Multi-Agent Collaboration in Competitive Scenarios

Florian Fuchs

Institut für Informatik
Technische Universität München
Orleansstr. 34, D-81667 München, Germany
phone: ++49 -89 48095 254, fax: ++49 -89 48095 250
email: fuchsf@informatik.tu-muenchen.de

Abstract

For many multi-agent scenarios one can assume that the agents behave cooperatively and contribute to a common goal according to their design. However, our work focuses on competitive scenarios which are characterized by the agents' strong local interests, their high degree of autonomy, and the lack of global goals. Therefore, two agents will cooperate if, and only if, both will gain — or at least expect to gain — from that cooperation.
This paper presents a conflict resolution mechanism which is appropriate for competitive resource allocation in dynamic environments. Its main issue is the integration of negotiation strategies in a distributed scheduling scenario. The basic ideas of the conflict resolution are a two-stage mechanism for the generation of counter-proposals within the course of a negotiation and a script representation for strategies.

1 Introduction

In contrast to multi-agent settings which follow a fully cooperative model, there are scenarios with quite different qualities. These are settings without a global goal function and with agents which follow exclusively local goals. Cooperation between two agents takes place only if both sides expect to profit. We denote such scenarios as competitive.

Our application is a cooperation scenario involving an amount of companies. A company will always try to maximize its own profits and improve its market position rather than pursueing a goal which is common to all companies in the market. For example, it will not support the goal to minimize the overall[1] work-in-progress inventory. Nevertheless, it may take advantage of cooperation with competitors. For example, an agent might offer orders to other agents if they can process them cheaper, or if the local capacities are not sufficient.

A restricted view to the global system state and agents pursueing different local goals are typical qualities in multi-agent systems, whereas in traditional centralized structured approaches these qualities hardly play any role. As a consequence of these problem characteristics conflicts arise among the agents which have to be resolved. I.e. mechanisms

[1] Here, 'overall' means taking into account the work-in-progress inventory of all companies in the scenario together.

for conflict resolution have to be integrated into a multi-agent approach in order to obtain global consistent solutions.

In competitive scenarios some problems make the conflict resolution more difficult than in fully cooperative ones. First of all, the amount of common knowledge is very small. In general, agents do not know the other agents' plans, goals, strategies, etc. An agent will not provide the others with that kind of information to avoid that they can take advantage of this knowledge. Thus, all the agents can do is to extract information out of the negotiation processes and build models of its competitors which are largely uncertain.

Furthermore, if an agent provides some information, it may not tell the truth. Interesting questions are: When is it profitable for agents to lie, and how can lies be discouraged. Palatnik and Rosenschein investigated some problems arising in that context [PR94].

The resolution of a conflict is achieved by the application of a conflict resolution strategy. There are different situations in which conflicts occur, e.g. the conflict may concern a part of the schedule which is either still coarse grained or where already much fine planning was involved. Furthermore, we have different scenarios, e.g scenarios with many alternative resources or such without any alternatives. Different situations and different problem scenarios require varying strategies to resolve a conflict successfully. An agent therefore needs conflict resolution knowledge that enables it to react appropriately when it is faced with a conflict.

This paper presents a conflict resolution mechanism for distributed resource allocation which is suitable for competitive scenarios. The rest of the paper is structured as follows: In section 2 different approaches are discussed and requirements for an appropriate conflict resolution mechanism are pointed out. Section 3 introduces the new concept, and finally, in section 4 some conclusive remarks are made.

2 Requirements

Competitive scenarios show some specific characteristics which require appropriate conflict resolution strategies. Most methods suitable for fully cooperative settings are not applicable.

Many approaches to conflict resolution among cooperative agents employ a more or less static set of strategies, one of which is chosen as a common strategy in the case a conflict occurs. This strategy might be either chosen through a negotiation between the agents (see for example the Cooperative Experts Framework [LLC91]), or might be agreed upon directly because the agents have identical conflict resolution units (see for example the Cooperative Design Engine [KB91]). With strategies which are common to all agents it is virtually impossible to model competitive scenarios where the agents are not willing to share the knowledge about their employed strategies. They have to be enabled to apply arbitrary strategies, however they believe their local interests are supported efficiently with.

