Exploiting Problem Structure for Distributed Constraint Optimization

JyiShane Liu and **Katia P. Sycara**
The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
e-mails: jsl@cs.cmu.edu, katia@cs.cmu.edu

Abstract

Distributed constraint optimization imposes considerable complexity in agents' coordinated search for an optimal solution. However, in many application domains, problems often exhibit special structures that can be exploited to facilitate more efficient problem solving. One of the most recurrent structures involves disparity among subproblems. We present a coordination mechanism, *Anchor&Ascend*, for distributed constraint optimization that takes advantage of disparity among subproblems to efficiently guide distributed local search for global optimality. The coordination mechanism assigns different overlapping subproblems to agents who must interact and iteratively converge on a solution. In particular, an anchor agent who conducts local best first search to optimize its subsolution interacts with the rest of the agents who perform distributed constraint satisfaction to enforce problem constraints and constraints imposed by the anchor agent. We focus our study on the well-known NP-complete job shop scheduling problem. We define and study two problem structure measures, disparity ratio and disparity composition ratio. We experimentally evaluated the effectiveness of the Anchor&Ascend mechanism on a suite of job shop scheduling problems over a wide range of values of disparity composition. Our experimental results show that (1) considerable advantage can be obtained by explicitly exploiting disparity (2) disparity composition ratio plays a more important role than disparity ratio in finding high quality solution with little computational cost.

Introduction

Constraint satisfaction (Mackworth 1987) provides a general framework for formalizing various AI problems such as scheduling, planning, etc., which are among the most commonly studied computational problems. A constraint satisfaction problem (CSP) involves a set of *variables* $X = \{x_1, x_2, \cdots, x_m\}$, each having a corresponding set of *domain values* $V = \{v_1, v_2, \cdots, v_m\}$, and a set of *constraints* $C = \{c_1, c_2, \cdots, c_n\}$ specifying which values of the variables are compatible with each other. A solution to a CSP is an assignment of values (an instantiation) to all variables, such that all constraints are satisfied. Recent work in DAI has considered the *distributed* CSPs (DCSPs) (Huhns & Bridgeland 1991) (Sycara *et al.* 1991) (Yokoo *et al.* 1992) (Liu & Sycara 1993) in which variables of a CSP are distributed among agents. Each agent has a subset of variables and coordinates with other agents in instantiating its variables so that a global solution can be found. DCSPs have been considered as a general framework for studying issues in agents' coordination.

In previous work (Liu & Sycara 1994), we developed a coordination mechanism, called *Constraint Partition and Coordinated Reaction (CP&CR)*, where problem constraints are partitioned by constraint type and constraint connectivity and are assigned to different agents. The agents' asynchronous local interactions achieve distributed constraint satisfaction. In this paper, we extend the work in (Liu & Sycara 1994) to the Constraint Optimization Problem (COP) in which a subset of the constraints are relaxed to achieve optimization of a given objective function. Distributed COPs introduce additional complexity in agents' coordination. However, in many application domains, problems often exhibit special structures that can be exploited to facilitate more efficient problem solving. One of the most recurrent structures involves *disparity* among subproblems. We present a coordination mechanism for distributed constraint optimization, called *Anchor&Ascend*, that takes advantage of disparity among subproblems to efficiently guide distributed local search for global optimality. Anchor&Ascend employs the notion of an anchor agent who conducts local optimization of its subsolution and interacts with the rest of the agents who perform constraint satisfaction through CP&CR to achieve global optimization. The idea of anchoring search by disparity is inspired by a well known strategy in constraint satisfaction algorithms (Purdom 1983) (Nudel 1983) where variables with the tightest constraints are instantiated first and used as anchors.

We evaluated Anchor&Ascend on the well known NP-complete (Garey & Johnson 1979) job shop scheduling problem. In general, job shop optimization for even small problem size is intractable and cannot be guaranteed. Hence the problem solving goal is to find high quality schedules with reasonable computational cost. We define two problem structure measures, disparity composition ratio and disparity ratio. We conduct computational experiments to study the behavior of the Anchor&Ascend procedure under different conditions of subproblem disparity. Our experimental results show that Anchor&Ascend is most effective under conditions of high disparity ratio and low disparity composition ratio. Exploiting these disparity measures to structure distributed coordination for constraint optimizations results in solutions of high quality with little computational cost. The results attest to the considerable advantage of exploiting problem structure in designing coordination mechanisms for distributed problem solving.

Job Shop Scheduling

Job shop scheduling (French 1982) involves synchronization of the completion of m jobs on n resources (machines). Each job J_i is composed of a sequence of activities (operations) $a_{ij}, j = 1, ..., n$, and can only be processed after its release date RD_i. Each activity a_{ij} has a specified duration dur_{ij} and requires the exclusive use of a designated resource for the duration of its processing. The problem (hard) constraints of job shop scheduling include (1) *activity temporal precedence* constraints, i.e., an activity must be finished before the next activity in the sequence for that job can be started, (2) *release date* constraints, i.e., the first activity of a job can only begin after the release date of the job, and (3) *resource capacity constraints*, i.e., resources have only unit processing capacity. A solution of the job shop scheduling problem is a feasible schedule, which assigns start times st_{ij} to each activity a_{ij}, that satisfies all problem constraints.

