Cooperative Transportation Scheduling: an Application Domain for DAI

Klaus Fischer, Jörg P. Müller, Markus Pischel*

DFKI GmbH, D-66123 Saarbrücken

Abstract

A multiagent approach to designing the transportation domain is presented. The MARS system is described which models cooperative order scheduling within a society of shipping companies. We argue why Distributed Artificial Intelligence (DAI) offers suitable tools to deal with the hard problems in this domain. We present three important instances for DAI techniques that proved useful in the transportation application: cooperation among the agents, task decomposition and task allocation, and decentralised planning.

An extension of the contract net protocol for task decomposition and task allocation 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. We demonstrate that the auction mechanism used for schedule optimisation can also be used for implementing dynamic replanning. Experimental results are provided evaluating the performance of different scheduling strategies.

*email: {kuf,jpm,pischel}@dfki.uni-sb.de

1 Introduction

In a time of constantly growing world-wide interdependence of trade and ware flow, logistics and the planning of freight transports are of crucial relevance both for economical and ecological reasons. The problems of vehicle routing and order scheduling are far from being satisfactorily solved in practice [Florian 93]. Moreover, in European shipping companies, factually no intelligent software support is available to the human scheduler. There exist efficient algorithms for solving static scheduling problems; however, classical computer science, Operations Research (OR), and also classical centralised AI have so far failed to provide adequate methodologies and algorithms to cope with open dynamic scheduling problems (see also [Bargl 92]). Unfortunately, virtually any problem of practical interest falls into the latter category. For example, one out of three trucks on Europe's roads drives empty; the average capacity utilisation in freight traffic is only about 55 % [Rittmann 91]. The process of political and economic integration in Europe and the abolishment of federal regulations on freight transportation services 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[1] simulation testbed (cf. [Kuhn et al. 93a]) 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 behaviour of the system is evaluated as follows: the quality measure are the costs for carrying out the orders. What is extraordinary about our approach is 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 to react quickly and without global replanning to unforeseen events, such as traffic jams or new transportation orders.

What is it that makes the transportation domain especially suitable for using techniques from (Distributed) AI ([Bond & Gasser 88], [Chaib-Draa et al. 92])? One reason is the complexity of the scheduling problem, which makes it very attractive for AI research (see Appendix B for a complexity-theoretic analysis of the domain). However there are more pragmatic reasons: *Commonsense knowledge* (e.g. taxonomic, topological, temporal, or expert knowledge) is necessary to solve the scheduling problems effectively. *Local knowledge about the capabilities* of the transportation company as well as knowledge about competitive (and maybe cooperative) companies massively influences the solutions. Moreover, since a global view is impossible (because of the complexity), there is a need to operate from a local point of view and thus to deal with *incomplete knowledge* with all its consequences. The last aspect leads to the DAI arguments:

- The domain is inherently distributed. Hence it is very natural to look at it as a multi-agent system. However, instead of tackling the problem from the point of view of the entities which are to be modelled and then relying on the emergence of the global solution, the classical approach to the problem is an (artificially) centralised one.

[1] **M**odeling **A**utonomous Coope**R**ating **S**hipping Companies

- There is a high degree of dynamics in the process of planning (new orders can be given to the system asynchronously) and execution (unforeseen events may occur, such as traffic jams). A recent direction in research on planning deals with exactly this class of problems (see e.g. [Haddawy & Hanks 90, Russell & Zilberstein 91, Kushmerick et al. 93, Boddy & Dean 94]). Architectures allowing agents to react to dynamics in execution while at the same time trying to achieve their longer-term goals have been provided by research in planning and DAI (e.g. [Cohen et al. 89, Firby 92, Ferguson 92, Lyons & Hendriks 92, Müller & Pischel 94a]).

- The task of a centrally maintaining and processing the knowledge about the shipping companies, their vehicles, and behaviour is very complex. Moreover, knowledge is often not even centrally available (real-life company are not willing to share *all* their local information with other companies). Therefore, modelling the companies as independent and autonomous units seems the only acceptable way to proceed.

- The existence of cooperative processes in the real transportation business makes the domain especially suitable for using DAI techniques such as task decomposition and task allocation [Davis & Smith 83, Kuhn et al. 93b, Decker & Lesser 94], decentralised planning[Ephrati & Rosenschein 93, Müller 94], and negotiation among agents [Zlotkin & Rosenschein 93, Rosenschein & Zlotkin 94].

The paper is organised as follows: We start with an overview of our multiagent system development environment and the underlying agent architecture in Section 2. In Section 3, the transportation domain is presented and analysed. Section 4 defines the multiagent approach for task decomposition, task allocation, and negotiation underlying the MARS system. An extension of the simulation system by a model for traffic jams is described in Section 5. Section 6 provides an auction-based mechanism for schedule optimisation and dynamic replanning in order to cope with the types of unforeseen events introduced in Section 5. Empirical results that have been obtained by running a series of scheduling benchmarks are reported in Section 7. Related work is described in Section 8. The paper ends with a discussion of the practical usefulness of the results presented so far.

2 The General Framework

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

2.1 The AGENDA Testbed

The AGENDA testbed [Fischer et al. 95] 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 a sense that it provides several important functionalities an agent should have; thus it supports a general template for agents that has to be filled by the designer of a DAI system with the domain-specific instantiation. The *system development level* provides the basic knowledge representation formalism, general inference mechanisms (such as forward and backward reasoning) which are used by the decision-making modules of the architectural layer, as well as a simulation toolbox supporting visualisation and monitoring of agents, and the gathering of performance statistics

(see [Hanks et al. 93] for a well-written discussion of properties, problems, and benefits of and examples for testbeds). The interrelationship between the two testbed levels is illustrated in figure 1.

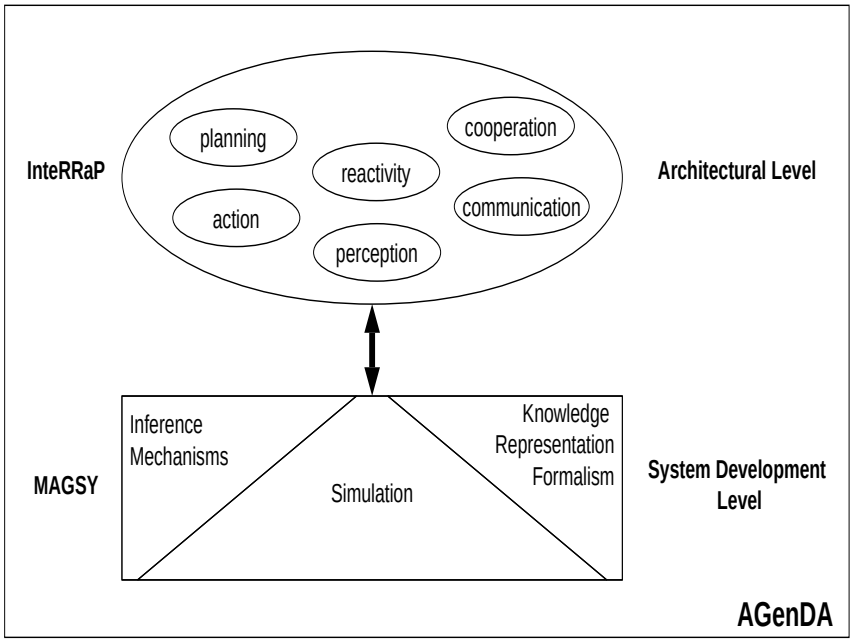

Figure 1: The AGenDA Testbed

The architectural level in the AGenDA Testbed is provided by the InteRRaP agent architecture [Müller & Pischel 94a, Müller & Pischel 94b]. 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 [Fischer 93]. MAGSY provides a frame-based knowledge representation formalism and a set of general purpose inference mechanisms. Moreover, it provides tools supporting the construction, visualisation, evaluation, and debugging of DAI 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 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 Mars system, the main challenge for our testbed was to provide different cooperation methods based on negotiation, leading to different scheduling mechanisms, and to experimentally evaluate these mechanisms.

An autonomous agent acting and interacting in a dynamic environment has to have certain properties, which should be reflected in its underlying design architecture. Firstly, agents are to behave in a *situated* manner, i.e. they have to perceive unexpected events

and to react appropriately to them (see e.g. [Brooks 86]). Secondly, they are to act in a *goal-directed* fashion in order to achieve their goals. In AI, this is normally achieved by devising plans for certain goals (see e.g. [McDermott 91]). Thirdly, they are to solve their tasks *efficiently* and often have to satisfy real-time constraints. This requires access to a set of "hard-wired" procedures [Georgeff & Lansky 86] with guaranteed execution properties. Fourthly, they are to cope with the presence of other agents. Whereas certain types of *interactions* can often be performed by employing local mechanisms (e.g. obstacle or collision avoidance in a robot scenario, see [Latombe 92, Müller & Pischel 94a, Müller & Pischel 94b]), others (e.g. 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) (see e.g. [Kinny et al. 92]), and thus the explicit representation of models of other agents in terms of beliefs, goals, plans, and intentions [Rao & Georgeff 91]. Finally, agents are to be *adaptive*, i.e. they must 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.

2.2 The INTERRAP Agent Architecture

The main idea of INTERRAP is to define an agent by a set of functional layers, linked by a communication-based control structure and a shared hierarchical knowledge base. The basic design elements of the agent are (1) its world interface facilities, (2) patterns of behaviour (PoB), as well as (3) local plans and (4) joint, multiagent plans.

