Sophisticated and Distributed: The Transportation Domain

– Exploring Emergent Functionality in a Real-World Application –

K. Fischer, N. Kuhn, H. J. Müller, J. P. Müller, M. Pischel

DFKI, Stuhlsatzenhausweg 3, D-66123 Saarbrücken

Abstract. In this paper, we present the MARS multi-agent system. MARS models a society of cooperating transportation companies. Emphasis is placed on how the functionality of the system as a whole - the solution of the global scheduling problem - emerges from local decision-making and problem-solving strategies, and on how variations of these strategies influence the performance of the system. We address three techniques of Distributed Artificial Intelligence (DAI) which are used for tackling the hard problems that occur in this domain, and which together give rise to the emergence of a solution to the global scheduling problem: (1) cooperation among the agents, (2) task decomposition and task allocation, and (3) decentralised planning. Finally, we briefly describe the implementation of the system and provide experimental results which show how different strategies for task decomposition and cooperation influence the behaviour of the system.

1 Introduction

Today, Distributed Artificial Intelligence (DAI) is rightfully regarded as one of the most dynamic branches within AI research. Using DAI techniques such as cooperation [6, 14], negotiation [27, 25, 17], task decomposition, and task allocation [8, 16], is believed to be promising for solving problems which are computationally and structurally complex, highly dynamic, which are characterised by incomplete and inconsistent knowledge, and where information and control are physically distributed. However, verifying this belief by implementing real-world applications remains a challenge for researchers in DAI [22].

In this paper, we explore the usefulness of several DAI techniques for modelling a real-world application domain: a scenario of 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. We evaluate the behaviour of the system as a whole in a straightforward manner: the measure of coherence is how well it can solve the problem of scheduling the orders, i.e. what cost are caused by carrying out the orders. What is extraordinary with our approach is that the companies themselves do not have facilities for planning orders. It is only the trucks which maintain local plans. The actual solution to the global order scheduling problem emerges from the local decision-making of the agents. There are three specific techniques used in order to bring about this emergent functionality:

- Task decomposition and task allocation is done in order to assign orders to appropriate trucks. Different models of task decomposition and task allocation are discussed in section 3.3
- Cooperation (1) among shipping companies and (2) between a company and its trucks helps solving the problem of task decomposition and allocation. Whereas the latter kind of cooperation can be implemented using a simple contract-net-like protocol [8], cooperation among shipping companies has to respect the autonomy of the single companies, and thus requires a full model of negotiation [3, 16].
- Task decomposition and task allocation is guided by local decision criteria. These criteria are derived from the local plans. Moreover, polynomial algorithms are used in order to solve the scheduling problem locally (see subsection 3.1).

In this paper, we do not treat in a detailed manner questions of protocols for cooperation and negotiation (see [16]). Rather, we provide an empirical investigation of how different forms of cooperation and different models of task decomposition lead to different solutions to the global scheduling problem.

The paper is organised as follows: In section 2.1, we present the transportation domain. We show the relevance of the domain, and we argue for choosing a multi-agent approach for modelling the domain. In section 3, three important DAI techniques are addressed: cooperation, task decomposition and allocation, and decentralised planning. The cooperation mechanisms are explained in more detail by means of examples in section 4. Finally, in section 5, a series of experiments is described, and results are are provided and discussed.

2 The MARS Multi-agent Scenario

2.1 The Domain of Application

In a time of constantly growing world-wide economical transparency and interdependency, logistics and the planning of freight transports get more and more important both for economical and ecological reasons. Many of the problems which must be solved in this area, such as the Travelling Salesman and related scheduling problems, are known to be $\mathcal{NP}$-hard. Moreover, not only since *just-in-time* production has come up, planning must be performed under a high degree of uncertainty and incompleteness, and it is highly dynamic. Standard Operations Research approaches (see [18, 7, 26] for an overview) can hardly cope with the dynamics of this domain (see [10] for a discussion of more recent approaches to fleet scheduling). In fact, also in reality these problems are far from being solved. Recent analysis [23] has revealed that more than one out of three trucks in the streets of Europe is driving without carriage, since it is on its way to pick up goods or on its way back home.

2.2 The (D)AI Aspects

Why is it adequate to use AI techniques and more specifically DAI approaches to tackle the transportation problems described above? One reason is the complexity of the scheduling problem, which makes it very attractive for AI research[1]. However there are more pragmatic reasons: *Commonsense knowledge* (e.g. taxonomical, 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:

1. 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.
2. 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.
3. In real business, companies usually solve capacity problems by contacting partners that might be able to perform the problematic tasks. Then the parties negotiate the contract. However, task allocation, contracting, negotiating and performing joint actions are main topics in DAI research.

2.3 The Scenario

The MARS scenario (Modelling a Multi-Agent Scenario for Shipping Companies) [2] implements a group of shipping companies whose goal it is to deliver a set of dynamically given orders, satisfying a set of given time and/or cost constraints[2]. The complexity of the orders may exceed the capacities of a single company. Therefore, cooperation between companies is required in order to achieve the goal in a satisfactory way. The common use of shared resources, e.g. train or ship, requires coordination between the companies. Although each company has a local, primarily self-interested view, cooperation between the shipping companies is necessary in order to achieve reasonable global plans (see section 5).