One of the commonly used objective function is the *weighted tardiness* in which each job is assigned a due date DD_i (soft constraints) and a tardiness weight w_i that introduces penalty if the job is finished later than DD_i. The tardiness $tard_i$ of a job J_i is either 0, when J_i is finished earlier than DD_i, or a distance between DD_i and the finish time of the last activity of J_i, when J_i is finished later than DD_i, i.e., $tard_i = max[0, (st_{in} + dur_{in} - DD_i)]$, where st_{in} is the start time of the last activity a_{in} in job J_i. The weighted tardiness cost c_i of a job J_i is the tardiness $tard_i$ of the job multiplied by its tardiness weight w_i, i.e., $c_i = tard_i \times w_i$. An optimal solution is a feasible schedule that minimizes the overall weighted tardiness cost, $C = \sum_{i=1}^{m} c_i$, where m is the number of jobs. Because of its tremendous complexity, job shop scheduling has been considered as one of the most difficult CSPs/COPs (Fox & Sadeh 1990).

Since each job J_i is assigned a release date RD_i, a due date DD_i, and together with the temporal precedence constraints between activities in the job, the notion of the *earliest/latest start times* (est_{ij}, lst_{ij}) of an activity a_{ij} arises, in which $est_{ij} = RD_i + \sum_{k=1}^{j-1} dur_{ik}$ and $lst_{ij} = DD_i - \sum_{k=j}^{n} dur_{ik}$. In order for a job J_i to be processed *after* RD_i, a_{ij} must not start earlier than est_{ij} (a hard constraint). In order for the job to be finished *before* DD_i, a_{ij} should not start later than lst_{ij}. In job shop scheduling COPs with tardiness as objective function, DD_i (and lst_{ij}) are relaxed. Note that whether a job would be finished before its due date depends only on the start time of the last activity a_{in} of the job.

In this paper, we focus on the *bottleneck job shops*. According to the definition given in (Morton & Pentico 1993), a bottleneck job shop is a subset of job shops in which jobs visit every resource exactly once. While their visiting sequences on other resources differ from each other, every job visits a bottleneck resource at the same sequence. In other words, every u-th activity of each job (a_{iu}) requires the use of a bottleneck resource. The processing times of activities on the bottleneck are on the average longer than those for other resources creating the biggest resource contention for bottleneck resources.

The domain of bottleneck job shops naturally involves disparity among subproblems (bottleneck resources versus non-bottleneck resources). We denote a bottleneck resource as BR_i, a non-bottleneck resource as NBR_i. The average processing time p_{av}^i of a resource (BR_i or NBR_i) is the average duration of activities requiring the use of the resource (BR_i or NBR_i). In addition, num_b is the number of bottleneck resources, and num_{nb} is the number of non-bottleneck resources in the shop. p_{AV}^b is the average p_{av}^i of bottleneck resources, i.e., $\sum_{i=1}^{num_b} p_{av}^i / num_b$. p_{AV}^{nb} is the average p_{av}^i of non-bottleneck resources, i.e., $\sum_{i=1}^{num_{nb}} p_{av}^i / num_{nb}$. In order to quantify disparity in bottleneck job shops, we define two disparity characteristics as follows.

Disparity Ratio is the ratio of the average p_{av}^i of bottleneck resources to the average p_{av}^i of non-bottleneck resources, i.e., p_{AV}^b / p_{AV}^{nb}.

Disparity Composition Ratio is the ratio of the number of bottleneck resources to the number of resources in the shop, i.e., $num_b / (num_b + num_{nb})$.

Disparity ratio and disparity composition ratio are two ways to quantify the disparity structure of the bottleneck job shops. In the following sections, we present a coordination mechanism, Anchor&Ascend, that exploits disparity among subproblems to direct local search for global optimality and examine the behavior of the coordination procedure over a range of disparity structures.

Coordination for Optimization

In the initial task categorization and distribution, CP&CR (Liu & Sycara 1994) for job shop constraint satisfaction assigns each resource to a *resource agent* responsible for enforcing capacity constraints on the resource, and each job to a *job agent* responsible for enforcing temporal precedence and release date constraints within the job. Resources are differentiated by comparing their average processing times. Resources with eminent average processing times are identified as bottleneck resources. An activity is governed by both a job agent and a resource agent, and each can change the start time of the activity in order to solve its own subproblem. Subsolutions of job agents that respect temporal precedence constraints may be partially invalidated by resource agents enforcing capacity constraints. Similarly, subsolutions of resource agents that respect capacity constraints may be partially invalidated by job agents enforcing temporal precedence constraints. Coordination information is exchanged between job and resource agents to facilitate an iterated modification process toward group convergence.

