Iterative Constraint-based Repair for Multiagent Scheduling

Kazuo Miyashita
ETL (Electrotechnical Laboratory)
1-1-4, Umezono, Tsukuba
Ibaraki 305, JAPAN
miyasita@etl.go.jp

Abstract

We propose a new integrated architecture for distributed planning and scheduling that exploits constraints for problem decomposition and coordination. Our goal is to develop an efficient method to solve densely constrained planning / scheduling problems in a distributed manner without sacrificing solution quality. We implemented a prototype system, called CAMPS, in which hierarchy of intelligent agents try to coordinate their actions for "satisficing" planning / scheduling results by handling several intra- and inter-agent constraints. In this paper, we show the repair-based methodology for distributed planning / scheduling and the constraint-based mechanism for dynamic coalition formation among agents.

Introduction

In the research of distributed artificial intelligence (DAI), there has been research on *cooperative distributed problem solving (CDPS)* (Decker, Durfee, & Lesser 1989) or coordinated problem solving (Gasser & Hill 1990). In those systems, a loosely coupled distributed network of semi-autonomous agents, each of which is capable of sophisticated problem solving, cooperatively interact with other agents to solve a problem with a single goal (Lesser & Corkill 1987).

The planning and scheduling activities in a manufacturing enterprise can be naturally modeled as CDPS. The goal of planning / scheduling is, in its broadest sense, to maximize profit of the enterprise. And in the planning / scheduling problem, each agent, who is in charge of different aspects of the problem, works cooperatively to attain the common goal of making a globally most profitable plan / schedule, and at the same time it acts distributedly to solve its own local sub-problem trying to maximize its own objectives. The most important distinction of the enterprise planning / scheduling problem from the other cooperative distributed problems studied so far is its constrainedness. In the planning / scheduling problem, any single decision of each agent can cause unpredictable ripple effects on other agents' decisions through constraint propagation. Therefore, constraints should be properly handled for efficient solving of distributed planning / scheduling problems.

There have been a couple of related research activities in the fields of constraint satisfaction problem (CSP) and constraint-based scheduling. Among theoretical research of distributed constraint satisfaction (Hirayama & Toyoda 1995; Yokoo *et al.* 1992), almost none has developed the effective method for solving densely constrained problems such as production scheduling problems. And in the study of distributed scheduling, the aim of the systems has been to find a feasible solution (CORTES (Sycara *et al.* 1991)) or a satisfactory solution in terms of a simple objective (e.g. time-shift preferences in DIS-DSS (Neiman, Hildum, & Lesser 1994)). In those systems, a solution is built constructively by scheduling agents which communicate abstract resource profiles and meta-level information (such as lending possibility) among them. The agent interaction is hard-coded a priori in the systems and can't be adjusted during the course of problem solving. But, in multiagent problem solving, the degree and type of desirable agents' interaction must differ among problem instances depending on the problem characteristics (such as its goal, constraints). And it is inherently difficult to find an optimal collaboration in distributed problem solving because of the following reasons: (1) several individual agents are in charge of the diverse aspects of the problem in an isolated or inter-dependent manner, (2) each agent should have its own local view of the profit, which is not necessarily consistent with the other agents' or global profit, and (3) each agent cannot have complete knowledge about the other agents because of resource limitation and privacy.

In our new architecture for distributed planning and scheduling, agents iteratively repair its local solution to pursue a globally "satisficing" result. In Anchor & Ascend approach by Liu (Liu 1996), the specific problem structure such as bottleneck resources is used to control iterative repair by several agents while keeping search space of the agents tractable. But this approach is effective only when the appropriate problem structure exists for target objectives (e.g. bottleneck resources for tardiness minimization). To meet more realistic conditions seen in everyday life of the factories, we need more flexible mechanism to define and refine the search space of agents in an effective and efficient manner. Thus we decide to have agents coordinate a way of decomposing a problem and resolving conflicts among themselves. Repair methods are decided collaboratively by negotiation among agents considering trade-offs of their preferences and global objectives. To improve quality and efficiency of the distributed repair process, the system needs to have sophisticated control over agents' negotiation process. Since the repair process by agents is strongly problem-dependent and the planning / scheduling problem is ill-structured by nature, it is difficult to develop a general procedure for agents negotiation in the distributed planning / scheduling problem. To solve the above difficulty, our system exploits constraints for two purposes in the distributed problem solving: (1) the problem is decomposed into a set of sub-problems based on its constraints and the sub-problems are distributed and assigned to the appropriate agents, and (2) intra- or inter-agent constraints are dynamically enforced or relaxed by the agents to control negotiation process considering the trade-off between concurrency and coordination in multiagent problem solving.

