alpha}$ BEFORE $A_q^{l\alpha}$). Additionally an order has a release date and a latest acceptable completion date, which may actually be later than the ideal due date. Each activity $A_k^{l\alpha}$ also requires one or several resources $R_{ki}^{l\alpha}$ $(1 \le i \le p_k^{l\alpha})$, for each of which there may be one or several alternatives (i.e. substitutable resources) $R_{kij}^{l\alpha}$ $(1 \le j \le q_{ki}^{l\alpha})$. There is a finite number of resources available in the system. Some resources are only required by one agent, and are said to be *local* to that agent. Other resources are *shared*, in the sense that they may be allocated to different agents at different times.

We distinguish between two types of constraints: activity precedence constraints and capacity constraints. The activity precedence constraints together with the order release dates and latest acceptable completion dates restrict the set of acceptable start times of each activity. The capacity constraints restrict the number of activities that a resource can be allocated to at any moment in time to the capacity of that resource. For the sake of simplicity, we only consider resources with unary capacity in this paper. Typically the limited capacity of the resources induces interactions between orders competing for the possession of the same resource at the same time. These interactions can take place either between the order of a same agent or between the orders of different agents.

With each activity, we associate preference functions that map each possible start time and each possible

resource alternative onto a preference. These preferences [Fox 83, Sadeh 88] arise from global organizational goals such as reducing order tardiness (i.e. meeting due dates), reducing order earliness (i.e. finished goods inventory), reducing order flowtime (i.e. in-process inventory), using accurate machines, performing some activities during some shifts rather than others, etc. In the cooperative setting assumed in this paper, the sum of these preferences over all the agents in the system and over all the activities to be scheduled by each of these agents defines a common objective function to be optimized. The sum of these preferences over all the activities under the responsibility of a single agent can be seen as the agent's local view of the global objective function. In other words, the global objective function is not known by any single agent. Furthermore, because they compete for a set of shared resources, it is not sufficient for an agent to try to optimize his own local preferences. Instead, agents need to consider the preferences of other agents when they schedule their activities. This is accomplished via a communication protocol described in section 6.

## 4.1. Activity-based Scheduling

In our model we view each activity $A_k^{l\alpha}$ as an aggregate *variable* (or vector of variables). A *value* is a reservation for an activity. It consists of a start time and a set of resources for that activity (i.e. one resource $R_{kij}^{l\alpha}$ for each resource requirement $R_{ki}^{l\alpha}$ of $A_k^{l\alpha}$, $1 \le i \le p_k^{l\alpha}$).

Each agent asynchronously builds a schedule for the orders he has been assigned. This is done incrementally by iteratively selecting an activity to be scheduled and a reservation for that activity. Each time a new activity is scheduled, new constraints are added to the agent's initial scheduling constraints that reflect the new activity reservation. These new constraints are then propagated (local constraint propagation step). If an inconsistency (i.e. constraint violation) is detected during propagation, the system backtracks. Otherwise the scheduler moves on and looks for a new activity to schedule and a reservation for that activity. The process goes on until all activities have been successfully scheduled.

If an agent could always make sure that the reservation that he is going to assign to an activity will not result in some constraint violation forcing him or other agents to undo earlier decisions, scheduling could be performed without backtracking. Because scheduling is NP-hard, it is commonly believed that such look-ahead cannot be performed efficiently. The most efficient constraint propagation techniques developed so far [LePape&Smith 87] for scheduling do not guarantee total consistency. In other words the reservation assigned by an agent to an activity may force other agents or the agent hims backtrack later on[3]. Consequently it is important to search in a way that reduces the chances of havi backtrack and minimizes the work to be undone backtracking occurs. This is accomplished via techniques, known as *variable* (i.e. activity) and *valu* reservation) ordering heuristics.

The variable ordering heuristic assigns a *criti measure* to each unscheduled activity; *the activity wi highest criticality is scheduled first.* The criti measure approximates the likelihood that the activity be involved in a conflict. The only conflicts tha accounted for in this measure are the ones that cann prevented by the constraint propagation mechanism. scheduling his most critical activity first, an agent re his chances of wasting time building partial schedule cannot be completed (i.e. it will reduce both the frequ and the damage of backtracking). The value ord heuristic attempts to leave enough options open t activities that have not yet been scheduled in ord reduce the chances of backtracking. This is don assigning a *goodness* measure to each possible reserv of the activity to be scheduled. Both activity criticality value goodness are examples of *texture measures.* next two paragraphs briefly describe both of measures[4].