[1] At this year's International Conference on AI and Applications (CAIA'93), seven out of sixty-one papers dealt with scheduling problems!

[2] MARS has been implemented for UNIX using the rule-based development tool MAGSY [11].

Apart from internal *system agents*, which perform tasks such as the representation and visualisation of the simulation world, the MARS agent society consists of two sorts of *domain agents*, which correspond to the logical entities in the domain: *shipping companies* and *trucks*. Looking upon trucks as agents allows us to delegate problem-solving skills to them (such as route-planning and local plan optimisation). Communication between agents is enabled by direct communication channels.

The *company* agent is responsible for the disposition of the orders that have been confided to him. Thus, it has to allocate the orders to its trucks, while trying to satisfy the constraints provided by the user as well as local optimality criteria. The shipping companies can be regarded as experts for cooperation and cooperative problem solving. They are equipped with additional global knowledge which is needed for cooperating successfully with other companies.

The *truck* agents represent the means of transport of a transportation company. Each truck agent is associated with a particular shipping company from which it receives orders of the form *"Load a goods g_1 at location l_1 and transport it to location l_2"*. Given such an order, the truck agent does the planning of the route ([15], see also section 3.1) according to its geographical knowledge and it will inform the shipping company agent about the deliverance of the goods. Furthermore, it is able to support the shipping company during the disposition phase: The truck reports remaining capacities, planned routes and it is able to estimate the effort (and the effects)[3] that are caused by an order.

3 DAI Techniques in the MARS Scenario

The description of the scenario reveals the autonomy of the agents as a necessary condition for a modelling that reflects the real world situation and that can even support the dispatcher in a real shipping company. In this section we describe several DAI methods that are used within MARS scenario.

The general idea of the solution is based on the paradigm of self-organisation, which is applied to a society of knowledge-based systems: at the beginning, the system is in a state of equilibrium. This equilibrium is disturbed by the distribution of a set of orders among the company agents. This stimulates the truck agents to devise local plans and to inform their company about the cost arising for carrying out an order in their local context. Based on this information the company agent allocates the orders to its trucks. Having done this, the society of agents has constructed a valid plan to deliver the set of initial orders and has reached a state of equilibrium in the sense of [24]. Following the terminology of Steels further on, there are two types of dynamics which can disturb this equilibrium: an internal dynamics which is due to that the trucks reflect on their plans, and an external dynamics caused by incoming orders. Both kinds of dynamic events result in message passing activities in the system. By passing messages among the agents, local disturbances spread out into the local plans

[3] i.e. cost, time, security of transport, ...

of other agents. To reach an equilibrium state again (i.e., another valid global plan for the actual set of orders) the messages are structured into *negotiation protocols*. The dissipative structure described by the different protocols is a cooperative task decomposition process based on the exchange of orders between the agents implementing a distributed and decentralised scheduling algorithm for this application domain.

For the rest of this section we stress three aspects of this approach: the *planning* of the single agents, the forms of *cooperation*, and the mechanisms for *task decomposition*.

3.1 The Distributed Scheduling Approach

The task of delivering several orders is basically a scheduling problem. What makes it even harder is the two-dimensionality of task decomposition resulting from the special domain. The goods to be transported can be distributed to several means of transport (truck, train, ship, plane), and the route between two cities on a road map can be splitted up into sub-routes which can be taken at different times using different conveyances. Due to the combinatorial explosion resulting from this, it is often impossible to devise a globally optimal plan. We tackle this problem by computing locally good solutions for each agent. Thus, we hope to get an reasonable overall solution for the given problem. This solution is further optimised by cooperation between the problem solving agents. Here two agents only agree to a solution if none of them gets a decrease in his local utility, and if at least one of them has an increase in utility by the deal. This process of negotiation leads to pareto-optimal solutions.

The problem of allocating a set of orders to a set of trucks is an $\mathcal{NP}$-hard problem. This can be shown by reducing the Rural Postman Problem (cf. [12]) which is known to be $\mathcal{NP}$-complete to the decision problem which corresponds to the order allocation problem. The Rural Postman Problem is defined as follows:

> **INSTANCE**: Graph $G = (V, E)$, length $l(e) \in Z_0^+$ for each $e \in E$, subset $E' \subseteq E$, bound $B \in Z^+$.
> **QUESTION**: Is there a circuit in G that includes each edge in E' and that has total length of at most B?

The Rural Postman Problem remains $\mathcal{NP}$-complete even if $l(e) = 1$ for all $e \in E$. The relationship between the modified problem and the task allocation problem becomes clear when we consider a company agent who only possesses a single truck, and who has for each edge $e_i = (v_{i_1}, v_{i_2}) \in E'$ an order o_i from location v_{i_1} to location v_{i_2} which needs the whole capacity of the truck. Then, there exists a circuit in G including each edge in E' which has a total length of at most B iff there exists a route for the truck which is at most of length B.
This reduction shows that both the task allocation problem and route planning of the trucks for a set of orders are $\mathcal{NP}$-hard problems (see [9] for the detailed proof). Thus, in order to keep them manageable in a computer implementation, heuristic algorithms have to be applied which do not guarantee optimality, but which in most cases provide good results in a reasonable amount of time.