In Anchor&Ascend, one of the bottleneck resources is selected and assigned the role of *anchor (agent)* by heuristic, e.g., the last one in jobs' processing sequence. The non-bottleneck resources together with the not selected bottleneck resources will be referred to as regular resources. Suppose every u-th activity in each job uses the selected bottleneck resource. For each job, J_i, we define pre-anchor activities to be $\{a_{ij}\}, j = 1, ..., u-1$, the anchor activity to be a_{iu}, and post-anchor activities to be $\{a_{ij}\}, j = u+1, ..., n$. A valid subsolution for the anchor agent is a sequence of a_{iu} with specified start times st_{iu}, in which none of the intervals $(st_{iu}, st_{iu} + dur_{iu})$ overlaps with others and all $st_{iu} \geq est_{iu}$. The local tardiness objective function for the anchor agent is, $C_{anchor} = \sum_{i=1}^{m}\{w_i \times max[0, (st_{iu} - lst_{iu})]\} = \sum_{i=1}^{m} c_{iu}$, where c_{iu} is the cost of a_{iu}, m is the number of jobs and $lst_{iu} = DD_i - \sum_{k=u}^{n} dur_{ik}$. The local cost C_{anchor} is used by the anchor agent to evaluate its subsolution and estimate the global weighted tardiness cost of the current solution. In other words, we assume all the tardiness of jobs is due to tardiness on the bottleneck resource and we use c_{iu} to predict the cost c_i of the job J_i by assuming all post-anchor activities in the job will be processed with no delay, i.e., $st_{ij} = st_{iu} + \sum_{k=u}^{j-1} p_{ik}, i = 1, ..., m, j = u+1, ..., n$. Of course, any actual delay will certainly increase c_i and increase the overall weighted tardiness cost C, i.e., $C_{anchor} \leq C$.

The assumption that C_{anchor} is responsible for the tardiness cost leads to the following intuitions regarding design decisions of the Anchor&Ascend coordination procedure: (1) the anchor agent should seek to optimize its local *anchor subsolution*, (2) other agents should coordinate with the anchor agent to resolve constraint conflicts imposed by their own problem constraints and by the current anchor subsolution, and (3) the current anchor subsolution should persist as long as possible; only when it becomes apparent that, despite the efforts of the non-anchor agents to adjust their subsolutions, the current anchor subsolution cannot become compatible with the rest of the subsolution, only then should the anchor agent modify its subsolution. A modified anchor subsolution, then serves as an anchor for further problem solving. Hence, the Anchor&Ascend coordination mechanism (see Figure 1) consists of two interacting components. The anchor agent (selected bottleneck resource) conducts a local best first search to generate an anchor subsolution. This subsolution imposes additional constraints, called *anchor constraints*, on the feasible subsolutions of other agents. In particular, the assignment of $\{st_{iu}\}$ to $\{a_{iu}\}$ in the anchor subsolution constrains pre-anchor activities $a_{ij}, j = 1, ..., u-1$, of a_{iu} to be finished before st_{iu}. Therefore, for a subsolution of a non-anchor agent to be feasible and compatible with the anchor subsolution, each pre-anchor activity must respect an effective due date, $d_{ij} = st_{iu} - \sum_{k=j}^{u-1} dur_{ik}$ [1]. To generate subsolutions that are feasible and compatible with the (current) anchor subsolution, all other agents (regular resources and jobs) perform asynchronous distributed constraint satisfaction using CP&CR.

A current anchor subsolution does not get changed as long as it appears possible that CP&CR can result in subsolutions that are feasible and compatible with the current anchor subsolution. Therefore, $\{st_{iu}\}$ in an anchor subsolution is not changed until a locally maintained history that records the number of start time changes of a pre-anchor activity reaches a threshold value[2]. In that case, when a precedence constraint violation occurs involving a pre-anchor activity, the job agent violates the effective due date anchor constraint on the pre-anchor activity in order to enforce temporal precedence constraints by changing the start time of the anchor activity to a later start time. The violation of an anchor constraint forces the anchor agent to modify its current anchor subsolution through a local search guided by local modification operators to seek another (perhaps less desirable) local optimum. The procedure repeats till a global solution that is compatible with the current local optimal anchor subsolution is found. From the distributed control viewpoint, Anchor&Ascend is a hybrid procedure. The anchor agent is activated first. After an anchor subsolution has been generated, the rest of the agents interact asynchronously using CP&CR for constraint satisfaction.

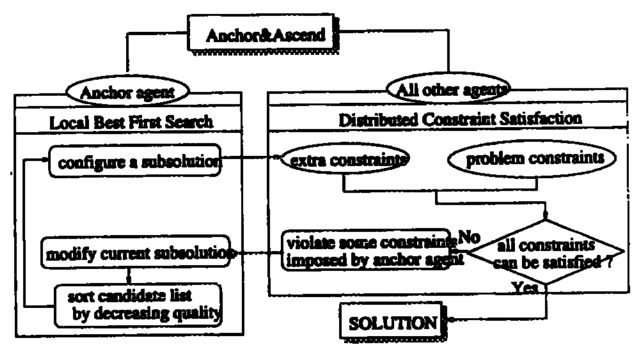

Figure 1: The Anchor&Ascend Coordination Mechanism

Anchor&Ascend controls the distributed local search for global optimization by assuming disparity among agents and going through a process of testing the feasibility of constructing a global solution based on different configurations of the anchor subsolution with *monotonically increased* objective costs. The completeness of the procedure depends on the completeness of the distributed constraint satisfaction session. Solution optimality depends on the procedure's completeness, the optimality of local optimization methods, and on how well the local anchor cost C_{anchor} reflects the global objective cost. We must emphasize that in job shop scheduling there is no guarantee for optimality for most

[1]Note that d_{ij} can be either less or greater than lst_{ij}.