Figure 2 shows the components of the INTERRAP agent model and their interplay. It consists of five basic parts: the World Interface (WIF), the Behaviour-Based Component (BBC), the Plan-Based Component (PBC), the Cooperation Component (CC), and the agent knowledge-base. The *world interface* holds the agent's facilities for perception, action, and communication. The *BBC* implements the reactive behaviour and the procedural knowledge of the agent. Basic building blocks of the BBC are *patterns of behaviour* which can be divided in two groups: *reactor patterns* and *procedure patterns*. Reactivity is obtained by providing a set of *reactor patterns* specifying hard-wired condition-action pairs. These are triggered by exogenous events. Procedural knowledge is contained in so-called *procedure patterns* which are activated by the plan-based component; these procedures are basically compiled plans which can be executed by the agent in order to perform some routine tasks.

The *PBC* contains a planning mechanism which is able to devise local single-agent plans. Depending on the requirements imposed by the application, the PBC may be instantiated with a suitable planning formalism. However, the interface definition between BBC and PBC requires that the latter can activate PoB, which are primitive actions from the perspective of the planner. The difference to classical AI planning systems is that PoB may be rather complex procedures (cf. [Firby 92]) incorporating a certain degree of execution intelligence (e.g. for dealing with different types of failures without explicit replanning). Finally, the *CC* contains a mechanism for devising joint plans (see [Kinny et al. 92, Müller 94]). It has access to protocols, and a multiagent planning mechanism which can access knowledge about other agents and about communication strategies. CC, PBC, and BBC establish the control of the agent. Their interaction (see e.g. [Müller & Pischel 94b]) defines the agent's overall behaviour.

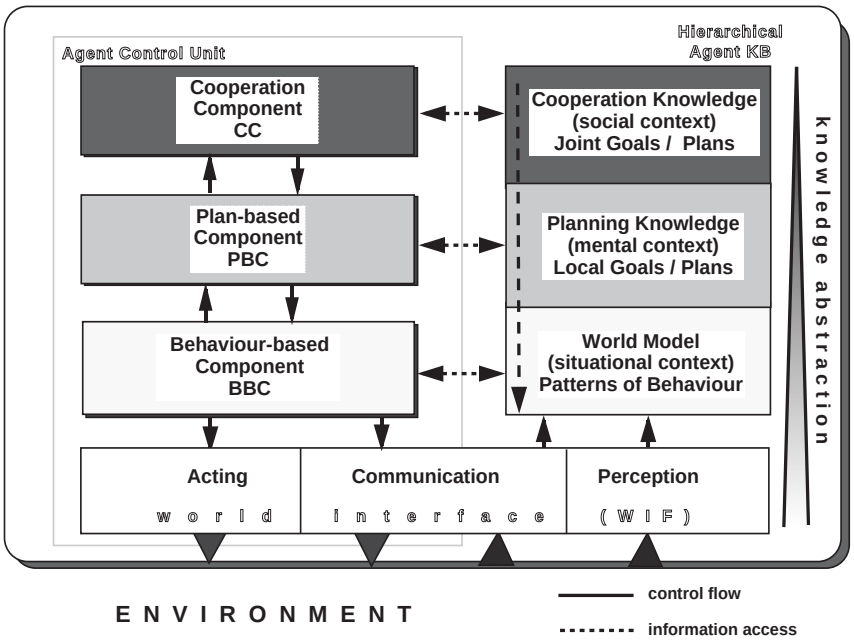

Figure 2: The InteRRaP Agent Architecture

The *knowledge base* is structured hierarchically. It consists of three layers which basically correspond to the structure of the agent control. The lowest layer contains facts representing the *world model* of the agent as well as representations of primitive actions and patterns of behaviour. The second layer contains the agent's *mental model*, i.e. representation of knowledge the agent has about its goals, skills, and plans. Finally, layer three comprises the agent's *social model*, i.e. knowledge of and strategies for cooperation, e.g. beliefs about other agents' goals. The basic idea is to restrict the information access and thus to reduce the practical complexity of reasoning in the lower (more reactive) system layers. For example, the plan-based component can access information about the world model, whereas the behaviour-based component does not have access to planning or cooperation information. This is supported by employing different reasoning mechanisms provided by MAGSY in different InteRRaP layers, namely data-driven forward reasoning in the BBC and goal-driven backward reasoning in the PBC and CC.

3 The Transportation Domain

In this section, the application domain of the Mars system is described and analysed. Some of the properties are derived that make dealing with this domain so difficult.

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 which must be solved in this area, such as the Travelling Salesman and related scheduling problems, are known to be NP-hard (see Appendix B for a formal analysis). As we have argued in the introduction, 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 to over-

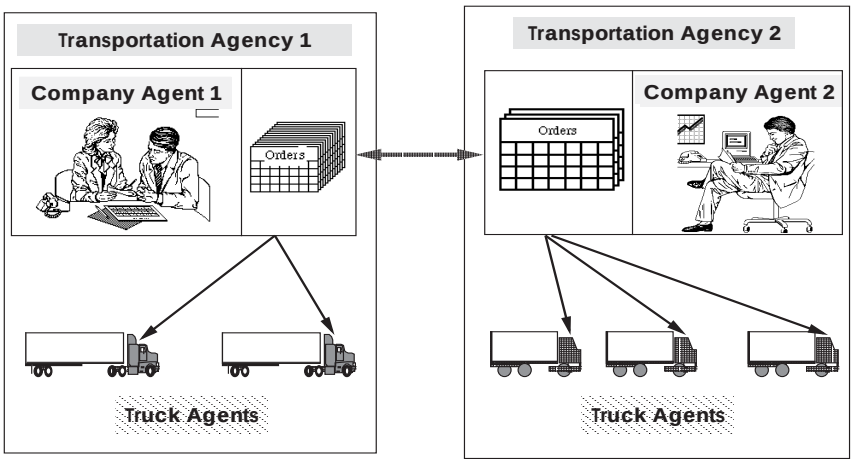

Figure 3: MARS: The Domain of Application

come the problems sketched above. Indeed, they are of increasing importance even in the highly competitive transportation business of today. Using the MARS system, several co-operation types such as the announcement of unbooked legs, order brokering, and different strategies for information exchange have been evaluated (see also [Fischer et al. 93]).

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

The *shipping company* agent (SCA) has to allocate orders to her[2] truck agents (TAs), 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 her own trucks. The functionality of an SCA is modelled in the INTERRAP architecture as follows:

- The BBC of a company is rather simple. It provides patterns of behaviour for recognising that a transportation order has been received, and for activating the communication primitives defined by the communication protocols (see section 4).

- Since an SCA does no scheduling on its own, the function of the PBC reduces to an algorithm that synthesises a plan for an order based on the partial bids received by the trucks.

- The CC contains the main part of the functionality of the SCA. The protocols for task allocation and negotiation (see Section 4) are represented as meta joint plans in a plan library (see [Müller 94]) and are executed by a plan interpreter.

Each TA is associated with a particular shipping company from which he receives orders of the form "`Load amount` s `of good` g_1 `at location` l_1 `and transport` it to

[2]We use 'she' to refer to shipping companies and 'he' to refer to trucks to resolve ambiguities.

`location` l_2 `while satisfying time constraints` $\{c_{t_1}, \ldots, c_{t_n}\}$". A TA is modelled as an InteRRaP agent as follows:

- The BBC of a truck agent contains PoB 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 recognising traffic jams based on data received by the travel information service (see Section 5). The primitive actions the TA is able to perform are driving, loading, unloading, as well as communicating with his company.

- A truck's PBC contains the local planning algorithm, which is a polynomial heuristic insertion algorithm; additionally, in order to compute a bit for an order, the TA has to evaluate the cost of his plan (see [Fischer et al. 94]).

- The CC of a truck contains the definition of the protocols used for communication with his SCA (see Section 4).

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 SCA even if they are not locally profitable for him. We call such a setting an instance of a *cooperative task-oriented domain* (cf. [Fischer 94]).

In the cooperation between SCAs 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. A practical example for this setting is the cooperation among different, geographically distributed branches of one shipping company. On the other hand, in a competitive task-oriented domain among the SCAs, the overall schedule which is computed will be far from optimal. In this setting we investigate how a single SCA can maximise her profits and how she can avoid being tricked by other agents.

4 The Mars Simulation System: A Multiagent Approach

In this section, we describe the multiagent approach underlying the Mars system; starting from the standard Contract Net protocol (Section 4.1), we define a framework that provides more powerful tools for task decomposition and task allocation (Section 4.2). A model for peer-to-peer negotiation among different SCAs is outlined in Section 4.3.

4.1 Vertical Cooperation: Task Decomposition and Task Allocation

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 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 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 fulfils 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 optimisation procedure. Our implementation is based on the Oz [Schulte et al. 94] system which was developed at DFKI in Saarbrücken and which provides powerful mechanisms for optimisation procedures in case the search space is defined by a constraint net.