The solution for the routing of the trucks is built up incrementally. If a truck has a plan to visit the locations $l_0, \ldots, l_n, l_0$ in order to deliver orders $o_1, \ldots, o_m$ and has to add a new order o from starting location s to target location t it inserts s and t into his plan such that the detour is minimised. This does not mean that s and t need not to be successive locations in the new plan. Rather, enough capacity has to remain to deliver the orders $o_{i+1}, \ldots, o_{j+1}$ together with order o. If s is inserted after l_i and t is inserted after l_j with $j > i$, the detour is computed by the formula if $j > i + 1$

$$detour = \begin{cases} dist(l_i, s) + dist(s, l_{i+1}) + dist(l_j, t) + dist(t, l_{j+1}) \\ \quad - (dist(l_i, l_{i+1}) + dist(l_j, l_{j+1})) & \text{if } j > i + 1 \\ dist(l_i, s) + dist(s, t) + dist(t, l_{i+1}) - dist(l_i, l_{i+1}) & \text{if } j = i + 1 \end{cases}$$

The time needed by this algorithm to insert one order into a delivery plan containing n locations to visit is bound by $\mathcal{O}(n^2)$. The number of locations in a plan depends linearly on the number of orders a truck has got. Therefore, by this algorithm the time needed for the planning of a route for m orders is bound by $\mathcal{O}(m^3)$. The allocation of the orders to trucks by the company agent is done using the contract net protocol: the company agent offers an order to some eligible trucks who evaluate their plans and inform the company agent about that. Based on this information it chooses the best offer and allocates the order to that truck. It follows from the above considerations that this algorithm for the task allocation within a shipping company is also of time complexity $\mathcal{O}(m^3)$.

Another interesting question is what kind of algorithm is implemented by this procedure. However, this depends on the company agent's strategy for processing the m contract nets to allocate the orders: if it always completes a protocol before it initiates the one for the next order, the procedure described above implements a *greedy* algorithm for the task allocation process. An alternative to that strategy is that the company agent can maintain several contract net protocols for different orders at the same time and can base its decision on the information that it receives by all the bids in the different protocols. Thus, the strategies of the agent allow to incorporate a broad range of heuristics into the allocation process. However, the second alternative requires more sophisticated planning mechanisms, since there is high uncertainty in the plans. For instance, if an order is open for allocation and the truck is asked to give a bid for a second order, *shall it believe that it is allocated the first one or not?* The different assumptions lead to different bids. Therefore, in our system, we preferred the former strategy.

The outcome of the task allocation can be improved by the cooperation between the different companies. A truck has some knowledge to find out weak points in his plan, e.g. empty rides or orders that lead to large detours. By telling its company agents about this he can initiate initiate different cooperation mechanisms by which the plans can be improved due to an exchange of orders between the agents.

3.2 Cooperation Settings

In the previous subsection, the local algorithms used by the truck agents were described. In order to coordinate the local activities, and in order to achieve a coherent global behaviour of the system, the agents have to cooperate. In this subsection, we define three basic cooperation settings, namely *vertical cooperation*, *horizontal cooperation*, and *enhanced cooperation*, which we implemented in the MARS domain.

Vertical Cooperation (VC) *Vertical cooperation* describes the process of task decomposition and task allocation between a shipping company and its trucks. This relation is hierarchical; the trucks are obliged to give their best to carry out orders given by their company, and they are obliged to provide the company with important information upon request. We use a slight variation of the contract net [8] in order to model this kind of interaction.

More precisely the procedure is as follows: The shipping company partitions orders into cargos that can be transported by single trucks. These cargos are offered to the trucks of the company. The bid of a truck describes the costs that will arise for it when carrying out this order. According to local decision functions and the incoming bids of the trucks the company allocates the subtasks to some of the trucks. This basic solution closely corresponds to the *centralised model of task decomposition* which is described in section 3.3. An advanced release of the protocol allows the trucks to bid also for only a part of an order; this is one step towards a decentralisation of task decomposition.

Horizontal Cooperation (HC) *Horizontal cooperation* means cooperation among a group of autonomous shipping companies. Companies can exchange orders and information about free loading capacity. This exchange, however, is not performed hierarchically; rather, it reveals all aspects of conflict, competition and cooperation which we find in human societies. The underlying model of cooperation has been described in [15, 19]. Agents are able to recognise so-called *patterns of cooperation* which describe situations where certain types of cooperation are both applicable and suitable. The execution of these patterns is described by cooperation protocols based on speech-acts. Agreements between companies (for example as to the price for delivering an order) are reached by negotiation.

If we try to describe the HC setting in terms of task decomposition, it implements a decentralised model of task decomposition: the transportation companies negotiate on how they might decompose the transportation orders. In a certain sense, task allocation is decentralised, too, since the companies generate proposals how to allocate the orders among their group.