Overview of The Problem

Our problem is formulated as a job shop planning / scheduling problem, which is a well-known NP-complete problem (French 1982). In the problem, each job consists of a set of operations with fixed duration to be scheduled according to a partial operation ordering. The dominant constraints in the problem are two-holds: (1) the operation precedence constraints along with a job's release date and due date restrict the set of acceptable start times for each operation, and (2) the capacity constraints restrict the number of operations that can use a resource at any particular point in time and create conflicts among operations that are competing for the use of the same resource at overlapping time intervals.

The goal of a planning / scheduling system is to set up plans that are feasible and direct to global optimality such as minimizing lead time and weighted tardiness, and to produce schedules that respect the given plan and other constraints, and also optimize a set of local objectives, such as minimize work in process inventory (WIP) (i.e. the time operations spend in a factory waiting to be processed), maximize resource utilization etc. To be noted is that global and local objectives may not necessarily be compatible so that the goal of the system is not to find an optimal but a satisficing result.

A Model of Distributed Planning / Scheduling

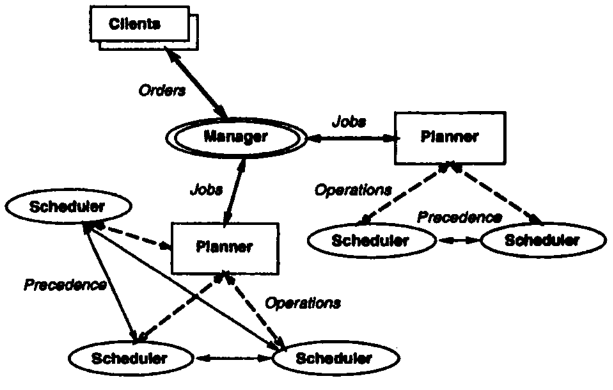


Figure 1: Agents Model

We model a planning / scheduling problem as a multiagent distributed problem, where we have three types of hierarchical agents in charge of different aspects of the planning / scheduling problem and those agents work cooperatively and asynchronously to achieve their goal (see Figure 1):

- A *manager agent* decomposes orders from clients into a set of jobs and distributes them to planner agents. Considering both clients' request and planners' capability, it sets a time-window for each job. A manager agent is thus in charge of load-balancing among planner agents.
- A *planner agent* distributes operations that consist of its assigned jobs to scheduler agents. Then, a planner agent integrates, evaluates and improves the solutions from the scheduling agents. A planner agent is responsible for constructing a solution which satisfies global constraints given by the manager agent.
- A *scheduler agent* administrates a single resource and sequences its assigned operations. A scheduler agent seeks for its local schedule respecting a planner's requirements as well as optimizing its own local objectives.

Decisions by one agent to achieve its tasks produce unpredictable effects on decisions of other agents through constraint propagation. The influence is not always exerted in top-down fashion, but agents in higher hierarchy are often required to modify their decisions to accommodate the feasibility of lower agents. Therefore, problem solving process is inherently iterative, and for fast problem solving, interactions among agents need to be controlled properly.

CAMPS Problem Solving Process

We propose the system called CAMPS (Constraint-based Architecture for Multiagent Planning / Scheduling), which is a distributed architecture for solving the planning / scheduling problem with constraint-based search.