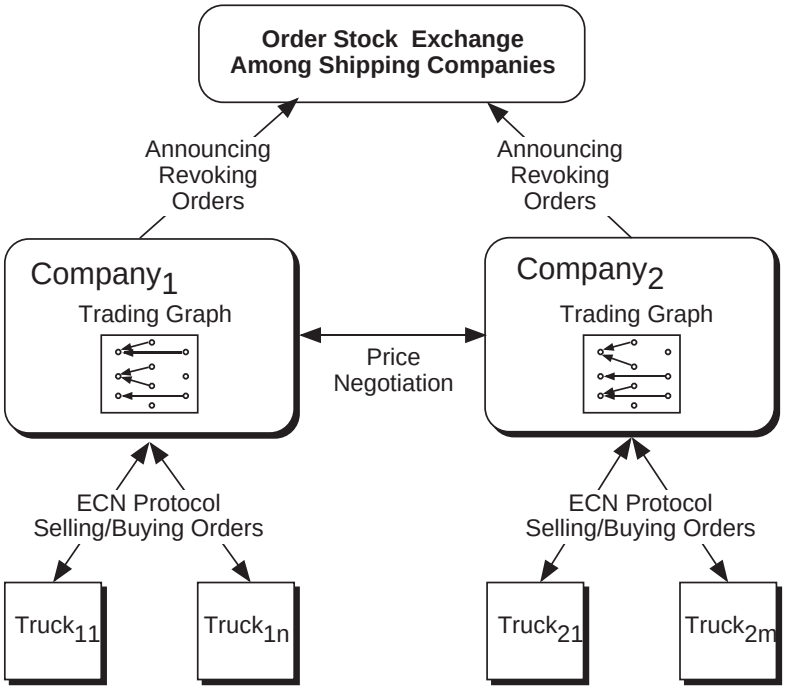

Figure 4: Hierarchical organisation of the agents in MARS.

For each order o announced by an SCA to her TAs, she receives a set of bids

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

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

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

and sends a grant to the TA a_{min}, notifying him 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 83]. Because the CNP provides time-out mechanisms it is easy to turn this communication protocol into an anytime algorithm (see for instance [Boddy & Dean 94, Russell & Zilberstein 93]), 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 4.2).

4.2 The 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, i.e. the SCA, has to solve a knapsack problem, which for itself is in general NP-hard. To overcome this problem, we decentralised task decomposition by developing and implementing an extension of the CNP, which is called the ECNP protocol. 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.

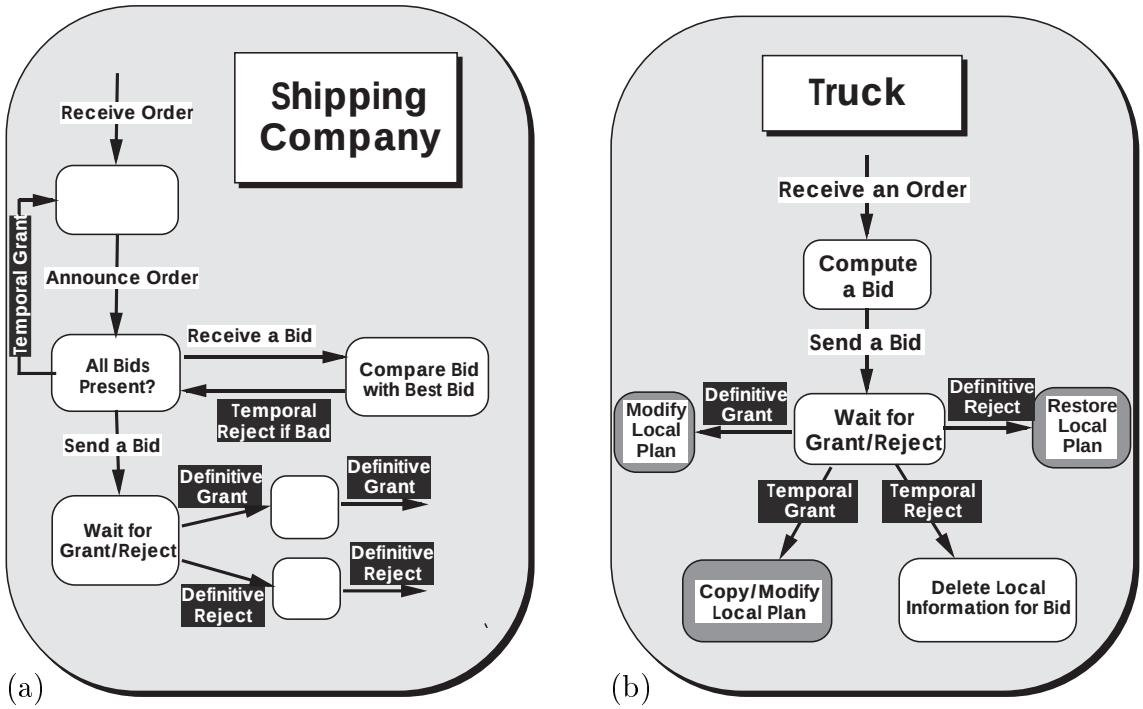

Figure 5: The ECNP from the point of view of a manager (a) and a bidder (b).

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. Figure 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, i.e. the TAs, are allowed to bid for only parts of an order.

In the ECNP the manager (SCA) announces an order o to its TAs. She 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 cover the total amount of the original order o. From this set of bids the SCA 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 et al. 94].

When a TA receives a temporal grant for the first time, he has to store a copy of his local situation, i.e. the currently valid plan, because he must be able to restore this situation in case he 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, he removes the copy created above and switches to the new plan. If a TA gets a definitive reject, he restores the situation before the first temporal grant.

In our framework, the ECNP is used to obtain a fairly good initial solution (see section 7 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 TAs will stick to decisions made in the past, the solution found is not even guaranteed to be pareto-optimal [Wellman 92].

There are different ways to optimise the ECNP solution. Currently, the *Simulated Trading* algorithm which is described in section 6 as a solution of the dynamic replanning problem is also used to optimise the order exchange among trucks. By coupling ECNP and Simulated Trading, we obtain an anytime algorithm [Boddy & Dean 88] $\mathcal{A}_{t_0}$ with a lowest time bound t_0 defined by the runtime of the ECNP process. I.e., $\mathcal{A}_{t_0}$ is an *interruptible anytime algorithm* [Russell & Zilberstein 91] for each $t \geq t_0$. Since the individual trucks employ a polynomial insertion algorithms for computing their bids within the ECNP, the time bound t_0 for the ECNP is polynomial.

4.3 Horizontal Cooperation: Negotiation

Optimising the utilisation 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. 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 an SCA and her trucks, 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. It is this peer-to-peer negotiation what we call horizontal cooperation and whose implementation is described in the sequel.

Negotiation Protocol: AGenDA supports the modelling of horizontal cooperation by providing a parametrised bargaining protocol which can be instantiated with the specific conditions of a negotiation. Figure 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 the connection to the vertical cooperation protocol with her 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

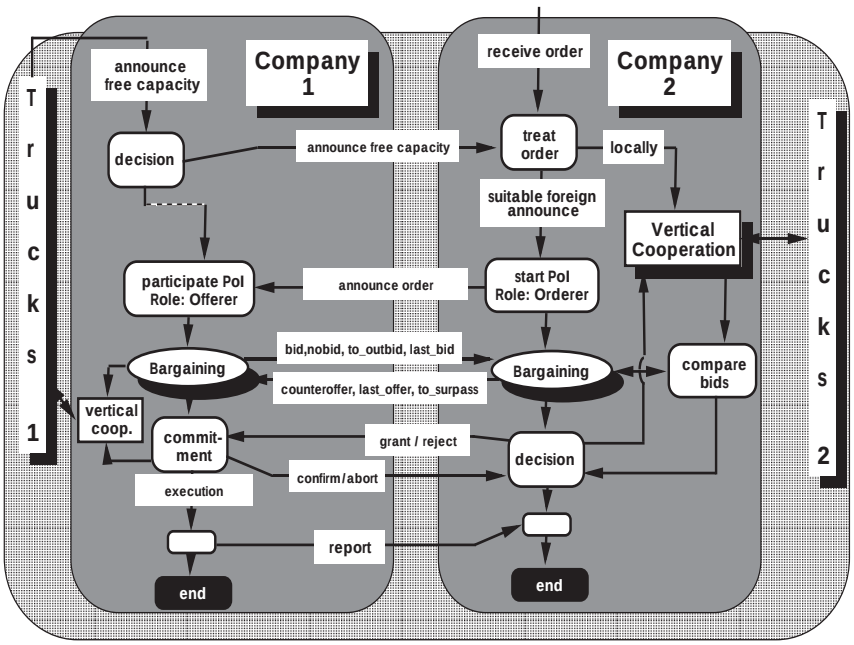

Figure 6: A Bargaining Protocol for Horizontal Cooperation

capacities c_1 has received by her trucks. Based on its local state, c_2 decides whether she wants to take up the announcement, and, if so, sends an order to c_1. This instantiates a bargaining protocol where c_1 takes the role of the offering agent, 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 figure 6 such as *last_bid, to_outbid, to_surpass* are special purpose features enabling an auction-like negotiation between more than two agents.

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 TAs, the SCAs make their decisions based on the utility of an order, which is computed as the difference between the worth (which is obtained from the customer or from other companies) and the costs. Based upon this information, a company determines in 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 modelling; 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 to 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 modelling* 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 behaviour 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; it 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 for 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 sequel, we will extend this framework to mechanisms allowing us to deal with dynamics in plan *execution*, too.

5 Introducing Execution Dynamics

In order to be able to explore methods for dealing with problems occurring due to unforeseen events happening during plan execution, the concept of traffic congestions has been integrated into the MARS system. It is described in the sequel. Firstly, the simulation environment is outlined in Section 5.1. Section 5.2 deals with how adequate cost functions can be defined. In Section 5.3, a probabilistic model for the generation of traffic jams is introduced.

