A MODEL FOR COOPERATIVE TRANSPORTATION SCHEDULING

Klaus Fischer, Jörg P. Müller, Markus Pischel, Darius Schier
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, D-66123 Saarbrücken

Abstract

The MARS system is described which models cooperative scheduling within a society of shipping companies as a multiagent system. Emphasis is placed on the functionality of the system as a whole — the solution of the global scheduling problem emerges from local decision-making and problem-solving strategies. An extension of the contract net protocol is presented; we show that it can be used to obtain good initial solutions for complex resource allocation problems. By introducing global information based upon auction protocols, this initial solution can be improved significantly. Experimental results are provided evaluating the performance of different cooperative scheduling strategies.

Although the concepts for resource scheduling are presented solely for the transportation domain, their abstraction is useful for a broad variety of resource allocation problems. The MARS system solves the dynamic scheduling problem where no complete specification of the problem is available a priori; thus, it is designed as an on-line system based upon anytime algorithms.

Topics:

Practical Applications of Multiagent Systems
Resource Allocation in Multiagent Systems

INTRODUCTION

Bidding protocols have been advocated as a valuable metaphor in the design of distributed problem solving for various problems. (Davis & Smith 1983) proposed the famous contract net protocol for task decomposition and task allocation in multiagent systems, assuming a setting where the agents are completely cooperative in the sense that they always tell the truth and that they pass utility to other agents without restrictions. In order to deal with a competitive setting the walrasian auction was introduced (Wellman 1992). (Lenting & Braspenning 1994) defined the all pay auction to increase the global performance of the walrasian auction.

In this paper we argue on the on hand that the solutions found by distributed task allocation can be significantly improved by introducing global information. On the other hand, we show that the transportation domain (Fischer *et al.* 1993; Sandholm 1993) offers both a cooperative and a competitive setting. We introduce a modification of the contract net protocol to solve the distributed task allocation problem and a procedure called simulated trading (Bachem, Hochstättler, & Malich 1993) to optimize a given solution iteratively. The performance of both strategies is evaluated by a set of benchmarks.

Our domain of application is the planning and scheduling of transportation orders which is done in everyday life by human dispatchers in transportation companies. Many of the problems which have to be solved in this area, such as the Traveling Salesman and related scheduling problems, are known to be NP-hard. Moreover, not only since *just-in-time* production has come up, planning must be performed under a high degree of uncertainty and dynamics. In reality these problems are far from being satisfactorily solved.

The MARS simulation testbed (cf. (Kuhn, Müller, & Müller 1993)) constitutes a multiagent approach to the transportation domain; it describes a scenario of geographically distributed transportation companies that have to carry out transportation orders arriving dynamically. For this purpose, they have a set of trucks at their disposal. We evaluate the behavior of the system as a whole in a straightforward manner: the measure of coherence is the quality (costs) of the schedule. Note that the companies themselves do not have facilities for scheduling orders; rather, it is their trucks that maintain local plans. The actual solution to the global order scheduling problem emerges from the local decision-making of the agents. There are two basic types of agents in MARS corresponding to the physical entities in the domain: *shipping companies* and *trucks*. Looking upon trucks as agents allows us to delegate problem-solving skills to them (such as route-planning and local plan optimization). The *shipping company* agent has to allocate orders to its trucks, while trying to satisfy the constraints provided by the customer as well as local optimality criteria (costs). A company also may decide to cooperate with another com-

pany instead of having an order executed by its own trucks. Each *truck* agent is associated with a particular shipping company from which it receives orders of the form "`Load amount` a_1 `of good` g_1 `at location` l_1 `and transport it to location` l_2 `while satisfying time constraints` $\{c_1, \ldots, c_n\}$".

In earlier versions of the system, dynamics occurred solely by the asynchronous arrival of transportation orders. Once a truck had accepted an order, it was sure to reach his destination in time; thus, there was no need of replanning. We dropped this restriction by introducing a model for simulating traffic jams in the system (Fischer *et al.* 1994): the time a truck needs in order to go from one place to another varies dynamically according to the output of a simulation model for traffic jams. Thus, a truck has to reconsider parts of its plan each time before it starts driving and possibly has to change it.

TRANSPORTATION SCHEDULING AND VERTICAL COOPERATION

Interaction of the agents within one shipping company (called *vertical cooperation*) is totally cooperative. This means that a specific truck agent (TA) will accept deals (i.e. results of negotiation processes) even if he[1] does not benefit from it. We call such a setting an instance of a *cooperative task-oriented domain* (cf. (Fischer 1994)). In the cooperation between shipping companies agents (SCA) we investigate in both a totally cooperative and a competitive setting (we call the latter setting an instance of a *competitive task-oriented domain*). If we assume a cooperative task-oriented domain, we are purely interested in the quality of the overall schedule which is emerging from the local problem solving done in the SCAs and TAs.

