A Simulation Approach Based on Negotiation and Cooperation Between Agents: A Case Study

Klaus Fischer, Brahim Chaib-draa, *Member, IEEE,* Jörg P. Müller, Marcus Pischel, and Christian Gerber

***Abstract*—This paper begins by presenting AGENDA, a simulation tool developed for the simulation and design of applications involving interacting entities. This testbed consists of two different levels, the architecture level and the system-development level. The architecture level describes a methodology for designing software agents by providing several important functionalities an agent should have. On the other hand, the system-development level provides the basic knowledge-representation formalism, general inference mechanisms, and simulation tool-box supporting visualization and monitoring of agents.**

Following this, the applicability of AGENDA to the transportation domain is presented in detail. The main challenge of AGENDA in the context of this domain has been to provide different cooperation-scalable methods based on negotiation, leading to different scheduling mechanisms, and to experimentally evaluate these mechanisms. This evaluation shows that: 1) AGENDA is suitable for the realistic application of the transportation domain; 2) mechanisms used for the vertical negotiation (between trucks considered as agents) and for the horizontal negotiation (between companies considered as agents) are applicable for the real-world application of the transportation domain. Finally, a complete study of the scalability of the simulation tool and the algorithms used for the negotiation is presented. This study, with the evaluation of the different mechanisms, can help designers of transportation companies, particularly in the case of large companies.

I. INTRODUCTION

AN exciting approach to using AI and simulation techniques together arises when we want to simulate interacting entities. For this type of simulation, we want to simulate the interacting entities as close to the the realistic environment as possible. Using deterministic and other crisp mathematical models for this type of interaction can be a hard choice. Indeed, interacting entities require properties such as autonomy, reactivity, rationality, reasoning on others, ability to plan and take the initiative by pursuing goals, etc., which are very difficult to deal with using classical mathematical methods. Furthermore, the agent behavior is not generally deterministic and is based on knowledge that could be uncertain, incomplete, and complex, and for which pragmatic methods are generally better. In addition, classical mathematical methods are generally more quantitative than qualitative, which could be another limitation. Finally, most of the mechanisms used for the negotiation and cooperation between agents are issued from pragmatic methods and generally are studied in the field of multi-agent systems (MAS's) [7], [23]–[26], [32]. For these reasons, we think that the appropriate way to simulate interacting entities consists of using simulation techniques and MAS methodologies.

Multi-agent systems deal with coordinating intelligent behavior among a collection of autonomous agents. Emphasis is placed on how these agents coordinate their knowledge, goals, skill, and plans jointly to take action or to solve problems. Apart from these issues, they also have to reason with regard to the process of interaction coordination itself. The MAS technologies have been a very active area since 1990. Their rapid development can be viewed in the light of the following considerations. The increasing complexity of organization and computer-controlled technical processes and systems makes it impossible to design them as monolithic entities and to maintain and monitor them by centralistic control systems. Examples are:

1) the open systems, which change over time and are always subject to new information from outside themselves, causing unanticipated events (banking systems, airline reservations, internet, business systems, etc.);
2) the optimization of the flow of work and information through cooperating companies;
3) international air-traffic control;
4) the coordination of logistic processes in shipping companies that schedule hundreds of transportation vehicles and that participate in multimodal transport;
5) the use of flexible transport systems (FTS's) in industrial manufacturing and assembly;
6) the design and the operation of traffic-guidance and control systems.

Most of these applications are based on the inherent distribution of competence, control, and information, as well as the complexity of the theoretical problems. It is therefore clear that this type of complex application needs a novel simulation methodology based on a combination of MAS methodologies with traditional simulation techniques.

In this paper, we propose an approach to this novel simulation methodology through a scalable application: the transportation domain. Firstly, we describe AGENDA, a testbed developed in the multi-agent research group at DFKI for the simulation and design of some specific distributed applications. This testbed consists of two different levels: the architectural

Manuscript received January 29, 1998; revised February 18, 1999. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and by Siemens AG, Germany.

K. Fischer and C. Gerber are with DFKI GmbH, the German Research Center for Artificial Intelligence, Saarbrücken, Germany (e-mail: fischer@dfki.de).

B. Chaib-draa is with the Computer Science Department, Laval University, Ste-Foy, P.Q., Canada (e-mail: chaib@ift.ulaval.ca).

J. P. Müller is with Zuno Ltd., International House, London, U.K. (e-mail: jpm@ dlib.com).

M. Pischel is a private consultant.

Publisher Item Identifier S 1094-6977(99)08202-4.

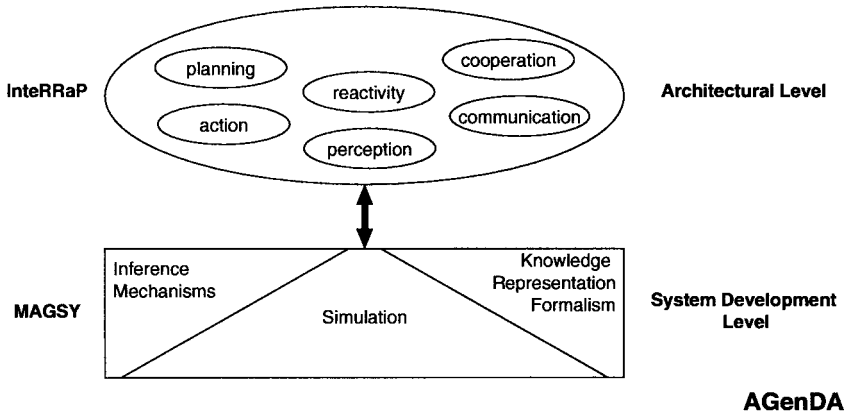


Fig. 1. AGENDA testbed.

level and the system-development level. The architecture level describes a methodology for designing agents by providing several important functionalities an agent should have. The system-development level provides the basic knowledge-representation formalism, general inference mechanisms, and the simulation tool box supporting visualization and monitoring of agents.

The transportation application has been implemented using the AGENDA framework. For this application, the main challenge for AGENDA was to provide different cooperation methods based on negotiation (leading to different scheduling mechanisms) and to experimentally evaluate these mechanisms. This evaluation shows that: 1) AGENDA is suitable to the realistic application of the transportation domain; 2) mechanisms used for the vertical negotiation (between trucks considered as agents) and for the horizontal negotiation (between companies considered as agents) are applicable for the real-world application of transportation domain.

Finally, a complete study of the scalability of the simulation tool and the algorithms used for the negotiation is presented. This study, with the evaluation of the different mechanisms, constitutes an efficient tool for designers of the transportation domains, particularly for large companies.

This paper is organized as follows. Section II describes AGENDA in detail. Section III presents Modeling Autonomous CoopeRating Shipping Companies (MARS), the system that simulates the transportation application using AGENDA. Section IV describes the multi-agent approach underlying the MARS system, particularly the negotiation and cooperation aspects. Section V explores traffic congestion as a method for dealing with problems occuring due to unforeseen events happening during plan execution. Section VI presents and analyzes the results of strategies and cooperation presented in this paper. Finally, Section VII discusses the scalability of the testbed, as well as the mechanisms proposed for the negotiation and cooperation.

II. AGENDA—A GENERAL TESTBED FOR MULTIAGENT APPLICATIONS

Before describing the MARS system, we would like to briefly sketch the framework underlying our multiagent applications.

A. AGENDA Testbed

The AGENDA testbed [15] serves as the development platform for our applications. The testbed consists of two different levels. The architectural level describes a methodology for designing agents in the sense that it provides several important functionalities an agent should have. It thus supports a general template for agents, which must be filled by the designer of a multiagent system with the domain-specific instantiation. The system-development level provides the basic knowledge-representation formalism, the general inference mechanisms (such as forward and backward reasoning) used by the decision-making modules of the architectural layer, and the simulation toolbox supporting visualization and monitoring of agents and the gathering of performance statistics (see [22] for a well-written discussion of properties, problems, benefits of, and examples for testbeds). The relationship between the two testbed levels is illustrated in Fig. 1.

The architectural level in the AGENDA testbed is provided by the integration of reactive behavior and rational planning (INTERRAP) agent architecture [34]. It defines the control within an agent as a hierarchical process, mapping different classes of situations to different reactive, deliberative, or cooperative-execution mechanisms. The system-development level is covered by the MAGSY system. MAGSY provides a frame-based knowledge-representation formalism and a set of general-purpose inference mechanisms. Moreover, it provides tools supporting the construction, visualization, evaluation, and debugging of multi-agent scenarios, including the lower layers of communication, on top of which more complex protocols such as the contract net or bargaining protocols can be defined.