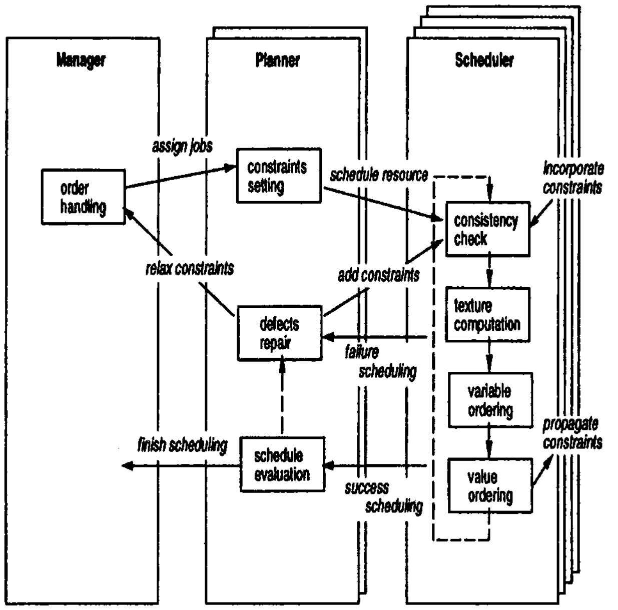


Figure 2: CAMPS Process Flow

Figure 2 depicts the schematic flow of agent interactions in CAMPS. Agents of upper level decompose a problem into sub-problems and assign them to the agents of lower level with the constraints to be respected. The assigned agents try to solve the given sub-problem and report the result back to the upper agents. When the result is failure or unsatisfactory, the upper agents try to modify the problem decomposition or the constraints. Therefore, in most of the cases, the same level of agents (planners, schedulers) can work independently and concurrently.

A manager agent takes orders from clients and decomposes the orders into jobs which are processed by a single planner agent. With each job, it makes decisions on a release date, the date that it will be ready for starting processing, and a due date, a date on which the job should finish, based upon several management data such as the client's requirements, current level of resource utilization, past records of jobs' lead time and so on. Then, the manager agent assign jobs to appropriate planner agents so that the planner agents can finish the jobs in the determined time-windows.

Given the jobs from the manager agent, each planner agent decomposes its assigned jobs into operations and assigns every operation to appropriate scheduler agents which can process the operations using the resource they supervise. Or, a planner agent assigns some part of its jobs to other planner agents as subcontractors imposing constraints to be satisfied. When assigning the operations to the scheduler agents, the planner agent adds constraints on each operation, which represent desirability of the time-interval over which the operations are scheduled, based on the estimated resource load and operation precedences. These constraints are decided through negotiation between agents considering both global objectives and local preferences. If mutual agreements on the constraints are not established, decomposition is revised based upon the contents of disagreement.

There have been several research activities which analyze global resource capacity and operation constraints for estimating contentions on each resource (Sadeh 1991; Berry 1991; Muscettola 1993). Most of them have been done under a single scheduling agent scenario with a few exceptions such as (Sycara *et al.* 1991; Neiman, Hildum, & Lesser 1994). Agents in CAMPS are more flexible than those systems since CAMPS can dynamically modify its constraints on its interacting agents in discourse with them.

After receiving time-interval constraints (desirability), each scheduler agent, which is in charge of a single resource, determines which reservation to assign to the operations so that they don't overlap each other over the time intervals. In fixing the reservation for operations, a scheduling agent can exploit the constraints and a set of heuristics. For example, a greedy heuristic selects a reservation based on local preferences associated with each operation. A least constraining heuristic selects the reservation that is the least likely to prevent other operations (of other agents) from being scheduled. Thus, the time-interval constraints help scheduler agents make sequencing decisions, which respect precedence and capacity constraints of the operations, without communicating with the other scheduler agents.

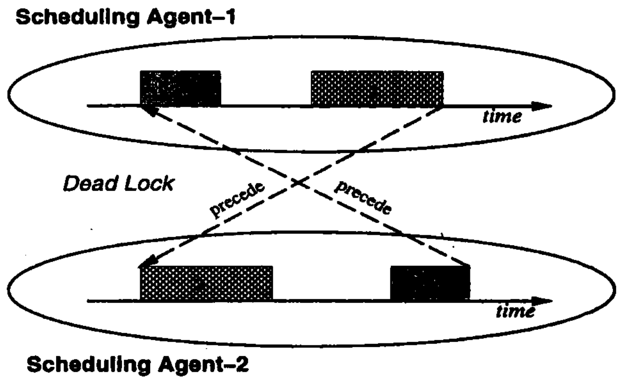


Figure 3: Inconsistency among Local Schedules

Repair-Based Planning / Scheduling

Rational load distribution to each scheduler agent by a planner agent can help scheduler agents make reservation of resources to the operations independently. Although it can reduce possibilities of inconsistencies among local schedules, they cannot be eliminated. Lots of inconsistencies among reservations of other agents such as a deadlock (Figure 3) can occur.