On the other hand, if a competitive task-oriented domain among the SCAs is assumed, it is clear that the overall schedule which is computed will be far from optimal. In this setting we investigate how a single SCA can maximize her profits and how she can avoid being tricked by other agents. In this paper we will concentrate on the cooperative setting and refer to (Fischer 1994) for the discussion of the latter setting.

Finding an Initial Solution

If an order o is announced to an SCA by a customer (which can also be another SCA), she has to compute a bid for executing the order. In order to determine the costs, she forwards the order to her TAs. Each TA $A_i, 1 \leq i \leq n \in I\!N$ computes a bid $(A_i, \text{insertioncosts}(T_i, o)^2, a)$, where T_i is the current tour of TA A_i and a is the amount of the order A_i is able to transport. Let $\mathcal{O}^i = \{o_1, \ldots, o_{m_i}\}, m_i \in I\!N$ be the current set of orders of TA A_i. A constraint net is derived from the information which is specified with the orders. Each solution to this constraint solving problem is a valid tour which fulfills all constraints specified by $\mathcal{O}^i$. A_i tries to find the best tour for $\mathcal{O}^i$ using a constraint solving and constraint optimization procedure[3]. For each order o an SCA announces to her TAs, she gets a set of bids

$$\mathcal{B} = \{(A_1, c_1, a_1), \ldots, (A_n, c_n, a_n)\}, n \in I\!N$$

where c_i specifies the costs that arise to TA A_i when executing amount a_i of order o. The SCA selects $(A_{min}, c_{min}, a_{min}) \in \mathcal{B}$ with

$$\forall (A, c, a) \in \mathcal{B} : \frac{c_{min}}{a_{min}} \leq \frac{c}{a}$$

and sends a grant to the TA A_{min}, notifying him that he will be granted the amount a_{min} provided that the SCA itself will actually receive a grant for o by the customer.

The procedure described so far is the well known Contract Net protocol (CNP) (Davis & Smith 1983). Because the CNP protocol provides time-out mechanisms it is easy to turn it into an anytime algorithm (Boddy & Dean 1994; Russell & Zilberstein 1993), producing a solution whose quality increases monotonically if more time for computation is available. Of course, it is possible that no solution is found within a specified time t_0. In this case the specified transportation order has to be rejected.

The Extended Contract Net Protocol

The pure contract net protocol runs into problems if the tasks exceed the capacity of a single TA, i.e. $a_{min} < \textit{amount-to-transport}(o)$. In this case, the manager of the task, i.e. the SCA, has to solve a knapsack problem, which for itself is in general NP-hard. To overcome this problem, we have decentralized task decomposition by developing an extension of the CNP, which is called the *extended contract net protocol* (ECNP). In the ECNP, the two speech acts *grant* and *reject* are replaced by four new speech acts: *temporal grant*, *temporal reject*, *definitive grant*, and *definitive reject* (see Figures 1 and 2).

In the ECNP the manager (SCA) announces an order o to its TAs. It then receives bids for the order and selects the best one as specified above. The best TA is sent a temporal grant. All others receive temporal rejects. If the best bid does not cover the whole amount of an order, the remaining part of the order is reannounced by the SCA. This procedure is repeated until there is a set of bids that covers the total amount of the original order o. From this set of bids the SCA

[1] We use 'he' to refer to truck agents (TA) and 'she' to refer to shipping company agents (SCA) to resolve ambiguities.

[2] $\text{insertioncosts}(T_i, o) =_{def} \text{cost}(T_i \oplus o) - \text{cost}(T_i)$. $\text{cost}(T_i \oplus o)$ denotes the additional costs for A_i when executing o given T_i.

[3] Our implementation is based on the Oz (Henz, Smolka, & Würtz 1993; Schulte, Smolka, & Würtz 1994) language developed at DFKI.

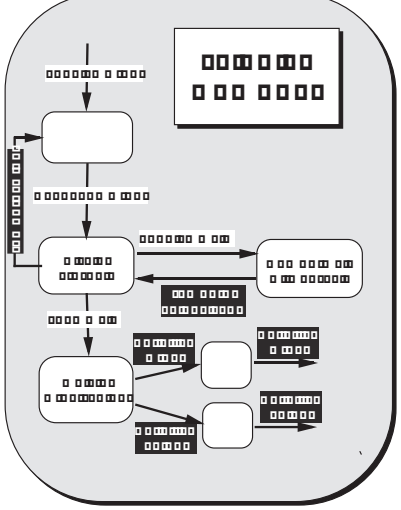

Figure 1: The ECNP Protocol (Manager)