Two applications have been implemented using the AGENDA framework. The first system is FORKS an interacting robots application. FORKS describes an automated loading dock, where forklift robots load and unload trucks while avoiding potential conflicts and resolving existing conflicts and exploiting possibilities to collaborate. The main requirement imposed on the testbed by this application was that it had to support reactivity and deliberation in the decision-making of an individual agent as well as perception and manipulation of the physical world. The second system is MARS, which is the topic of this article. In the case of the

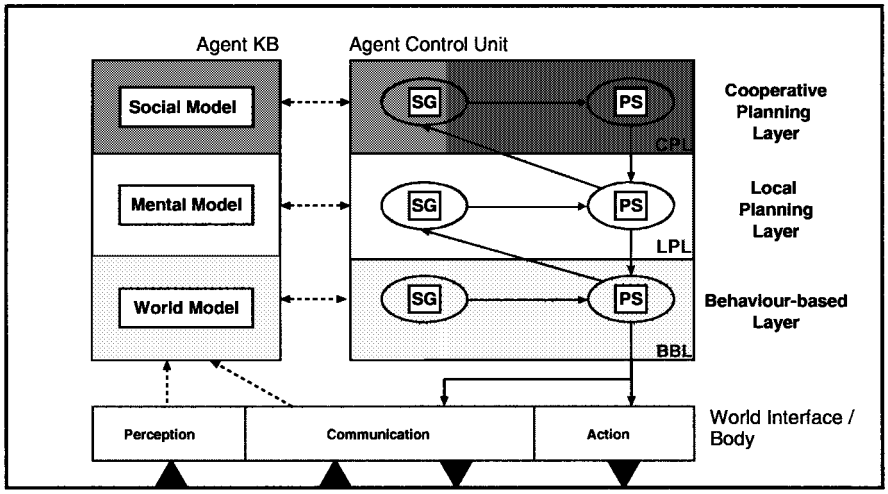


Fig. 2. INTERRAP Agent architecture.

MARS system, the main challenge for our testbed was to provide different cooperation methods based on negotiation. This led to different scheduling mechanisms, which were experimetally evaluated.

An autonomous agent acting and interacting in a dynamic environment must have certain properties, which should be reflected in its underlying design architecture. Firstly, agents are to behave in a situated manner (i.e., they must perceive unexpected events and react appropriately to them [3]). Secondly, they should not simply act in response to their environment, but should be able to exhibit opportunistic, goal-directed behavior and take the initiative. Thirdly, they should be able to solve their tasks efficiently and must often satisfy real-time constraints. This requires access to a set of "hard-wired" procedures [18] with guaranteed execution properties. Fourthly, they are to cope with the presence of other agents. Whereas certain types of interactions often can be performed by employing local mechanisms (i.e., obstacle or collision avoidance in a robot scenario [30], [34]), others (i.e., collaboration) require the adoption of joint goals, the generation and execution of joint plans, the exchange of relevant information (i.e., about goals and plans [28]), and the explicit representation of models of other agents in terms of beliefs, goals, plans, and intentions [17]. Finally, agents should be able to exhibit an adaptive behavior. To achieve this, they should learn in order to improve their performance and to survive even if the environment changes. These requirements have led to the development of the agent architecture INTERRAP, a layered architecture describing the individual agent.

B. INTERRAP Agent Architecture

The development of INTERRAP has been guided by the following general design decisions.

Layered Control: An agent is described by different levels of abstraction, and of representational and inferential complexities.

Layered Knowledge Base: The beliefs of an agent are stored in a hierarchical knowledge base. This allows us to restrict the amount and the representation of information available to the lower control layers.

Bottom-Up Activation: Control is shifted bottom-up. Layer i gains control only if layer $i-1$ is not competent to deal with the situation.

Top-Down Execution: Each layer uses operational primitives defined at the next lower layer to achieve its goals.

Fig. 2 shows the components of the INTERRAP agent model and their interplay. It consists of three modules: a world interface (WIF), a knowledge base (KB), and a control unit (CU). The agent's WIF provides the agent's sensoric, communicative, and actoric links to its environment. The KB and the CU are structured in three layers. Control layers are the behavior-based layer (BBL), the local planning layer (LPL), and the the cooperative planning layer (CPL). Each control layer consists of two processes called situation recognition and goal activation (SG) and planning, scheduling, and execution (PS). The knowledge base is partitioned accordingly into a world model (WM), a mental model (MM), and social model (SM).

Notice that the BBL implements the reactive behavior and the procedural knowledge of the agent. Basic building blocks of the BBL are patterns of behavior (PoB), which can be divided in two groups: reactor patterns and procedure patterns. Reactor patterns specify hard-wired, condition-action pairs occurring in reactive behavior, whereas procedure patterns specify procedural knowledge (i.e., compiled plans), and they are activated by the plan-based component in order to perform routine tasks.

The LPL contains a planning mechanism that is able to devise local single-agent plans. Depending on the requirements imposed by the application, the LPL may be initiated with a suitable planning formalism. However, the interface definition between BBL and LPL requires that the latter can activate PoB's, which are primitive actions from the perspective of the planner. The difference to classical AI planning systems is that PoB's may be rather complex procedures [11] incorporating a certain degree of execution intelligence (e.g., for dealing with different types of failures without explicit replanning). Finally, the CPL contains a mechanism for devising joint plans [28], [33]. It has access to protocols and a multiagent planning mechanism that can access knowledge about other agents and

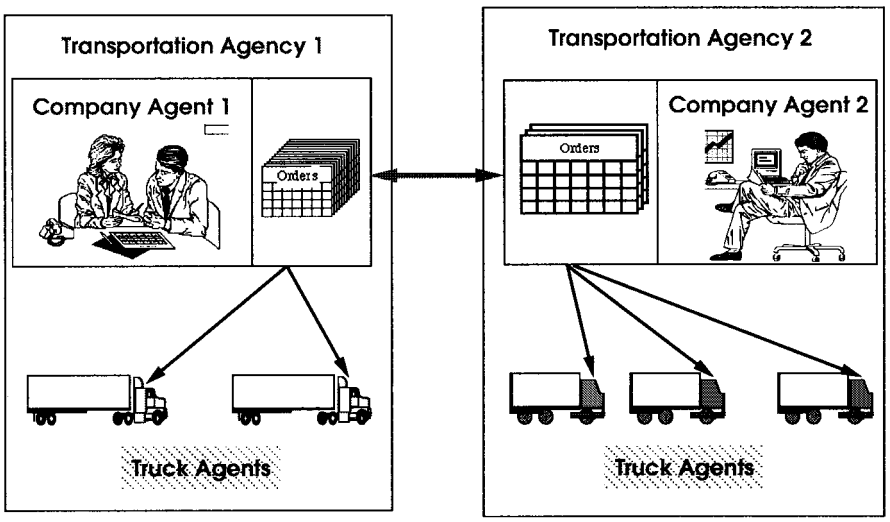


Fig. 3. MARS: The domain of application.

about communication strategies. CPL, LPL, and BBL establish the control of the agent. Their interaction (see [35]) defines the agent's overall behavior.

INTERRAP is a vertically layered architecture. Control is triggered by the BBL, which has immediate access to the agent's sensory system, and is shifted upward incrementally until a "competent" layer for execution has been found. From these, activity spreads back downward to the BBL, which is the only layer with direct access to the actoric functions defined in the agent's world interface. This is an important difference to the other layered architectures such as those by Ferguson [10] and Kaebling [27], in which there is concurrent access both to sensory input and actoric output, and conflicts among the layers must be solved by applying appropriate global filtering and suppression mechanisms to ensure that a specific layer only sees those parts of the input that are relevant to it. This also suppresses harmful interactions among the decisions of different layers.

Each control layer is allowed to access a specific portion of the agent KB. This access is organized incrementally in that each control layer may use information stored in its corresponding KB layer and in lower KB layers but is not allowed to access information stored in higher layers. Thus, the BBL has access only to the WM part of the KB containing the agent's object-level beliefs about the world. The LPL may access the MM, in which information relevant for planning, such as goal-specific information, plan libraries, or meta-descriptions of patterns of behavior, is stored. Finally, the CPL may reason about the whole knowledge base, including the agent's SM, which contains descriptions of negotiations protocols, joint plan libraries, and information referring to the goals of other agents.

III. SIMULATING THE TRANSPORTATION DOMAIN WITH AGENDA: THE MARS SYSTEM

A. Why the Transportation Domain?

The process of economic integration in the world and the abolishment of federal regulations on freight transportation services in Europe has led to a dramatically increased competition in the logistics business. Thus, new technologies are required to keep with the complexity and the dynamics of the domain.

The MARS simulation testbed [29] presented in this paper constitutes a multiagent approach to these problems. A scenario of geographically distributed transportation companies is described. The companies have to carry out transportation orders, which arrive dynamically. For this purpose, they have a set of trucks at their disposal. The global behavior of the system is evaluated as follows. The quality measure is the cost of carrying out the orders. We address this problem with an innovative approach in the sense that the companies themselves do not have facilities for scheduling orders. It is the trucks that maintain local plans, and the actual solution to the global-order scheduling problem emerges from the local decision making of the agents. Thus, one very complex plan is replaced by several smaller and simpler plans, allowing one to react quickly and without global replanning to unforeseen events such as traffic jams or new transportation orders.