Therefore, interaction among agents is necessary to resolve such inconsistencies. In the past research of distributed scheduling, agents interact with other agents tightly during construction of a partial schedule to prevent reservations that cause inconsistencies or to perform global backtracking for conflict resolution. But, in those systems the cost of communication among scheduling agents is immense especially when frequent global backtracking occur. In CAMPS, we adopt a repair-based approach, i.e. each scheduler agent constructs its own local schedule without dense communication among the other scheduler agents and resolves the conflicts in agents' schedule through negotiation after every agent builds its own local schedule.

Our repair-based approach is motivated by the following considerations:

1. Quick detection of over-constrainedness

   When any single scheduling agent cannot make a feasible schedule even without considering conflicts among other agents, the given problem is possibly too tight and a planning agent must relax some constraints to make the problem solvable. This situation can be efficiently captured if each scheduling agent tries to make its own schedule before interacting with other agents at the first step of building a consistent schedule. Although clearly this is not a complete testing to detect over-constrained problems, in some cases this might help reducing redundant computation resulting from the poor capabilities of a planning agent which assign over-constrained problems to scheduling agents.

2. Efficient solution finding

   Once every agent makes an initial schedule to be repaired, agents can get the global view (or, at least a meta-level view because of communication bandwidth limitation) of overall solution structure, which cannot be acquired correctly in the middle of constructive problem solving. Therefore, there is a possibility for a repair-based method to find a solution efficiently by exploiting information of the solution structure for focusing computational efforts on the most constrained parts of the conflicted solution.

3. Acquisition of context-dependent preferences

   In realistic planning / scheduling problems, each agent needs to make decisions based upon context-dependent preferences and constraints that have not been represented explicitly in the problem solving model. It is difficult to capture this contextual information even from a human expert without giving him/her a specific situation as examples. But repairing one's schedule to make it compatible with others' and maximize its own local objectives as much as possible through negotiation can provide a rich set of actual problem solving situations for each agent (either human or computer). In negotiations each agent determines its actions considering the trade-offs of local and global (other agents') preferences and revises them by getting feed-back from other agents. Hence we hypothesize that, by storing the course of the inter-agent negotiations, context-dependent preferences of agents can be acquired with reasonable cost and they can be applied to solve the future problems.

Agent Coordination in CAMPS

Each scheduler agent's schedule should be coordinated and inconsistencies should be resolved through interactions among agents. Agents do not pursue merely a feasible solution, but quality of the global plan / schedule can be iteratively improved through agents' coordination. In the course of coordination, the agents make negotiations trying to find optimal trade-offs among their local preferences and other agents' preferences and make commitments based on the negotiation results.

Problem solving process in CAMPS consists of the following four steps: (1) decomposing a problem into mutually interrelated sub-problems and imposing some constraints on sub-problems from a global objective point of view, (2) solving sub-problems independently, (3) detecting conflicts and defects by integrating local sub-problem solutions, and (4) defining and refining the shared search space of interacting agents to remove conflicts and defects. Among the above steps, the second and third step can be executed without sophisticated interactions between agents. Careful coordination is required at the first and fourth step: at these steps agents must cooperatively accomplish several tasks such as problem decomposition and conflict resolution.