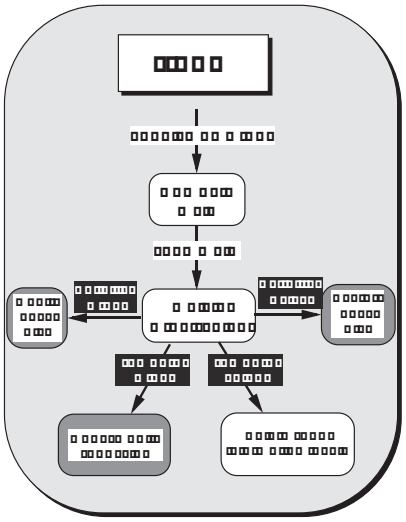

Figure 2: The ECNP Protocol (Bidder)

computes a bid which is passed to the customer. Based on the answer of the customer, the SCA sends a definitive grant (or definitive reject, respectively) to all TAs which got temporal grants before. It is possible to prove that in general all but the last bid selected are locally optimal choices for the SCA (Fischer, Kuhn, & Müller 1994).

When a TA receives a temporal grant for the first time, it has to store a copy of its local situation, i.e. the currently valid plan, because it must be able to restore this situation in case it obtains a definitive reject. All subsequent temporal grants and temporal rejects are handled like the grants and rejects in the pure CNP. If a TA is sent a definitive grant for an order, it removes the copy created above and switches to the new plan. If a TA gets a definitive reject, it restores the situation before the first temporal grant.

Simulated Trading: An Auction Procedure for Further Optimization

Using the ECNP an SCA distributes incoming orders to her set of TAs. However, because the situation changes by new orders coming in and because the TAs will stick to decisions made in the past, the solution found is not even guaranteed to be pareto-optimal (Wellman 1992). At any point in time, when no ECNP bidding process is active, each TA has a valid tour for the suborders granted to him in the ECNP. For further optimizing this solution, we use an auction mechanism called simulated trading (ST) (Bachem, Hochstättler, & Malich 1993).

The main idea is to let the SCA simulate a stock exchange (see figure 3) where her TA can offer their current orders at some specific "saving price" and may buy orders at an "insert price". While getting sell and buy offers form her TAs the SCA tries to find an order exchange that optimizes the global solution. This is done by assigning each offer of a TA to a node in a so-called trading graph $TG = (V, E)$. A node v can be a buy node $v \in V_b$ or a sell node $v \in V_s$, i.e. $V = V_b \cup V_s$. Each node has a label $v = (A, l, o)$ which denotes the name A of the TA, the level l denoting the number of preceding offers A has sent to his SCA, and the order o. For each buy node v_b, the SCA inserts a directed edge (v_b, v_s) to a sell node v_s referring to the same order. The price of a trade (the weight of each edge) is the difference between the selling and the buying price of the order. Thus, a global interchange of k customers between all of the current tours of the TA corresponds to a matching in the trading graph. The weight of the matching is defined by the profit of this global interchange. Searching for a trading matching is done by a complete enumeration of the trading graph. Though this requires exponential time in the worst case, it turned out to be feasible in practice since normally the trading graph does not have too many branches. Whereas we allowed the splitting of orders into suborders in the ECNP, we forbid it in the simulated trading process, to restrict combinatorial explosion.

The buy- and sell-offers of the TAs are divided in different decision levels. That means that the decision process entering level l requires all previous offers in lower levels $1, \ldots, l-1$ to be successfully finished. For later use we will refer to this constraint as the *level constraint*. The corresponding action of decision level l changes the tour of the acting agent.

We define l_i^{min} and l_i^{max} as the first and last decision level of TA A_i, respectively. If we have $l_i^{min} > l_i^{max}$ the agent A_i does not get involved into the simulated trading procedure, at all. Else we can recursively define

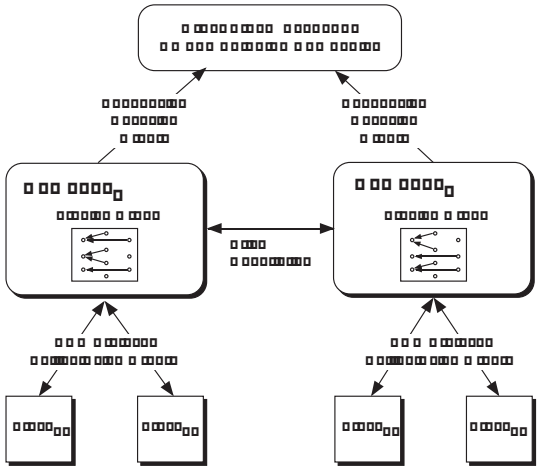

Figure 3: Hierarchical organization of the agents in MARS.

the tour of TA A_i of level $l \in I\!N$:

$$T_i^{(l_i^{min}-1)} = T_i \text{ and for } l \geq l_i^{min}:$$

$$T_i^{(l)} = \begin{cases} T_i^{(l-1)} \ominus \{o\} & \text{if agent } A_i \text{ sells} \\ & \text{order } o \text{ in level } l \\ T_i^{(l-1)} \oplus \{o\} & \text{if agent } A_i \text{ buys} \\ & \text{order } o \text{ in level } l \end{cases}$$

with l_{max} we define the maximal number of decision levels such that the inequality $l_i^{max} - l_i^{min} < l_{max}$ always holds. Each TA decision of buying or selling orders is represented as a quintuple denoting tour, level, customer, price, and decision type (buy or sell).