### 4.1.1. Variable Ordering

Each agent's constraint propagation mechanism is b on the technique described in [LePape&Smith 87 always ensures that unscheduled activities within an c can be scheduled without violating activity preced constraints. This is not the case however for capa constraints: there are situations with insufficient capa that may go undetected by this constraint propaga technique. Accordingly a critical activity is one wh resource requirements are likely to conflict with resource requirements of other activities. [Sadeh 88, Sa 89] describes a technique to identify such activities. technique starts by building for each unscheduled activi probabilistic *activity demand*. An activity $A_k^{l\alpha}$'s demand a resource $R_{kij}^{l\alpha}$ at time $t$ is determined by the ratio reservations that remain possible for $A_k^{l\alpha}$ and require us $R_{kij}^{l\alpha}$ at time $t$ over the total number of reservations

[3]This is already the case in the centralized version of the schedu problem. Because of the additional cost of communication it is even n so in the distributed case.

[4]For a more complete description of these measures, the reade referred to [Sadeh 90].

remain possible for $A_k^{l\alpha}$. Clearly activities with many possible start times and resource reservations tend to have smaller demands at any moment in time, while activities with fewer possible reservations tend to have higher ones. In a second step, each agent aggregates his activity demands as a function of time, thereby obtaining his *agent demand*. This demand reflects the need of the agent for a resource as a function of time, given the activities that he still needs to schedule[5]. Finally, for each shared resource, agent demands are aggregated for the whole system thereby producing *aggregate* demands that indicate the degree of contention among agents for each of the (shared) resources in the system as a function of time. Time intervals over which a resource's aggregate demand is very high correspond to violations of capacity constraints that are likely to go undetected by the constraint propagation mechanism. The contribution of an activity's demand to the aggregate demand for a resource over a highly contended-for time interval reflects the reliance of the activity on the possession of that resource/ time interval. It is taken to be the *criticality of the activity*.

To choose the next activity to schedule, each agent looks among the resource/ time intervals that he may need and selects the one with highest aggregate demand. He then picks his activity with the highest criticality) to the aggregate demand for that resource/time interval. In other words, each agent looks for the resource/time interval over which he has some demand that the most likely to be involved in a capacity constraint violation. He then picks his activity with the highest probability of being involved in the conflict.

### .2. Value Ordering

Once an agent has selected an activity to schedule next, must decide which reservation to assign to that activity. re several strategies can be considered. In particular, we inguish between two extreme strategies: (1) a least straining value ordering strategy (LCV) and (2) a edy value ordering strategy (GV). Under LCV an agent select the reservation that will be the least constraining to itself and to other agents. LCV is a mechanism for *iding conflicts* over resources and over time intervals. heuristic can be viewed as resulting in *altruistic* avior on the part of an agent. Under the GV strategy, gent can select reservations based solely on its local erences, irrespectively of its own future needs as well hose of other agents. This heuristic results in *egotistic/myopic* behavior on the part of the agent. In this paper, we report experimental results obtained using the LCV value ordering strategy.

[5] tice that, an agent's demand at some time $t$ for a resource is ed by simply summing the demand of all his unscheduled activities $t$. Because these probabilities do not account for limited capacity, um may actually be larger than 1

## 5. Using Textures for Decentralized Scheduling

This section describes additional theoretical concerns and new mechanisms that arose in our application of the texture approach to decentralized, multiagent, resource-constrained scheduling. The issues that we addressed include:

- scheduling with incomplete information about the intentions and future behavior of other agents.
- scheduling with uncertain/changing information (i.e. even when detailed information regarding other agents' intentions is communicated, this information is not stable over time, since agents are scheduling asynchronously), and
- scheduling *without* the help of coordinating agents for avoiding conflicts and achieving global goals.

The following subsections describe our approach to addressing these issues.

