A MULTI-AGENT MODEL BASED ON RESIDUAL RESOURCES

Gaël Hette, Sylvia Estivie, Emmanuel Adam and René Mandiau
Univ Lille Nord de France, F-59000 Lille, France
UVHC, LAMIH, F-59313 Valenciennes, France
CNRS, FRE 3304, F-59313 Valenciennes, France

Keywords: Multi-agent system, Task allocation, Residual resources, Resources pooling.

Abstract: We propose a model to solve the disturbances which could occur during the execution of agents task schedules in a dynamic multi-agent system. Our model is based on pooling residual resources to reallocate tasks in case of incoming tasks or disturbances during the execution of the system. In our context, initial task schedules of agents are defined such that, during some given time windows, agents can perform additional tasks with their residual resources. In this paper, we propose a model for this problem. In order to validate this model, we also propose a first resolution method attempt and an experimental study of this framework.

1 INTRODUCTION

In this paper we consider the case of task reallocation over a multi-agent system, where each agent has non shareable resources to achieve tasks. These resources are still available during the execution of such system instances, nevertheless these resources are not used continuously by the agents. As a consequence, we focus on the context of a resource pooling system, where agents can achieve tasks for others by using their own allocated resources for time periods during which these resources are not needed by them. We denote these kind of resources as *residual resources*. We propose to use multi-agent task allocation techniques (Chevaleyre et al., 2006) to reallocate tasks during the execution of a defined schedule. We focus on a special case of task reallocation problem for which global schedule modification is not allowed. We aim to propose a framework to deal with incoming tasks or task failures by pooling *residual resources*. Considering an agent facing a loss of its allocated resource, our approach is based on a model and a set of methods to find corresponding available resources (according to other agents schedule) and insert additional tasks into another agent's schedule, in order to solve this problem. We assume that resource allocation and task schedule are such that agents could perform additional tasks with their *residual resources* during given time windows. So, the main idea of this work is to consider that agents pool their *residual resources* in order to enable task reallocation with a view to execute tasks for other agents. We focus on the case of task reallocation during the execution of a defined schedule. The main constraints of our problem are that any modification of the initial resource allocation is forbidden and no major modification of agents schedule is allowed. As a matter of fact, we propose in this paper a framework for this problem. In order to validate this model, we also propose a first resolution method attempt and an experimental study of this framework. Section 2 details our problem of adaptive task scheduling within the context of a resource pooling system and gives a brief state of the art. In Section 4 we propose a computational model and procedures to deal with the problem of adaptive task scheduling. In order to validate our model, Section 5 concludes with an experimental study.

2 OUR PROBLEM

We consider a multi-agent system with an initial resource allocation amongst agents computed in order to accomplish an associated task schedule. Each task of this schedule has to be performed by a specific agent within a given time window, and implies the use of some of its allocated resources. This problem includes situated agents moving in an environment, agents and their allocated resources share the same locations. Thus when agents move from a location to another, they are accompanied by their allocated resources. Furthermore, the agent schedules define a set of tasks to execute under location constraints. In addition, we assume that the initial resource allocation and task schedule are such that some allocated

Hette G., Estivie S., Adam E. and Mandiau R..
A MULTI-AGENT MODEL BASED ON RESIDUAL RESOURCES.
DOI: 10.5220/0003698600850090
In *Proceedings of the 4th International Conference on Agents and Artificial Intelligence* (ICAART-2012), pages 85-90
ISBN: 978-989-8425-96-6

resources are not fully used by agents and agents can perform additional tasks during given finite time windows. Moreover, the problem occurs in a dynamic environment where tasks appear during the execution. These dynamic tasks are new task orders or initially scheduled tasks that agents fail to perform due to the system dynamics. So, the main idea of this work is to consider that agents pool their *residual resources* in order to execute additional tasks for others and to be able to delegate the achievement of tasks to other agents. As we focus on the case of task reallocation during the execution of a defined schedule, the main constraint of our problem is that we cannot modify the resource allocation. We must rather use this allocation to assign the execution of additional tasks to agents based upon their resources and task schedule.