5.1 The Simulation Framework

The enhanced MARS simulation environment can be divided in two parts: the simulation world and the agent society. The agent society has been described in Section 3. The simulation world consists of three parts: the world simulator module, the traffic jam generator, and the travel information service. The actual *world simulator module* maintains the state of the world, i.e. current positions of TAs and goods, the road map which is maintained as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, state of traffic, time etc. The *traffic jam generator* incorporates a statistical model for traffic jams. Its output is given as a function

$$\phi : \mathcal{T} \times \mathcal{E} \mapsto [0, 1],$$

where $\mathcal{T}$ is the set of time instants. Thus, ϕ computes for each connection between two cities of the road map at a time instant t_i the degree of traffic disturbance that ranges from $\phi \to 0$ (traffic jam) to 1 (no disturbance). In the sequel, we write ϕ_{ij}^t for the disturbance factor $\phi(t, e_{ij})$ on the edge connecting nodes $i, j \in \mathcal{V}$ at a given time instant t. Sensory data about the traffic density is provided to the third module of the simulation world, the *travel information service* (TIS). TIS information can be accessed from the TAs; it is used by them in order to compute the best route for a given situation (see Section 5.2).

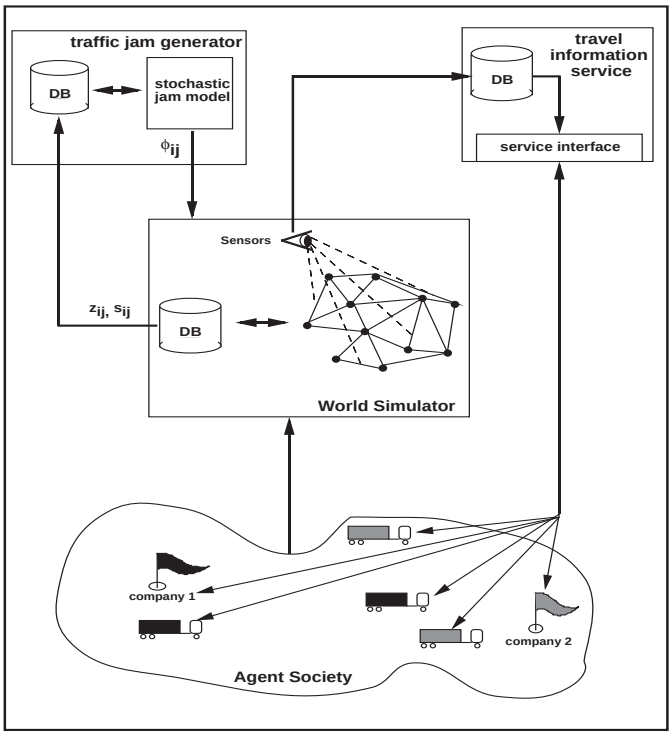

Figure 7: The Simulation Environment

5.2 Cost Functions

There are two different levels for describing how the TAs make their plans: the *rough planning* level maintaining which tasks to perform and in what order to perform them, and the *fine planning* level describing the actual route taken to perform a specific order. Since the rough planning takes place several hours before the start of a journey, it makes no sense to take into account an extensive amount of information regarding the traffic situation during that phase, because the situation is very likely to have changed by the time the order execution actually starts.

In the enhanced model which we present in the sequel, distances and disturbance factors ϕ_{ij}^t on an edge connecting nodes i and j are considered for the fine planning. ϕ_{ij}^t ranges over $[0, 1]$, where $\phi = 0$ means *traffic jam* and 1 means *no disturbance*. The objective function which is to be minimised is to be defined via costs. Criteria which have to be considered for this purpose are e.g. transportation time dur and distance d. The effect of the disturbance factor in our model is that it increases transportation time. The criteria are evaluated using factor prices. A model assumption is that the fixed costs C_{fix} are the same for all TAs and SCAs[3]. In the following, we set $C_{fix} = 0$ to simplify the model further. In this model, assuming the state of the world at time t, the time dur_{ij} a TA needs to get from a place i to a place j is given as

$$dur_{ij} = \left(\frac{\overline{d_{ij}}}{\overline{v_{ij}}} \right) \cdot \frac{1}{\phi_{ij}^t} \tag{1}$$

[3]Changing this assumption solely means shifting the calculated costs along the X-axis.

where

- $\overline{d_{ij}}$ is the distance $[km]$ between i and j,
- $\overline{v_{ij}}$ is the average speed $[km/h]$ of a TA under optimal conditions, and
- ϕ_{ij}^t is the value of the disturbance factor for the edge connecting nodes i and j at time t.

In the following, let $x_1, \ldots, x_k$ be the set of factors relevant for decision-making. In our case, $k = 2$: both the time needed and the distance travelled are considered as factors in the cost function. Now, for time t, given a tour T_t, we can define the objective function,

$$\begin{aligned} cost(T_t) &= \sum_{i=1}^{n} \sum_{j=1}^{n} \left(dur_{ij} \cdot p_1 + \overline{d_{ij}} \cdot p_2 \right) \cdot y_{ij} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \overline{d_{ij}} \cdot \left(\frac{p_1}{\phi_{ij}^t \cdot \overline{v_{ij}}} + p_2 \right) \cdot y_{ij} \rightarrow \min \end{aligned}$$

where

$$y_{ij} = \begin{cases} 1 & : \quad \text{if } e_{ij} \in T_t \\ 0 & : \quad \text{otherwise} \end{cases}$$

$p_1 =$ costs per hour, $p_2 =$ costs per km.

Using this formula, the current route as well as alternative routes to bypass a traffic jam are evaluated. For different types of disturbances, the evaluation can lead to different replanning strategies, which include either local replanning or global reallocation as described in Section 6.

5.3 A Model For Traffic Jam Generation

In the sequel, the disturbance variable ϕ which is used in order to generate and to simulate traffic jams is defined by means of a probabilistic model. Based on ϕ, the TAs then evaluate alternative routes according to the cost function defined in Section 5.2. For the disturbance variable ϕ_{ij}^t on an edge between i and j at time t, we have

$$\phi_{ij}^t = \begin{cases} 1 & : \quad \textit{no disturbance} \\ [0.6\text{–}1[& : \quad \textit{little disturbance} \\ [0.4\text{–}0.6[& : \quad \textit{medium disturbance} \\ [0.2\text{–}0.4[& : \quad \textit{heavy disturbance} \\ ]0\text{–}0.2 & : \quad \textit{stop and go traffic} \\ 0 & : \quad \textit{total jam} \end{cases}$$

The traffic density z_{ij}^t is used to compute ϕ_{ij}^t. z_{ij}^t denotes the number of vehicles going on edge e_{ij} at time t. For an assumed speed v a vehicle is allowed to go on edge e_{ij}, z_{ij}^t reaches its maximum value $\overline{s_{ij}^v}$. Note that for calculating the maximum value $\overline{s_{ij}^v}$ for z_{ij}^t which allows a TA to go with speed v (in this case $\phi_{ij}^t > 0$ holds), the reaction time $t_r(v)$ of the truck driver has to be considered when computing the average distance $\overline{dist_{cars}^v}$. In case of a traffic jam at edge e_{ij} ($\phi_{ij}^t \rightarrow 0$), z_{ij}^t reaches the maximum value $\overline{s_{ij}^0}$. In a simple approximation, $\overline{s_{ij}^v}$ is computed as

$$\overline{s_{ij}^v} = \frac{len(e_{ij})}{\overline{len_{cars}} + \overline{dist_{cars}^v}}, \tag{2}$$

where $\overline{len_{cars}}$ is the average length of a vehicle, $\overline{dist_{cars}^v}$ is the average distance between two vehicles going at speed v, $len(e_{ij})$ is the length of edge e_{ij}. The traffic density z_{ij}^t only affects the disturbance variable ϕ_{ij}^t if the TA cannot drive the planned average speed on this segment. For $z_{ij}^t \leq \overline{s_{ij}^v}$, there is $\phi_{ij}^t = 1$, i.e. no disturbance. We capture this by a function s_{ij}^v with

$$s_{ij}^v(z) = \begin{cases} \overline{s_{ij}^v}, & : \quad if\ 0 \leq z < \overline{s_{ij}^v}, \\ z, & : \quad if\ \overline{s_{ij}^v} \leq z \leq \overline{s_{ij}^0} \end{cases} \tag{3}$$

Based on this, the disturbance factor ϕ_{ij}^t can be defined as follows:

$$\phi_{ij}^t = \phi(t, e_{ij}) = \frac{\overline{s_{ij}^v} \cdot (\overline{s_{ij}^0} - s_{ij}^v(Z(t, e_{ij})))}{s_{ij}^v(Z(t, e_{ij})) \cdot (\overline{s_{ij}^0} - \overline{s_{ij}^v})}$$

where v is a global constant; $Z(t, e_{ij})$ is a random variable specifying the number of vehicles on edge e_{ij} which is computed by the formula:

$$Z(t, e_{ij}) = P(\varepsilon) \cdot \overline{s_{ij}^v} \cdot \lambda + (1 - \lambda) \cdot Z(t - 1, e_{ij}).$$

$P(\varepsilon)$ is a random variable with a normal distribution and a mathematical expected value of ε. ε is time-dependent (e.g. rush hours). λ determines the influence of random variable $P(\varepsilon)$ on $Z(t, e_{ij})$.

Up to now, we have described how jam information is generated and how it is integrated into the cost functions of the TAs, allowing them to derive information needed for decision-making. Section 6 describes a mechanism that allows the agents to react to unforeseen situations caused by the occurrence of traffic jams by initiating a dynamic reallocation process.