### 5.1. Incomplete information

In a multi-agent system, complete information is unavailable to each agent about the constraints, partial plans/schedules and heuristic analyses of other agents. Incomplete information results because of limitations in the amount of inter-agent communication that can reasonably occur. Hence, some level of summarization and abstraction is needed. In our approach, summarization information is expressed in terms of the texture measures that have been effective for centralized problems. Specifically, we represent an agent's intentions with respect to resource use in terms of that *agent's demand density* for the resource for different time intervals. All agent densities are further abstracted to produce an *aggregate density*, which represents the system-wide expectation for resource utilization over time.

An important outcome of this approach is the ability to efficiently communicate those aspects of an agent's partial schedules which are most relevant to each of the other agents in a system, without the need to explicitly determine relevance. An element of a partial schedule is relevant to another agent if it influences the agent's expectations regarding the demand for resources the agent requires. Since the effects of most scheduling decisions indirectly

influence the computation of an agent's expected demand for shared resources, these implicitly include an abstraction of all relevant decision making elements.

### 5.2. Rapidly changing information

The continuous, asynchronous behavior of agents can reduce the validity of information they exchange, regardless of how complete that information may be. Therefore, an agent cannot depend on the certainty of information when it elects to use it, because other agents' decisions interact with its already constructed partial schedules as well as with its future scheduling decisions thus producing new expectations. In addition, because of the associated communication costs, agents cannot afford to communicate, update and evaluate information with every change that occurs. Hence the information communicated must remain predictive robustly in the face of communication lags.

There are several aspects of the texture approach that address the problem of rapid information obsolescence in asynchronous, multi-agent systems: First, texture measures produce relatively accurate early predictions of agent behavior, as long as expectations are communicated by all agents at the initiation of scheduling and constraints remain constant. Second, the uniform representation of expectations as densities and the incremental nature of activity scheduling allows changes in expectations to be incorporated as soon as they are received. Third, agents can monitor their current expectations to determine when these have changed significantly from those that were last communicated.

In the multi-agent system, other agents can make reservations throughout an agent's search, making it difficult to determine which set of previous reservations were responsible for a constraint violation when it is eventually detected. The task facing an agent at this point is to find the last set of reservations it made which, together with those made by other agents, does not violate constraints. A simple backtracking procedure will eventually find this state, but is extremely inefficient.

In order to deal with this problem, we have developed a variation of backjumping [Dechter 89] for uncertain, multi-agent environments. In our approach, backjumping involves iteratively undoing each activity's scheduled reservation and determining whether constraint violations remain, until the set of acceptable activity reservations has been partitioned. No alternative values are tried for any one activity until this set has been determined. This procedure avoids the inefficient testing of alternate values for variables when, in fact, violations already exist for values assigned to previously addressed variables. Our version of backjumping locates the appropriate search point with computation that is just a linear function of the number of variables traversed, a tremendous saving over chronological backtracking.

### 5.3. Absence of explicit coordination

Coordination within the texture approach to multi-agent scheduling is achieved through mutual acceptance and adherence to shared policies of decision-making. In our system, the goal of supporting other agents' attempts to achieve a solution to their portions of the global scheduling problem is realized through three policies. First, agents use information about other agents' expectations to avoid over-constraining them through the application of LCV value ordering heuristics. Second, reservations for resources are granted without contest when requested by an agent (i.e. reservations granted on a first-come, first served basis). Reservations are also surrendered promptly by agents if they decide not to use them as a result of local constraint violations. Third, once an agent has made a reservation, it is not required to surrender it i.e., no provision is made for one agent to request another to backtrack. An important principle is that all agents assume that the global good is best realized through the application of these policies and therefore, do not depart from them to maximize local objective functions.

## 6. A Communication Protocol for Distributed Scheduling

The agents make decisions using local available knowledge as well as information communicated by the other agents. In our model, resources are passive objects that are *monitored* by active agents. Each resource has a monitoring agent and each agent monitors one or more resources. Thus, monitoring responsibility is distributed among many agents. Monitoring resources does not give an agent either a global view or preferential treatment concerning the allocation of the monitored resources but is simply a mechanism that enables agents to perform load balancing in bookeeping efforts and efficient detection of capacity constraint violations. Since there is no single agent that has a global system view, the allocation of the shared resources must be done by collaboration of the agents that require these resources (one of which is the monitoring agent).