The current decision level has to take into account preceding decisions. The prize for buying or selling an order is calculated as savings or insertion costs depending on the type of decision. When an order is going to be sold, the savings are calculated; otherwise the minimal insertion costs for this order are determined.

At every decision level, each TA sends at least one of the above mentioned quintuples to the SA. The sell and buy decisions of all TAs now form a bipartite graph where a sell- and a buy-node are adjacent if they refer to the same order.

This directed bipartite graph G is called *trading graph* and is defined by:

$$G = (V = V_b \cup V_s, E) \text{ with}$$
$$V \subset \{A_1, \ldots, A_n\} \times \{1, \ldots, l_{max}\} \times \{o_1, \ldots, o_m\}:$$
$$v = (A_i, l, o) \in V_b \iff \text{TA } A_i \text{ with tour } T_i^{(l-1)} \text{ buys order } o \text{ in level } l$$
$$v = (A_i, l, o) \in V_s \iff \text{TA } A_i \text{ with tour } T_t^{(l-1)} \text{ buys order } o \text{ in level } l$$
$$(v, w) \in E \iff v \in V_b, w \in V_s : v = (\_, \_, k), w = (\_, \_, k)$$

In this definition, '_' denotes an anonymous variable. Further let $\rho : V \to I\!R^+$ denote the price of a decision defined as:

$$\rho(v = (A_i, l, o)) = \begin{cases} \text{savings}(T_i^{(l-1)}, o)^4 & v \in V_s \\ \text{insertioncosts}(T_t^{(l-1)}, k) & v \in V_b \end{cases}$$

So if the SA receives a buy-offer a new node is added to V_b and connected to all sell-nodes referring to that order. In case she receives a sell-offer, a new node in V_s is inserted and, if there is already an buy-offer for that order, a link from the inserted node to the corresponding buy-node is created.

Since our objective is to implement the exchange of orders among TAs, it may occur that edges and nodes have to be deleted from the trading graph. A property that we always want to preserve is *admissibility*:

A *trading graph* $G = (V = V_s \cup V_b)$ is called *admissible* $:\Longleftrightarrow$

$$\forall v \in V_b \exists w \in V_s \text{ with } (v, w) \in E$$
$$\forall l_1 < l_2 < l_3 \text{ with } (A, l_1, \_), (A, l_3, \_) \in V :$$
$$\exists v \in V \text{ with } v = (A, l_2, \_)$$

This definition implies that each buy-node in an admissible trading graph is adjacent to at least one sell-node and that all decision levels are continuously planned. Note that the admissibility condition is not violated if the SA inserts a node or an edge in the graph because each TA is continuously planning his decisions from one level up to the next higher one and is allowed to buy only the orders that are offered in the trading graph.

Now the SCA has to search for a set of node pairs, each of which consists of a buy-node and a proper sell-node under the conditions that every node is included in at most one such pair and that the level constraint is not violated. The level constraint implies that if a node pair contains the decision in level l of tour T_i every foregoing decision of TA A_i must be included into another pair. Because every pair corresponds to an edge of the trading graph we can define the set of pairs we are looking for by a matching under constraints: Let $G = (V = V_b \cup V_s, E)$ be an admissible trading graph with $V \subset \{A_1, \ldots, A_n\} \times \{1 \ldots, l_{max}\} \times \{o_1, \ldots, o_n\}$. Further let $\emptyset \neq M \subset E$ denote a set of edges. Let M_V denote the set of nodes as follows:

$$M_V = \{v \in V \mid \exists w \in V \text{ with } (v, w) \in M \vee (w, v) \in M\}$$

The set of edges $M \subset E$ is called a *trading matching* if it satisfies the following conditions:

$$\forall ((t_v, l_v, k_v), (t_w, l_w, k_w)) \in M :$$
$$u_v = (t_v, l, \_) \in V \wedge l \leq l_v \Rightarrow u_v \in M_V \text{ and}$$
$$u_w = (t_w, l, \_) \in V \wedge l \leq l_w \Rightarrow u_w \in M_V \text{ and}$$
$$\forall v \in M_V \cap V_b : \exists (v, w) \in M \text{ and}$$

[4] $\text{savings}(T_i^{(l-1)}, o) =_{def} \text{costs}(T_i) - \text{costs}(T_i \ominus o)$

$$\forall v \in M_V \cap V_s : \exists (w, v) \in M$$

Thus, according to this Definition a trading matching now is an admissible trading graph that is a matching. The gain of a trading matching M_V is defined as:

$$\text{gain}(M_V) = \sum_{v \in M_V \cap V_s} \rho(v) - \sum_{v \in M_V \cap V_b} \rho(v)$$

Important for the ST procedure are the decision criteria for the TA to decide which orders to sell or buy. This is done using heuristics like "buy nearest" and "sell farthest" combined with randomization techniques.