6 Simulated Trading: An Auction Mechanism for Dynamic Task Reallocation

The Simulated Trading (ST) [Bachem et al. 93] procedure which is presented in this section can be used for two different purposes:

- **Dynamic replanning:** if a TA realises that he 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 optimisation:** starting from the initial ECNP solution (see section 4.2), ST may be initiated to yield a better order allocation. The experimental results in Section 7 demonstrate the usefulness of ST as an optimisation technique.

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

6.1 Principles of ST

In [Bachem et al. 92], 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.

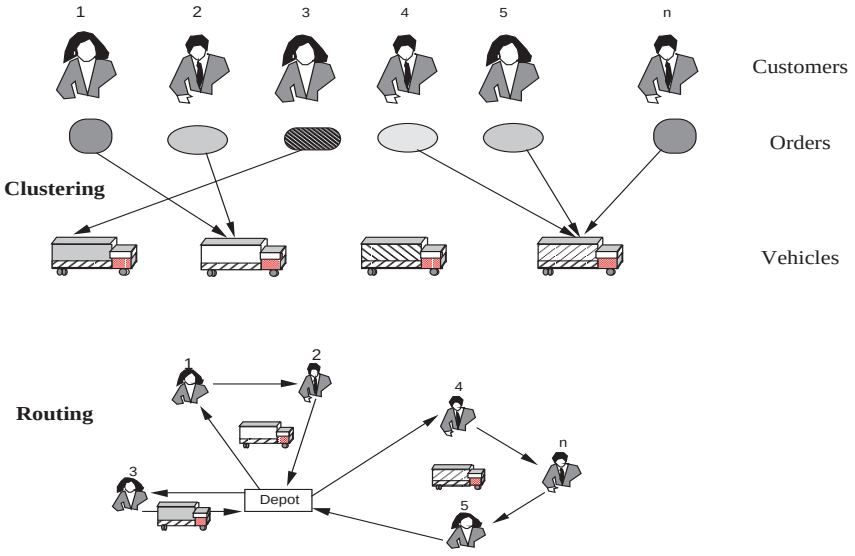

Figure 8: The Standard Vehicle Routing Problem

The solution to this problem is constructed using the Simulated Trading procedure. It starts with a set of feasible tours $T_1, \ldots, T_{t_0}$ which may e.g. be obtained by a conventional heuristic which is applicable to this domain. The tours are represented as an ordered list of costumers that have to be visited. Parallelism is achieved by that the data of each tour T_i can be 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 costumers orders between the different processors. To 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 as the price Pr to i, where

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

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 costumer i in T_p, i.e.

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

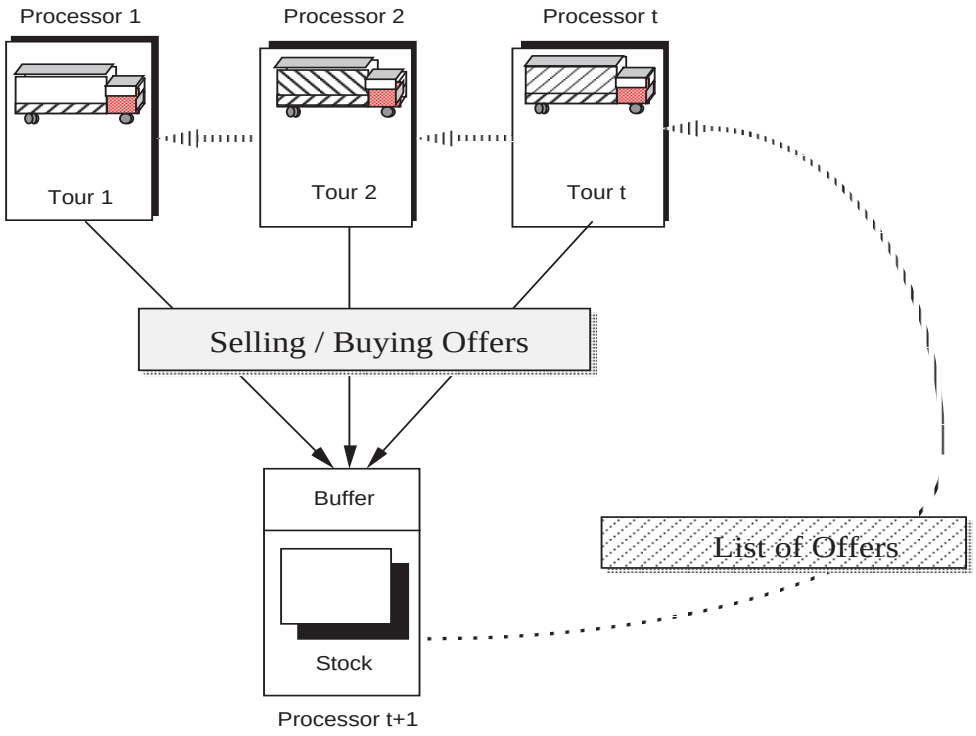

Figure 9: Stock Exchange for Orders in the ST Procedure

The exchange of orders is synchronised 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, Costumer, Price).

The stock manager maintains a data structure, called *Trading Graph* whose nodes are the selling and buying offers of the processors. Furthermore, there exists an edge between vertices v_i = (processor$_1$, l_i, Selling, c_i, Pr_i) and v_j = (processor$_2$, l_j, 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 labelled) by the difference of the prices Pr_j - Pr_i, giving 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 with offering a selling of an order a processor believes that this order eventually will be bought by another processor, and it will base its future price calculations on its reduced tour. Thus, an admissible exchange must ensure, that with each node $v_i \in M$, all nodes of the processors selling or buying v_i and which have a smaller level than v_i have to be also 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.

6.2 Adapting ST for the MARS System

The main idea is to let the SCA simulate a stock exchange 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 maintains the trading graph and tries to find an order exchange that optimises 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 turned 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 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 randomisation 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 which 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.

6.3 Using ST for Dynamic 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 he 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 (see Section 5). 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 where he has to deliver orders. 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 paralysed 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.

7 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 et al. 92], 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 modelled.
- 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 10 shows the results from a class of experiments comparing the relative performance of our solution before and after the optimisation 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%.

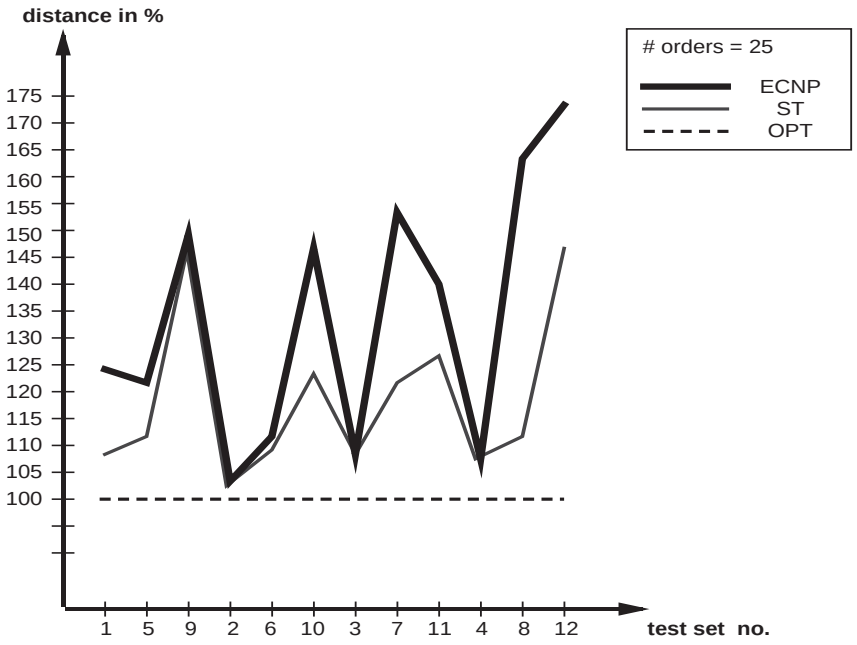

Figure 10: Comparison of ECNP and ST with the Optimal Solution

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 11a) to 11d). A more detailed table displaying the results for the benchmark examples in numerical form is enclosed in the appendix of this article.