Background. We present the two most similar problems to our, as they include situated agents and their solutions imply schedule such that some allocated resources are unused by agents during given time periods. The *Vehicle Routing Problem with Time Windows* (*VRPTW*) (Laporte and Osman, 1995) has been widely studied in both operational research (Solomon, 1987; Cordeau et al., 2001) and multi-agent system (Fischer et al., 1999; Barbucha and Jedrzejowicz, 2009). The *VRPTW* concerns the design of least cost routes over a vehicles fleet, in order to deliver goods from a central depot to a set of geographically scattered points during given time windows. The Dynamic Pickup and Delivery Problem (*DPDP*) (Berbeglia et al., 2010) is another vehicle routing problem of interest. This problem concerns the transport of people or goods from an origin to a destination. Now we give an overview of the different approaches for these problems. These approaches have led us in our study even if they differ in terms of constraints on the modification of the resource allocation and task schedule of agents. As a matter of fact, they were not reusable for our work as they imply major modification that are not allowed in our case. In the multi-agent literature, the use of market based allocation procedures is the main approach adopted to solve VRPTW. These procedures are often based on Contract Net Protocol (Davis and Smith, 1983; Fischer et al., 1999) or on Vickrey auction model (Vickrey, 1961; Gorodetski et al., 2003), but only few propose an approach for dynamic task reallocation. Fischer et al. (Fischer et al., 1999) presented a multi-agent simulation framework where agents interactions are driven by negotiation protocols. Here, the problem is similar to our in the sense of resource pooling and as it proposes a procedure between companies to buy and sell free loading capacities to overcome disturbances during the execution. Outcomes of this procedure are interesting because they can lead to local modifications of the schedule. But these outcomes differs from ours, since they lead to global modification of agents schedule in the worst cases and thus give no guarantee of minimum changes. Studies concerning the *DPDP* are based on the same two steps approach. In the first step, the problem is resolved in its static version according to requests known before execution. The second step occurs during execution, when a new request is known. During this step, heuristic methods such as insertion methods, interchange moves and deletion heuristics are performed. As a result, the previous solution of the schedule is updated with minor or global schedule modification. All these approaches concern problems for which a global modification of the schedule is allowed. But we consider a problem where such global modification is forbidden. So, in the next section we propose a model to enable dynamic task reallocation under temporal and location constraints without global modification of the schedule.

3 COMPUTATIONAL MODEL

We consider a multi-agent system populated by n situated agents $a_1, ..., a_n$ and m resources. The agents and their allocated resources share the same location. Thus when agents move from a location to another, they are always accompanied by their allocated resources. Each task must be achieve at a given location of the environment. The knowledges of an agent a_i are represented by a tuple $< \mathcal{E}_i, \mathcal{S}_i, \mathcal{R}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{P}_i >$, with :

- $\mathcal{E}$: environment representation,
- $\mathcal{S}_i$: the task schedule of agent a_i,
- $\mathcal{R}_i$: finite set of a_i allocated resources,
- $\mathcal{O}_i$: finite set of a_i *task offers*,
- $\mathcal{D}_i$: finite set of a_i *task demands*,
- $\mathcal{P}_i$: a_i preferences to select the best *task offer* among others.

Environment and Communication. The environment of the agents is defined by a spatial grid divided in small regions. This spatial grid is modeled by an undirected graph $\mathcal{E} = (V, A)$, where each vertex $v_n \in V$ represents a small region of the grid. The edge set is defined as $A = \{(i, j) | i, j \in V, i \neq j\}$. Each pair of adjacent regions are represented by a valued edge $(i, j) \in A$. The weight of each edge (i, j) has a non-negative value $t_{i,j}$ and denotes the time travel of an agent from region i to j. In the context of our study, we assume that there is no communication constraint

between agents. In addition, we consider that agent are able to perform *point to point* and *broadcast* communication protocols.

Task Schedule. Let $\mathcal{S}_i = \{t_{i1}, ..., t_{in}\}$ denote the task schedule of an agent a_i. Each primitive task $t_{im} \in \mathcal{S}_i$ is defined by a tuple $t_{im} =< R_{im}, V_{im}, SDate_{im}, EDate_{im}, Loc_{im} >$, where :

- R_{im} : type of resource needed to achieve t_{im},
- V_{im} : amount of resource needed to achieve t_{im},
- $SDate_{im}$: start date of t_{im} execution,
- $EDate_{im}$: end date of t_{im} execution,
- Loc_{im} : location constraint on t_{im} performance.