[2]The value is set experimentally. It represents the amount of effort allowed for the job agents and the regular resource agents to find a compatible solution to the current anchor subsolution.

objective functions. In practice, the goal is to find a high quality solution with reasonable computational cost.

In the Anchor&Ascend process, the anchor agent iteratively configures an anchor subsolution until the coordinated local interactions of all other agents result in a globally feasible solution. The amount of search (conducted by the local interactions of agents) allowed for each configuration of the anchor subsolution has a fixed limit, which is controlled by a threshold value[3]. Therefore, the complexity of the Anchor&Ascend process is reduced to the complexity of the iterative process of configuring anchor subsolutions. The anchor agent takes a constant time to configure an anchor subsolution. Each anchor subsolution is a processing sequence on the anchor bottleneck resource. The number of possible processing sequences is the factorial of the number of activities that use the anchor bottleneck resource. Therefore, the worst case complexity of the Anchor&Ascend approach is approximately $m!$, where m is the number of activities that use the anchor bottleneck resource, i.e., the number of jobs in the problem. This is considerably less than the general complexity of the job shop, which is $(m!)^n$, where n is the number of resources in the shop.

The following subsections present the parts of the Anchor&Ascend coordination procedure in more detail as well as experimental results that show the behavior of the procedure under different disparity conditions.

Initial Optimization of Anchor Subsolution

Initially, the anchor agent generates a sequence of a_{iu} according to earliest start time est, i.e., a_{iu} with earlier est_{iu} is earlier in the sequence. Being a highly contended resource, the bottleneck resource usually processes the sequence of a_{iu} without any slack (gap) between adjacent activities, a^{k-1}_{pu}, a^{k}_{qu}, and a^{k+1}_{ru}, where the superscript $(k-1, k, k+1)$ denotes the processing sequence on the bottleneck resource. In other words, $st_{pu} + dur_{pu} = st_{qu}$, and $st_{qu} + dur_{qu} = st_{ru}$. To optimize the sequence $\{a_{iu}\}$ with minimal C_{anchor}, the anchor agent goes through an iterative process of switching pairs of anchor activities a_{iu} to reduce C_{anchor}.[4] During this process, two heuristic subroutines[5], *jump forward* and *jump backward*, are used. In jump forward (or backward) an anchor activity a_{iu} in the sequence is repeatedly moved forward (or backward) toward time origin (or time infinity) by switching with one of the preceding (or succeeding) activities to reduce C_{anchor}. Given a sequence of activities, $S = \{a^k_{iu}\}$, the subroutine is described as follows:

1. Calculate c_{iu} for each a_{iu}. Select an activity, a^v_{pu}, in S with the largest (or smallest) c_{pu}.
2. For each a^k_{iu}, k from $v-1$ to 1 (or from $v+1$ to m). If $st_{iu} < est_{pu}$, go to 3. Otherwise, if switching a_{pu} with a^k_{iu} would reduce C_{anchor}, then switch them.
3. Remove a_{pu} from S. If S is empty, stop. Otherwise, go to 1.

To obtain a near optimal subsolution, the anchor agent iteratively applies the jump forward/backward subroutines to the current sequence of a_{iu} until C_{anchor} can no longer be reduced.

Distributed Constraint Satisfaction

Once the anchor agent has generated an anchor subsolution, the non-anchor agents engage in distributed constraint satisfaction to find subsolutions that satisfy the effective due date anchor constraints imposed on pre-anchor activities as well as release date, temporal precedence and capacity constraints. Recall that a job agent's subsolution may be modified and interfered by any resource agents and vice versa. The distributed search is conducted through CP&CR, a process of coordinated local reaction of agents. The two groups of agents (resources and jobs) are iteratively activated in turn. Whenever an agent is activated, it will check and ensure the validity of its subsolution (satisfying its own constraints) by making necessary changes to the start times of the activities under its jurisdiction. In particular, when a new anchored subsolution is presented by the anchor agent, each job agent assigns start times to its activities such that pre-anchor activities are processed as soon as possible and post-anchor activities are processed immediately after the anchored activity a_{iu}, e.g., $st_{ij} = RD_i + \sum_{k=1}^{j-1} p_{ik}$, for $j = 1, ..., u-1$, $st_{ij} = st_{iu} + \sum_{k=u}^{j-1} p_{ik}$, for $j = u+1, ..., n$. Then regular resource agents are activated to resolve any capacity constraint violations followed by job agents to resolve any precedence constraint violations caused by resource agents' modification. Therefore, the global solution is repeatedly and locally modified by the agents searching for an instantiation that satisfies all constraints.