Enhanced Cooperation (EC) In the *enhanced cooperation* setting, the relationship among the companies is the same as in HC in a sense that they can exchange orders and information about free capacities. What is new is that it extends the vertical cooperation. Trucks are assigned the ability to reflect on their plans. If a truck agent realises that it has a poor plan, it can cancel the contract made with its company for a specific order, i.e. it may give back its part of that order to its company. In this case, the order is offered again to the agent society. This procedure makes sure that the order will be executed in any case: if the society does not find a better way to process the order, the order automatically falls back to the truck (here, of course, infinite loops and cycles must be avoided by checking appropriate conditions). Thus, there is some risk in following the EC strategy: if no other company can do the task better, the truck has gained nothing, but only lost time.

EC can be regarded as a decentralisation of the hierarchical task allocation implemented by the VC setting. According to EC, the trucks do not always and unconditionally have to accept a task allocation proposed by their company, but can undo this decision in certain cases.

3.3 Task Decomposition and Task Allocation in MARS

The development of multi-agent systems like MARS is motivated by the goal of providing a special purpose system that is able to accomplish a certain set of tasks. In the transportation domain these tasks are the transportation orders given by the users. An important question for the modelling of this domain is how these orders are allocated to the local resources of the companies, namely to their trucks.

In section 2.1, it was already mentioned that the complexity of an order may exceed the capacity of a single shipping company. Therefore, the handling of an order consists of two phases: the *task decomposition phase* and the *task allocation phase*. During the former phase, a decomposition of a task (or an order) into a set of subtasks is computed recursively, until every subtask is small enough to be directly given to some truck. In the latter phase, particular agents have to be determined who will commit themselves to accomplishing the task. In general, each of these phases can be implemented in either a centralised or a decentralised manner, yielding four possible methods for task handling.

Within our current implementation of the scenario at least three of these possible approaches may be found on different levels of the task handling process: the *centralised task decomposition model* (contract-net model), the *decentralised task decomposition model*, and the *completely decentralised model*. The first one, which is characterised by centralised decomposition and centralised allocation, is mainly used between a shipping company and its truck. It is implemented according to a contract net protocol (cf. [8]) as described in chapter 3.1.

In general, the division of a task into a set of subtasks will be done by some heuristic that is available to the decomposition process. But as we are going to deal with open systems in the sense of [13], the heuristics *cannot* take care of the situation of all the agents that are currently part of the system. Moreover, in [16] we presented an example that the decomposition using a heuristic that is not adequate at the moment may fail to compute a solution to a decomposition problem, although an appropriate decomposition is obvious from a more global point of view. To overcome this, we proposed a decentralisation of the task decomposition process: Instead of offering subtasks to the trucks, the whole transportation order is offered to them. Their bids now consist of two parts: firstly, a part of the order they are able to accomplish, and secondly, an estimation of the cost associated with this partial order. The shipping company collects the proposals given by the trucks and uses this information to allocate the subtasks.

In the completely decentralised model, task decomposition as well as task allocation phase are implemented as decentralised processes. It is used e.g. to describe cooperation between autonomous shipping companies, e.g. for dealing with a transportation order that exceeds the capacities of a single company. A company announces its interest in such an order to other companies and specifies a possible subtask of the order she would like to accomplish. The other companies may respond in the same manner, i.e. by announcing their interest in another possible subtask of this order, or they may respond by proposing a modification of the actual state of task decomposition and task allocation. The central technique for achieving a task decomposition and a task allocation that is commonly accepted is the negotiation of the different proposals among the companies that are involved in this process. This process was described in more detail in [16].

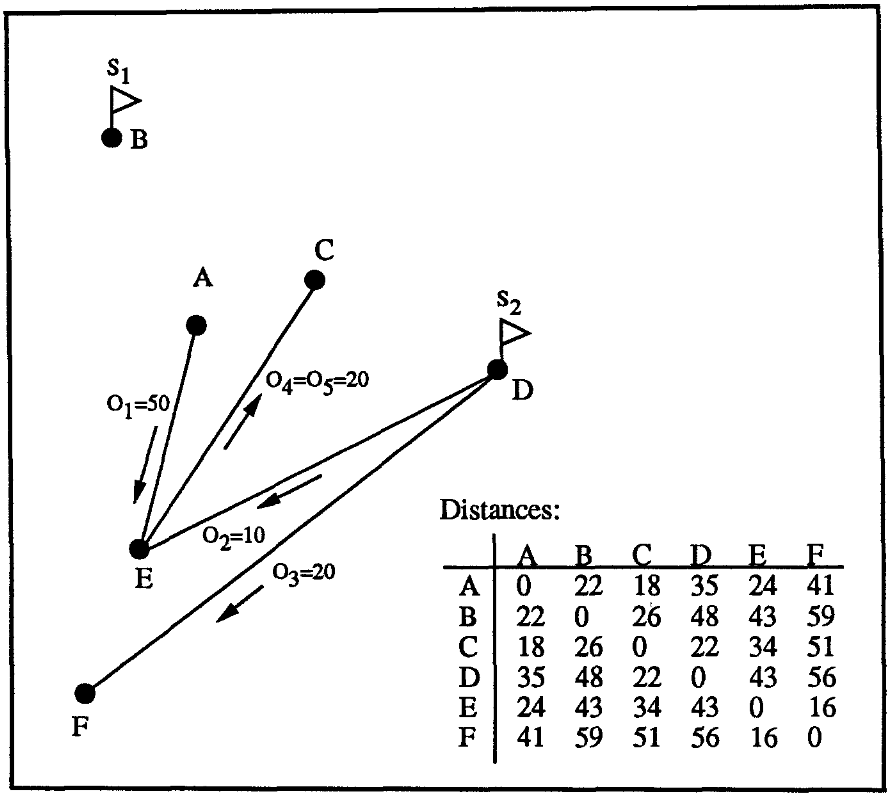
In the following section, we present an example and discuss how these different methods can influence the final solution.