The constraint Loc_{im}, defines the location in the environment where the task t_{im} must be performed by an agent a_i. Thus, a_i must perform the task t_{im} in regard with the location constraint specified by Loc_{im}. In addition, $\mathcal{S}_i = \{t_{i1}, ..., t_{in}\}$ is defined as a strictly ordered set, such that:

$$SDate_{i1} < EDate_{i1} < SDate_{i2} < ... < EDate_{in}$$

Agents schedule are defined such that, during some time periods, all allocated resources are not needed by them (i.e. *residual resources*). We assume that during such periods, agents can perform additional tasks by using these *residual resources*. Thus, each agent a_i has a *task offers* set $\mathcal{O}_i$, as a way to propose to other agents to execute tasks for them with its residual resources. According to the dynamics, *disturbances* may occur during the execution of schedules. We define a *disturbance* as a loss of one or more resources initially allocated to an agent. So, when a *disturbance* occurs, the agent is unable to perform the tasks requiring these resources. To overcome such situations each agent a_i has a *task demands* set $\mathcal{D}_i$, defining the tasks it cannot perform and needs to delegate to others. At the beginning, $\mathcal{D}_i$ the *task demands* set of an agent a_i is empty, whereas its set of *task offers* $\mathcal{O}_i$ contains elements according to its *residual resources*. These contents are computed along the dynamics of the system. We define each element of $\mathcal{O}_i$ and $\mathcal{D}_i$ as finite sets of data describing the usage constraints of the resources associated to a *task offer* and to a *task demand*.

Task Offers Representation. Here we give a detailed description of the finite data set used by agents to describe their *task offers* and corresponding usage constraints. Given an agent a_i, each *task offer* $o_{ik} \in \mathcal{O}_i$, is defined by the tuple $o_{ik} =< a_i, T_k, R_k, V_k, Sdate_k, Edate_k, Loc_k, C_k >$, where:

- a_i : the agent proposing the *task offer*,
- T_k : identifier of the *task offer*,
- R_k : offered resource type,
- V_k : offered resource volume or amount,
- $Sdate_k$: start date of resource availability,
- $Edate_k$: end date of resource availability,
- Loc_k : constraint(s) on a_i location,
- C_k : acceptance criteria to contract a task execution for another agent with this offer.

Task Offers Acceptance Criteria. The decision criteria C_k relative to a *task offer* $o_{ik} \in \mathcal{O}_i$ of an agent a_i are used by a_i during the task reallocation procedure. Given the *task demand* $d_{jq} \in \mathcal{D}_j$ of an agent a_j, these criteria are used by a_i to decide to execute a task for a_j in regard with the constraints of o_{ik} and d_{jq}. Considering o_{ik} and d_{jq}, $Pmax_k$ denotes the maximum postpone total delay allowed over a_i schedule and induced by the execution of a task for another agent, $Dmax_k$ denotes the maximum distance induced by the location constraints of o_{ik} and d_{jq}. Let $Dist_q$ the distance induced by o_{ik} and d_{jq} location constraints and P_q the time delay occurring in a_i schedule if it executes the *task demand* d_{jq}. Thus the acceptance criteria used by a_i to execute the task associated to d_{jq} for a_j are defined by the following condition :

$$(P_q \leqslant Pmax_k) \wedge (Dist_q \leqslant Dmax_k)$$

Task Demands Representation. We define a *task demand* as the finite data set used by agents to describe a task they can no longer perform and to define the constraints and preferences regarding its execution by another agent. Given an agent a_i, each *task demand* $d_{iq} \in \mathcal{D}_i$ is defined by the tuple $d_{iq} =< a_i, T_q, R_q, V_q, Sdate_q, Edate_q, Loc_q, C_q >$ where:

- a_i : the agent willing to delegate a task execution,
- T_q : identifier of the *task demand*,
- R_q : needed resource type,
- V_q : needed resource volume or amount,
- $Sdate_q$: start date of task execution,
- $Edate_q$: end date of task execution,
- Loc_q : location constraint(s) on T_q performance,
- C_q : acceptance criteria to determine corresponding feasible *task offers*,