During this process, the agents exchange local views to facilitate the convergence to a global solution and to recognize the situation in which the search fails to meet anchor constraints. When a job (or resource) agent is making changes to the assignment of its activities, it takes into consideration information associated with these activities by resource (or job) agents in order to minimize the possibility of causing constraint violations for other agents. This exchange of information allows agents to coordinate their local modifications and facilitate group convergence. A very simple scenario is shown in Figure 2 where each rectangular box represents an activity with corresponding processing duration. Resource Y is the anchor agent. In (a), the anchor agent constructs the initial configuration of the anchor subsolution. In (b), each job agent assigns start times to its activities such that pre-anchor activities (A11, A21, A31) are processed as soon as possible and post-anchor activities (A13, A23, A33) are processed immediately after the anchored activities (A12, A22, A32). In (c), A21 within

[3] It is described in previous paragraph and in footnote 2.

[4] For other objective function (e.g., makespan), exact methods are available for this one machine sequencing problem. However, to guarantee optimality for weighted tardiness where no exact method is available, it requires more elaborate branch and bound procedure which is exponential in the worst case. We choose to rely on heuristics for efficiency.

[5] Similar heuristic, pairwise interchange, is also used in neighborhood search(Morton & Pentico 1993).

dotted rectangular box represents the start times assigned by job agent Job2. Res.X agent finds a capacity constraint violation between A31 and A21 (shown by dotted rectangular box before conflict resolution), and changes the start time of A21. All agents are satisfied with the current instantiation of variables in (d) which represents a solution to the problem. For more details on CP&CR, see (Liu & Sycara 1993).

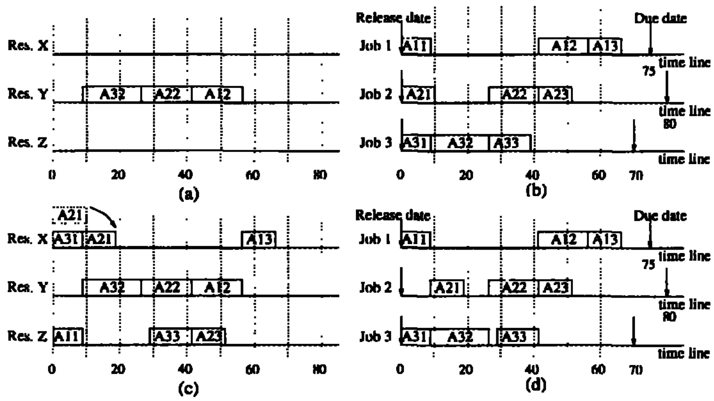

Figure 2: A Simplified Scenario

The iterated process continues until it becomes apparent that it is not possible to obtain a global solution consistent with the current anchor subsolution. This happens when a job agent sees that a start time change frequency threshold for an activity under its jurisdiction has been exceeded. These thresholds are maintained locally by each job agent for each activity. When the threshold has been exceeded, the job agent may violate the effective due date constraint on some pre-anchor activities by changing the start time of an anchor activity to a later time (the time when its preceding activity will be finished). As a result of the effective due date constraint violation, the anchor agent must conduct a local search to modify its current anchor subsolution.

Modification of Anchor Subsolution

The changes to a subset of st_{iu} by a subset of job agents signal to the anchor agent that a reconfiguration of the set of st_{iu} is required. In job shop scheduling, two operators, $exchange(i,j)$ and $right\text{-}shift(i)$, have been used (Morton & Pentico 1993) (Miyashita & Sycara 1993) (Nakakuki & Sadeh 1994) to modify the processing sequence of activities on a resource. In determining the scope of applying the two operators, we consider the following: (1) since the current anchor subsolution has the minimal local objective cost, it is desirable to modify it as little as possible in order to limit the increase in the objective cost, (2) the modification should also correspond to the changes made by job agents so that new anchor subsolutions have better chance of leading to a successful constraint satisfaction session. Therefore, $right\text{-}shift(i)$ is applied to only the changed anchor activity, while $exchange(i,j)$ is applied to a heuristic neighborhood of the changed anchor activity.

The local search guided by application of the two modification operators generates a set of candidate anchor subsolutions. These new sequences are put into the list of candidate sequences if they do not duplicate existing sequences in the list. The list of candidate sequences is sorted by increasing local objective cost. Then the anchor agent chooses the first one (with the least cost) from the list to be the next anchor subsolution and the process of constraint satisfaction by the non-anchor agents repeats until a global solution is found where all constraints are satisfied. Since the anchor agent searches for a proper sequence with monotonically increased objective cost, the global solution represents the best solution that the Anchor&Ascend procedure can find.