B. MARS System

The application domain for the MARS system is the planning and scheduling of transportation orders as performed by dispatchers in shipping companies. Many of the problems that must be solved in this area, such as the traveling salesman and related scheduling problems, are known to be $\mathcal{NP}$-hard. Moreover, the domain is highly dynamic, and decisions have to be made under a high degree of uncertainty and incompleteness.

Cooperation and coordination are two very important processes that may help overcome the problems sketched previously. Indeed, they are of increasing importance even in the highly competitive transportation business of today. Using the MARS system, several patterns of cooperation, such as the announcement of unbooked legs, order brokering, and different strategies for information exchanges, have been experimentally evaluated (see [6], [29]).

Corresponding to the physical entities in the domain are two basic types of agents in MARS: transportation companies and trucks, and they are designed according to the INTERRAP agent architecture. Companies can communicate with their trucks and among each other. The user may dynamically dedicate transportation orders to specific companies and groups of companies. Looking upon trucks as agents allows us to delegate problem-solving skills to them (such as route-planning and local plan optimization).

Shipping company agents (SCA's) must allocate orders to their truck agents (TA's) while trying to satisfy the constraints provided by the user as well as local optimality criteria (costs). An SCA also may decide to cooperate with another company instead of having an order executed by his or her own trucks. The functionality of an SCA is modeled in the INTERRAP architecture as follows.

1) The BBL of a company is rather simple. It provides patterns of behavior for recognizing that a transportation order has been received and for activating the communication primitives defined by the communication protocols (see Section IV).
2) Since an SCA does no scheduling on its own, the function of the LPL reduces to an algorithm that synthesizes a plan for an order based on the partial bids received by the trucks.
3) The CPL contains the main part of the functionality of the SCA. The protocols for task allocation and negotiation (see Section IV) are represented as meta joint plans in a plan library [33] and are executed by a plan interpreter.

TA's are associated with particular shipping companies, from which they receive orders of the form "Load amount s of good g_1 at location l_1 and transport it to location l_2 while satisfying time constraints $\{c_{t_1}, \ldots, c_{t_n}\}$." A TA is modeled as an INTERRAP agent as follows.

1) The BBL of a TA contains PoB's for checking the existence of new orders, for deciding when to begin the execution of a plan step based on the temporal information kept in the plan, for performing the actual plan execution, and for recognizing traffic jams based on data received by the travel-information service (see [16]). The primitive actions the TA is able to perform are driving, loading, unloading, and communicating with its company.
2) A truck's LPL contains the local-planning algorithm, which is a polynomial heuristic-insertion algorithm. Additionally, in order to compute a bit for an order, the truck has to evaluate the cost of its plan [13].
3) The CPL of a truck contains the definition of the protocols used for communication with its SCA (see Section IV).

Interaction of the agents within one shipping company (called vertical cooperation) is totally cooperative. This means that a specific TA will accept deals (i.e., results of negotiation processes) with his or her SCA even if they are not locally profitable. We call such a setting an instance of a "cooperative task-oriented domain" (cf. [14]). In the cooperation between SCA's, 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 interested purely in the quality of the overall schedule that is emerging from the local problem solving done in the SCA's and TA's.

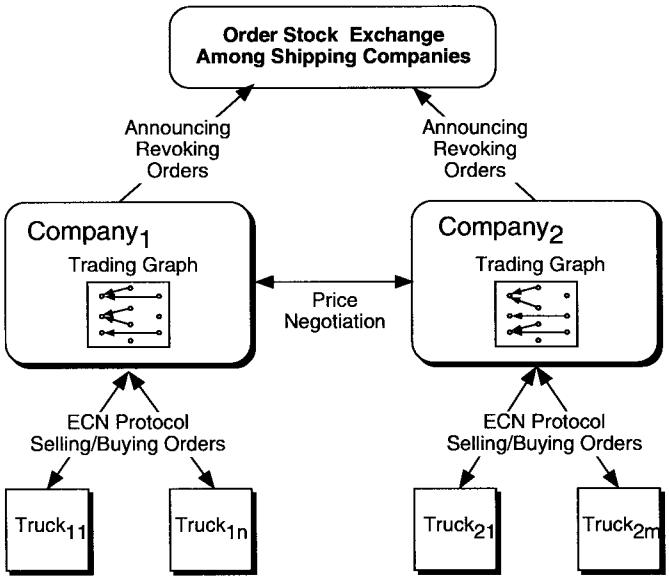


Fig. 4. Hierarchical organization of the agents in MARS.

On the other hand, if a competitive task-oriented domain among the SCA's is assumed, it is clear that the overall schedule that is computed will be far from optimal. In this setting, we investigate how a single SCA can maximize profits and how he or she can avoid being tricked by other agents. In this paper, we will concentrate on the cooperative setting and refer to [14] for the discussion of the latter setting.

IV. COOPERATION AND NEGOTIATION IN THE MARS SYSTEM

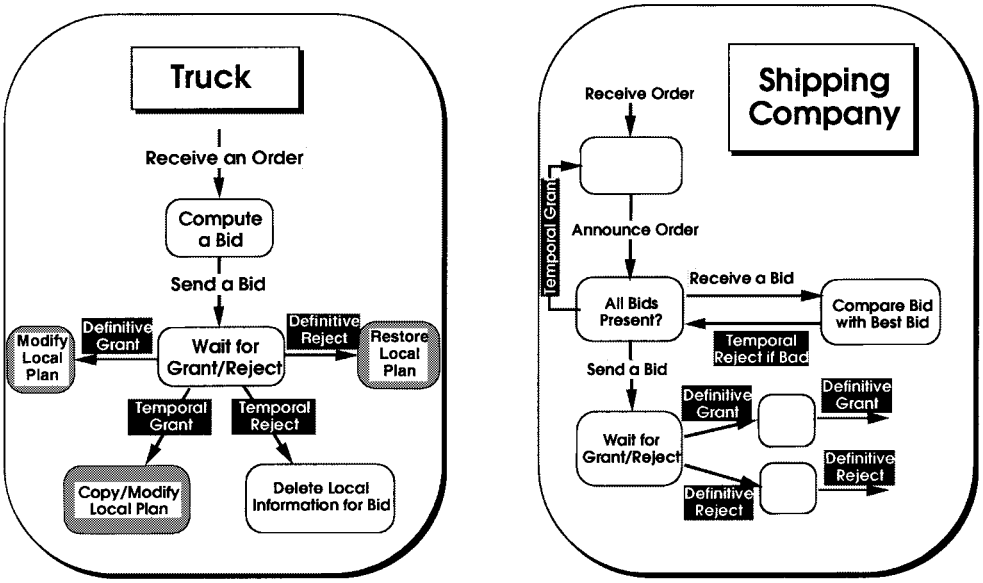
In this section, we describe the multi-agent approach underlying the MARS system. Starting from the standard contract net protocol (Section IV-A), we define a framework that provides more powerful tools for task decomposition and task allocation (Section IV-B). A model for peer-to-peer negotiation among different SCA's is outlined in Section IV-C. Finally, a sophisticated procedure based on the simulated trading is described in detail in Section IV-D.

A. Vertical Cooperation: Task Decomposition and Task Allocation

When an order o is announced to an SCA by a customer (which can also be another SCA), this SCA has to compute a bid for executing the order. In order to determine the costs, the order was forwarded to the TA's. Each TA a computes a bid

$$(a, cost(T_a \oplus o) - cost(T_a), w)$$

where T_a is the current tour of a, and w is the amount of the order a is able to transport. $cost(T_a \oplus o)$ denotes the additional costs for a when executing o given T_a. Let $\mathcal{O}^a = \{o_1^a, \ldots, o_n^a\}, n \in I\!N$ be the current set of orders for a.



(a) (b)

Fig. 5. ECNP from the point of view of (a) a manager and (b) a bidder.

A constraint net is derived from the information specified with the orders. Each solution to this constraint-solving problem is a valid tour that fulfills all constraints specified by $\mathcal{O}^a$. Then, a tries to find the best tour for $\mathcal{O}^a$, using a constraint-solving and constraint-optimization procedure. Our implementation is based on the Oz [38] system, which was developed at DFKI, and which provides powerful mechanisms for optimization procedures in case the search space is defined by a constraint net.

For each order o announced by an SCA to the TA's, this SCA receives a set of bids

$$\mathcal{B} = \{(a_1, c_1, w_1), \ldots, (a_n, c_n, w_n)\}, \quad n \in I\!N$$

where c_i specifies the costs truck a_i will produce when executing amount w_i of order o $1 \leq i \leq n$. The SCA selects

$$(a_{\min}, c_{\min}, w_{\min}) \in \mathcal{B} \quad \text{with } \forall (a, c, w) \in \mathcal{B} : \frac{c_{\min}}{w_{\min}} \leq \frac{c}{w}$$

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

The procedure described so far is the well known contract net protocol (CNP) [8]. Because the CNP provides time-out mechanisms, it is easy to turn this communication protocol into an "anytime algorithm" [5], [37] (i.e., the system will produce a solution, if there is one, within a specified time t_0). The quality of the solution may be increased if more time for computation is available (see also Section IV-B).