The multi-agent communication protocol is as follows:

1. Each agent determines required resources by checking the process plans for the orders it has to schedule. It sends a message to each monitoring agent informing it that it will be using shared resources.

2. Each agent calculates its demand profile for the resources (local and shared) that it needs.

3. Each agent determines whether its new demand profiles differ significantly from the ones it sent previously for shared resources. If its demand has changed, an agent will send it to the monitoring agent.

4. The monitoring agent for each resource combines all *agent demands* when they are received and communicates the *aggregate demand* to all agents which share the resource[6].

5. Each agent uses the most recent aggregate demand it has received to find its most critical resource/time-interval pair and its most critical activity (the one with the greatest demand on this resource for this time interval). Since agents in general need to use a resource for different time intervals, the most critical activity and time interval for a resource will in general be different for different agents. The agent communicates this reservation request to the resource's monitoring agent and awaits a response.

6. The monitoring agent, upon receiving these reservation requests, checks the calendar of the resource it is monitoring to find out whether the requested intervals are available. There are two cases:

- If the resource is available for a requested time interval, the monitoring agent of the resource (a) communicates "Reservation OK" to the requesting agent, (b) marks the reservation on the resource calendar, and (c) communicates the reservation to all concerned agents (i.e. the agents that had sent positive demands on the resource).
- If the resource had already been reserved for the requested interval, the request is denied. The agent whose request was denied will then attempt to substitute another reservation, if any others are feasible, or otherwise perform backjumping.

7. Upon receipt of a message indicating its request was granted, an agent will perform consistency checking to determine whether any constraint violations have occurred. If none are detected, the agent proceeds to step 2. Otherwise, backjumping occurs with undoing of reservations until a search state is reached which does not cause constraint violations. Any reservations which were undone during this phase are communicated to the monitor for distribution to other agents. After a consistent state is reached, the agent proceeds to step 2.

The system terminates when all activities of all agents have been scheduled i.e. when all demands on resources become zero. In this version of the protocol we assume that reservations are not changed because of backtracking.

[6]With the exception of the first time demands are exchanged, agents do not wait for aggregate demands to be computed and returned prior to continuing their scheduling operations (although they can postpone further scheduling if desired).

## 7. Experimental Results

The main goal of our experiments was to determine the feasibility of the texture approach to multi-agent scheduling across a number of different scheduling experiments and across a variety of system configurations. We have developed a testbed and performed experiments with 1-, 2- and 3-agent configurations. The experiments were run asynchronously on a number of machines corresponding to each of the agent configurations. In addition, we wanted to test particular mechanisms and parameters that influence system performance. In particular, our experiments considered:

- the effects of agents' incomplete knowledge of each other's plans (i.e. the robustness of texture measures when aggregated across multiple agents and with the resulting loss of detailed information),
- the effects of rapidly changing expectations on performance (i.e. the robustness of these measures with respect to delays in the communication of densities),
- the consequences of asynchronous scheduling (e.g., asynchronous use of variable-ordering strategies) without external coordination.

The experiments summarized here were created from problems found to be difficult in previous research on centralized scheduling [Sadeh 89] and they reflect system performance with respect to search efficiency rather than schedule optimality. We selected problems on which more traditional constraint satisfaction approaches performed poorly (e.g. Purdom's dynamic search rearrangement technique [Sadeh 89, Purdom 83]). The problems were also selected and distributed across the agents in a way that maximized *resource coupling* within orders and across agents. The problems were constructed so that a change in reservation for any activity or resource would influence expectations for every other.

All experimental problems were selected so that orders could be distributed evenly between two agents, all resources were shared by the two agents (high inter-agent resource coupling), every order used all resources (high intra-order resource coupling), and problems ranged from 40 - 100 activities.