The main results of these experiments can be summarised as follows: Firstly, ST improves the ECNP solutions in most cases. Secondly, presorting improves the behaviour 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

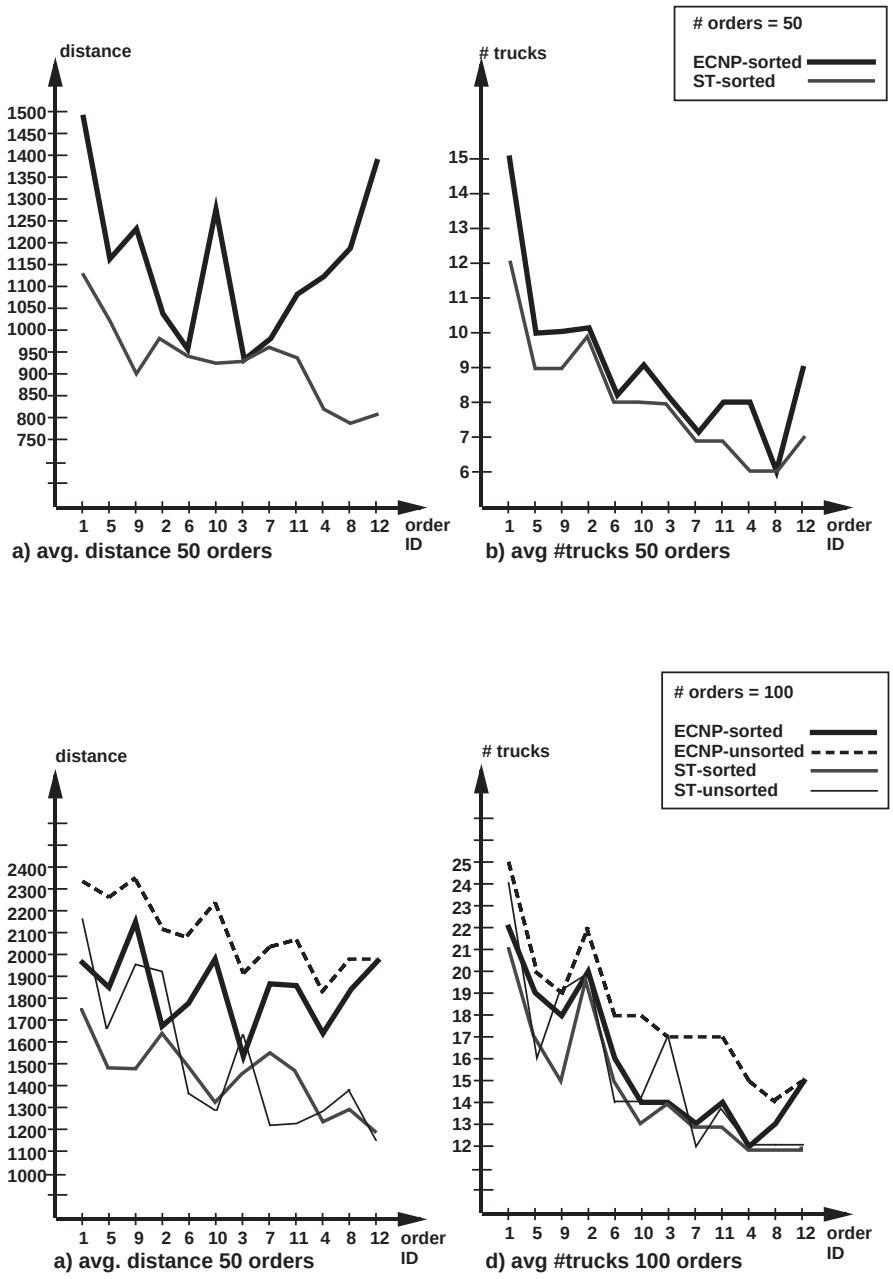

Figure 11: Comparison of ECNP and ST on Random and Sorted Input Sets

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 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. 93].

8 Related Work

The problem of delivering a set of orders is often regarded as a scheduling or a routing task, or a combination of both (see [Bodin et al. 83]). The difference between routing and scheduling tasks is that routing problems have no restriction on delivery time nor are there precedence relationships between stops. Hence, routing problems focus exclusively on the spatial or geometrical aspects of the problem. On the other hand, scheduling focusses exclusively on the time constraints of the problem. Combined routing and scheduling problems incorporate both spatial and temporal characteristics.

Compared to the large number of investigations dealing with static scheduling or routing problems, the dynamic problem instance, where new orders are allowed to be input into the system at any point in time, is only weakly explored. Particular attention has been given to dynamic aspects at the "Centre for Transportation Studies" at MIT where solutions to this problem have been developed since the mid seventies. An overview of several of these approaches is contained e.g. in [Bodin et al. 83].

Most of the MIT approaches rely on applying OR-based methods. However, it turns out that problems arise when the number of constraints to deal with grows or when real-time response of the system is required, e.g. in case system support should be given to a dispatcher who has to tell customers an estimated cost of an order at the phone. For this class of problems often knowledge-based approaches are used as those of Bagchi and Nag ([Bagchi & Nag 91]). They deal with the problem that a vehicle scheduler at a centralised facility receives customer requests for truck capacities at specific dates and times. The scheduler has to assign these loads to trucks obtained from contract carriers. Based on a study of the concepts of a human scheduler Bagchi and Nag have derived a set of rules which are used to build up a plan incrementally and to do some repairing if necessary. To implement these rules and to develop their dynamic load scheduling system EXLOAD they decided to use a rule-based expert system shell. Within their system, global optimisation is reduced to assigning a new shipment to a contract with minimal incremental cost caused by that insertion. This is based on a result of Psaraftis ([Psaraftis 88]) who shows that in a dynamic scheduling environment global minimisation over a period of time is best achieved by minimising the incremental cost of each assignment.

In contrast to our approach, Bagchi and Nag offer a centralised solution concept that covers only the problem solving for one dispatcher in a single shipping company. MARSextends this approach by providing an inter-company perspective that could be combined with Bagchi and Nag's solution to model the intra-company situation.

OR-based approaches have been applied successfully to solve static instances of the Vehicle Routing Problem. However, in order to be used in a dynamic environment these methods have to be enhanced with mechanisms providing a real-time behaviour of the corresponding algorithms. Furthermore, usually OR-based methods are difficult to use if the number of constraints is high (see [Golden & Assad 83, Psaraftis 88, Bagchi & Nag 91]).

Falk, Spieck, and Mertens (cf. [Falk et al. 93]) pursue an approach based on the integration of knowledge-based mechanisms and OR algorithms. This combination of two methodologies is expressed by the term *Partial Intelligent Agents* (PIAs) used to denote

components of distributed, cooperating systems having a hybrid structure, i.e. modules that include a "conventional" (usually OR-based) and a knowledge-based part. Each agency is represented by a *dispatching PIA* that is responsible for the allocation of the orders of its agency to the trucks. The dispatcher knows the current location of its trucks and it bases its decision on this knowledge. Its objective function considers maximising the utilisation of the trucks' capacity, minimising the idle time and rides without carriage, and minimising the length of the route for a single order.

The process of cooperative planning for a new order is basically handled by the Contract Net Protocol as proposed in [Davis & Smith 83]. Falk et al. do not only use the Contract Net for the task allocation process but also for task decomposition.

Compared to our modelling the approach described above considers an instance of our domain, namely a single company which is geographically distributed. Thus, the dispatching agents are willing to exchange all the information (in this case, the complete route plans) in a cooperation process. A further difference to our approach is that Falk et al. do not allow the actual transportation resources to take an active role by modelling them as agents. Doing this allows us to parallelise the scheduling process and thus, to reduce the practical problem complexity (see Section 3).

9 Conclusion and Outlook

In this paper, we have presented a multiagent approach to the design of the transportation problem. Techniques developed in Distributed AI, such as task decomposition and task allocation, decentralised planning, and negotiation have been applied to the scheduling of transportation orders among an agent society consisting of shipping companies and their trucks. The applicability and suitability of these techniques for the real-world application of transportation scheduling in medium-size and large shipping companies has been demonstrated by developing the simulation system MARS.

The paper provides experimental results indicating that the multiagent approach to scheduling achieves acceptable solutions that are comparable to those of heuristic search Operation Research algorithms. Moreover, the multiagent approach as implemented in the MARS system has some fundamental advantages over standard OR algorithms: It provides increased flexibility, since it allows to vary dynamically the number of agents, even during the simulation. Moreover, whereas the scope of the available Operations Research techniques is limited to static scheduling problems, the multiagent approach results in an on-line system, which can cope with open, dynamic scheduling problems and with the dynamics in plan execution. Especially the latter argument makes DAI tools for task decomposition, task allocation, planning, and negotiation a powerful and promising alternative for solving industrial scheduling problems.

The current enormous advances in telecommunication and sensor technology establish the necessary preconditions to put DAI concepts into practice in the transportation domain over the next few years: trucks are equipped with board computers which – via wireless modem (see e.g. MODACOM [Preissner-Polte 93]) – maintain the connection to their company, and which allow the truck to obtain traffic information recorded by sensors installed along the roads. A decision-support system for the driver computes the currently 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 concepts such as the ECNP 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 order dispatching tasks in a real shipping company is an important medium-term goal of our research.

As regards theoretical aspects of our research, 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. Preliminary work published in [Fischer 94] shows that the general results presented by Zlotkin and Rosenschein [Zlotkin & Rosenschein 93] for task-oriented domains are not fully applicable to the transportation domain as presented in this paper. A more detailed treatment of this issue will be subject to future work.

Acknowledgements

We would like to thank Norbert Kuhn for intensive discussions and helpful comments on earlier drafts of this paper. Darius Schier implemented the Simulated Trading algorithm. Martin Malich (University of Cologne) helped us obtaining the benchmark data and was very cooperative in all matters concerning Simulated Trading. We thank the anonymous reviewers of the International Journal of Applied Artificial Intelligence for comments that contributed to improving the quality and the readability of the paper.

References

[Bachem et al. 92] A. **Bachem**, W. **Hochstättler**, and M. **Malich**. *Simulated Trading: A New Approach For Solving Vehicle Routing Problems.* Technical Report 92.125, Mathematisches Institut der Universität zu Köln, Dezember 1992.

[Bachem et al. 93] A. **Bachem**, W. **Hochstättler**, and M. **Malich**. *The Simulated Trading Heuristic for Solving Vehicle Routing Problems.* Technical Report 93.139, Mathematisches Institut der Universität zu Köln, 1993.

[Bagchi & Nag 91] P. **Bagchi** and B. **Nag**. *Dynamic Vehicle Scheduling: An Expert System Approach.* Journal of Physical Distribution and Logistics Management, 21(2), 1991.

