Abstract Architecture for Task-oriented Multi-agent Problem Solving

Jiří Vokřínek, Antonín Komenda, and Michal Pěchouček

***Abstract*—Problem solving and planning in decentralized environments is a key technical challenge in numerous industrial applications, ranging from manufacturing, logistics, virtual enterprizes to multirobotics systems. We present an abstract architecture of a multiagent solver and respective algorithm providing decomposition, task allocation, and task delegation. Various features of the abstract architecture, such as computational complexity or admissibility of the underlying optimization heuristics, are analyzed in the paper. Four instances of the abstract architecture implementations are given to demonstrate the applicability of the abstract solver in a wide variety of real-problem domains.**

***Index Terms*—Algorithms, complexity, distributed planning and problem solving, multiagent applications, multiagent systems, task allocation, task delegation.**

I. Introduction

THE PROBLEM of distributed decision making, decentralized planning, and controlling entities in heterogeneous distributed environments is crucial for many application domains [1]. Existing centralized methods depend on one central-planning system that gathers all required data about decentralized entities before the planning process starts. This approach faces various difficulties. One problem is the need for sharing private knowledge and information about the actual status of these entities. The other problem is the need for real-time replanning based on environments and conditions changing dynamically in time. We present an approach, where each entity is in charge of suggesting their individual plans, where cooperation, resource sharing, and deconflition is solved by methods of negotiation.

The problem of distributed planning and problem solving has been often discussed in the artificial intelligence planning and multiagent research communities recently (e.g., [1]–[4]). Distributed planning has been viewed as either 1) planning for activities and resources allocated among distributed agents; 2) distributed (parallel) computation aimed at plan construction; or 3) plan-merging activity. In this paper, we are solving the problem by algorithms based on task allocation and local resource planning in cooperative environments and use the delegation for continual solution improvement, which is performed in noncritical time. The paper is divided into two parts presenting 1) abstract multiagent solver, problem definition, solver architecture, and algorithms, including discussion of its computational complexity and conditions of strategies admissibility and 2) examples of application areas and description of implemented multiagent systems.

Manuscript received November 19, 2009; revised May 9, 2010; accepted July 26, 2010. Date of current version December 17, 2010. This work was supported by Czech Ministry of Education, Youth and Sports under Grant 6840770038, by Student Grant of Czech Technical University in Prague under Grant SGS10-801890, and by U.S. Army Grant W911NF-08-1-0521 and W15P7T-05-R-P209 under Contract BAA 8020902.A. This paper was recommended by Associate Editor R. W. Brennan.

The authors are with the Agent Technology Center, FEE, Czech Technical University in Prague, Prague, Czech Republic.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCC.2010.2073465

II. Multiagent Solver

Multiagent-planning approaches are used for solving a wide variety of planning problems. As analyzed by Brafman and Domshlak [5], the multiagent-planning techniques can be beneficial for such problems, where the domain sizes of individual agents are considerably smaller (e.g., in logarithmic relation to each other) than the overall size of the problem (even if the planning complexity of an individual agent is exponential) and the number of dependencies between agents is low.

The distributed planning and problem solving has been analyzed by Durfee [1]. One of the related strategies discussed is a *task-sharing* approach. The principle is based on passing of tasks from a busy agent to a vacant agent(s). The process can be summarized in four basic steps.

1) *Task decomposition:* The tasks of agents are decomposed into subtasks. Sharable subtasks are selected.
2) *Task allocation:* The selected tasks are assigned to the vacant agents or agents, which ask for them.
3) *Task accomplishment:* Each agent tries to accomplish its (sub)tasks. The tasks, which need further decomposition are recursively processed and passed to other agents.
4) *Result synthesis:* The results of the tasks are returned to the allocating agent, since it is aware how to use it in the context of the higher tasks.

From the perspective of distributed problem solving, task allocation and the result synthesis are the most crucial parts. However, from the planning perspective, the other two phases are more important. The allocation problem is usually solved by contracting and negotiation techniques, which imply problems related to the resource allocation domain, e.g., cross booking, overbooking, backtracking, and others. In the allocation phase, a hierarchy of agents is established, which may not be fixed in heterogeneous multiagent systems.

The decomposition and delegation principle is widely used in agent-based approaches for problem solving and planning, and shows great applicability to realistic problems. Taking into account Brafman and Domshlak's analysis, Durfee's task-sharing approach efficiency is tightly bound to the solver's ability to

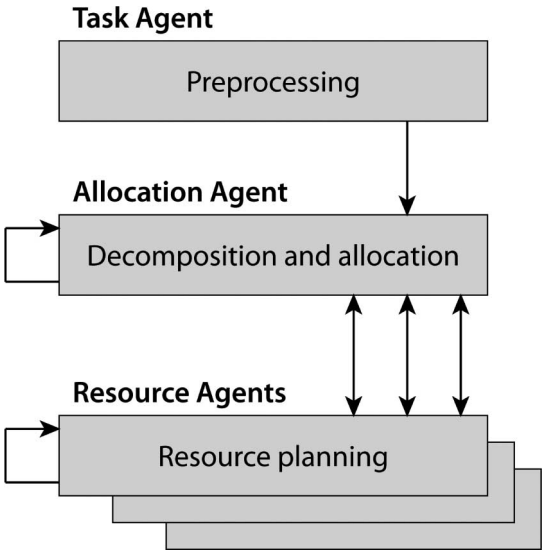

Fig. 1. Abstract architecture of agent-based solver/planner.

reduce the problem sizes for individual agents and keeping the dependencies between agents low.