Another popular method in cooperative domains is to let one of the conflicting parties play a special role in the conflict resolution process. In the work of Polat and Guvenir, for instance, the agents have different roles in the negotiation process according to their knowledge and problem-solving capabilities [PG92]. As far as competitive scenarios are concerned, this is not a feasible way. Here, agents will refuse to take part in negotiations

in which they do not have equal rights, and therefore, may be put at a disadvantage or may be cheated.

Quite similar drawbacks have approaches in which the conflict resolution is done by an arbitrator agent (see for example [Wer90] or [SRSF90]). The existence of such an arbitrator is a quite unrealistic assumption in most cases. Conflicts rather have to be resolved by direct negotiation between the conflicting parties.

Let us briefly summarize the requirements for a conflict resolution mechanism suitable for our scenario which we have found so far:

- **Symmetry**
  None of the conflicting parties must play an extraordinary role in the conflict resolution process, they must be given equal rights.
- **Private Strategies**
  Strategies have to be private in the sense that they are freely choosable by each of the conflicting parties, and they are not accessible by the other agents.
- **No Arbitrators**
  The agents have to resolve conflicts without help from some higher level instance.

Furthermore, there are several requirements which are not specific for competitive scenarios, but have to be considered in other settings as well. The most important ones are listed in the following:

- **Stability**
  The conflict resolution should not yield an unstable behavior of the overall system, since instability results in undesired unpredictability of future market trends. In our scenario a stable system behavior is in the first place expressed by stable prices and a low contract cancelation rate.
- **Exchangeable Strategies**
  An important characteristic of a market scenario is its dynamic, the company goals are subject to constant change. Therefore, an agent must have the capability to choose appropriate strategies, taking into consideration its own goals, the model it has about its negotiation partners, as well as the needs of its clients.

3 A Conflict Resolution Concept

In the following, an approach for conflict resolution in the scheduling domain is presented which meets the above-mentioned requirements. First, we will briefly describe the underlying scheduling model, then the employed agent model will be characterized, and last, the integration of conflict resolution strategies is addressed.

3.1 Scheduling model

The underlying scheduling model corresponds with the general job-shop scheduling model and is very similar to the model described in [Win93]. It is described briefly in the following.

Time is considered discrete with a granularity Δt equal to 1. I.e. a point in time t in our model is actually an interval $[t, t + \Delta t[$. Therefore, points in time can be represented as integer numbers.

First of all we have a set of resources $\mathcal{R} = \{R_1, \ldots, R_s\}$ which are available at the shop floor. Then we have resource groups G_i and $\forall i : G_i \subseteq \mathcal{R}$. One resource group is bound to every activity which can be performed at the shop floor. Thus a resource group describes the set of alternative resources which can be employed for a certain activity. In general, resource groups contain more than one resource, i.e. more than one resource has the capability for performing the specific activity. Otherwise, resources can be elements of more than one resource group, i.e. they can perform more than one activity.

A resource request $rr = \{G_1, \ldots, G_m\}$ is a set of resource groups, a resource assignment is a set of resources. A resource assignment ra fulfills a resource request rr if and only if a bijective mapping f exists from rr to ra and $f(G) = R \Rightarrow R \in G$. From this follows immediately that a resource cannot perform more than one activity at the same time.

A task T is a triple (sd_T, rr_T, d_T), with sd_T is a range of valid start dates of T, rr_T is the resource request of T, and d_T is the duration of T. A task assignment for a task T is a pair (sda_T, ra_T), where sda_T is a start date assignment of T, and ra_T is a resource assignment of T.

Furthermore we have a set of jobs $\mathcal{J} = \{J_1, \ldots, J_k\}$ which are orders from some clients. Each job J is a pair $(\mathcal{T}_J, <_J)$, with $\mathcal{T}_J$ is a set of tasks and $<_J$ is a partial order defined on $\mathcal{T}_J$. For two tasks $T_1, T_2 \in \mathcal{T}_J$, $T_1 <_J T_2$ holds true if $sda_{T_1} + d_{T_1} \leq sda_{T_2}$. This partial order reflects the temporal dependence between tasks and can be seen as a kind of production plan.