[Bargl 92] Michael **Bargl**. *Tourenplanung in der Transportwirtschaft - viel gelobt, selten genutzt?* Logistik im Unternehmen, 6(6), Juni 1992.

[Boddy & Dean 88] M. **Boddy** and T. **Dean**. *An Analysis of Time-dependent Planning.* In: Proceedings of the 7th National Conference on Artificial Intelligence, pp. 49–54, 1988.

[Boddy & Dean 94] M. **Boddy** and T. L. **Dean**. *Deliberation scheduling for problem solving in time-constrained environments.* Artificial Intelligence, 67:245–285, 1994.

[Bodin et al. 83] L. **Bodin**, B. **Golden**, A. **Assad**, and M. **Ball**. *Routing and Scheduling of Vehicles and Crews.* Computers and Operations Research, 10(2):63 – 211, 1983.

[Bond & Gasser 88] A. **Bond** and L. **Gasser**. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, Los Angeles, CA, 1988.

[Brooks 86] Rodney A. **Brooks**. *A Robust Layered Control System for a Mobile Robot.* In: IEEE Journal of Robotics and Automation, volume RA-2 (1), pp. 14–23, April 1986.

[Chaib-Draa et al. 92] B. **Chaib-Draa**, B. **Moulin**, R. **Mandiau**, and P. **Millot**. *Trends in Distributed Artificial Intelligence.* Artificial Intelligence Review, 1(6):35–66, 1992.

[Cohen et al. 89] P. R. **Cohen**, M. L. **Greenberg**, D. M. **Hart**, and A. E. **Howe**. *Trial by Fire: Understanding the Design Requirements for Agents in Complex Environments.* AI Magazine, 10(3), 1989.

[Davis & Smith 83] R. **Davis** and R. G. **Smith**. *Negotiation as a metaphor for distributed problem solving.* Artificial Intelligence, 20:63 – 109, 1983.

[Decker & Lesser 94] K. **Decker** and V. R. **Lesser**. *Designing a Family of Coordination Algorithms.* In: Proceedings of the 13th International Workshop on Distributed Artificial Intelligence, pp. 65–84, Lake Quinalt, Washington, July 1994.

[Desrochers et al. 92] M. **Desrochers**, J. **Desrosiers**, and M. **Solomon**. *A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows.* Operations Research, 40(2), 1992.

[Ephrati & Rosenschein 93] E. **Ephrati** and J. **Rosenschein**. *Multi-Agent Planning as a Dynamic Search for Social Consensus.* In: Proc. of IJCAI-93, pp. 423–429. Morgan Kaufmann, San Mateo, CA, August 1993.

[Falk et al. 93] J. **Falk**, S. **Spieck**, and P. **Mertens**. *Unterstützung der Lager- und Transportlogistik durch Teilintelligente Agenten.* Information Management, 2, 1993.

[Ferguson 92] I. A. **Ferguson**. *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents.* PhD thesis, Computer Laboratory, University of Cambridge, UK,, 1992.

[Firby 92] R. James **Firby**. *Building Symbolic Primitives with Continuous Control Routines.* In: J. Hendler (ed.), Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems (AIPS-92). Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[Fischer & Kuhn 93] K. **Fischer** and N. **Kuhn**. *A DAI Approach to Modeling the Transportation Domain.* Research Report RR-93-25, DFKI, 1993.

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

[Fischer et al. 93] K. **Fischer**, N. **Kuhn**, H. J. **Müller**, J. P. **Müller**, and M. **Pischel**. *Sophisticated and Distributed: The Transportation Domain.* In: Proceedings of MAAMAW-93, Neuchatel, CH, August 1993. Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World.

[Fischer et al. 94] K. **Fischer**, N. **Kuhn**, and J. P. **Müller**. *Distributed, Knowledge-Based, Reactive Scheduling in the Transportation Domain.* In: Proceedings of the Tenth IEEE Conference on Artificial Intelligence and Applications, pp. 47–53, San Antonio, Texas, March 1994.

[Fischer et al. 95] K. **Fischer**, J. P. **Müller**, and M. **Pischel**. *Unifying Control in a Layered Agent Architecture.* Technical Memo TM-94-05, DFKI GmbH, Saarbrücken, January 1995.

[Fischer 93] K. **Fischer**. *The Rule-based Multi-Agent System MAGSY.* In: Proceedings of the CKBS'92 Workshop. Keele University, 1993.

[Fischer 94] K. **Fischer**. *Decision Theoretic Analysis of the Transportation Domain.* In: [Fischer & O'Hare 94], Amsterdam, August 1994.

[Florian 93] Michael **Florian**. *Highway Helden in Not. Ein Beitrag zum Verständnis der sozialen Reproduktion arbeits- und berufsbedingter Risiken von Fernfahrern.* PhD thesis, Universität Münster, 1993.

[Garey & Johnson 79] M.R. **Garey** and D.S. **Johnson**. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W.H. Freeman and Company, 1979.

[Georgeff & Lansky 86] M. P. **Georgeff** and A. L. **Lansky**. *Procedural knowledge.* In: Proceedings of the IEEE Special Issue on Knowledge Representation, volume 74, pp. 1383–1398, 1986.

[Golden & Assad 83] B. **Golden** and A. **Assad**. *Vehicle Routing: Methods and Studies.* Studies in Management Science and Systems. North Holland, 1983.

[Haddawy & Hanks 90] P. **Haddawy** and S. **Hanks**. *Issues in Decision-Theoretic Planning: Symbolic Goals and Numeric Utilities.* In: Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control. Morgan Kaufmann, November 1990.

[Hanks et al. 93] S. **Hanks**, M. E. **Pollack**, and P. R. **Cohen**. *Benchmarks, Test Beds, Controlled Experimentation, and the Design of Agent Architectures.* AI Magazine, Winter:17–42, 1993.

[Kinny et al. 92] D. **Kinny**, M. **Ljungberg**, A. **Rao**, E. **Sonenberg**, G. **Tidhar**, and E. **Werner**. *Planned Team Activity.* In: A. Cesta, R. Conte, and M. Miceli (eds.), Pre-Proceedings of MAAMAW'92, July 1992.

[Kuhn et al. 93a] N. **Kuhn**, H. J. **Müller**, and J. P. **Müller**. *Simulating Cooperative Transportation Companies.* In: Proceedings of the European Simulation Multiconference (ESM-93), Lyon, France, June 1993. Society for Computer Simulation.

[Kuhn et al. 93b] N. **Kuhn**, H. J. **Müller**, and J. P. **Müller**. *Task Decomposition in Dynamic Agent Societies.* In: Proceedings of the International Symposium on Autonomous Decentralized Systems (ISADS-93), Tokyo, Japan, 1993. IEEE Computer Society Press.

[Kushmerick et al. 93] N. **Kushmerick**, S. **Hanks**, and D. **Weld**. *An Algorithm for Probabilistic Planning.* Technical Report 93-06-03, University of Washington, Department of Computer Science, June 1993.

[Latombe 92] J. P. **Latombe**. *How To Move (Physically Speaking) in a Multi-Agent World.* In: Y. Demazeau and E. Werner (eds.), Decentralized A.I., volume 3. North-Holland, 1992.

[Lyons & Hendriks 92] D. M. **Lyons** and A. J. **Hendriks**. *A Practical Approach to Integrating Reaction and Deliberation.* In: Proceedings of the 1st International Conference on AI Planning Systems (AIPS), pp. 153–162, San Mateo, CA, June 1992. Morgan Kaufmann.

[McDermott 91] D. **McDermott**. *Robot Planning.* Technical Report 861, Yale University, Department of Computer Science, 1991.

[Müller & Pischel 94a] J. P. **Müller** and M. **Pischel**. *An Architecture for Dynamically Interacting Agents.* International Journal of Intelligent and Cooperative Information Systems (IJICIS), 3(1):25–45, 1994.

[Müller & Pischel 94b] J. P. **Müller** and M. **Pischel**. *Integrating Agent Interaction into a Planner-Reactor Architecture.* In: M. Klein (ed.), Proceedings of the 13th International Workshop on Distributed Artificial Intelligence, Seattle, WA, USA, July 1994.

[Müller 94] J. P. **Müller**. *Evaluation of Plans for Multiple Agents (Preliminary Report).* In: K. Fischer and G. M. P. O'Hare (eds.), Working Notes of the Workshop on Decision Theory for DAI Applications at ECAI-94, Amsterdam, NL, August 1994.

[Preissner-Polte 93] A. **Preissner-Polte**. *Funk-Verkehr.* manager magazin, 5, 1993.

[Psaraftis 88] H. **Psaraftis**. *Dynamic Vehicle Routing Problems.* In: Vehicle Routing: Methods and Studies. North-Holland, 1988.

[Rao & Georgeff 91] A. S. **Rao** and M. P. **Georgeff**. *Modeling Agents Within a BDI-Architecture.* In: R. Fikes and E. Sandewall (eds.), Proc. of the 2rd International Conference on Principles of Knowledge Representation and Reasoning (KR'91), pp. 473–484, Cambridge, Mass., April 1991. Morgan Kaufmann.

[Rittmann 91] R. **Rittmann**. *Die Macht der Trucks.* Bild der Wissenschaft, 9:112–114, 1991.

[Rosenschein & Zlotkin 94] J. S. **Rosenschein** and G. **Zlotkin**. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers.* MIT Press, 1994.

[Russell & Zilberstein 91] S. J. **Russell** and S. **Zilberstein**. *Composing Real-Time Systems.* In: Proceedings of IJCAI-91, pp. 212–217. Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1991.