Note that simulated trading can only be active during a period of time when no new orders arrive at the SCA. Nevertheless, while the ST process is active the system maintains a valid solution because ST is done using a copy of the current plan of a TA and the current plan is replaced by the new one computed via the simulated trading procedure only if that was successful, i.e. a trading match was found which led to a new optimum. Thus, reactivity is guaranteed: when a new order arrives, the TA always uses the consistent original plan to compute a bid for the ECNP. If a new order occurs while simulated trading is active, the procedure has to be aborted, unless the order fits into the plan which has been used for the ST process.

Plan Execution and Replanning

An important feature of the MARS system is that TAs do not only *compute* plans: when time is up, they actually start *executing* the orders. Executing an order includes the steps of loading, driving, and unloading. Note, that even after the TA already has started the execution of his local plan, it is possible for him to participate in the ECNP protocol. However, in the ST process the TA is not allowed to sell orders it has already loaded.

A problem in plan execution is that planning is done on statistical data which may be too optimistic. For instance, when the plan is actually executed the TA may get stuck in a traffic jam. Therefore, replanning might be necessary because the TA may run into problems with respect to the time constraints which are specified with the orders. Fortunately, this situation can be nicely handled in our framework. We distinguish two cases:

Firstly, there are disturbances that can be resolved using local replanning. In some cases, the TA can do this by selecting an alternative route to the next city he has to deliver orders to. This is done by computing the shortest path in a dynamically changing graph using Dijkstra's algorithm. In other cases, this can force the TA to completely recompute his local plan using his local planning procedure. Even if the TA is able to successfully derive a new plan which satisfies all constraints, the quality of the plan may drop and thus, some orders may be sold within the next ST process. Therefore, restricted global rescheduling may occur already in this case.

Secondly, if the TA cannot fix the problem by local replanning, the procedure depends on whether the order is already loaded on the TA or if it is not. In the latter case, the TA initiates a simulated trading process to sell the orders that he is no longer able to execute. If a trading matching is found, this is a solution to the problem. If the simulated trading process does not find a valid solution for the situation, the TA has to report the problem and return the respective orders to his SCA. In this case the SCA herself can decide whether to sell the order to another SCA (see below) or to contact the customer, report the problem, and try to negotiate about the violated constraints. In the worst case, the company has to pay a penalty fee.

If the orders that are causing trouble are already loaded on the TA, it is not possible to just return the order to the SCA or to sell it in a simulated trading process. In this case, the only chance for the TA is to report the problem to the SCA which then has to find a solution by contacting the client, trying to relax the constraints of the order. If a TA runs into this situation he is paralyzed in the sense that he cannot participate in the ECNP or in the simulated trading process until he receives instructions from his SCA. Fortunately, the ECNP and the simulated trading procedure can deal with this situation because they do not require participation of all TAs.

HORIZONTAL COOPERATION

Optimizing the utilization of transport capacities is the foremost goal for an SCA. Due to the spatial and temporal distribution of incoming orders, cooperation with other SCAs (so-called *horizontal cooperation*) may be a beneficial operation. Although it would be possible to use the simulated trading approach also for global optimization by switching orders between SCAs, we claim that such an approach is inadequate, for the following reasons:

1. Within one shipping company, the setup for the simulated trading process can be provided easily; however, even in this case the current plans of the TAs have to be frozen and each TA involved in this process can only accept new orders after the simulated trading process is finished. A simulated trading process between SCAs would require an immense synchronization effort and a considerable communication overhead.

2. In contrast to the coordination between a company and its TAs, cooperation between companies is a peer-to-peer process where a solution (e.g. a price to be paid for an offer) can only be found if all the participants agree, and where the conditions of the solution have to be negotiated among the companies. Thus, there is no global decision authority in order to control the negotiation process.

3. SCAs will behave more selfishly than TAs in negotiation. Therefore, global optimality of the overall schedule which emerges from local problem solving is no longer the key criterion to guide negotiation. Rather the SCAs will try to maximize their own profits by selling and buying orders among each others. Selfishness is also the reason why, in general, the information about orders, costs, and prices necessary for the simulated trading algorithm cannot be assumed to be publicly available.