4 Cooperation Settings: an Example

In this section, by means of the example shown in figure 1, we explain how vertical (VC), horizontal (HC), and enhanced (EC) cooperation work in practice: we consider two shipping companies, S_1 and S_2. S_1 resides in city B and S_2 in city D. Each shipping company controls a set of two trucks each of which has a capacity of 40 units. The map contains 6 cities, named A, B, C, D, E, and F. The distances between the cities are given by the table shown in figure 1. Let us assume that the system receives a set of orders in the following sequence:

O_1 : 50 units from city A to city E; offered to S_1
O_2 : 10 units from city D to city E; offered to S_1
O_3 : 20 units from city D to city F; offered to S_2
O_4 : 20 units from city E to city C; offered to S_1
O_5 : 20 units from city E to city C; offered to S_1

Example 1: Vertical Cooperation: In this first example, the shipping companies try to solve the problems on their own. The solution in this example is produced using purely vertical cooperation (VC) between the companies and their trucks. In doing so, truck $T_1^{S_1}$ of shipping company S_1 starts from city B to city A to collect 40 units of order O_1, it then goes down to city E to drop 40 units of order O_1. It collects the orders O_4 and O_5 to bring them to city C. Another truck $T_2^{S_1}$



Distances:

	A	B	C	D	E	F
A	0	22	18	35	24	41
B	22	0	26	48	43	59
C	18	26	0	22	34	51
D	35	48	22	0	43	56
E	24	43	34	43	0	16
F	41	59	51	56	16	0

Fig. 1. Example Scenario.

starts from city B to city A to collect the 10 units which were left by truck $T_1^{S_1}$. It then heads for city D to collect O_2 and goes to city E to drop the 20 units of orders O_1 and order O_2. Truck $T_1^{S_2}$ transports order O_3 from city D to city F.

Let $l(T)$ denote the length of the plan which was executed by truck T. $l(T)$ specifies the costs which were necessary for truck T to fulfil its task. We get $l(T_1^{S_1}) = 22 + 24 + 34 = 80$, $l(T_2^{S_1}) = 22 + 35 + 43 = 100$, and $l(T_1^{S_2}) = 56$. Therefore, total costs of 236 were necessary to fulfil the whole set of orders. It it easy to see that, from a global point of view, this solution is not very good. But in fact, it is an optimal solution from the local point of view of each shipping company.

Example 2: Horizontal Cooperation In this example everything remains the same as in example 1 except that the shipping companies do cooperate by offering orders to each other (see section 3.2). In our example, an offer made by another company is accepted if it is better than the offers made by the own trucks. If the shipping companies do behave like this, the solution for truck $T_1^{S_1}$ is the same as in example 1. Also, truck $T_2^{S_1}$ still starts from city B to city A to collect the 10 units of order O_1 which were left by $T_1^{S_1}$. This time, however, it heads directly to city E, because order O_2 is passed from shipping company S_1 to shipping company S_2.

In this example, $l(T_1^{S_1}) = 22 + 24 + 34 = 80$, $l(T_2^{S_1}) = 22 + 24 = 46$ and $l(T_1^{S_2}) = 43 + 16 = 59$ holds. Therefore, total costs of 185 result from fulfilling the whole set of orders.

Example 3: Enhanced Cooperation The last example shows the effect of enhanced cooperation (EC). Here, the behaviour of the trucks is altered with respect to example 2. In example 2, $T_2^{S_1}$ has a very poor plan: It starts from city B to city A to collect 10 units of order O_1. It then goes to city E using just a quarter of its capacity. In this example now trucks are able to reflect on the quality of their plans. This means that $T_2^{S_1}$ realises before it starts from city B that its plan is poor. It gives back its part of order O_1 to shipping company S_1. Because trucks will only start going when there is no new order announced for some amount of time, all orders are already known in the system. This means that truck $T_1^{S_2}$ already has its local plan. At this time $T_2^{S_1}$ is no longer the best one to transport the 10 units of order O_1. Due to cooperation between the shipping companies the 10 units of order O_1 will be passed over to truck $T_1^{S_2}$.

The result is that $T_1^{S_1}$ executes the same plan as in example 2. Truck $T_1^{S_2}$ picks up order O_2 and O_3 in D and starts for city A, where it collects the remaining 10 units of order O_1. It goes down to city E and unloads the 20 units of orders O_1 and O_2. Finally, it visits city F to drop order O_3. $T_2^{S_1}$ has no longer a local plan and therefore stays in city B. This time, $l(T_1^{S_1}) = 22+24+34 = 80$ and $l(T_1^{S_2}) = 35+24+16 = 75$ holds and therefore total costs of 155 result from fulfilling the whole set of orders. With respect to the first example 34 % of the costs could be saved.