Task Demands Acceptance Criteria. Let $o_{jk} \in \mathcal{O}_j$ the *task offer* of an agent a_j and $d_{iq} \in \mathcal{D}_i$ the *task demand* of an agent a_i. The acceptance criteria C_q relative to a *task demand* d_{iq} are used by a_i during the task reallocation procedure to determine if o_{jk} satisfies the constraints of d_{iq} (i.e. if o_{jk} could be a feasible *task offer* in regard with the execution constraints specified by d_{iq}). Considering a_i and d_{iq}, $Pmax_q$ denotes the maximum postponement allowed for the execution of

d_{iq}, $Dmax_q$ denotes the maximum distance induced by the location constraints of d_{iq} and o_{jk} and allowed to contract the *task offer* o_{jk} of agent an a_j. Let us consider $Dist_k$ the distance induced by d_{iq} and o_{jk} location constraints, and P_k the postponement induced if a_j executes d_{iq}. Thus the decision criteria used by a_i to determine if o_{jk} is a feasible offer are defined by the following condition :

$$(P_k \leqslant Pmax_q) \wedge (Dist_k \leqslant Dmax_q)$$

Agent Preferences. Given the *task demand* $d_{iq} \in \mathcal{D}_i$ of an agent a_i, we define a method to determine the best offer among a set of feasible offers and with regard to both of their usage constraints. Thus, the preferences $Pref_q$ associated to d_{iq} define the selection method used in the determination of the best offer. This selection method uses a multi-criteria aggregation function (Bellosta et al., 2008). These preferences define criteria regarding distance travel et delivery time delay. Thus, an offer inducing a short travel distance will be preferred to a longer distance offer. The same selection method is applied for postponements criteria. The expression of preferences over alternative solutions is based on a preferences aggregation defined on these criteria. Each alternative solution o_{jk} is defined by a pair of criteria (D_{jk}, R_{jk}), denoting the alternative o_{jk} which induces a distance D_{jk} and a delivery postponement R_{jk}. The set of alternatives can be represented by the set of pairs (D_{jk}, R_{jk}). Upon this set, we use a lexicographical order to determine the best solution. Thus, we define a total order over the alternatives set such that :

$$((D_1, R_1) \succ (D_2, R_2))$$

$$\text{iff } (D_1 < D_2) \vee ((D_1 = D_2) \wedge (R_1 < R_2))$$

Task Reallocation Procedure. In the case of *disturbances* occurring during the system execution, the main goal of the task reallocation procedure is to find *residual resources* allowing the execution of a task under temporal and location constraints. We propose a simple heuristic method for such problems (Algorithm 1 *delegateTask*) and used to validate our model. This heuristic method is based on a greedy algorithm which aims to reduce the total amount of information exchanges and to shorten the time needed to find a solution by reducing the total number of queried agents. This algorithm determines successive small sets of agents to query until a feasible solution is found. Thus, when an agent needs to delegate a task execution, at first it makes a local search by asking his nearest neighboring agents about their *task offers*. If no corresponding *task offer* is found, then the agent performs a more global research by selecting and asking agents in a farther neighborhood. The notion of

Algorithm 1: delegateTask(demand).

```
1: contract ← false
2: fSet ← null
3: qSet ← getQueryAgentSet(i=0)
4: while (qSet ≠ null) ∧ ¬ contract do
5:     fSet ← feasibleOfferSet(qSet, demand)
6:     contract ← contractOffer(fSet, demand)
7:     fSet ← getQueryAgentSet(i+1)
8: end while
```

neighboring agents defines how to determine the successive sets of agents to query about their *task offers* (method *getQueryAgentSet*). The set of feasible *task offers*, corresponding to the demand d_{iq}, is selected upon the procedure *feasibleOfferSet* and according to the acceptance criteria C_q of d_{iq}. This set of feasible *task offers* is used as input of the procedure *contractOffer* in order to propose and conclude a task delegation with another agent. The procedure *contractOffer* determines the best offer o_{jk} according to the preferences of d_{iq}. Then a task delegation request is sent to a_j the agent proposing o_{jk}. In return, a_j accepts or refuses the request if the realization of d_{iq} implies a delay in its schedule. If the request is refused, the procedure is repeated with the next best offer, until the feasible offer set is empty or a task delegation is concluded.

4 EXPERIMENTAL STUDY

In this Section, we present an experimental study with an implementation of our framework and two simple task reallocation methods, in order to validate our model for dynamic task allocation.