Because of these reasons we chose a model providing a stock exchange for transportation orders among SCAs which is organized as a blackboard to which SCAs can post orders they would like to sell specifying a price they would like to achieve (see figure 3). If another SCA wants to buy an order, a bilateral negotiation process between these SCAs is started determining the actual price to be paid for the order in a decentralized manner.

The decision-making of the companies during the negotiation process is based on information they obtain by their TAs, e.g. information about free capacities and costs. This allows a company to determine in how far cooperation will lead to an increase of its local utility, and thus, to determine its range of negotiation. Another important issue for decision-making is partner modeling; for example, if all the agents had complete knowledge about the decision criteria of all other agents, each agent could locally compute whether there is a solution accepted by all the partners. In the case where all the agents have the same decision criteria, two agents could directly agree on the mean value of the first bid and the first counter-offer, since negotiation will converge towards this value. However, in reality, agents do not have complete knowledge about each other; this makes the bargaining process interesting. In the current system, *partner modeling* is restricted to agents making simple assumptions on the parameters of other agents; future research will aim at enhancing this model. There are several configurable parameters that can be used to vary the decision-making behavior of an agent, e.g.:

ω_d desired profit in per cent for an order.

ω_m minimal profit in per cent accepted by an agent.

Δ function determining the amount to which an agent's next offer is modified given its current offer p; for example, it can be set to a constant k or to $max(k, \frac{(\omega_d - \omega_m) \cdot p}{n})$, where n is a scaling factor determining the speed of convergence; the *max* function guarantees termination of the negotiation independent of the size of n.

σ_c threshold denoting the agent's cooperation sensitivity (i.e. how uneconomic does an order have to be for an agent to offer it to another agent); $\sigma_c \in [0, 1]$

Providing a set of different configurations and strategies for agents is one important functionality of a testbed; however, it has to be complemented by tools for performing, monitoring, and evaluating experiments in order to derive general properties of the features that are producible by the testbed. This is discussed in the following subsection.

EXPERIMENTAL RESULTS

In order to evaluate the influence of the strategies presented so far on the solution of the global scheduling problem, we ran benchmarks developed by (Desrochers, Desrosiers, & Solomon 1992), consisting of 12 test sets à 100 orders describing instances of the *vehicle routing problem with time windows*. This is a static scheduling problem that does not challenge the full expressiveness of MARS:

- There is only one depot from where a set of clients has to be served.
- In each example there are 100 orders for 100 clients where no client occurs twice.
- In the test data, it is assumed that only unloading at the location of the client does need time. There are no time restrictions specified for the process of loading a truck.
- There is only a single company modeled.
- It is assumed that there is always a direct line connection between two cities.

However, despite these restrictions, optimal solutions are known for only a small portion of the examples.

In general, optimal solutions can only be computed if a problem is treated as a closed planning problem. In this case, when the planning processes is started all input data must be known. Throughout the planning process the input data is not allowed to be changed. It is clear that there exist special purpose algorithms which perform more efficient than our system for this specific problem, but these algorithms are not able to deal with the more general problem solved by MARS.

The parameters to be observed are the distance needed by the trucks (the primary quality criterion in the benchmark) and the number of trucks required by the solution (which is an important criterion from an economic point of view). The parameters varied were the number of orders (25, 50, and 100, respectively), the percentage of orders with time constraints (25, 50, 75, and 100 %), the strategy (pure ECNP or ST) and the structure of the input set (random or pre-sorted by the earliest start time). The latter parameter is of special importance: randomness simulates dynamics in a sense that the agent has no knowledge about the temporal ordering of transportation orders. Since no benchmark for a dynamic problem was available, this helps us to evaluate how graceful the performance of our strategies degrades in the dynamic (non-ordered) case with respect to the static (ordered) case.

Figure 4 shows the results from a class of experiments comparing the relative performance of our solution before and after the optimization using ST with the optimal solution for some examples where this solution is known (assuming a sorted input set). It shows that the ECNP solution is between 3% and 74 % worse than the optimal solution and thus is comparable to heuristic OR algorithms; in our experiments ST improves this solution by an average of ca. 12%.