B. Extended-Contract Net Protocol

The pure contract net protocol as described so far runs into problems if the tasks exceed the capacity of a single truck, i.e.,

$$a_{\min} < \textit{amount-to-transport}(o).$$

In this case, the manager of the task (the SCA) has to solve a knapsack problem, which in general is $\mathcal{NP}$-hard. To overcome this problem, we decentralized task decomposition by developing and implementing an extension of the CNP, called the extended-contract net protocol (ECNP). ECNP is available as a standard protocol in MARS. In ECNP, the two speech acts "grant" and "reject" are replaced by four new speech acts: "temporal grant," "temporal reject," "definitive grant," and "definitive reject." The ECNP is a natural, straightforward solution of the task-decomposition problem.

A flow-chart representation is used to represent the negotiation protocols provided by the MARS testbed. The protocols describe the roles of the individual agents in the negotiation process. Fig. 5 shows the flow charts for the ECNP protocol, from (a) the manager's and (b) the bidders' point of view. The main difference to the CNP is that now the bidders (the TA's) are allowed to bid for only parts of an order.

In the ECNP, the manager (SCA) announces an order o to the TA's. Bids for the order are then received and the best one is selected 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 cover the total amount of the original order o. From this set of bids, the SCA computes a bid that is passed to the customer. Based on the answer of the customer, the SCA sends a definitive grant (or definitive reject, respectively) to all TA's who received temporal grants before. It is possible to prove that in general all but the last bid selected are locally optimal choices for the SCA [13].

When a TA receives a temporal grant for the first time, he or she must store a copy of the local situation, (i.e., the currently valid plan) because the situation must be restored in the event a definitive reject is received. All subsequent temporal grants and temporal rejects are handled like the grants and rejects in

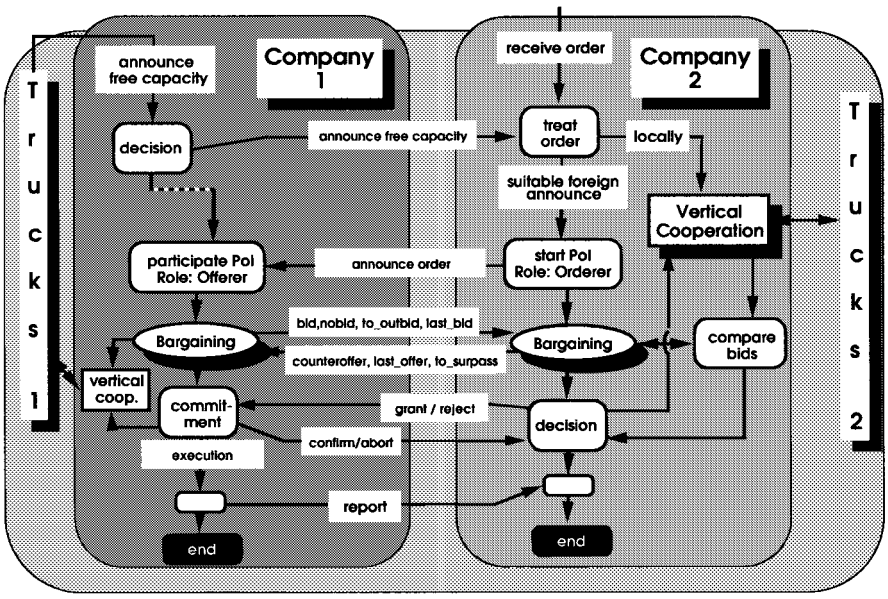


Fig. 6. Bargaining protocol for horizontal cooperation.

the pure CNP. If a TA is sent a definitive grant for an order, he or she removes the copy created above and switches to the new plan. If a definitive reject is received, the situation is restored before the first temporal grant.

In our framework, the ECNP is used to obtain a fairly good initial solution (see Section V for a quantification of this claim) for the contract net protocol. Having a quick algorithm to determine a rather good upper bound for the costs induced by an order is important for the agent, since it provides a basis for its future decisions. However, because the situation changes if new orders arrive, and because the TA's will stick to decisions made in the past, the solution found is not even guaranteed to be Pareto-optimal [39].

There are different ways to optimize the ECNP solution. Currently, the simulated-trading algorithm, described in Section IV-D as a solution to the dynamic replanning problem, is also used to optimize the order exchange among trucks. By coupling ECNP and simulated trading, we obtain an anytime algorithm [4] $\mathcal{A}_{t_0}$ with a lowest-time bound t_0 defined by the runtime of the ECNP process. In other words, $\mathcal{A}_{t_0}$ is an interruptible anytime algorithm [36] for each $t \geq t_0$. Since the individual trucks employ polynomial-insertion algorithms for computing their bids within the ECNP, the time bound t_0 for the ECNP is polynomial.

C. Horizontal Cooperation: Negotiation

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 SCA's (so-called "horizontal cooperation") may be a beneficial operation. For example, companies may exchange orders and information about free-loading capacities, and they may apply for orders offered by other companies. However, in contrast to the coordination between SCA and trucks, cooperation between companies is a peer-to-peer process in which a solution (e.g., a price to be paid for an offer) can only be found if all the participants agree. In addition, the conditions of the solution must be negotiated among the companies. It is this peer-to-peer negotiation that we call horizontal cooperation and whose implementation is described in the sequel.

1) Negotiation Protocol: AGENDA supports the modeling of horizontal cooperation by providing a parameterized bargaining protocol that can be initiated with the specific conditions of a negotiation. Fig. 6 illustrates the protocol by means of a flow chart.

It shows both the types of messages exchanged between the companies as well as the connection between local reasoning within a company (represented by local decision nodes and by connection to the vertical-cooperation protocol with its trucks) and cooperative reasoning in the course of the negotiation. A company (company 1 or c_1 in the example) may decide to announce free transport capacity to another company (let us say company 2 or c_2). This decision can be made based on information about free capacities c_1 has received by its trucks. Based on its local state, c_2 decides whether it wants to take up the announcement, and if so, sends an order to c_1. This instantiates a bargaining protocol in which c_1 takes the role of the offering agent, and c_2 takes the role of the orderer. c_1 will start by sending an offer (bid) to c_2, c_2 will decide whether to accept, reject, or to modify the bid by making a counteroffer. The bargaining process continues either until both parties have agreed on a common solution or until it becomes clear that no compromise can be found. The other communication acts shown in Fig. 6 such as "last bid," "to outbid," and "to surpass" are special-purpose features enabling an auction-like negotiation between more than two agents.

2) Decision Making: The decision making of the companies during the negotiation process is based on information they obtain by their trucks (e.g., information about free capacities and costs). Whereas the costs of an order were the decision criterion for the TA's, the SCA's make their decisions based on the utility of an order, which is computed as the

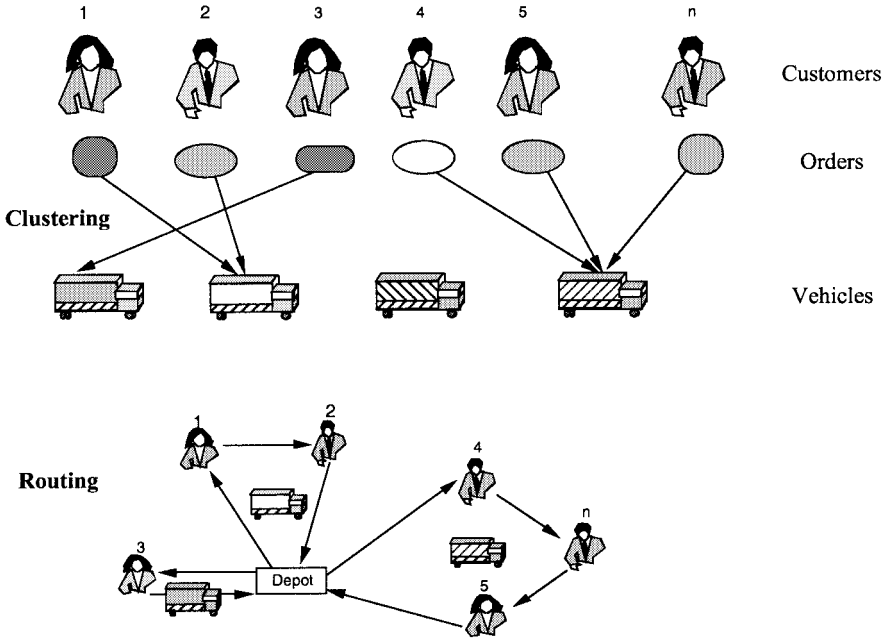


Fig. 7. Standard vehicle-routing problem.

difference between the worth (which is obtained from the customer or from other companies) and the costs. Based on this information, a company determines how far cooperation will lead to an increase of its local utility, and thus determines 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 counteroffer, since negotiation is to converge at 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.

- ω_d Desired profit in percent for an order;
- ω_m Minimal profit in percent accepted by an agent;
- Δ Function determining the amount to which an agent's next offer is modified given its current offer p, which can be set to either constant k or $\max(k, \frac{(\omega_d - \omega_m) \cdot p}{n})$. 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 (which is a measure of how uneconomic an order has to be for an agent to be offered to another agent) $\sigma_c \in [0, 1]$.