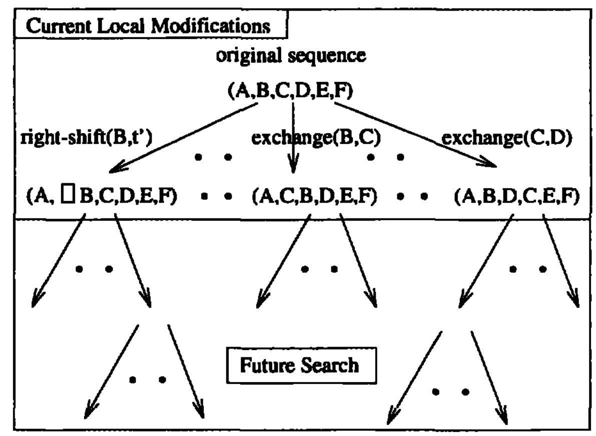

Figure 3: Anchor Agent's Local Modification Operations

For example, as shown in Figure 3, let the anchor agent have a current processing activity sequence of (A,B,C,D,E,F). Suppose B was changed by a job agent to a later start time $\acute{t}$. The anchor agent applies $right\text{-}shift(B,\acute{t})$ to the current processing sequence. This results in a new sequence with gap between A and B, and all activities after B are right-shifted accordingly with B. The anchor agent also applies $exchange(i,j)$ to the current processing sequence with $i = B, C$, $j = i+1, i+2$, and $i = D, j = i+1$. In other words, exchange operations, exchange(B,C), exchange(B,D), exchange(C,D), exchange(C,E), and exchange(D,E), are individually performed on the current processing sequence which results in five new sequences, e.g., exchange(B,C) results in (A,C,B,D,E,F), exchange(C,D) results in (A,B,D,C,E,F). These six applications of the two modification operators (five exchanges and one right-shift) represent a heuristic balance between minimizing cost increase (solution quality) and increasing chances of leading to successful search by other agents (search efficiency). In particular, exchange(B,C), exchange(B,D), and right-shift(B,$\acute{t}$) directly respond to the job agent's constraint violation (unable to meet constraint imposed by the start time of B) by assigning B to later start times. Exchange(C,D), exchange(C,E), and exchange(D,E) attempt to change the condition of resource contention of regular resources such that the preceding activity of B in the job can start earlier and B can start at its original start time. For example, if one of the pre-anchor activities of C was competing with the preceding activity of B for the same resource such that the preceding activity of B could not meet its effective due date imposed by B, by exchanging C with D and moving C to a later start time, the resource may schedule the preceding activity of B being processed before the pre-anchor activity of C so that the effective due date imposed by B can be met. When there are more than one anchor activity being changed by job agents, the anchor agent only performs modification on the subsolution resulting from the earliest changed anchor activity in order to limit cost increase.

Experimental Results and Evaluation

To study the behavior of the Anchor&Ascend coordination procedures under different disparity conditions, we construct a set of test problems with disparity ratio, ranging from 1.25 to 5.0 with a granularity of 0.25, and disparity composition ratio, ranging from 0.2 to 0.8 with a granularity of 0.2. Each combination of the two parameters is represented by a subset of 10 problems that are randomly generated while controlling the disparity condition. Therefore, the problem set includes 64 subsets and a total of 640 problems. Each problem consists of 10 jobs on 5 resources with 50 activities to be scheduled. The disparity ratio represents the ratio of average processing times between the subgroup of bottleneck resources and the subgroup of non-bottleneck resources. A disparity composition ratio of 0.4 means that 2 out of 5 resources are bottleneck resources. The following table specifies the sequence (in bold) of using bottleneck resources in the job for each disparity composition ratio. For example, in the disparity composition ratio of 0.4, the second activity uses the first bottleneck resource, the fourth activity uses the second bottleneck resource, in each job. In cases of more than one bottleneck resources, the Anchor&Ascend mechanism requires the selection of a particular bottleneck resource as the anchor agent. Analytically, selection of a later bottleneck resource would improve the solution quality (tighter control due to less downstream processing) but would certainly demand more computational cost (harder for the distributed constraint satisfaction session). This is verified by our partial experimental results in the case of two bottleneck resources. In this study, the latest bottleneck resource is selected to be the anchor agent.

disparity composition ratio	*sequences in the job*
0.2	1 2 3 **4** 5
0.4	1 **2** 3 **4** 5
0.6	**1** 2 **3** 4 **5**
0.8	**1** **2** 3 **4** **5**

Experimental results are evaluated by both computational cost and solution quality. We use the total number of states[6] the anchor agent explored before a global solution is found as an estimate of the computational cost. CPU times of (1 100 300 500) states explored approximately correspond to (0.5 22 105 420) seconds in Common Lisp implementation on an HP-715/100 workstation. For problems of high disparity composition ratio and low disparity ratio, the anchor agent usually needs to explore a large number of states before a solution can be found. In order to keep the experimentation in a reasonable time frame, we set a limit of 500 states upon which the procedure would terminate even if a global solution has not been found. Solution quality of Anchor&Ascend (S_A) is evaluated by comparing to that of naive First Come First Serve (FCFS)[7] dispatch rule (S_F) and indicated by an improvement measure, $(S_F - S_A)/S_F$. As a more serious evaluation, solution quality of Anchor&Ascend is also compared to that of X-R&M heuristic dispatch rule (Morton & Pentico 1993), which is well respected and widely used in the Operation Research community for weighted tardiness problems. For each subset of problems representing a combination of a disparity ratio and disparity composition ratio, an average computational cost and an average solution quality are obtained from results of 10 problems in the subset. For problems that were not solved within 500 states, a computational cost of 500 states is included in calculating the average computational cost, while not contributing to the average solution quality.