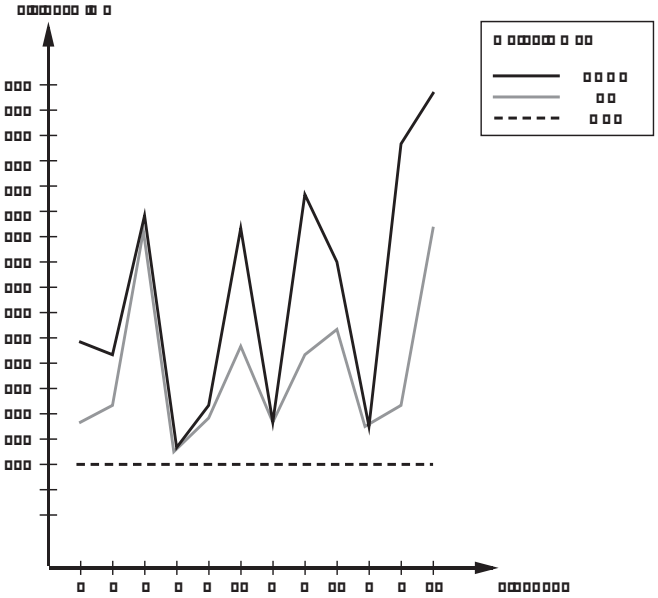

Figure 4: Comparison of ECNP and ST with the Optimal Solution

A second class of experiments compares the performance of ECNP with ST for different problem sizes and different degrees of constrainedness, making a distinction between random and sorted input. The results of these experiments are illustrated by figure 5*a*) to 5*d*).

The main results of these experiments can be summarized as follows: Firstly, ST improves the ECNP solutions in most cases. Secondly, presorting improves the behavior of both algorithms; however, ST yields much better results in the unsorted case than pure ECNP; this implies that ST is a good strategy for dealing with dynamic problems, since the trading process is likely to resolve suboptimal order assignments in the ECNP solutions. On the other hand, ECNP which implements a greedy strategy is very sensitive with respect to the ordering of the transportation orders.

Thirdly, note that the orders drawn along the x-axis are sorted according to how strong they are constrained: 100 % of the orders in the test sets 1, 5, 9 are constrained, 75 % of order sets 2, 6, 10, and so on, where test set 1 denotes the set named $R101$ in the original benchmark data, 2 stands for $R102$ and so on. It is an interesting observation that compared to

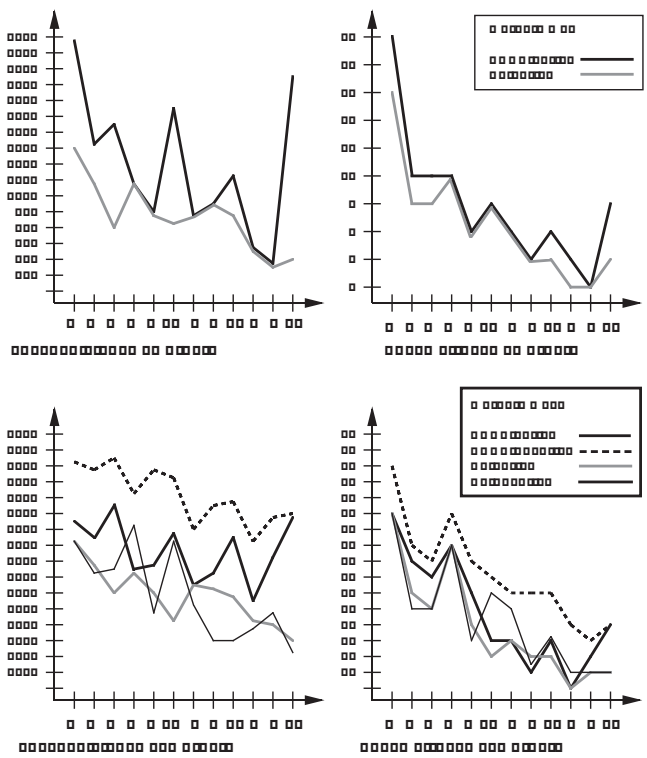

Figure 5: Comparison of ECNP and ST on Random and Sorted Input Sets

ST, ECNP behaves relatively better for strongly constrained orders than for weaker constrained ones: for 25 orders, ST is only 7.2% better than ECNP (in savings of distance on an average) in the 100% constrained case, whereas it saves 22.4% for 25 % constrained order sets. We might speculate that this is a general property of greedy, contract-net-like algorithms; however, this speculation still needs being confirmed by further theoretical and empirical results. For results comparing different horizontal cooperation settings at the SCA layer, we refer to (Fischer *et al.* 1993).

CONCLUSION

In this paper, we have presented the MARS system modeling the transportation domain, a real-world application that has been designed and implemented as a multiagent system. The main contribution of the paper is that it presents a combination of a contract-net-like protocol and an auction procedure which provides promising solutions to difficult scheduling problems.

To evaluate the performance of the MARS system, we have provided a comparison of our approach with Operations Research (OR) solutions, using a set of benchmark examples. The quality of the results achieved by the multiagent approach has been shown to be comparable to those of heuristic OR algorithms. In ad-

dition, the multiagent approach is more flexible: it allows to vary the number of agents on-line and can cope with open, dynamic scheduling problems and with uncertainty in plan execution, whereas the scope of the available Operations Research techniques is limited to static scheduling problems.