So far, we have described methods for task decomposition and task allocation implemented in the MARS system, which allow us to deal with dynamics and uncertainty in planning. In the following section, we will extend this framework to mechanisms allowing us to deal with dynamics in plan execution, too.

D. Simulated Trading

The ST [2] procedure presented in this section can be used for two different purposes.

Dynamic replanning: If a TA realizes that he or she cannot satisfy the time constraint of an order because of an unforeseen traffic jam, he can initiate an ST process leading to an order reallocation satisfying the time constraints.

Iterative optimization: Starting from the initial ECNP solution (see Section IV-B), ST may be initiated to yield a better order allocation. The experimental results in Section V demonstrate the usefulness of ST as an optimization technique.

In the following, the principle of ST and its application in the MARS system are explained.

1) Principles of ST: In [1], Bachem, Hochstättler, and Malich present a parallel improvement heuristic for solving vehicle-routing problems with side constraints. Their approach deals with the problem that n customers order different amounts of goods, which are located at a central depot. The task of the dispatcher is to cluster the orders and to attach the different clusters to trucks, which then in turn determine a tour to deliver the cluster allocated to them.

The solution to this problem is constructed using the ST procedure. It starts with a set of feasible tours $T_1, \ldots, T_{t_0}$, which may be obtained by a conventional heuristic for example, which is applicable to this domain. The tours are represented as an ordered list of custumers who must be visited. Parallelism is achieved through the data of each tour T_i being assigned to a single processor i (the tour manager) of a parallel multiple instruction multiple data (MIMD) computer. To guide the improvement of the initial solution, an additional processor (the stock manager) is added to the system. The task of the stock manager is to coordinate the exchange of customers orders between the different processors (Fig. 7). To

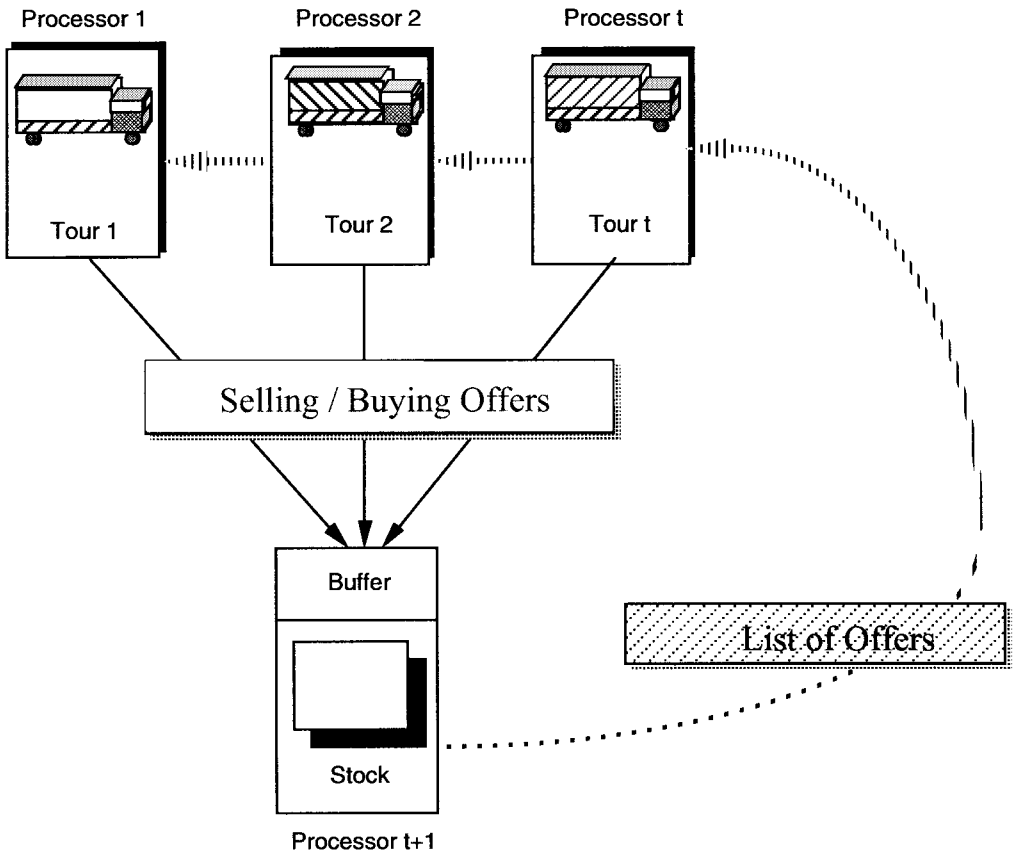


Fig. 8. Stock exchange for orders in the ST procedure.

do this, it collects offers for buying and selling orders coming from the processors in the system.

A price system is introduced, providing a quality criterion for order exchanges to the stock manager. If processor p sells an order i (i.e., an order from the depot to customer i), its cost should decrease. This saving of costs is associated with the price *Pr* to i, where

$$Pr \stackrel{\text{def}}{=} cost(T_p) - cost(T_p')$$
$$T_p' \stackrel{\text{def}}{=} T_p \ominus \{i\}.$$

Here, the term $T_p \ominus \{i\}$ denotes the tour that evolves from T_p if customer i (or order i, respectively) is deleted from processor p's tour list. Accordingly, the price *Pr* for processor p buying a customer i is computed as the difference of costs for the old tour T_p and the costs for the new tour $T_p \oplus \{i\}$, which evolves from the insertion of customer i in T_p, i.e.,

$$Pr \stackrel{\text{def}}{=} cost(T_p') - cost(T_p)$$
$$T_p' \stackrel{\text{def}}{=} T_p \oplus \{i\}.$$

The exchange of orders is synchronized by the stock manager according to levels of exchange situations. At each level, it asks each processor for a selling or buying order. Having done this, it updates a list of the offers and sends it to all tour managers. Each offer is associated with a quintuple (processor, level, selling or buying, customer, price).

The stock manager maintains a data structure, called trading graph, whose nodes are the selling and buying offers of the processors (Fig. 8). Furthermore, there exists an edge between vertices $v_i = (\text{processor}_1, l_i, \text{Selling}, c_i, Pr_i)$ and $v_j = (\text{processor}_2, l_j, \text{Buying}, c_i, Pr_j)$ if processor_2 wants to buy customer c_i from processor_1. l_i and l_j indicate the levels of negotiation. The edge is weighted (or labeled) by the difference of the prices $Pr_j - Pr_i$, leading to the global saving of an exchange of the order between these tours. In this graph, the stock manager now looks for a so-called trading matching (i.e., a subset M of the nodes specifying admissible exchanges of orders between tours).

One problem here is that, in offering a selling of an order, a processor believes that this order eventually will be bought by another processor, and so it will base its future price calculations on its reduced tour. Thus, an admissible exchange must ensue, in which in each node $v_i \in M$, all nodes of the processors that are selling or buying v_i and that have a smaller level than v_i also must be in M.

The gain of the matching is obtained by summing up the weights of the edges between nodes in M. A trading matching is then defined to be an admissible matching whose gain is positive.

2) Adapting ST for the MARS System: The main idea is to let the SCA simulate a stock exchange in which the TA can offer the current orders at some specific "saving price" and make buy orders at an "insert price." While getting sell and buy offers form TA's, the SCA maintains the trading graph

and tries to find an order exchange that optimizes the global solution. 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 turns out to be feasible in practice, since the trading graph normally 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.

Important for the ST procedure is 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 matching was found that led to a new optimum). Thus, reactivity is ensured. 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 TA's plan used for the ST process.

3) Using ST for Dynamic Replanning: An important feature of the MARS system is that TA's 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 the local plan, it is possible for him or her to participate in the ECNP protocol. However, in the ST process, the TA is not allowed to sell orders he or she has already loaded.

A problem in plan execution is that planning is done on statistical data that may be too optimistic. For instance, when the plan is actually executed, the TA may get stuck in a traffic jam [16]. Therefore, replanning might be necessary, because the TA may run into problems with respect to the time constraints that are specified with the orders. Fortunately, this situation can be handled nicely 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 where orders will be delivered. 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 that 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 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 must report the problem and return the respective orders to the SCA. In this case, the SCA 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 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, who then has to find a solution by contacting the client and trying to relax the constraints of the order. If TA's run into this situation, they are paralyzed in the sense that they cannot participate in the ECNP or in the simulated trading process until they receive instructions from their SCA. Fortunately, the ECNP and the simulated trading procedure can deal with this situation, because they do not require participation of all TA's.