A resource schedule σ of a resource R is a partial mapping of the set of tasks to integer numbers: $\sigma : \mathcal{T} \mapsto \mathbb{N}_0^{+-}$. This mapping gives the start date assignments for all tasks which are planned for this resource.

Now, scheduling means to find a valid schedule for a given set of jobs. A valid schedule is a set of resource schedules $\{\sigma_1, \ldots, \sigma_s\}$ — one for each resource — with the following restrictions holding:

1. For all tasks the start date assignments are in the range of valid start dates[2].
2. For each job J the partial order $<_J$ is fulfilled.
3. For all tasks the resource assignment fulfills the resource request.
4. All resources which are assigned to a task are assigned for the same period of time.
5. Resources are not assigned to more than one task for the same period of time[3].

The model allows that jobs can be brought into the system dynamically (online-scheduling). For this reason not a fixed set of tasks has to be scheduled but orders arriving at any point in time have to be included in the scheduling process immediately. Therefore another restriction must be considered: All activities have to be scheduled with start dates which are not prior to the point in time the schedule is released.

[2]This also ensures that due dates are respected which are implicitly given by the valid start dates.

[3]I.e. all resources have the capacity one. But this is not a major restriction as resources with higher capacities may be modeled as multiple resources with capacity one.

For the representation of schedules we do not use an exact representation of time points or intervals. Instead, we represent the schedules with a dense function over time for each resource. This dense function describes the capacity usage of the resources and provides a feasible way for the evaluation of resource allocation requests. Maintaining exact schedules would be virtually impossible due to the large uncertainty during the early stage where inter-company cooperations are planned and negotiated.

3.2 Agent model

This section gives an overview of the applied agent model. Therefore, we will describe our agent design and how the agent is situated within its environment.

3.2.1 Agent roles

Typical for agents is that they are able to be involved in several different roles. An role of an agent forms its basic view, its goals, and behavior patterns (see [Sun93]). Agents are able to choose a certain role from their repertoire of roles according to the requirements of the current situation. In the following the two roles *order agent* and *resource agent* are essential which correspond largely to the roles *manager* and *bidder*, introduced in [Smi88] within the scope of the *contract net protocol*.

The agent roles do not determine the agent's behavior completely, thus the agent is enabled to realize different strategies. On the one hand an agent might make use of variations on the resource allocation protocols. On the other hand an agent can freely choose values for the conditions of a contract (see section 3.2.3).

3.2.2 Agent architecture

Figure 1 shows the conceptual structure of the agent model which is used in our cooperation scenario. The agent is embedded in its environment which consists of three parts: The client environment from where jobs are brought into the system, the execution environment functioning as an outlet for jobs[4], and finally the agent environment where jobs may be distributed via some cooperation mechanisms.

An agent encapsulates all the knowledge it needs to realize its goals and also the appropriate processing mechanisms for that knowledge. The knowledge is kept explicitly represented in a local knowledge base which contains in the first place knowledge about other agents, communication protocols, local goals, and skeleton plans that are instructions on how to manufacture products.

In addition to the knowledge base there are four agent components which operate quite independently and are connected via some internal interfaces.

The planning unit maintains a set of jobs it receives in irregular intervals from some clients. The planning unit has to build a production plan for a job — mostly by getting a prebuilt plan from the knowledge base — and determines a task which is to be scheduled next.

An agent is able to communicate through its communication unit with other agents. From a received message it extracts information and updates its local knowledge base

[4]This does not really have to mean that jobs are physically executed but they are rather passed on to some subordinate planning instance.