The MARS system has been implemented at the DFKI using the AGENDA development environment (Fischer, Müller, & Pischel 1995) for multiagent systems. AGENDA supports the design of agents according to the INTERRAP agent architecture (Müller & Pischel 1994a; 1994b) and provides several desirable functionalities of a simulation system and testbed, such as statistics and visualization tools. The fact that the MARS system can be distributed over a large number of physical machines makes it a powerful scheduling tool, and it turned out to be useful to solve industrial scheduling problems consisting of 800 transportation orders. Currently, a joint project with a shipping company with a daily dispatch volume of about 700 trucks is envisaged, where a tuned version of MARS is to be used as the kernel of an online scheduling assistant.

An important issue for future work are decision-theoretic problems: Using the concepts presented in this paper as a basis for decision-making, the SCAs will start negotiation processes among each others. In this negotiation processes, strategies must be found that guarantee that agents will not benefit e.g. from lying. In (Fischer 1994), it has already been shown that the general results presented in (Zlotkin & Rosenschein 1993) for task-oriented domains are not directly applicable to the transportation domain as presented in this paper. Thus, a separate analysis of this setting will be necessary.

References

Bachem, A.; Hochstättler, W.; and Malich, M. 1993. The Simulated Trading Heuristic for Solving Vehicle Routing Problems. Technical Report 93.139, Mathematisches Institut der Universität zu Köln.

Boddy, M., and Dean, T. L. 1994. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence* 67:245–285.

Davis, R., and Smith, R. G. 1983. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence* 20:63 – 109.

Desrochers, M.; Desrosiers, J.; and Solomon, M. 1992. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* 40(2).

Fischer, K., and O'Hare, G. M. P., eds. 1994. *International Workshop on Decision Theory for DAI Applications*. Amsterdam: ECAI'94.

Fischer, K.; Kuhn, N.; Müller, H. J.; Müller, J. P.; and Pischel, M. 1993. Sophisticated and distributed: The transportation domain. In *Proc. of MAAMAW-93*. Neuchatel, CH: Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World.

Fischer, K.; Müller, J. P.; Pischel, M.; and Jacob, E. 1994. Modeling traffic jams in a logistics simulation environment. In *Proc. of the European Simulation Multiconference (ESM'94)*. Barcelona: Society for Computer Simulation International (SCS).

Fischer, K.; Kuhn, N.; and Müller, J. P. 1994. Distributed, knowledge-based, reactive scheduling in the transportation domain. In *Proc. of the Tenth IEEE Conference on Artificial Intelligence and Applications*.

Fischer, K.; Müller, J. P.; and Pischel, M. 1995. AGenDA: A General Testbed for DAI Applications. In Jennings, N. R., and O'Hare, G. M. P., eds., *Foundations of DAI*. John Wiley & Sons, Inc.

Fischer, K. 1994. Decision theoretic analysis of the transportation domain. In *(Fischer & O'Hare 1994)*.

Henz, M.; Smolka, G.; and Würtz, J. 1993. Oz - a programming language for multi-agent systems. In *Proceedings of the IJCAI*.

Kuhn, N.; Müller, H. J.; and Müller, J. P. 1993. Simulating cooperative transportation companies. In *Proceedings of the European Simulation Multiconference (ESM-93)*. Lyon, France: Society for Computer Simulation.

Lenting, J., and Braspenning, P. 1994. An all-pay auction approach to reallocation. In *Proc. of ECAI-94*, 259–263.

Müller, J. P., and Pischel, M. 1994a. An architecture for dynamically interacting agents. *International Journal of Intelligent and Cooperative Information Systems (IJICIS)* 3(1):25–45.

Müller, J. P., and Pischel, M. 1994b. Integrating agent interaction into a planner-reactor architecture. In Klein, M., ed., *Proc. of the 13th International Workshop on Distributed Artificial Intelligence*.

Russell, S. J., and Zilberstein, S. 1993. Anytime sensing, planning, and action: A practical model for robot control. In *Proc. of IJCAI'93*, 1402–1407. Chambery, F: Morgan Kaufmann Publishers Inc., San Mateo, CA, USA.

Sandholm, T. 1993. An implementation of the contract net protocol based on marginal cost calculations. In *Proc. of the 12th International Workshop on Distributed Artificial Intelligence*, 295–308.

Schulte, C.; Smolka, G.; and Würtz, J. 1994. Encapsulated search and constraint programming in Oz. In *Second Workshop on Principles and Practice of Constraint Programming*, 116–129.

Wellman, M. 1992. A general-equilibrium approach to distributed transportation planning. In *Proc. of AAAI-92*, 282–290.

Zlotkin, G., and Rosenschein, J. S. 1993. A domain theory for task-oriented negotiation. In *Proc. of the 13th International Joint Conference on Artificial Intelligence*, volume 1.