V. Transportation-Domain Simulation: Execution Analysis

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 [9], consisting of 12 test sets of 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, and no client occurs twice.
- In the test data, it is assumed that only unloading at the location of the client needs 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 process 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 that perform more efficiently 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

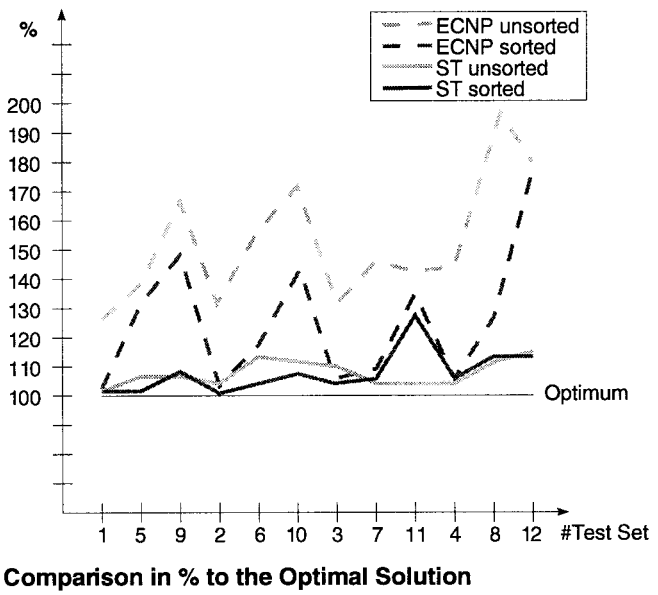


Fig. 9. Comparison of ECNP and ST with the optimal solution.

constraints (25%, 50%, 75%, and 100%), the strategy (pure ECNP or ST), and the structure of the input set (random or presorted by the duedate of the orders). The latter parameter is of special importance. Randomness simulates dynamics in the sense that the agent has no knowledge of the temporal ordering of transportation orders. Since no benchmark for a dynamic problem was available, this helps us to evaluate how gracefully the performance of our strategies degrades in the dynamic (nonordered) case with respect to the static (ordered) case.

Fig. 9 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 the static case). It shows that the ECNP solution is between 3% and 96% 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. 27% in the nonordered case and ca. 11% in the ordered case. 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 Fig. 10(a)–(d).

The main results of these experiments can be summarized as follows. ST improves the ECNP solutions in most cases, and 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 that implements a greedy strategy is very sensitive with respect to the ordering of the transportation orders.

Finally, note that the orders drawn along the x-axis are sorted according to how strongly they are constrained. 100% of the orders in test sets 1, 5, and 9 are constrained, as are 75% of order sets 2, 6, and 10, and so on. In this case, 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 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 have similar results for 50 orders and 100 orders as we see in Fig. 10(a) and (c). We might speculate that this is a general property of greedy, contract-net-like algorithms. However, this speculation still needs confirmation by further theoretical and empirical results. For results comparing different horizontal cooperation settings at the SCA layer, we refer to [12].

VI. Scalability

As seen in Section II, AGENDA is used for simulation on both the microlevel and macrolevel by introducing agents to represent individuals in the system (e.g., trucks in the transportation domain) as well as groups of individuals (e.g., companies). In this section, we show the feasibility of our approach, which is based on rather complex agents as means of representation of entities in the simulation. In particular, we demonstrate the adaptability of our system to arbitrary application size. In the following section, we examine scalability of the transportation domain and scalability of the agent architecture and society.

A. Scalability of the Transportation Domain

In this section, we investigate complexity dependence of "agent communication" on the "number of agents" in the system. We use this measurement for the following reasons. In the transportation domain, shipping companies of arbitrary turnovers have to be modeled. The concrete number of agents in the system, in particular the number of TA's, reflects the abstract notions of all-over turnover and work load. Agent communication turns out to be a good performance indicator, because in physically distributed domains such as the transport domain, opening and using communication channels has proven to be the most important limiting factor. As a measurement, we do not use the simple amount of communication acts, but the overall number of communication primitives of which an act consists. Notice that our estimation is based on a pure-computational complexity. Estimating pure-computational complexity however, would be insufficient, because the complexity of the task an agent has to perform does not always have to effect performance of another agent and overall performance in a distributed system.

We now discuss communication complexity of the three previously presented communication protocols: CNP, ECNP, and ST. For the sake of independence of the underlying computational model, we assume that agent communication is only possible in a point-to-point fashion. So in this model, broadcast communication can only be realized by sequentially sending messages to communication partners. Of course, broadcasting may be available for realistic applications. In such cases, communication will be even less complex.

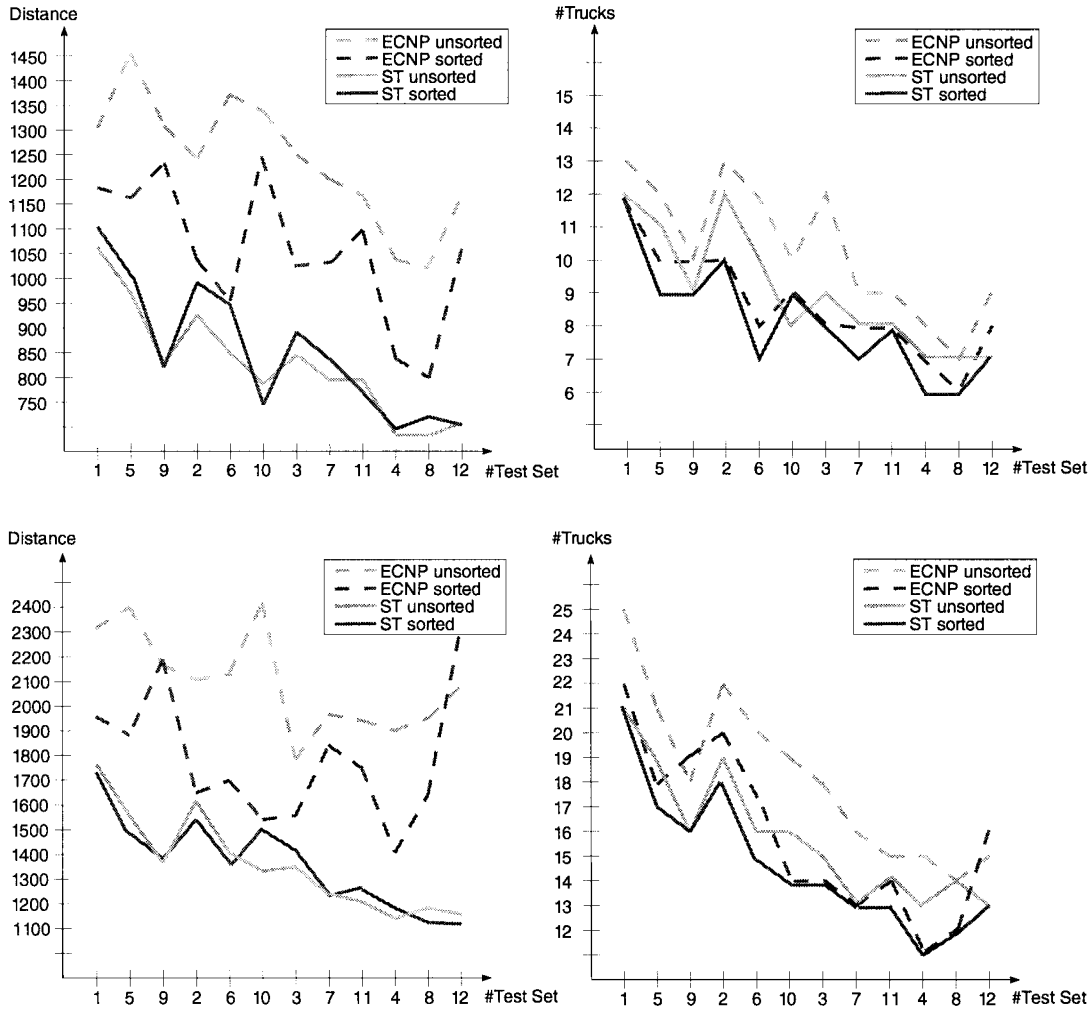


Fig. 10. Comparison of ECNP and ST on random and sorted input sets.

1) Contract Net Protocol (CNP): The classic CNP has been introduced in Section IV-A. During CNP, the following communication acts are performed. A manager sends bid requests for a certain good or order to all bidders who reply with their bids. Then, the manager selects an appropriate partner, confirms the cooperation, and sends rejects to all other bidders. Let n be the number of communicating agents. According to the point-to-point communication assumption, $(n-1)$ requests (each of which consist of one communication primitive) are made, followed by $(n-1)$ bids, one confirmation, and $(n-2)$ rejects, all consisting of one communication primitive. Hence, CNP has a communication complexity of $O(n)$ in terms of number of participating agents.

2) Extended Contract Net Protocol (ECNP): The ECNP (presented in Section IV-B) extends CNP in the sense that an order-partitioning strategy is now included in the protocol. If a chosen bidder is not willing to accept the whole order, the remaining part is reannounced by the manager. At first glance, ECNP occurs to be much more complex than CNP, because this partitioning strategy can be seen as a suboptimal solution procedure of the well-known NP-hard knapsack-problem. However, using CNP, the manager has to solve the knapsack problem before starting the protocol. Using ECNP, computational complexity is transferred to communication complexity. For each reannouncement procedure, a complete CNP must be performed. Hence, complexity increases to $O(nm)$ for the order size m. If, however, order size is *a priori*-limited up to a constant, ECNP's communication complexity is $O(n)$, just as in the case of CNP.