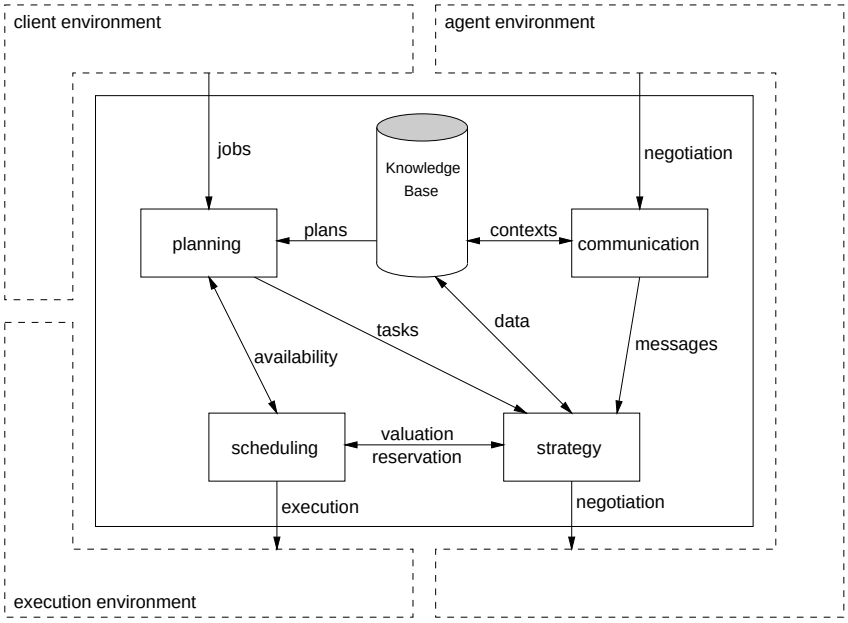

Figure 1: agent architecture

accordingly. Furthermore, the message is put into a certain context according to its type and contents. The agent then changes to the role intended for this context.

The strategy unit contributes the strategic part to the conflict management. This unit is responsible for generating an appropriate response to a received message. Therefore, it uses strategy scripts which are interpreted by an integrated interpreter (see section 3.3).

The scheduling unit manages schedules, evaluates proposals for resource allocations from the strategy unit, and is able to find alternative schedules to a given one which perform better. This unit is also the one which is connected to the execution environment and makes the necessary steps for the execution[5] of tasks.

3.2.3 Coordination and cooperation

In the distributed scheduling system each resource will be represented by a resource agent and each order by an order agent. These agents look after the goals concerning resources or orders respectively. The goals concerning a resource are mainly to use resources to capacity and to coordinate processing activities with servicing activities. The goals of an order agent can be manifold: The client's interest can be to produce a cheap product, to produce it as quickly as possible, etc.

The basis for the cooperation model is a monetary system. Order agents receive some currency for each order they are responsible for. They are free in how they spend their means of payment. For the realization of an order the order agent has to allocate a set of resources taking into consideration all temporal dependencies between the different tasks

[5] or the simulation of the execution respectively

which are necessary for a single order. For an allocation by a resource agent it has to pay a certain price. In reaction costs arise for the resource agent for processing activities and for maintaining the resources — in particular also for resources lying idle.

For any kind of work, i.e. processing of orders or usage of resources, exist standard prices. But in a concrete case prices can go over or remain under these standard prices any way you like. On the one hand prices are subject to negotiation and conflict resolution strategies, and on the other hand prices may be adjusted if, for example, due dates are exceeded.

Order agents can freely decide upon the usage of any received currency. I.e. there is no need at all that they spend their fund received for a certain order on the allocation of resources needed for that particular order.

The allocation of resources is subject to negotiation. The negotiation protocol which is made use of corresponds largely with the well-known *contract net protocol* [Smi88]. This protocol consists of the four phases *request*, *offer*, *order*, and *confirmation*. For an example for an employment of this protocol in a distributed planning system see also [HFL96].

An order agent sends a request for a task to a set of resource agents with the required capabilities. This request proposes a cooperation between two agents and consists of some elements which give a description of that proposed cooperation — e.g. a requested resource type, a offered price, etc. The resource agents in turn generate an offer as an alternative proposal. This offer possibly accepts the request totally or makes some modifications on parts of the request. These two negotiation phases are repeated until a solution acceptable to both is found. The message types *order* and *confirmation* finish a negotiation.

We assume that the agents behave rationally. In our approach, an agent is rational if, and only if, it pursues the goal to maximize its profits. Thus, the termination of a negotiation is ensured if at least one of the agents uses a 'reasonable' strategy. This follows from the fact that an exceeding of due dates — because of long negotiations — lowers the order agent's profit, and eventually it is cheaper to accept an inconvenient offer or to interrupt the negotiation. However, having two agents which both behave irrationally, a reasonable negotiation result cannot be expected.