A Transportation Network Application. This work is a part of an industrial project for an urban pooling capacity transportation network. Our experimental study is a simulation of the multi-agent system used to monitor the network execution. The simulation is based on a set of agents representing transportation vehicles. The agents schedule are computed as the solution instance of a *VRPTW*. Time and space are discretized. Time is divided in 10 minutes rounds. The environment is described by a spatial grid divided into a set of 89 non overlapping areas. During rounds, agents travel in areas according to their schedule. Agents can handle a limited amount of freight. During the execution, the amount of freight handled by agents evolves. Thus, at given time rounds, agent load capacity are not fully used and freight transportation capacity is still available. During schedule execution, failures are randomly generated in order to observe the system behavior. These failures imply the

inability of an agent to execute a freight delivery initially planned in it schedule. For example, truck driver may be missing at the beginning of the day, a vehicle may suffer of breakdown during its travel,... In the context of our experimental study, we consider two different failure types. Given t the time round of a failure detection, the first type of failure induces the inability to execute a task scheduled in round $t+n$, whereas the second induces the inability to execute a task scheduled in round t. Our experimentations concern the second type of failure, which implies to quickly find a solution.

Experimental Protocol. Our experiments have been led over two sets of 10 delivery schedules of two different types. Both of the delivery schedules types involve a set of 100 agents. The first delivery schedule type (denoted *type I*) includes 800 delivery orders over a 5 hours period. The second delivery schedule type (denoted *type II*) includes 1600 delivery orders overs a 10 hours period. The delivery schedules of *type II* occur in a far more congested road time period than the schedules of *type I*. For a number of failures ranging from 1 to 90 (one failure maximum per agent), we conducted 100 random draws for each different range of failures. Two different task reallocation procedures have been applied separately to solve each of the schedules thus obtained.

Task Reallocation based on Contract Net Protocol. This reallocation procedure (denoted *global research* in the following), starts with the broadcast of a message containing the *task demand*. Then all the agents reply with a refusal message or a proposal message containing a non empty list of their available *task offers* matching the constraints of the *task demand*. The preferred feasible offer is selected and contracted according to agent preferences and acceptance criteria.

Task Reallocation based on Local Research. In regard with our applicative context, the location constraints of *task offers* and *task demands* refer to areas of the spatial grid modeling the environment. So, given an agent a_i, we define the nearest neighboring agents of a_i as the set of agents located in areas adjacent to a_i location area and agents located in the area of a_i. Starting from the nearest neighboring set, the farther neighboring sets of agents are defined step by step by selecting agents located in the adjacent areas of the previous neighboring set.

Experimental Settings. The main settings used in our experimental study concern the value of agents acceptance *task offers* and *task demands* acceptance criteria. Here we give the acceptance criteria assignment used in our experiment. We consider in this case, that given an agent a_i the acceptance criteria of all its *task offers* and *task demands* have the same value. More formally, given an agent a_i, $\forall o_{ik} \in O_i$ and $\forall d_{jq} \in \mathcal{D}_j$ the assignment of a_i acceptance criteria are defined such that $Pmax_k = Pmax_q = Pmax$ and $Dmax_k = Dmax_q = Dmax$. Distribution of the assignment values of our experiments are given in table 1.

Table 1: Acceptance criteria assignments distribution.

number of agents	*Pmax* (minutes)	*Dmax* (km)
33	30	5
33	60	10
34	90	∞

4.1 Experimental Results

Proportion of Resolved Failures. For a number of failures ranging from 1 to 10 the mean proportions of resolved failures with the local research are similar to those obtained with the global research. The related ratio of the local research over the global research method have almost equal values for a number of failures ranging from 1 to 20. According to ratio values, the local research procedure is less efficient for a number of failures exceeding 30%. In regard with the observations made upon the proportion of resolved failures and their corresponding ratio, we assume that road congestion periods have little influence on the mean proportion of resolved failures. Indeed, the entire ratio result values are similar whatever the research method and type of schedule studied.

Amount of Information Exchanges. In regard with the definition of the task reallocation procedures, the amount of information exchanges depends on the number of messages exchanged. This number of messages is defined by the number of agents queried about their *task offer* and also by the number of agents requested to accept additional task until a solution is found. In case of a global research the number of agents queried about their *task offers* is always equal to 99 agents, whereas this number presents a very high variability level in the case of a local research. The mean number of queried agents needed to solve a failure varies between 15 and 18 agents. Considering a number of failures evolving from 1 to 40%, the mean number of agents requested to accept an additional task varies from 1 to 1.5 for both procedures.