Practically speaking, ECNP has proven to be very useful, even for very large scale agent societies, because usually only a strongly limited set of agents appear as protocol participants. In the transportation domain, only agents representing trucks of the same company participate at an ECNP.

3) Simulated Trading (ST): An ST protocol (described in Section IV-D) consists of several levels of exchange simulation. At each level, every tour manager may announce a selling request or place a buying bid to the stock manager. After having computed the trading graph, the stock manager has to inform all tour managers of received selling requests. Again, let n be the number of agents participating in the negotiation. In the buying/selling announcement phase of each level, $O(n)$ primitive communication acts are performed. In the information phase, the stock manager has to send at most $O(n)$ messages. Such a message may contain at most $O(n)$ offers, which are primitive communication acts. As the

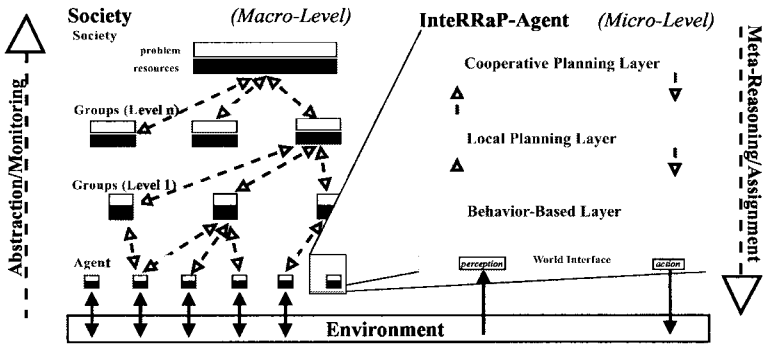


Fig. 11. Hierarchical control on the macrolevel and microlevel.

number of levels is fixed in advance, we get (according to the above assumption) an overall ST communication complexity of $O(n^2)$.

Similar to the case of ECNP, we can argue that usually only a few tour managers are involved in an ST procedure. The stock manager's task to process the trading graph (which is known to be NP-hard) does not influence communication complexity.

As a summary, we are able to show that from a theoretical point of view, all three protocols have linear, or at most, squared complexity in terms of communication acts. Speaking in practical terms, only a few agents join in such communication protocols, ensuring no communication bottlenecks, which ensures good scalability.

B. Scalability of the Agent Architecture and Society

Regarding agent architecture and society, complexity-theoretic claims on efficiency can hardly be made, especially because no clear mapping between agent architecture/structure and overall performance of the system can be achieved. However, resource-adapting mechanisms can be employed in order to enable the system to organize itself optimally according to current workload. A unified approach for resource adaptation on both the microlevel (the agent architecture) and the macrolevel, the agent society (see Fig. 11) can be found in [20]. In the following, we briefly summarize ideas presented there.

Resource control on the macrolevel focuses on distribution of resources among the whole agent society. This can be achieved by on-line controlling parameters such as number of agents in the society, overall society structure, formation of smaller groups, communication between these groups, etc. [19] gives a detailed enumeration on a domain-independent perspective. In the concrete, transportation domain-scalable parameters are, for instance, the following.

Number of TA's in a Company: Depending on the current turnover, a company has to vary the number of trucks in its fleet.

Formation of Negotiation Partners: Not all TA's need to participate in a negotiation. Depending on the freight type (e.g., liquids, dangerous goods, etc.) and current location of the physical trucks, only a limited number of agent representatives need to join the protocol.

Use of Communication Protocols: In our approach, ST is used to improve the results obtained from an ECNP. However, in some cases, the overhead of applying the additional ST protocol may not outweigh the gained performance improvement.

In the general case, subgroups of agents can again be decomposed to even smaller subgroups. Similar, but refined, resource control has to be performed on these group levels. In groups containing single agents, additional controllable features are abstract properties of agent members. In particular, agent characteristics contribute to the diversity of a group. They describe inhibitory or stimulating guidelines on resource allocation at the agent-architecture level. Actually, this establishes the so-called micro–macro link (MML). Concretely, in the transportation domain, agents representing trucks may require facilities other than agents representing shipping companies.

The microlevel of individual agents is finally responsible for transforming abstract task and resource guidelines that are given by the society into computational and external actions. The INTERRAP architecture implements a smooth transition from subsymbolic reactivity to symbolic deliberation and social capabilities by the notions of abstraction and meta-reasoning. As stated previously, the agent's state and computation is divided into three different layers. By performing meta-reasoning on a certain layer, resources are allocated to the next layer below. For instance, if an agent has to perform much reactive behavior at the moment, its BBL will obtain additional computational resources from the layer above (the local planning component), and if an agent needs to perform much reasoning, its inference module will receive additional resources.

On both the microlevel and macrolevel, mechanisms are installed to control resource distribution among agents in a group or among layers inside an agent. The goal is to find an optimal resource distribution, depending on a measurement of performance inherent to the overall task specification. This measurement is based on several factors such as operating time, quality of the result, etc. Zoutendjk's steepest ascent method [40] for finding local optima in some search space can well be adapted for that task for the following reasons. It is able to react fast to situation changes in order to maintain high, but not necessarily optimal, performance during the complete run of the application. A (local) optimal configuration can be found by moving from some arbitrary starting point in the configuration space iteratively to the currently optimal

configuration, modifying resource distribution in the (locally) most promising way.

VII. Conclusion and Future Work

In this paper, we have presented a multi-agent approach to the simulation of the transportation domain. Techniques developed in the context of MAS's, such as task decomposition and task allocation, decentralized planning, and negotiation have been applied to the scheduling of the transportation orders among an agent society consisting of shipping companies and their trucks. These techniques have been mixed with classical techniques from simulation in order to have an efficient simulation tool for studying interacting entities. The applicability and suitability of this tool for the real-world application of transportation scheduling in medium and large companies has been demonstrated.

This paper provides experimental results that show that the multi-agent approach as implemented in AGenDA has some fundamental advantages as a simulation tool. Thus, the multi-agent approach provides increased flexibility, since for instance, it allows dynamic variation of the number of agents during the simulation. In addition to this aspect, the multi-agent approach has many advantages that will result in on-line systems. Among these advantages, we can cite:

1) the ability to cope with open, dynamic scheduling problems and with the dynamics in plan execution;
2) the ability to cope with specific properties of agents such as autonomy, rationality, ability to plan and take initiative by pursuing goals, reasoning about others, and about the coordination, etc.

Finally, the study of scalability shows that techniques developed in this paper easily can be used for medium and large companies.

The current enormous advances in telecommunication and sensor technology establish the necessary preconditions to put techniques developed in this paper into practice in the transportation domain over the next few years. Trucks are now equipped with board computers that maintain the connection to their company and which allow them to obtain traffic information recorded by sensors installed along the roads. A decision-support system for the driver computes the currently (near) optimal route to go, based on this sensor information and on information it receives from the driver's company. Thus, new transportation orders can be allocated very flexibly and quickly to the appropriate resource. Tools based on algorithms such as the ST or the ENCP can be used to assist the human dispatcher in its allocation decisions.

An extension of the MARS system allowing the use of the methods presented in this paper for ordering dispatching tasks in a real shipping company is on the verge of being achieved.

References

[1] A. Bachem, W. Hochstättler, and M. Malich, "Simulated trading: A new approach for solving vehicle routing problems," Mathematisches Institut der Universität zu Köln, Germany, Report 92.125, 1992.

[2] ——, "The simulated trading heuristic for solving vehicle routing problems," Mathematisches Institut der Universität zu Köln, Germany, Report 93.139, 1993.

[3] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. RA-2, pp. 14–23, Jan. 1986.

[4] M. Boddy and T. Dean, "An analysis of time-dependent planning," in *Proc. 7th Nat. Conf. Artificial Intelligence*, St. Paul, MN, 1988, pp. 49–54.

[5] M. Boddy and T. L. Dean, "Deliberation scheduling for problem solving in time-constrained environments," *Artif. Intell.*, vol. 67, pp. 245–285, 1994.

[6] M. Buchheit, N. Kuhn, J. P. Müller, and M. Pischel, "MARS: Modeling a multiagent scenario for shipping companies," in *Proc. European Simulation Symp. (ESS-92)*, Dresden, Germany, 1992.

[7] B. Chaib-draa, "Industrial applications of distributed artificial intelligence," *Commun. ACM*, vol. 38, no. 11, pp. 49–53, 1995.

[8] R. Davis and R. G. Smith, "Negotiation as a metaphor for distributed problem solving," *Artif. Intell.*, vol. 20, pp. 63–109, 1983.

[9] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Oper. Res.*, vol. 40, no. 2, 1992.

[10] I. A. Ferguson, "TouringMachines: An architecture for dynamic, rational, mobile agents," Ph.D. thesis, Comput. Sci. Lab., Univ. Cambridge, U.K., 1992.