If one task requires more than one resource all these resources are allocated through only one negotiation. The reason is that these resources must be reserved for the same time interval. Several independent negotiations for one task would require a much more complex synchronization mechanism.

One might think that the communication overhead is an issue in such a decentral approach to the resource allocation problem. But at least in the flexible manufacturing domain — which is our main concern, but also in others — the execution times for the activities are very long in contrast to the negotiation times. Similar results could be proven with practical experiments in [HFL96].

3.3 Conflict resolution strategies

Distributed scheduling yields conflicts which result from inconsistent allocation requests of different agents for the same resource. These conflicts have to be resolved. In other words, such a resource conflict means requests for one resource by at least two agents for

overlapping periods of time. The reason for these inconsistencies lies both in the agents' restricted local views and in their differing local interests.

3.3.1 The resolution mechanism

The agents use their capability to communicate for the resolution of any arising conflicts. The syntactic, semantic, and procedural aspects of negotiation are specified in communication protocols which also describe a set of message types. Essential for strategies are the message types *request* and *offer*, both representing cooperation proposals. These proposals consist of several conditions of a contract. Such a condition is an attribute like a price or a time interval. A favorable assignment of values to the terms of a contract is used for managing the conflict resolution and thus for scheduling strategies.

At first we have to clarify what kind of strategies we are considering. In the first place a strategy supports the decision whether a request — or an offer respectively — for a resource allocation should be accepted or not. An efficient strategy must not only determine a threshold which is to be exceeded by the valuation of a proposal. Beside a valuation also other aspects should be considered including the current supply and demand for the requested resource, order priorities, or the course of the negotiation so far. However, strategies should be designed for generating an agent behavior which supports the agent's goals.

If an agent rejects a proposal addressed to it, it has to generate an alternative suggestion. This alternative suggestion is built by adjusting some or all of the conditions of the contract in a way that the valuation of the so corrected proposal becomes more favorable. In principle, an alternative always exists because one can always shift a time interval to some later start date — although the price may become very high if due dates are exceeded. It is also possible that negotiations are interrupted without having found a feasible solution. An order agent might do so if alternative resources exist.

An alternative proposal is found through a two-stage mechanism: In a first phase the received proposal is analyzed according to its utility for the agent's local goal. Therefore, the profit is computed dependent on varying values for the conditions of the contract. In other words, we have a cost function which maps contract conditions to the profit resulting from those values. Note that the outcome may have a negative value as well. This can happen whenever an offered price is not high enough to compensate the expenditures which arise from the insertion of the new task into the schedule[6].

The overall gain is the difference between the values of the schedule with the requested task inserted according to the proposal and the current schedule. If no overlappings occur when the task is inserted, a simple balance can be calculated. If in contrast the task overlaps one or more other tasks, first a cancelation must be found which allows the insertion. The costs for any cancelation have to be considered in the balance. Note that it may be a complex problem to find an optimal cancelation which is the cheapest concerning the expense it imposes. But one can find simple heuristics which lead to suboptimal solutions and achieve good approximations.

With this cost function we can build a proposal evaluation by calculating the overall gain for some discrete values for the conditions of the contract around the proposed values.

[6]These expenditures especially become very high if other tasks have to be canceled in order to make the insertion possible.

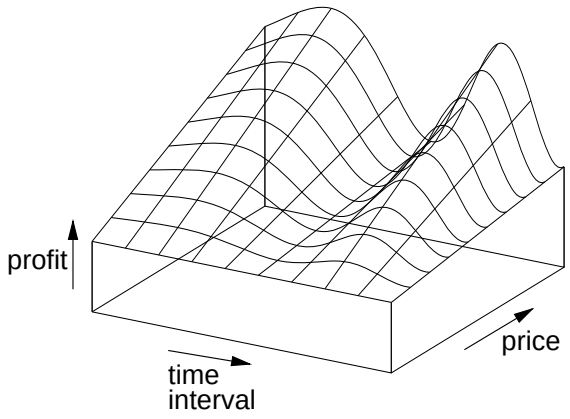

Figure 2: proposal evaluation

Figure 2 shows an example of such a proposal evaluation taking into account just two contract terms 'time interval' and 'price'[7].