Over 100 experiments were run in order to vary several properties of each problem. The asynchrony in the system prevents exact replication of experiments. So, we repeated each experimental run a minimum of three times. If different runs of the same experiment produced wide

variations in the results, we repeated the experiment five times. The reported results (see figure) are the average of these runs. In each case, the dependent variable was the efficiency with which the scheduling system found a solution. Efficiency is expressed in terms of the total number of states needed to reach a solution. For example, for a problem with 40 activities, the minimum number of states needed to assign a reservation to each activity is 40. Every reservation that needed to be redone added an additional state to the total. This allowed comparing a 40-activity 1-agent problem to a pair of 20-activity problems solved simultaneously by 2 agents, or a 10, 15, 15 split of the 40 activities among three agents. There were 5 resources (all shared among all the agents). These resources were used by 8 orders, each having 5 activities.

Problem versions differed in several ways. First, to establish a baseline, we created a 1-agent system, which was similar to the 2-agent and 3-agent systems in every way, except that the aggregate densities were constructed from a single agent. This was still different from the original centralized system in that decisions were based on an *abstract aggregate demand profile* that did not include detailed information about the number of activities which contributed to the densities. Furthermore, we varied the frequency with which the aggregate was computed, thereby isolating the effect of uncertain expectations caused by infrequent and delayed communication of densities in the 2-agent and 3-agent systems.

Specifically, we implemented several simplified versions of the heuristic used by agents to determine when to communicate their changed densities. In the *minimum* delay condition, a single reservation on *any resource by any agent* initiated the exchange of densities for all resources. In the *increased* delay conditions, densities were exchanged *for each resource independently*, whenever N reservations were made on it, where N = 1, 3, and 5. This provided a way to observe the effects of wide ranges in communication bandwidth in the 2-agent and 3-agent systems and comparable conditions in a 1-agent system.

Another version of the 1-agent system was created which used a *semi-random* version of the variable-ordering heuristic. The goal was to isolate and assess the effects of less accurate variable ordering that might occur in a multi-agent system. Recall that variable ordering is performed in parallel in a multi-agent system (each agent selects the best activity to schedule from its subset of all activities which require a critical resource). Agents do not coordinate the selection of activities to schedule to ensure that the globally most critical ones are scheduled first. As a result, variable ordering is probably less effective than in a 1-agent system. The semi-random heuristic still selects activities to schedule from those which require the most critical resource/time-interval (which narrows the selection to a maximum of 20% of the activities in these problems). However, it then randomly selects from this subset, instead of selecting the activity with the greatest demand for the critical resource. Relative to completely random variable ordering, the semi-random condition is still highly selective in that only activities which use the most critical resource/time interval are considered. In fact, we found that random variable ordering resulted in terrible performance, even in the 1-agent case. Solutions were not found in over 500 states.

Two system versions were created to compare the use of backtracking and backjumping search techniques. As expected, the use of a backjumping strategy substantially reduced the search in the 2- and 3-agent systems. The results presented in the Figure are results of the backjumping version. The reported results are for a representative group of 40-activity experiments (8 orders, 5 activites per order, and 5 shared resources). The four curves represent the effects of increasing the delay (from 0 to 5) prior to initiating creation of aggregate demand densities for 1-, 2- and 3-agent configurations and for a 1-agent case with a semi-random variable ordering strategy (labelled 1-agent SR variable ordering in the figure).

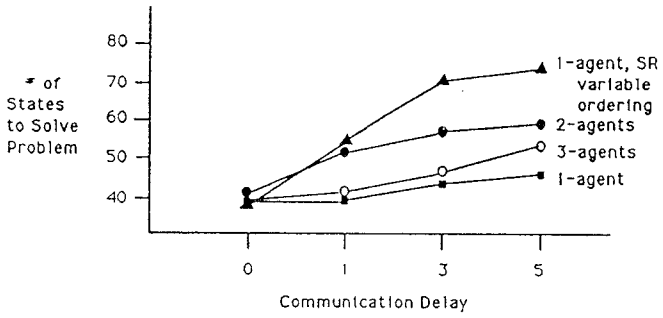


**Figure 7-1:** Experimental Comparisons of Distributed Scheduling Systems

The first important observation is that the use of abstracted texture measures was sufficient to allow near perfect performance (solving the problem in 40 states) when the texture information was updated frequently (minimum delay conditions, expressed as 0 on the x-axis) in all experimental configurations. This matches performance obtained in the original centralized scheduling system [Sadeh 89]. Thus, despite the incompleteness of information available in the 2- and 3-agent systems, texture measures provide satisfactory summarizations. Second, as expected, performance of the 2- and 3-agent systems does

deteriorate as the communication of changing texture information is delayed. Since current texture information is used to perform both variable and value ordering, it is likely that both these processes deteriorate. An interesting observation is that in this set of experiments, the 3-agent system did better in terms of search efficiency than the 2-agent system[7].