Coordination for Repair

In CAMPS, coordination mechanism among agents works for repairing the defects caused by the lack of agents' knowledge about the environment (including other agents). Since such knowledge can't be fully acquired owing to dynamic nature of the environment and privacy of agents, coordination through repair is inevitable for pursuing globally consistent solutions. In the system, we have three types of repair strategies for resolving inconsistencies or improving quality of the solutions:

1. Intra-Agent Constraint Adjustment

   After schedule integration, when conflicts are detected among each agent's schedule, they are to be resolved through negotiations among scheduler agents by revising schedules locally. This process is similar to the repair-based scheduling approach (Minton *et al.* 1990; Zweben *et al.* 1992; Miyashita & Sycara 1995), but it is more complicated because of distributed nature of the system, i.e. many conflicts can/should be detected and solved simultaneously, thus resolution processes on

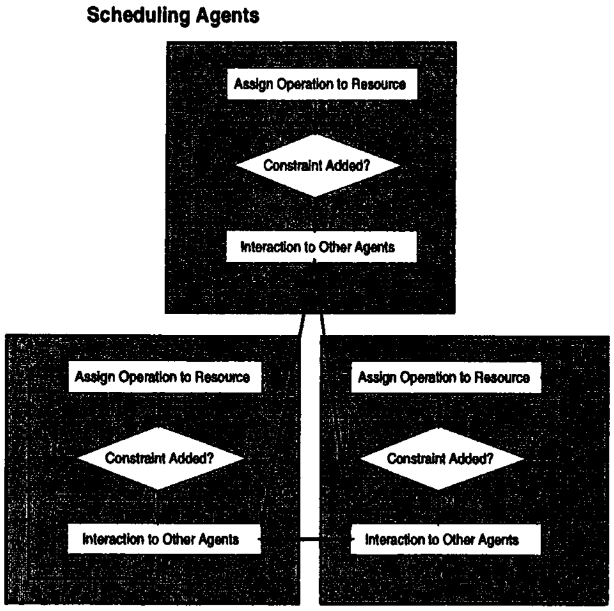
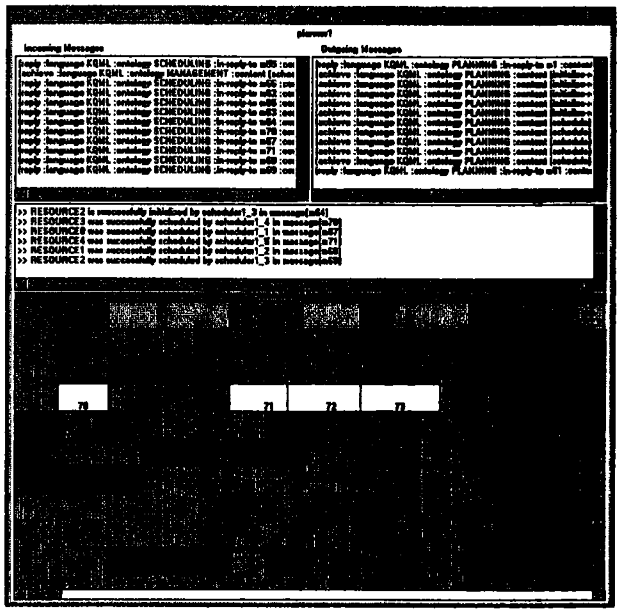

Scheduling Agents
Assign Operation to Resource
Constraint Added?
Interaction to Other Agents
Assign Operation to Resource
Constraint Added?
Interaction to Other Agents
Assign Operation to Resource
Constraint Added?
Interaction to Other Agents