Figure 4 Computational Cost

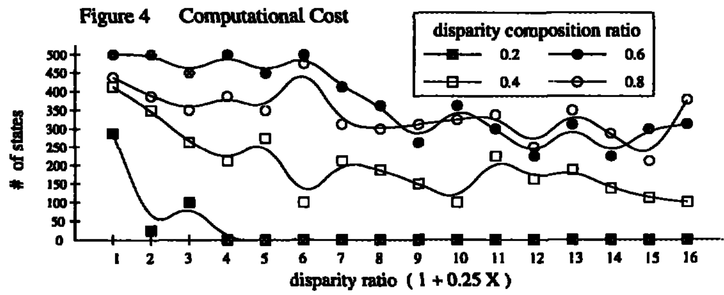

Figure 5 # of problems solved within 500 states

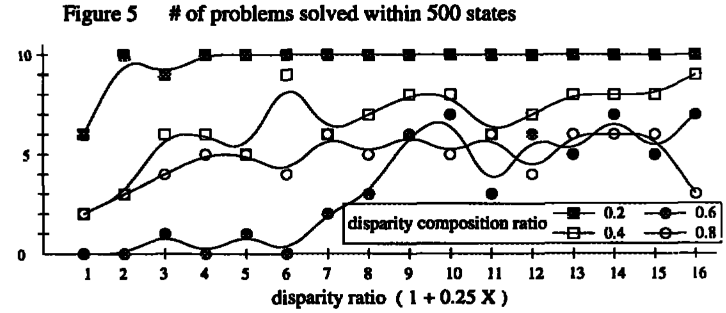

Figure 4 shows the computational cost over a range of combinations of disparity ratio and disparity composition ratio. Generally, computational cost increases with increasing disparity composition ratio and/or decreasing disparity ratio. Anchor&Ascend is especially effective for problems of disparity composition ratio of 0.2 (1 bottleneck resource out of 5 resources) and disparity ratio of 1.75 and above, finding a solution of high quality (see Figures 6 and 7) within

[6]Each state represents a configuration of the anchor agent's subsolution.

[7]Activity with the earliest ready time is dispatched first.

1 second and mostly in 1 state. The results on the computational cost also suggest that Anchor&Ascend is less applicable to problems with disparity composition ratio higher than 0.4. Figure 5 shows the number of problems solved within 500 states in each subset. The number generally decreases with an increase in disparity composition ratio and a decrease in disparity ratio. The percentage of problems of disparity composition ratio (0.2 0.4 0.6 0.8) that were solved within 500 states are (97 66 34 48) percents, respectively.

Figure 6 Solution Quality (vs. FCFS)

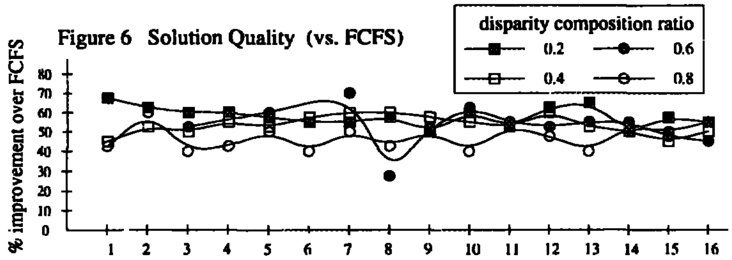

Figure 7 Solution Quality (vs. X-R&M)

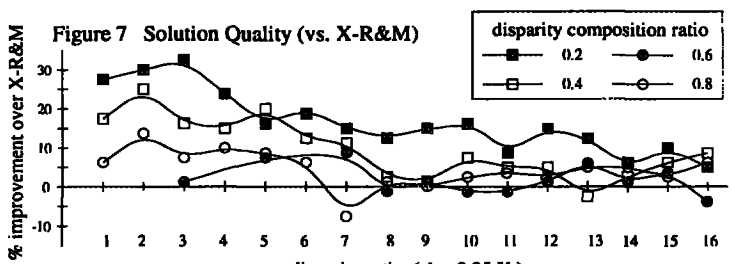

Figure 6 depicts the solution quality obtained by Anchor&Ascend comparing to that of FCFS. In general, solutions obtained by Anchor&Ascend have an improvement of 50 to 60 percents over that of FCFS. Solution quality of problems with disparity composition ratio of 0.6 and 0.8 has more variations and is less conclusive since the number is obtained from an average of fewer solved problems (since more problems need more than 500 states to solve). In addition, lower disparity composition seems to facilitate higher solution quality. However, no obvious relation between solution quality and disparity ratio is observed. Figure 7 depicts the solution quality obtained by Anchor&Ascend comparing to that of X-R&M. Most problems have an improvement of 5 to 20 percents over that of X-R&M. Relation of solution quality with disparity composition ratio and disparity ratio can not be drawn from this comparison because X-R&M is a parameterized heuristic dispatch rule and requires a trial of optimal values of parameters for different scheduling conditions. We use the same values of parameters through out the experiments.

In general, our experiments show that Anchor&Ascend is most effective in high disparity ratio and low disparity composition ratio, in which solutions of very good quality can be easily obtained with little computational cost. No obvious relation is observed between the solution quality and the disparity ratio, although an increase in disparity composition ratio generally results in a decrease in solution quality. In addition, disparity composition ratio is more important than disparity ratio in the applicability of Anchor&Ascend. The experimental results seem to indicate a cutoff value of 0.4 ($\leq$ 0.4) as the disparity composition ratio in the applicability of Anchor&Ascend.