Computational Cost. We define the computational cost as the mean CPU time [1] needed to solve a failure. Local research is 6 times faster than global protocol in the worst case and 8.5 faster in the best case. The

[1]The computer used to run the experimentations is equipped with a 2.66GHz Intel Core i7 processor and a 4Go 1067MHz memory

very high CPU time needed by the global research and compared to the local research, comes with the size of the *task offers* set used to find a solution.

Solution Quality. The quality of the solutions is characterized by three different indicators: the total postponement over agents initial schedule, the delay of resolved freight deliveries and the additional travel distance. Here we focus our study and observations for a failure proportion varying from 1 to 30%. Indeed, we consider that for any failure proportion superior to 30% the simulation results concerns exceptional cases of disturbances in a transportation network. Under this assumption and according to results obtained, the mean total postponement induced by additional tasks in agents schedule is higher with the global protocol than with the local protocol. In addition, the mean value of postponement is higher with schedule *type II* than with *type I*, this observation rely on the differences of road congestion characterizing the two types of delivery schedules. Concerning the mean postponement of resolved freight deliveries, we observe higher mean postponement result values with the global research procedure (from 17 to 27 minutes) than with the local research procedure(from 13 to 21 minutes). The mean additional travel distance of agents obtained with the local research is higher (from 1.5 km to 2,2 km) than with the global research procedure (from 1,3 km to 1,5 km). In short, and with regard to the global research, the local research maximizes the additional travel distances, but enables to minimize the postponement over the delegated deliveries and the agents task schedule.

5 CONCLUSIONS

In this paper, after the presentation of a task reallocation problem and a state of the art, we proposed a computational model that allows agents to delegate the execution of tasks to others. This model aims to capture all the the descriptors needed for task delegation. It gives a description of the various constraints for the performance of tasks under location and temporal constraints by a set of situated agents. In order to validate our model, we introduced a method to enable the exploitation of *residual resources* in a multi-agent system where agents have non shareable resources to execute a defined task schedule. To overcome *disturbances* during the execution of such schedule, we propose to reallocate tasks to agents over time periods where they can perform additional tasks without modifying their initial schedule. Our first experimental results show that the use of local research is more efficient than global search and gives solutions of equivalent quality, in term of additional distance and postpone delay. The main lines of our future research will focus on optimizing the research method by introducing a negotiation protocol. We also aim to propose a task reallocation procedure which enable the use of multiple *task offers* to solve one failure. Another part of our work will concern the influence of acceptance criteria on the method efficiency and the solution quality.

ACKNOWLEDGEMENTS

The present research work has been supported by the Ile-de-France Region, the ”Ministère de l’Économie, de l’Industrie et de l’Emploi” and ADVANCITY.

REFERENCES

Barbucha, D. and Jedrzejowicz, P. (2009). Agent-based approach to the dynamic vehicle routing problem. In *7th International Conference on PAAMS’09*, pages 169–178. Springer-Verlag.

Bellosta, M.-J., Kornman, S., and Vanderpooten, D. (2008). A unified framework for multiple criteria auction mechanisms. *Web Intelligence and Agent Systems*, 6(4):401–419.

Berbeglia, G., Cordeau, J.-F., and Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15.

Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Lemaître, M., Maudet, N., Padget, J. A., Phelps, S., Rodríguez-Aguilar, J. A., and Sousa, P. (2006). Issues in multiagent resource allocation. *Informatica*, 30(1):3–31.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., and Soumis, F. (2001). *VRP with Time Windows*, pages 157–193. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

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

Fischer, K., Chaib-draa, B., Müller, J. P., Pischel, M., and Gerber, C. (1999). A simulation approach based on negotiation and cooperation between agents: A case study. *IEEE Trans. on Systems, Man, and Cybernetics*, 29:531–545.

Gorodetski, V. I., Karsaev, O., and Konushy, V. (2003). Multi-agent system for resource allocation and scheduling. In Marík, V., Müller, J. P., and Pechoucek, M., editors, *CEEMAS*, volume 2691 of *Lecture Notes in Computer Science*, pages 236–246. Springer.

Laporte, G. and Osman, I. H. (1995). Routing problems: A bibliography. *Annals of Operations Research*, 61(1):227–262.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265.

Vickrey, W. (1961). Counterspeculation, auctions and competitive sealed bids. *Journal of Finance*, 16:8–37.