[Russell & Zilberstein 93] S. J. **Russell** and S. **Zilberstein**. *Anytime Sensing, Planning, and Action: A Practical Model for Robot Control.* In: Proceedings of IJCAI'93, pp. 1402–1407, Chambery, F, 1993. Morgan Kaufmann Publishers Inc., San Mateo, CA, USA.

[Schulte et al. 94] Christian **Schulte**, Gert **Smolka**, and Jörg **Würtz**. *Encapsulated Search and Constraint Programming in Oz.* In: Second Workshop on Principles and Practice of Constraint Programming, pp. 116–129, Orcas Island, Washington, USA, 2-4 May 1994.

[Wellman 92] M.P. **Wellman**. *A General-Equilibrium Approach to Distributed Transportation Planning.* In: Proceedings of AAAI-92, pp. 282–290, 1992.

[Zlotkin & Rosenschein 93] G. **Zlotkin** and J. S. **Rosenschein**. *A Domain Theory for Task-Oriented Negotiation.* In: Proc. of the 13th International Joint Conference on Artificial Intelligence, volume 1, Chambéry, France, 28.8.–3.9. 1993.

A Benchmark Results

Legend for table 1:

TS	Number of test set	#O	Number of orders
E	ECNP solution	S	Simulated Trading solution
O	Optimal solution	Dist.	Distance travelled
WT	waiting time	Std.	sorted?
T	Total time needed (incl. waiting time)		

B Computational Complexity of Transportation Problems

In this Section, we summarise basic complexity results showing that the transportation problems tackled by our research are $\mathcal{NP}$-hard.

The class of problems we are interested in is characterised by the fact that orders may be entered into the system at any time, and thus defines an open routing or scheduling problem. Usually, this problem instance is called the *Dynamic Vehicle Routing Problem* (DVRP). We will now formalise the intuitive notion of the transportation problem described so far.

Definition 1 *The Routing Decision Problem RDP:*

> **INSTANCE***:*
> *Graph* $G = (V, E)$,
> *Length* $l(e) \in Z_0^+$ *for each* $e \in E$,
> *Set of orders* $O = \{\ o_i=(s_i,d_i,w_i) \mid i = 1, \ldots, m,$ *with* $s_i \in E$ *being the starting point of* o_i, $d_i \in E$ *being the target point of* o_i, *and* $w_i \in Z^+$ *giving the weight (or volume) of* $o_i\ \}$,
> *Trucks* $A = \{a_1, \ldots, a_m\ \}$,
> *Function Capacity:* $A \rightarrow Z^+$ *giving the capacity of each truck, and*
> *Bound* $B \in Z^+$.
>
> **QUESTION***:*
> *Is there a disposition function* D*:* $O \rightarrow A$ *and a routing of the trucks* $a_i \in A$*, such that all orders are delivered and that the sum of the length of the route of the trucks is at most* B*?*

Note that the Routing Decision Problem RDP is only a simplified version of the problem which we actually deal with, since there are no precedences between orders, nor are time windows specified. However, the following theorem 1 states that already this problem is $\mathcal{NP}$ complete.

Theorem 1 *RDP is NP-complete.*

We refer to [Fischer & Kuhn 93] for the proof of theorem 1. It is done in two steps. Firstly, a polynomial time reduction of the *Modified Rural Postman Problem* which is known to be NP-complete (cf. [Garey & Johnson 79]) to the RDP is given. This shows that the Routing Decision Problem is at least NP-hard. Then, a nondeterministic polynomial-time algorithm for RDP is provided, showing that RDP itself belongs to NP. □

Finally, theorem 2 states that the RDP is $\mathcal{NP}$ complete even if there is only a single truck, and if the length of each edge e in the graph G is equal to one.

Theorem 2 *The RDP is* $\mathcal{NP}$ *complete even if* $|A| = 1$ *and* $l(e) = 1$ *for all* $e \in E$.

The theoretical results obtained so far show that the scheduling problems in the transportation domain are intractable, i.e. we should not hope to find complete algorithms running in less than exponential time. Therefore, there is a need for using heuristic algorithms, new methodologies, and new techniques for dealing with the transportation problems. In the sequel, we present a heuristic approach to a solution of these problems based on the multi-agent paradigm.

TS	#O	# Trucks			Dist.			T		WT		Std.
		E	S	O	E	S	O	E	S	E	S	
R101	25	9	8	8	768.5	665.3	617.1	1464.5	1365.8	44.6	450.5	y
R101	50	15	13	12	1483.2	1150.4	1035.2	2664.1	2282.0	68.1	631.5	y
R101	100	22	22	18	1959.9	1833.9	1607.7	3891.2	3887.5	931.2	1053.5	y
R101	100	22	21	-	2327.4	1835.4	-	4514.7	3891.0	1187.2	1055.5	n
R102	25	7	7	7	566.0	566.0	547.1	1243.3	1243.3	427.2	427.2	y
R102	50	10	10	11	1041.7	1041.7	904.6	1917.7	1917.7	375.9	375.9	y
R102	100	20	20	17	1655.1	1655.1	1434.0	3617.1	3617.1	962.0	962.0	y
R102	100	22	20	-	2116.1	1922.6	-	4141.6	3652.2	980.3	729.6	n
R103	25	4	4	5	494.1	494.1	454.4	840.5	840.5	96.4	96.4	y
R103	50	8	8	9	935.3	935.3	772.5	1655.9	1655.9	220.5	220.5	y
R103	100	14	14	-	1544.1	1544.1	-	2937.7	2937.7	393.6	393.6	y
R103	100	17	16	-	1907.3	1436.6	-	3497.9	3116.6	590.5	679.9	n
R104	25	4	4	4	447.1	447.1	416.9	826.7	826.7	129.6	129.6	y
R104	50	7	6	-	842.4	826.2	-	1363.0	1351.4	20.5	25.1	y
R104	100	11	11	-	1449.1	1333.9	-	2461.9	2442.3	12.7	108.4	y
R104	100	15	12	-	1824.3	1275.7	-	3280.5	2512.9	456.2	237.2	n
R105	25	7	6	6	645.7	593.8	530.5	1185.5	1036.8	289.7	193.0	y
R105	50	10	9	9	1159.7	1027.9	891.7	1846.2	1674.9	186.4	147.0	y
R105	100	19	17	-	1850.0	1682.5	-	3421.2	3130.2	570.7	447.6	y
R105	100	20	16	-	2269.7	1630.9	-	3665.6	2948.1	395.9	317.2	n
R106	25	6	6	5	519.5	511.8	467.4	1094.3	1070.8	324.8	309.0	y
R106	50	8	8	8	953.6	953.5	785.2	1607.3	1601.0	153.6	147.5	y
R106	100	17	15	-	1677.4	1489.6	-	3108.2	2845.2	430.8	355.5	y
R106	100	19	14	-	2279.6	1371.1	-	3736.8	2782.1	457.1	410.9	n
R107	25	5	4	4	664.0	515.7	423.0	1046.4	857.9	132.4	92.2	y
R107	50	7	7	7	977.2	977.2	703.2	1539.8	1539.8	62.6	62.6	y
R107	100	12	13	-	1629.7	1544.3	-	2668.6	2692.9	38.9	148.5	y
R107	100	17	12	-	2040.4	1207.2	-	3471.9	2551.3	431.4	344.0	n
R108	25	4	4	4	652.0	441.4	396.2	971.1	823.6	69.1	132.2	y
R108	50	6	6	-	798.0	785.3	-	1312.9	1308.1	149.3	22.8	y
R108	100	13	12	-	1729.1	1296.0	-	2729.5	2487.1	0.4	191.1	y
R108	100	14	12	-	2015.5	1388.8	-	3123.6	2461.6	108.1	72.7	n
R109	25	6	6	5	658.9	658.9	441.3	1104.5	1104.5	195.6	195.6	y
R109	50	10	9	-	1232.8	900.6	-	1883.6	1620.6	150.8	219.9	y
R109	100	18	15	-	2141.8	1490.4	-	3443.4	2792.2	301.6	301.7	y
R109	100	19	16	-	2349.7	1641.0	-	3614.3	2930.4	264.6	289.3	n
R110	25	5	5	4	632.9	531.4	429.5	1016.9	927.9	134.0	146.4	y
R110	50	9	9	7	1275.2	932.5	697.0	1845.1	1698.5	69.8	265.9	y
R110	100	14	13	-	1980.9	1316.1	-	3099.6	2617.3	118.6	301.1	y
R110	100	18	17	-	2233.8	1821.7	-	3517.3	2985.7	283.4	164.0	n
R111	25	4	4	4	599.5	541.7	428.8	886.1	838.3	36.6	46.5	y
R111	50	8	7	-	1085.6	941.7	-	1620.7	1459.2	35.1	17.4	y
R111	100	14	13	-	1860.6	1480.1	-	3136.3	2809.4	275.6	329.2	y
R111	100	17	14	-	2064.7	1211.1	-	3399.1	2676.5	334.4	465.3	n
R112	25	5	5	4	683.7	572.6	393.0	1075.0	963.9	141.2	141.2	y
R112	50	9	7	-	1381.7	803.9	-	1924.6	1380.7	42.9	76.8	y
R112	100	15	12	-	1978.7	1205.8	-	3091.8	2333.6	113.0	127.8	y
R112	100	15	12	-	1988.1	1139.7	-	3101.8	2338.7	113.1	199.0	n

Table 1: Benchmark Results for ECNP and ST Algorithms