The above examples can be demonstrated by our implementation of a multi-agent system for the transportation domain. The question is whether these nice problem-solving strategies will work in practice, too. Section 5 describes a series of experiments, where examples of 50 and 400 orders are simulated and evaluated.

5 Experimental Results

In this section, we describe a number of experiments we have run with the MARS system. As mentioned before, the main question is how different kinds of cooperation bring about different solutions to the task decomposition problem, and thus, lead to different behaviour of the system as a whole. In the previous section, three interesting cooperation settings have been discussed, namely VC, HC, and EC. We have seen how these settings correspond to the models of task decomposition and allocation presented in section 3.3. In the following, we will evaluate the different settings by means of a series of experiments.

5.1 Description of the Experiment

In our experiment, three transportation companies with their trucks had to carry out a number of orders. The following parameters could be varied:

- The number of trucks per company varied from 1 to 20.
- The number of orders varied: we tested the system with order loads of 50 and of 400 orders. Moreover, in order to ensure general validity of the results, each experiment was repeated 10 times with randomly generated loads.
- Each experiment was carried out for the VC, the HC, and the EC setting.

The experiments were run on a network of SUN SPARC stations. For the biggest experiment, the agents were run on eight SUN workstations in parallel. What was measured in the experiments were the costs caused by the trucks for carrying out the orders, as well as the average percentage of capacity load of the trucks. The costs were computed by a simple cost model: the costs caused by a truck are proportional to the distance covered while carrying out the orders.

5.2 Results

Figure 2 shows the result of *experiment 1* (which actually has been a series of experiments), which was run with an order load of 50 orders. Figure (2.1) displays the cost caused for solving the scheduling problem. Figure (2.2) shows the average capacity load of the trucks. The x-axis denotes the number of trucks per company. The y-axis is labelled with the cost and the load percentage, respectively. The curves show the behaviour of the system for the three cooperation

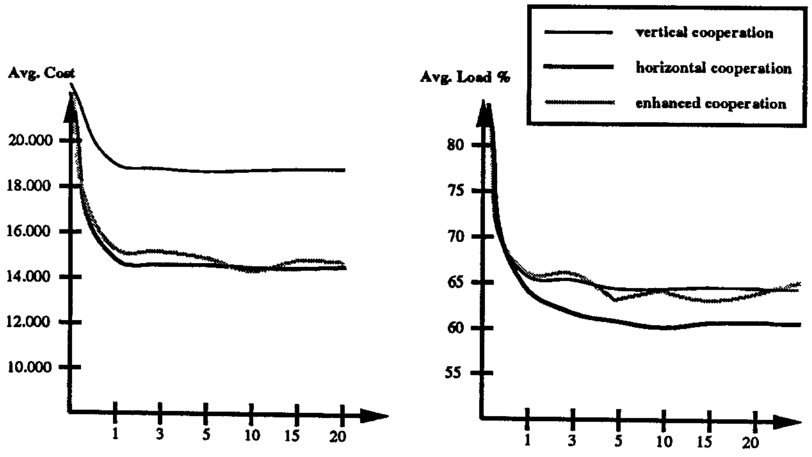


Fig. 2. Experiment 1

settings defined in section 4.

Experiment 2 is a reiteration of the former experiments with a bigger order load. This time, the society of transportation companies had to deliver a set of 400 orders. Figure 3 reveals the results for this scenario. Again, in figure (3.1), the costs are displayed, whereas in figure (3.2), the average capacity load of the trucks is shown.

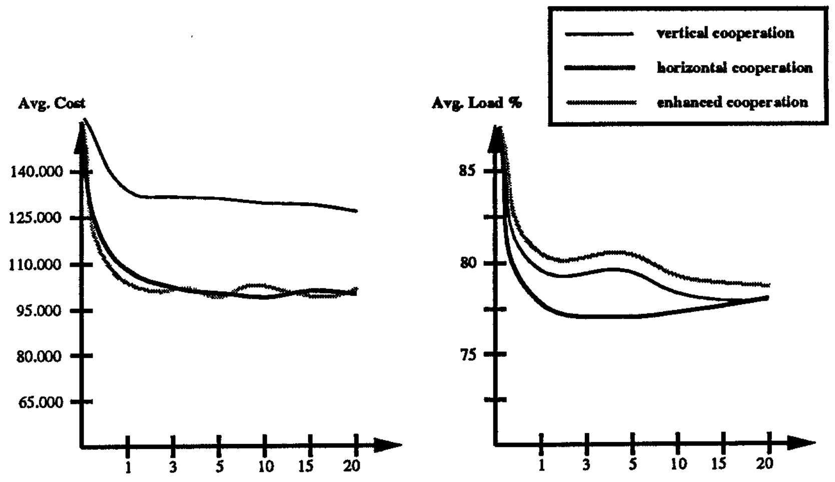


Fig. 3. Experiment 2

In figure 4, some statistical indices characterising the solutions found by VC and HC for experiment one with a number of five trucks per company are shown. Note, that the standard deviation can be regarded as a measure for the stability of the different forms of cooperation, i.e. for how much the computed solutions depend on variations of the input.