Figure 4: Interactions among Scheduler Agents



```
(reply :language KQML :ontology SCHEDULING :in-reply-to m55 :co
(achieve :language KQML :ontology MANAGEMENT :content (sche
(reply :language KQML :ontology SCHEDULING :in-reply-to m56 :co
(reply :language KQML :ontology SCHEDULING :in-reply-to m62 :co
(reply :language KQML :ontology SCHEDULING :in-reply-to m65 :co
(reply :language KQML :ontology SCHEDULING :in-reply-to m63 :co
(reply :language KQML :ontology SCHEDULING :in-reply-to m64 :co
(reply :language KQML :ontology SCHEDULING :in-reply-to m70 :co
(reply :language KQML :ontology SCHEDULING :in-reply-to m67 :co
(reply :language KQML :ontology SCHEDULING :in-reply-to m71 :co
(reply :language KQML :ontology SCHEDULING :in-reply-to m68 :co
(reply :language KQML :ontology SCHEDULING :in-reply-to m69 :co
```

```
(reply :language KQML :ontology PLANNING :in-reply-to m1 :content
(achieve :language KQML :ontology PLANNING :content (initialize-
(achieve :language KQML :ontology PLANNING :content (initialize-
(achieve :language KQML :ontology PLANNING :content (initialize-
(achieve :language KQML :ontology PLANNING :content (initialize-
(achieve :language KQML :ontology PLANNING :content (initialize-
(achieve :language KQML :ontology PLANNING :content (schedule)
(achieve :language KQML :ontology PLANNING :content (schedule)
(achieve :language KQML :ontology PLANNING :content (schedule)
(achieve :language KQML :ontology PLANNING :content (schedule)
(achieve :language KQML :ontology PLANNING :content (schedule)
(reply :language KQML :ontology PLANNING :in-reply-to m11 :conte
```

```
>> RESOURCE2 is successfully initialized by scheduler1_3 in message(m64)
>> RESOURCE3 was successfully scheduled by scheduler1_4 in message(m70)
>> RESOURCE0 was successfully scheduled by scheduler1_1 in message(m67)
>> RESOURCE4 was successfully scheduled by scheduler1_5 in message(m71)
>> RESOURCE1 was successfully scheduled by scheduler1_2 in message(m68)
>> RESOURCE2 was successfully scheduled by scheduler1_3 in message(m69)
```

Figure 5: A Planner Agent

the different sets of conflicts may interact each other and result in divergence.

The simplest way of revising a schedule is to change constraints of a scheduler agent. An example of this type of repair is adding precedence constraints between operations assigned to a scheduler agent. By adding or deleting constraints of the agent, search space of the agent can be controlled so that a desirable schedule might be achieved.

2. Inter-Agent Constraint Adjustment

When a desirable schedule can't be achieved by intra-agent constraint adjustments, next coordination mechanism for repair is changing constraints among scheduler agents. In an initial stage of CAMPS problem solving, there is no (or very few, if any) constraints among scheduler agents. It means scheduler agents can work on their own problem independently and concurrently. If load is distributed to agents ideally, this will produce a feasible solution most efficiently. However, ideal load distribution is seldom realized because of intractable interactions among agents and operations. Therefore, in CAMPS, coordination among scheduler agents is necessary and implemented as inter-agent constraints.

An example of inter-agent constraints is a precedence constraint between operations assigned to different agents. With inter-agent constraints, agents need to communicate with related agents when a constrained variable is processed in the agents (see Figure 4). By dynamically adjusting inter-agent constraints, CAMPS can enforce coordination among agents with loss of concurrency. Thus CAMPS can control the granularity of distributed problem solving.

3. Problem Adjustment

When conflicts are decided unsolvable or cannot be resolved in a reasonable amount of time among scheduler agents, a planner agent starts negotiation with agents to remove conflicts. In the negotiation, a planner agent can relax some constraints or change problem decomposition to give scheduler agents more flexibility in modifying their local schedule for conflict resolution. The examples of remedies executable by a planner agents are:

- prolong lead time of jobs (i.e. allow tardiness)
- improve performance of the resource (e.g. overtime)
- cancel jobs
- subcontract jobs to other planner agents
- subcontract some operations to other scheduler agents

And even when a conflict-free schedule is generated for entire scheduler agents, a planner agent can ask re-schedule or repair of schedule for scheduler agents when the planner agent cannot be satisfied with quality of the current schedule from the global point of view.