The effect of delaying communication/computation of demand densities is greater for the 2- and 3-agent systems than the 1-agent system. This interaction may reflect the compensatory relation between variable and value ordering observed in [Sadeh 89]. Note that 2- and 3-agent performance is still better than the semi-random condition, suggesting that variable ordering strategy is robust with respect to the conditions of the multi-agent environment (incomplete, changeable information and asynchronous behavior without external coordination).

## 8. Concluding Remarks

In this paper we have presented mechanisms to guide distributed search. The domain of investigation is distributed job-shop scheduling. In particular, we have presented measures of characteristics of a search space, called textures, that are used to focus the attention of agents during search and allow them to efficiently find scheduling solutions that satisfy all constraints. In addition, the textures express the impact of local decisions on system goals and allow agents to form expectations about the needs of others. This ability is critical in multi-agent complex environments, such as the factory floor, where agents have to plan under considerable uncertainty. We have presented two types of textures (activity criticality and value goodness), their operationalization into variable and value ordering heuristics and their use in distributed problem solving. In addition, a communication protocol that enables the agents to coordinate their decisions has been presented.

A testbed has been implemented that allows for experimentation with a variety of distributed protocols that use variable and value ordering heuristics. The testbed also provides unique opportunities to compare closely matched single- and multi-agent scheduling systems. This comparison helps establish baseline performance measures and isolate conditions that influence performance in multi-agent systems.

Our results demonstrated that a texture approach to multi-agent scheduling can produce search efficiency that approximates that of a centralized system, even for problems that are difficult for traditional approaches to constraint satisfaction. Furthermore, the texture approach proved to be robust in the face of decreasing communication frequency, thus substantially decreasing communication overhead.

[7]This was true in the majority of comparisons between the 2- and 3-agent systems. No easy generalization can be made, however, since in some of the experimental groupings, the 3-agent system performance was very bad in the increased delay conditions, whereas the 2-agent system performed with graceful deterioration for the corresponding increased delay conditions.

## References


[Baker 74] K.R. Baker. *Introduction to Sequencing and Scheduling.* Wiley, 1974.

[Bitner 75] J.R. Bitner and E.M. Reingold. Bakctrack Programming Techniques. *Communications of the ACM*18(11):651-655, 1975.

[Cammarata 83] Cammarata, S. McArthur, D. and Steeb, R. Strategies of cooperation in distributed problem solving. In *IJCAI-83*, Pages 767-770. IJCAI, Karlsruhe, W. Germany, 1983.

[Conry 86] Conry, S.E., Meyer, R.A., and Lesser, V.R. *Multistage Negotiation in Distributed Planning.* Technical Report COINS-86-67, COINS, University of Massachusetts, december, 1986.

[Dechter 88] Rina Dechter and Judea Pearl. Network-Based Heuristics for Constraint Satisfaction Problems. *Artificial Intelligence*34(1):1-38, 1988.

[Dechter 89] Dechter, R., and Meiri, I. Experimental Evaluation of Preprocessing Techniques in Constraint Satisfaction Problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Pages 271-277. American Association of Artificial Intelligence, Detroit, MI., August, 1989.

[Durfee 87a] Durfee, E.H.
*A Unified Approach to Dynamic Coordination: Plannign Actions and Interactions in a Distributed Problem Solving Network.*
PhD thesis, COINS, University of Massachusetts, 1987.

[Durfee 87b] Durfee, E.H, Lesser, V.R., and Corkill, D.D.
Coherent Cooperation Among Communicating Problem Solvers.
*IEEE Transactions on Computers*C(36):1275-1291, 1987.