[11] R. J. Firby, "Building symbolic primitives with continuous control routines," in *Proc. 1st Int. Conf. Artificial Intelligence Planning Systems (AIPS-92)*, J. Hendler, Ed. San Mateo, CA: Morgan Kaufmann, 1992.

[12] K. Fischer, N. Kuhn, H. J. Müller, J. P. Müller, and M. Pischel, "Sophisticated and distributed: The transportation domain," in *Proc. 5th Eur. Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-93)*, Neuchatel, Switzerland, Aug. 1993.

[13] K. Fischer, N. Kuhn, and J. P. Müller, "Distributed, knowledge-based, reactive scheduling in the transportation domain," in *Proc. 10th IEEE Conf. Artificial Intelligence and Applications*, San Antonio, TX, Mar. 1994, pp. 47–53.

[14] K. Fischer, "Decision theoretic analysis of the transportation domain," in *Int. Workshop on Decision Theory for DAI Applications, ECAI'94*, K. Fischer and G. M. P. O'Hare, Eds. Amsterdam, The Netherlands: Wiley, Aug. 1994.

[15] K. Fischer, J. P. Müller, and M. Pischel, "AGenDA: A general testbed for DAI applications," in *Foundations of DAI*, N. R. Jennings and G. M. P. O'Hare, Eds. New York: Wiley, 1996, pp. 401–447.

[16] ——, "Cooperative transportation scheduling: An application domain for distributed AI," *Appl. Artif. Intell.*, vol. 10, no. 2, 1996.

[17] M. P. Georgeff and A. Rao, "Rational software agents: From theory to practice," in *Agent Technology, Foundations, Applications, and Markets*, N. Jennings and M. Wooldridge, Eds. Berlin, Germany: Springer Unicom, 1998, pp. 139–160.

[18] M. P. Georgeff and A. L. Lansky, "Procedural knowledge," in *Proc. IEEE Knowledge Representation*, vol. 74, pp. 1383–1398, 1986.

[19] C. Gerber, "An artificial agent society is more than a collection of "social" agents," in *Socially Intelligent Agents—Papers from the 1997 AAAI Fall Symp.*, AAAI, Portland, OR, 1997.

[20] C. Gerber and C. G. Jung, "Toward the bounded optimal agent society," in *Working Notes of the KI'97 Workshop on Distributed Cognitive Systems*, K. Fischer, C. G. Jung, and S. Schacht, Eds., 1997.

[21] S. Hanks, "Modeling a dynamic and uncertain world: Action representation and plan evaluation," Dept. Comp. Sci., Univ. Washington, Seattle, Tech. Rep. 93-09-07, 1993.

[22] S. Hanks, M. E. Pollack, and P. R. Cohen, "Benchmarks, test beds, controlled experimentation, and the design of agent architectures," *AI Mag.*, pp. 17–42, Winter 1993.

[23] M. N. Hunhs and M. P. Singh (Eds.), *Reading in Agents*. San Mateo, CA: Morgan Kaufmann, 1998.

[24] ICMAS-95, *1st Int. Conf. Multi-Agent Systems*, San Francisco, CA, 1995.

[25] ICMAS-96, *2nd Int. Conf. Multi-Agent Systems*, Kyoto, Japan, 1996.

[26] ICMAS-98, *3rd Int. Conf. Multi-Agent Systems*, Paris, France, 1998.

[27] L. P. Kaelbling, "An architecure for intelligent reactive systems," in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate, Eds. San Mateo, CA: Morgan Kaufmann, pp. 713–728, 1990.

[28] D. Kinny, M. Ljungberg, A. Rao, E. Sonenberg, G. Tidhar, and E. Werner, "Planned team activity," in *Proc. Eur. Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-92)*, A. Cesta, R. Conte, and M. Miceli, Eds. Springer-Verlag: Italy, July 1992.

[29] N. Kuhn, H. J. Müller, and J. P. Müller, "Simulating cooperative transportation companies," in *Proc. Eur. Simulation Multiconference (ESM-93)*, Lyon, France, 1993.

[30] J. C. Latombe, "How to move (physically speaking) in a multi-agent world," in *Decentralized AI 3*, E. Werner and Y. Demazeau, Eds. Kaiserslantern, Germany: North-Holland, pp. 263–280, 1992.

[31] D. McDermott, "Regression planning," *Int. J. Intell. Syst.*, vol. 6, pp. 357–416, 1991.
[32] B. Moulin and B. Chaib-draa, "An overview of distributed artificial intelligence," in *Foundations of Distributed Artificial Intelligence*, G. M. P. O'Hare and N. R. Jennings, Eds. New York: Wiley, 1996, pp. 3–55.
[33] J. P. Müller, "Evaluation of plans for multiple agents (preliminary report)," in *Working Notes of the Workshop on Decision Theory for DAI Applications at ECAI-94*, K. Fischer and G. M. P. O'Hare, Eds. Amsterdam, The Netherlands: Wiley, Aug. 1994.
[34] J. P. Müller and M. Pischel. "An architecture for dynamically interacting agents," *Int. J. Intell. Cooperative Inform. Syst.*, vol. 3, no. 1, pp. 25–45, 1994.
[35] ——, "Integrating agent interaction into a planner-reactor architecture," in *Proc. 13th Int. Workshop Distributed Artificial Intelligence*, Seattle, WA, July 1994.
[36] S. J. Russell and S. Zilberstein, "Composing real-time systems," in *Proceedings of 12th Int. Joint Conf. on AI (IJCAI-91)*. San Mateo, CA: Morgan Kaufmann, 1991, pp. 212–217.
[37] ——, "Anytime sensing, planning, and action: A practical model for robot control," in *Proc. 13th Int. Joint Conf. AI (IJCAI'93)*, Chambery, France, 1993, pp. 1402–1407.
[38] C. Schulte, G. Smolka, and J. Würtz, "Encapsulated search and constraint programming in Oz," in *2nd Workshop Principles and Practice of Constraint Programming*, Orcas Island, WA, May 2–4, 1994, pp. 116–129.
[39] M. P. Wellman, "A general-equilibrium approach to distributed transportation planning," in *Proc. AAAI-92*, San Jose, CA, 1992, pp. 282–290.
[40] G. Zoutendijk, *Mathematical Programming Methods*. Amsterdam, The Netherlands: North–Holland, 1976.

Klaus Fischer received the M.Sc. degree in computer science and the Ph.D. degree in computer engineering in 1992 from the Technische Universitaet (TU), Munich, Germany.

From 1986 to 1991, he worked in the joint research project SFB 331 Information Processing with Autonomous Mobile Robot Systems, Department of Computer Science, TU. In 1992, he joined the Department of Deduction and Multi-Agent Systems, Multi-Agent System Research Group, DFKI GmbH, Saarbrücken, Germany, and assumed the responsibility of Group Leader in Nov. 1993.

Brahim Chaib-draa (M'93) received the Ingénieur degree in computer engineering in 1978 and the Ph.D. degree in computer science in 1990, both from École Supérieure d'Électricité (SUPELEC) de Paris, France.

In 1990, he joined the Computer Science Department of Laval University, Ste-Foy, P.Q., (Canada), where he is Professor and Group Leader of the Data mining, Agents, and Multi-Agent Systems (DAMAS) group. He has published several technical publications in these areas. His funding sources have included the Natural Sciences and Engineering Research Council of Canada (NSERC), the "Fonds pour la Formation de Chercheurs et l'Aide la Recherche (FCAR)" of Québec Province, and the Social Sciences and Humanities Research Council of Canada (SSHRC). He is on the Editorial Boards of *Concurrent Engineering: Research and Applications (CERA)* and *The International Journal of Advanced Manufacturing systems*. His research interests include agent and multiagent technologies, natural language for the interaction, formal systems for agents and multiagent systems, distributed practical reasoning, and real-time and distributed systems.

Dr. Chaib-draa is member of the ACM and the AAAI.

Jörg P. Müller received the M.Sc. degree in computer science and the the Ph.D. degree in artificial intelligence in 1996, both from the Unversitaet des Saarlandes, Saarbrücken, Germany.

He is currently with Zuno Ltd., International House, London, U.K. From 1992 to 1996, he worked as a Researcher in the Multi-Agent System Research Group, DFKI GmbH, Saarbrücken. In 1996, he joined the Mitsubishi Electric Digital Library Group, London, U.K., where he conducted research and development work pertaining to the use of multiagent systems in the context of digital libraries. His main research interests are agent and multiagent systems and their applications.

Marcus Pischel, photograph and biography not available at the time of publication.

Christian Gerber received the M.Sc. degree in computer science from the Unversitaet des Saarlandes, Saarbrücken, Germany. He received the M.S. degree in artificial intelligence from Rutgers University, New Brunswick, NJ, in 1995. He is currently pursuing the Ph.D. degree in artificial intelligence at the Unversitaet des Saarlandes, Saarbrücken, Germany.

Since 1996, he has been a Researcher with the Multi-Agent System Research Group, DFKI GmbH, Saarbrücken, Germany.

Mr. Gerber was a Fulbright scholar in 1995.