Discussion

The Anchor&Ascend approach revolves around disparity among subproblems and the subsequent dominance effects on solution construction and global optimality. We expected the approach to be most effective in problems that exhibit extreme disparity among subproblems, e.g., low disparity composition ratio and/or high disparity ratio. For a given problem under condition of low disparity composition ratio, there are few agents with dominance effects on solution construction. For low disparity composition ratio, the amount of interactions required to construct a feasible global solution is much less extensive. This also implies that disparity ratio (relative task difficulty between agent categories) has only secondary effect on problem solving efficiency as compared to disparity composition ratio. The experimental results confirm our theoretical analysis. However, the scope of applicability of Anchor&Ascend exceeds our original expectation. We were surprised that the approach is able to solve even partial sets of problems with high (0.6 and 0.8 in the range of 0.2 to 0.8) disparity composition ratio with moderate computational cost. In addition, disparity ratio can be as low as 1.25 (in the range of 1.25 to 5.0) for the approach to work. Overall, the experimental results attest to the considerable advantage of exploiting problem structure in designing coordination mechanisms for distributed problem solving.

An additional, perhaps, even more interesting finding in our experiments is related to the bimodal characteristics of most NP-complete problems. (Cheeseman, Kanefsky, & Taylor 1991) conducted a study on constraint satisfaction problems and showed that there is at least an "order parameter" that separates problems into regions of solvability. It was conjectured that constraint optimization problems might exhibit similar phase transition. Essentially, the Anchor&Ascend approach iteratively attempts to solve a series of constraint satisfaction problems until it succeeds. In our experimental results, we noticed that problems tend to diverge into subsets that are either easy or hard to solve. The tendency of divergence is more evident at high disparity composition ratio and low disparity ratio where Anchor&Ascend is less effective. This observation provides empirical evidence to the phase transition property of NP-complete optimization problems.

Finally, our approach and the experimental results also enrich the practice of job shop scheduling where consideration of shop conditions has been focused on the number of bottlenecks, resource utilization rate, and job tardy factor. We provide two additional measures of shop conditions that emphasize the relative loadings on shop resources. Our experimental results show that these two parameters (disparity composition ratio and disparity ratio) can allow us to identify shop conditions when Anchor&Ascend is most likely to be applicable and effective. Furthermore, in many real job shops where shop conditions can be adjusted ahead of time, our experimental results could guide decision on varying the mix of job batches, changing resource loadings, such that high quality solution can be found efficiently.

Conclusions

In this paper, we exploited characteristics of problem structure in designing coordination mechanisms for distributed constraint optimization. In particular, we exploited disparity among subproblems. This characteristic often occurs in problems from many application domains. We present a particular coordination mechanism, Anchor&Ascend, that involves the notion of an anchor agent. Local search for global optimization is based on a combination of the anchor agent conducting local best first search and all other agents performing distributed constraint satisfaction to enforce given problem constraints as well as constraints imposed by the anchor subsolution. Experimental results in the domain of job shop scheduling show that (1) considerable advantage can be obtained by explicitly exploiting disparity (2) disparity composition ratio plays a more important role than disparity ratio in finding high quality solution with little computational cost.

References

Cheeseman, P.; Kanefsky, B.; and Taylor, W. 1991. Where the *really* hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence.*

Fox, M. S., and Sadeh, N. 1990. Why is scheduling difficult? a CSP perspective. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, 754–767.

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

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

Huhns, M., and Bridgeland, D. 1991. Multiagent truth maintenance. *IEEE Transactions on System, Man, and Cybernetics* 21(6):1437–1445.

Liu, J., and Sycara, K. P. 1993. Distributed constraint satisfaction through constraint partition and coordinated reaction. In *Proceedings of the 12th International Workshop on Distributed AI.*

Liu, J., and Sycara, K. P. 1994. Distributed problem solving through coordination in a society of agents. In *Proceedings of the 13th International Workshop on Distributed AI.*

Mackworth, A. K. 1987. Constraint satisfaction. In Shapiro, S. C., ed., *Encyclopedia in Artificial Intelligence.* New York: Wiley. 205–211.

Miyashita, K., and Sycara, K. 1993. Case-based incremental schedule revision. In Fox, M., and Zweben, M., eds., *Knowledge-Based Scheduling*. Morgan Kaufmann.

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

Nakakuki, Y., and Sadeh, N. 1994. Increasing the efficiency of simulated annealing search by learning to recognize (un)promising runs. In *Proceedings of AAAI-94*, 1316–1322.

Nudel, B. A. 1983. Consistent-labeling problems and their algorithms: expected-complexities and theory-based heuristics. *Artificial Intelligence* 21:135–178.

Purdom, P. W. 1983. Search rearrangement backtracking and polynomial average time. *Artificial Intelligence* 21:117–133.

Sycara, K.; Roth, S.; Sadeh, N.; and Fox, M. 1991. Distributed constraint heuristic search. *IEEE Transactions on System, Man, and Cybernetics* 21(6):1446–1461.

Yokoo, M.; Durfee, E.; Ishida, T.; and Kuwabara, K. 1992. Distributed constraint satisfaction for formalizing distributed problem solving. In *Proceedings of the 12th IEEE International Conference on Distributed Computing Systems*, 614–621.