Coop. Setting	Cost_{avg}	Cost_{max}	Cost_{min}	Standard Deviation
VC	16,859	31,604	7,706	5677.6
HC	13,365	21,200	7,210	3560.0

Fig. 4. Statistical Indices for Experiment One, Five Trucks per Company

Finally, figure 5 displays a comparison of the estimated number of messages sent by the agents using the VC, HC, and EC settings for experiment 1. We will use this as a measure for the run-time efficiency. An interpretation of the experimental data is provided in the following subsection.

5.3 Discussion

Let us now have a closer look at the basic results of the experiments described in the previous section.

Quality of the Solutions: If we compare the costs in figures (2.1) and (3.1), the most obvious result is that in both experiments, the average cost can be reduced

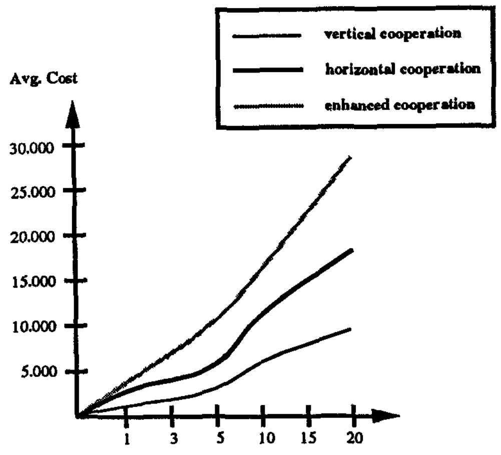


Fig. 5. Message Statistics for Experiment 1

considerably by introducing horizontal cooperation. Moreover, as problems get more complex, using horizontal cooperation pays off more and more: if we look at the case of 20 trucks, using the HC setting in the experiment 1 reduces the cost by about 21% compared to the VC solution, whereas in the bigger experiment 2, cost are reduced by 28%. On the other hand, the EC strategy does not seem to yield considerably better results than HC. This is discussed below.

As regards the average capacity load of the trucks, there are three remarkable points: Firstly, the VC strategy yields a slightly better load percentage than the HC strategy. Secondly, with the number of trucks increasing, the load percentage tends to decrease. Thirdly, the EC strategy yields the best capacity loads, on an average. What seems to be a bit confusing at a first glance is that the poor VC strategy results in better capacity loads than the sophisticated HC. Intuitively, we could think that lower costs and a better utilisation of capacity go hand in hand. In fact, the relationship is not as straightforward as it seems to be. A truck can go long ways - as long as it is fully loaded, the capacity load will be ok. Therefore, a single truck can reach a high capacity load by combining orders in a clever manner - this, however, does not automatically imply that the plan causes little costs. This is the main reason why VC results in a good capacity usage. On the other hand, EC combines the low cost of HC with the good capacity load of VC.

Runtime Efficiency: Run-time efficiency was measured by the number of messages which were sent by agents to other agents. Here, the results clearly confirmed our assumptions: a higher communication overhead is the price to pay for the better solutions obtained by using the EC and HC settings. However, this overhead drastically depends both on the internal decision criteria of the companies and on the negotiation protocols used. In the example, each company offers each order to any acquainted other company. By using more intelligent (heuristic) criteria for partner selection, the amount of messages can be drastically reduced. In our experiments, we could not state considerable differences in run-time between VC and HC, whereas EC is considerably more time-consuming.

Stability and Convergence: Obviously, the solution obtained by HC converges against a stable local state very quickly. For example, in experiment one, this stable state is reached with only five trucks, and the cost does not change when more trucks are used. Moreover, the behaviour of HC systems seems to be less sensitive to changing input data: figure 4 shows that both the difference between minimal and maximal cost values and the standard deviation is much smaller if HC is used than if VC is used. This, however, satisfies the evaluation criteria of graceful degradation and flexibility proposed by [24]. EC is both the least stable and the least predictable strategy, since trucks decide to drop their plans based solely on local quality criteria. EC produces cost comparable to HC, a capacity usage comparable to VC, but a very high computation and communication overhead. Thus, the experience is confirmed that modelling in the small requires one to be very careful, since locally reasonable decisions lead to factually no global improvement, or even to a decrease of the system behaviour, unless the design and the parameter setting are done in a very cautious way. Finding better decision criteria for EC is a subject of our future work.

Optimal Solution Up to now, we have not said anything about optimal solutions to the scheduling problems solved by our system in the two experiments. Of course, it would be fine to know about the real optima. However, since the orders arrive asynchronously and are scheduled dynamically, and since message deliverance times have to be taken into consideration, such a reference solution cannot be obtained by using static OR methods like Branch and Bound algorithms (see e.g. [5]). Up to now, and as far as we know, no practicable approaches towards solving this problem exist. This corresponds to the fact that evaluation of large distributed systems is a big problem in general.

6 Conclusion

The transportation domain was introduced as a multi agent application, The use of specific DAI techniques for the cooperation between the agents and for task decomposition and allocation were described. By means of a series of experiments, first results as regards the influence of local criteria to the emerging functionality of the system as a whole were reported.