[Fox 83] Fox, M.S.
*Constraint-Directed Search: A Case Study of Job Shop Scheduling.*
PhD thesis, Computer Science Department, Carnegie-Mellon University, 1983.

[Fox 89] Mark S. Fox, Norman Sadeh, and Can Baykan.
Constrained Heuristic Search.
In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Pages 309-315. 1989.

[French 82] S. French.
*Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop.*
Wiley, 1982.

[Garey 79] M.R. Garey and D.S. Johnson.
*Computers and Intractability: A Guide to the Theory of NP-Completeness.*
Freeman and Co., 1979.

[Golomb 65] Solomon W. Golomb and Leonard D. Baumert.
Backtrack Programming.
*Journal of the Association for Computing Machinery*12(4):516-524, 1965.

[Haralick 80] Robert M. Haralick and Gordon L. Elliott.
Increasing Tree Search Efficiency for Constraint Satisfaction Problems.
*Artificial Intelligence*14(3):263-313, 1980.

[LePape&Smith 87] LePape, C. and S.F. Smith.
Management of Temporal Constraints for Factory Scheduling.
In *Proceedings IFIP TC 8/WG 8.1 Working Conference on Temporal Aspects in Information Systems (TAIS 87)*, Elsevier Science Publishers, held in Sophia Antipolis, France, May, 1987.

[Mackworth 77]
Consistency in Networks of Relations.
*Artificial Intelligence*8(1):99-118, 1977.

[Mackworth 85] Mackworth, A.K., and Freuder, E.C.
The Complexity of some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems.
*Artificial Intelligence*25(1):65-74, 1985.

[Montanari 71] Ugo Montanari.
*Networks of Constraints: Fundamental Properties and Applications to Picture Processing.*
Technical Report , Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 1971.

[Nadel 88] Bernard Nadel.
Tree Search and Arc Consistency in Constraint Satisfaction Algorithms,
In L. Kanal and V. Kumar, *Search in Articial Intelligence*. Springer-Verlag, 1988.

[Ow et. al. 88] Ow, P.S., S.F. Smith, and A. Thiriez.
Reactive Plan Revision.
In *Proceedings AAAI-88*, St. Paul, Minnesota, August, 1988.

[Parunak 86] Parunak, H.V., P.W. Lozo, R. Judd, B.W. Irish.
A Distributed Heuristic Strategy for Material Transportation.
In *Proceedings 1986 Conference on Intelligent Systems and Machines*, Rochester, Michigan, 1986.

[Prosser 89] Prosser, P.
A Reactive Scheduling Agent.
In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Pages 1004-1009. American Association of Artificial Intelligence, Detroit, MI., August, 1989.

[Purdom 83] Paul W. Purdom, Jr.
Search Rearrangement Backtracking and Polynomial Average Time.
*Artificial Intelligence*21:117-133, 1983.

[Rinnooy Kan 76] A.H.G. Rinnooy Kan.
*Machine Scheduling Problems: Classification, complexity, and computations.*
PhD thesis, University of Amsterdam, 1976.

[Sadeh 88] N. Sadeh and M.S. Fox.
*Preference Propagation in Temporal/Capacity Constraint Graphs.*
Technical Report CMU-CS-88-193, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, 1988.
Also appears as Robotics Institute technical report CMU-RI-TR-89-2.

[Sadeh 89] N. Sadeh and M.S. Fox.
Focus of Attention in an Activity-based Scheduler.
In *Proceedings of the NASA Conference on Space Telerobotics*, 1989.

[Sadeh 90] N. Sadeh, and M.S. Fox.
Variable and Value Ordering Heuristics for Activity-based Job-shop Scheduling.
In *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, Hilton Head Island, S.C.*, 1990.

[Smith 85] Stephen F. Smith and Peng Si Ow.
The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks.
In *Proceedings of the Ninth International Conference on Artificial Intelligence*, Pages 1013-1015. 1985.

[Smith&Hynynen 87] Smith, S.F. and J.E. Hynynen.
Integrated Decentralization of Production Management: An Approach for Factory Scheduling.
In *Proceedings ASME Annual Winter Conference: Symposium on Integrated and Intelligent Manufacturing*, Boston, MA, December, 1987.