This proposal evaluation constitutes the basis for the second phase of our generation of counter proposals. Any proposal can be seen as a point in the proposal space defined by the conditions of the contract. Profit is a characteristic of each proposal. Now, the task of the second phase is to move the cooperation partner's proposal to some alternative point in the proposal space. For that, the profit characteristic of each point is used which was computed in the first phase.

Two aspects have to be considered for the counter-proposals. First, the agent has to take into account its local goals and guide the search for the counter-proposal accordingly. Second, if an agent wants to increase its profit by cooperating it has to make that cooperation possible by generating acceptable proposals. This requires the capability to compromise, and hence, the capability to maintain appropriate models of the competitors.

Example

Figure 3 illustrates an example for a search for a counter-proposal. It is an isoline version of figure 2 with a received proposal marked as P. It is assumed that the agent's goal is to increase the utilization of its resources and it regards the following strategy as appropriate for achieving this goal: It tries to make its offers as attractive as possible for its negotiation partners. Therefore, it asks the lowest price which is still acceptable in combination with a time interval which is not too far from the original one.

An agent's negotiation strategy has to be expressed by an intelligent search strategy in the proposal space. Therefore, we provide a script language with high level operations such as 'climb up a hill' or 'go along an isoline'. Those operations can be restricted by some constraints like a maximum distance or a range of search directions if appropriate.

[7]For simplicity reasons, just two dimensions are considered. In general, the evaluation can have more dimensions.

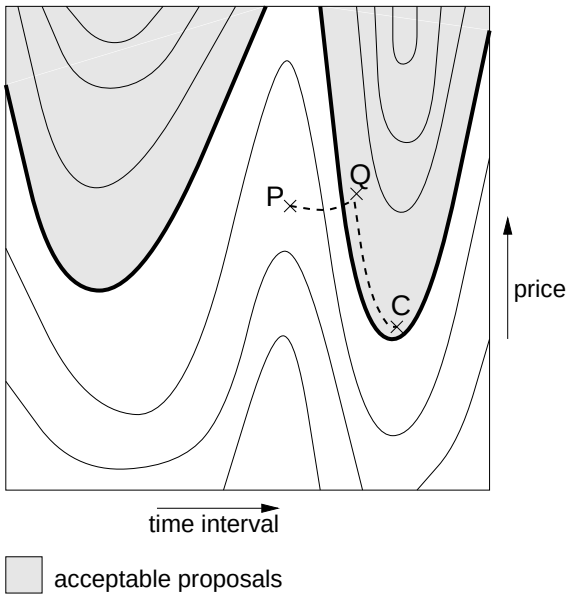

Figure 3: proposal evaluation

The terms of a contract are estimated as acceptable if their valuation exceeds an acceptance threshold. In our example, the first goal of the search strategy is to reach a near and acceptable point. This is accomplished by applying the hill-climbing instruction which is provided by our script language together with an additional constraint on the maximum altitude to climb and yields point Q. The reachability of a certain altitude is ensured if an agent is rational, an therefore, uses a proposal valuation which is strong monotonous in the price dimension. The goal of the second part of our search strategy is to lower the price condition of the contract without changing its overall valuation. This is achieved by walking along an isoline through Q as long as the price still gets down — to point C in figure 3. □

Note that a proposal evaluation does not have to be computed completely. Due to our negotiation mechanism it is sufficient to compute some values in the near environment of a received proposal first and then enlarge this region in the desired direction.

3.3.2 Strategy scripts

Strategies are represented as scripts which are to be interpreted by the strategy unit. They are embedded in the negotiation protocols as far as the strategy scripts are restricted to those respecting the valid protocols. I.e. the negotiation protocols provide skeletons for strategy scripts. The computations concerning the negotiation strategies have to be filled into these skeletons.

An agent is able to take part in several negotiations simultaneously, i.e. the interpreter must handle several strategy scripts at the same time. This is achieved by the interruptability of scripts and by introducing negotiation contexts. Scripts are interruptible in a way that whenever a message has to be waited for according to the employed negotiation

protocol the interpreter saves the current state — in order to make it possible to resume this negotiation eventually — and suspends the interpretation of the current script. At the beginning of a negotiation a new negotiation context is created and each message belonging to that particular context is labeled with this context. Thus on the receipt of a message an agent can decide which negotiation to resume.