Thus, there are three main contributions of the paper: Firstly, we showed how different models and mechanisms of task decomposition, task allocation, and of cooperation can be used in order to model a real-world application and in order to solve hard real-world problems such as the scheduling problem by applying the paradigm of emergent functionality. Secondly, our empirical results confirm that the multi-agent approach is a *suitable* approach for modelling and solving this kind of problems. Thirdly, our experiments can be useful in a more practical sense: cooperation among transportation companies appears to be an important subject when it comes to solve today's world-wide traffic problems. The reduction of costs which has been observed in our experiments can turn out to be a strong and convincing argument in favour of this type of cooperation.

References

1. 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.
2. M. Buchheit, N. Kuhn, J. P. Müller, and M. Pischel. MARS: Modelling a Multiagent Scenario for Shipping Companies. In *Proceedings of the European Simulation Symposium (ESS-92)*, Dresden, 1992. Society for Computer Simulation (SCS).
3. S. Bussmann and H. J. Müller. A Negotiation Framework for Cooperating Agents. In *Proc. of the 2nd Workshop on Cooperating Knowledge Based Systems*. Keele, GB, September 1992.
4. P. Bagchi and B. Nag. Dynamic Vehicle Scheduling: An Expert System Approach. *Journal of Physical Distribution and Logistics Management*, 21(2), 1991.
5. M. Desrochers, J. Desrosiers, and M. Solomon. A new optimisation algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2), 1992.
6. E. H. Durfee and V. R. Lesser. Negotiating task decomposition and allocation using partial global planning. In *Distributed Artificial Intelligence, Volume II*, pages 229–244, San Mateo, CA, 1989. Morgan Kaufmann Publishers, Inc.
7. W. Domschke. *Logistik (Bd.2): Rundreisen und Touren.* Oldenbourg-Verlag, München - Wien, 3rd edition, 1990.
8. R. Davis and R.G. Smith. Negotiation as a methaphor for distributed problem solving. *Artificail Intelligence*, 20:63 - 109, 1983.
9. K. Fischer and N. Kuhn. A DAI Approach to Modelling the Transportation Domain. Technical Report RR-93-25, DFKI, 1993.
10. K. Fischer, N. Kuhn, and J. P. Müller. Distributed, knowledge-based, reactive scheduling in the transportation domain. In *Proc. of the Tenth IEEE Conference on Artificial Intelligence and Applications*, San Antonio, Texas, March 1994.
11. K. Fischer and H. M. Windisch. MAGSY- Ein regelbasiertes Multi-Agentensystem. In H. J. Müller, editor, *KI1/92, Themenheft Verteilte KI.* FBO-Verlag, 1992.
12. M. R. Garey and D. S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.
13. C. Hewitt. Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence*, 47:79–106, 1991.
14. N. R. Jennings. *Joint Intentions as a Model of Multi-Agent Cooperation.* PhD thesis, Queen Mary and Westfield College, London, August 1992.
15. 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.
16. N. Kuhn, H. J. Müller, and J. P. Müller. Task decomposition in dynamic agent societies. In *Proceedings of the International Symposium on Autonomous Decentralised Systems (ISADS-93)*, Tokyo, Japan, 1993. IEEE Computer Society Press.
17. S. E. Lander and V. R. Lesser. Understanding the role of negotiation in distributed search among heterogeneous agents. In *Proc. of the 12th International Workshop on*

Distributed Artificial Intelligence, pages 249–262, Hidden Valley, Pennsylvania, May 1993.

18. H. Müller-Merbach. *Operations Research*. Verlag Franz Vahlen, München, 3rd edition, 1973.
19. J. P. Müller and M. Pischel. The Agent Architecture INTERRAP: Concept and Application. Technical Report RR-93-26, German Artificial Intelligence Research Center (DFKI), Saarbrücken, June 1993.
20. J. P. Müller and M. Pischel. Integrating agent interaction into a planner-reactor architecture. In *Proc. of the 13th International Workshop on Distributed Artificial Intelligence*, Seattle, WA, USA, July 1994.
21. J. P. Müller and M. Pischel. Modelling interacting agents in dynamic environments. In *Proc. of the European Conference on Artificial Intelligence (ECAI94)*, pages 709–713. John Wiley and Sons, August 1994.
22. H. Van Dyke Parunak. Industrial applications of multi-agent systems. In *Proc. of INFAUTOM-93*, Toulouse, February 1993. Association Colloque SUP'AERO.
23. R. Rittmann. Die Macht der Trucks. *Bild der Wissenschaft*, 9:112–114, 1991.
24. L. Steels. Cooperation between distributed agents through self-organisation. In Y. Demazeau and J.-P. Müller, editors, *Decentralised A.I.*, pages 175–196. North-Holland, 1990.
25. K. P. Sycara. Multiagent compromise via negotiation. In L. Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence, Volume II*, pages 119–137. Morgan Kaufmann, San Mateo, California, 1989.
26. H.-J. Zimmermann. *Methoden und Modelle des Operations Research*. Vieweg-Verlag, Braunschweig - Wiesbaden, 2nd edition, 1992.
27. G. Zlotkin and J. S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In *Proc. of the Eleventh IJCAI*, pages 912–917. Detroit, Michigan, August 1989.