Implementation

Figures 5 and 6 show the snapshots of a planner agent and scheduler agents in CAMPS. Current version of CAMPS system is implemented using JAVA language. As shown in the windows of the above figures, agents in

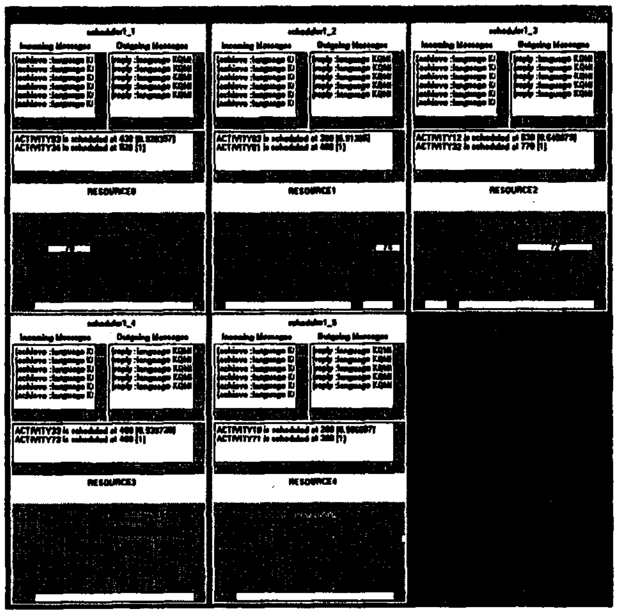


Figure 6: Scheduler Agents

CAMPS communicate each other by means of message passing. Agent's messages are coded in KQML language. The contents of the message are written with domain ontologies which model the task and domain structure of the problem. More sophisticated design of ontologies, which can be a fundamental tool for both problem solving and knowledge acquisition, is still a target of future research.

Conclusions

In this paper, we propose a new architecture for distributed planning and scheduling that exploits constraints for problem decomposition and coordination. In the research we view distributed planning/scheduling as a negotiated search process that iteratively builds a feasible and mutually satisfactory schedule among agents. The feasibility and superiority of our approach should be further evaluated empirically. Our next plan is to extend our view of domain model in terms of planning and scheduling problems, and capture the preferences of individual agents and search control knowledge for efficient and high quality problem solving by applying case-based reasoning method, which stores the past course of negotiation and reuses it to solve the current problems.

References

Berry, P. M. 1991. *A Predictive Model for Satisfying Conflicting Objectives in Scheduling Problems.* Ph.D. Dissertation, University of Strathclyde.

Decker, K. S.; Durfee, E. H.; and Lesser, V. R. 1989. Evaluating research in cooperative distributed problem solving. In Gasser, L., and Huhns, M. N., eds., *Distributed Artificial Intelligence, Vol.2.* London: Pitman. 485–519.

French, S. 1982. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop.* London: Ellis Horwood.

Gasser, L., and Hill, R. W. 1990. Engineering coordinated problem solvers. *Annual Review of Computer Science* 4:203–253.

Hirayama, K., and Toyoda, J. 1995. Forming coalitions for breaking deadlocks. In *Proceedings of the First International Conference on Multi-Agent Systems*, 155–162. AAAI.

Lesser, V., and Corkill, D. 1987. Distributed problem solving. In Shapiro, S., ed., *Encyclopedia in Artificial Intelligence.* New York: Wiley.

Liu, J.-S. 1996. *Coordination of Multiple Agents in Distributed Manufacturing Scheduling.* Ph.D. Dissertation, The Robotics Institute, School of Computer Science, Carnegie Mellon University.

Minton, S.; Johnston, M. D.; Philips, A. B.; and Laird, P. 1990. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 17–24. Boston, MA: AAAI.

Miyashita, K., and Sycara, K. 1995. CABINS: A framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair. *Artificial Intelligence* 76(1-2):377–426.

Muscettola, N. 1993. Scheduling by iterative partition of bottleneck conflicts. In *Proceedings of the Ninth Conference on Artificial Intelligence for Applications*, 49–55. Orlando, FL: IEEE.

Neiman, D. E.; Hildum, D. W.; and Lesser, V. R. 1994. Expoliting meta-level information in a distributed scheduling system. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 394–400. Seattle, WA: AAAI.

Sadeh, N. 1991. *Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling.* Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University.

Sycara, K. P.; Roth, S. F.; Sadeh, N.; and Fox, M. S. 1991. Resource allocation in distributed factory scheduling. *IEEE Expert* 6(1):29–40.

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

Zweben, M.; Davis, E.; Daun, B.; and Deale, M. 1992. Rescheduling with iterative repair. In *Proceedings of AAAI-92 workshop on Production Planning, Scheduling and Control.* San Jose, CA: AAAI.