The simultaneous handling of strategy scripts causes side-effects between them in a way that a decision made within a script A may have impact on a negotiation realized through a script B. To reduce the number of side-effects the agents do not negotiate simultaneously about tasks which are constrained by a partial order, i.e. not about two tasks $T_1, T_2 \in \mathcal{T}_J$ with $T_1 <_J T_2 \vee T_2 <_J T_1$. Remaining conflicts between negotiations are resolved in a reactiv way. Negotiations which are finished first will be inserted into the resource schedule.

4 Conclusion and future work

This paper presented a conflict resolution concept for multi-agent systems which is suitable for competitive scenarios.

The most important feature of this concept is that the agents' conflict resolution strategies are private, i.e. not accessible by other agents. Therefore, an agent is enabled to preserve its local interests. Of course the possibility to share a common strategy remains. Furthermore, the agents have equal rights concerning the resolution mechanism — no coordinating instance at a higher level is needed. The strategies are represented in a script form which provides easy readability and modifiability together with a high expressiveness. Strategies are not hard coded within the algorithms like in previous works in that field, and thus allowing dynamic adaption of negotiation strategies at run time.

Currently effort is spent on building a prototypical implementation of the described system. This prototype will also include a simulation environment which serves as a testbed for the efficiency of different strategies in different manufacturing scenarios. The next goal is to investigate the effects of some strategies on the scheduling results — especially the impact on job lateness, in-process inventory, job idle time, and machine utilization. We hope to finish the first prototype implementation in a few months and will come out with the first experimental results soon afterwards.

References

[HFL96] S. Hahndel, F. Fuchs, and P. Levi. Distributed Negotiation-based Task Planning for a Flexible Manufacturing Environment. In J. W. Perram and J.-P. Müller, editors, *Distributed Software Agents and Applications*, number 1069 in Lecture Notes in Artificial Intelligence. Springer, 1996.

[KB91] M. Klein and A. B. Baskin. A Computational Model for Conflict Resolution in Cooperative Design Systems. In S. M. Deen, editor, *CKBS'90: Proc. of the International Working Conference on Cooperating Knowledge Based Systems, October 1990, Univ. of Keele, UK*, pages 201–219. Springer, Berlin u.a., 1991.

[LLC91] S. Lander, V. R. Lesser, and M. E. Connell. Conflict Resolution Strategies for Cooperating Expert Agents. In S. M. Deen, editor, *CKBS'90: Proc. of the International Working Conference on Cooperating Knowledge Based Systems, October 1990, Univ. of Keele, UK*, pages 183–200. Springer, Berlin u.a., 1991.

[PG92] F. Polat and H. A. Guvenir. A Conflict Resolution Based Cooperative Distributed Problem Solving Model. In *Proc. of AAAI-92*, pages 106–115, 1992.

[PR94] M. Palatnik and J. S. Rosenschein. Long Term Constraints in Multiagent Negotiation. In 13^{th} *International DAI Workshop*, pages 265–279, Seattle, Washington, 1994.

[Smi88] R. G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. In A. H. Bond and L. Gasser, editors, *Readings in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., San Mateo, California, 1988.

[SRSF90] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Managing Resource Allocation in Multi-Agent Time-Constrained Domains. In *Proc. of a Workshop on Innovative Approaches to Planning, Scheduling, Control*. Kaufmann, 1990.

[Sun93] K. Sundermeyer. Modellierung von Agentensystemen. In J. Müller, editor, *Verteilte Künstliche Intelligenz — Methoden und Anwendungen*, pages 22–44. BI Wissenschaftsverlag, 1993.

[Wer90] K. Werkman. Design and fabrication problem solving through cooperative agents. Technical report, Lehigh University Bethlehem, 1990.

[Win93] A. Winkelhofer. *Zeitrepräsentation und merkmalsgesteuerte Suche zur Terminplanung*. PhD thesis, Technische Universität München, 1993.