In the domains, where the optimization/planning problem can be decomposed into independent tasks the multiagent approach shows its benefits. Such a task can be allocated and executed by different agents with low or no influence on each other. In this paper, we assume that such problem decomposition exists. In the rest of the paper, we assume that tasks are independent if not stated otherwise.

We can define the abstract multiagent solver architecture as a composition of three types of agents (see Fig. 1).

1) *Task agent:* This agent is for preprocessing of the problem. It should use a domain-specific heuristic, generic ordering strategy, and randomized method.
2) *Allocation agent:* This agent is for problem decomposition into tasks and delegation of the tasks to resource agents. It maintains task allocation and result synthesis. This agent's strategies and algorithms are domain-independent.
3) *Resource agent:* This agent is for individual case-specific resource planning. In case of further decomposition, the task is handed over to another task agent.

The multiagent system built upon this architecture is composed of one task Agent, one allocation agent, and a set of resource agents. The resource agents represent distributed nature of the multiagent problem.

For complex hierarchical systems, this abstract architecture can be reflected in the multiagent system recurrently, it can be reduced (i.e., one agent undertakes a role of more than one abstract agent type), or it can be parallelized (more abstract solvers are instantiated with potentially overlapping agents, e.g., several task agents or allocation agents handling various problems in parallel). In large systems, concurrent interactions may arise that need to be handled. The agents' interactions are guided by interaction protocols, which are mostly built on Smith's contract net protocol (CNP) [6].

A. Multiagent Problem

The multiagent solver uses the principles of problem decomposition and delegation to autonomous agents that solve parts of the problem individually. The overall solution is then obtained by merging the individual agents' results.

The optimization based on CNP interactions in cooperative environments is usually described as utilitarian social welfare maximization [7]. Therefore, the abstract algorithm objective function can be defined as maximization of social welfare, which is as follows:

$$\text{sw} = \sum_{a \in \mathcal{A}} u_a \tag{1}$$

where $\mathcal{A} = a_1, \ldots, a_n$ is the population of agents and u_a is the utility of agent a. In our case, the social welfare can be computed as a sum of resource agents ($\mathcal{R} \subset \mathcal{A}$) utilities that can be defined as follows:

$$u_a = \sum_{t \in \mathcal{T}_a} (\text{rew}(t) - \text{cost}(t, a)) = \left(\sum_{t \in \mathcal{T}_a} \text{rew}(t) \right) - \text{cost}(\mathcal{T}_a) \tag{2}$$

where $\mathcal{T}_a$ is a set of tasks allocated to the agent $a \in \mathcal{R}$, $\text{rew}(t)$ is a reward for fulfilling task t, $\text{cost}(t, a)$ is the cost of agent a to perform task t, and

$$\text{cost}(\mathcal{T}_a) = \sum_{t \in \mathcal{T}_a} \text{cost}(t, a) \tag{3}$$

is the cost of the overall plan of an agent. The total reward for fulfilling a set of all tasks $\mathcal{T}$ is as follows:

$$\text{rew}(\mathcal{T}) = \sum_{a \in \mathcal{R}} \text{rew}(\mathcal{T}_a) = \sum_{a \in \mathcal{R}} \sum_{t \in \mathcal{T}_a} \text{rew}(t) \tag{4}$$

so the social welfare can be expressed as follows:

$$\text{sw} = \text{rew}(\mathcal{T}) - \sum_{a \in \mathcal{R}} \text{cost}(\mathcal{T}_a) = \text{rew}(\mathcal{T}) - \sum_{t \in \mathcal{T}} \text{cost}(t, a). \tag{5}$$

Since we assume the same quality of task fulfilling by any agent, the reward $k = \text{rew}(\mathcal{T})$ is not influenced by the allocation of tasks to the agents. We can derive social welfare as follows:

$$\text{sw} = k - \sum_{t \in \mathcal{T}} \text{cost}(t, a). \tag{6}$$

As denoted earlier, the goal of CNP-based multiagent optimization in cooperative environments is social welfare maximization. Given by (6), it is the same as minimization of solution cost, where $\text{cost}(t, a)$ is evaluated by the resource agent a undertaking task t. The *objective function* of the abstract solver is then

$$\sum_{t \in \mathcal{T}} \text{cost}(t, a). \tag{7}$$

The task allocation stage of the solver searches for the best suitable mapping of the tasks $\mathcal{T}$ to the resource agents $\mathcal{R}$ that minimizes the objective function given by (7). We can define the goal of the allocation as finding such a partition $\mathcal{P}$ of the set of tasks $\mathcal{T}$ that

$$\underset{\mathcal{P}}{\text{argmin}} \sum_{i=1}^{v} \text{cost}(\mathcal{T}_i) \tag{8}$$

Algorithm 1 The abstract algorithm of a multi-agent solver.

```
Input: Set of tasks T, set of Resource Agents R
Output: T allocated on R
        and local plans of Resource Agents exist

function solve(T, R) begin
  apply ordering heuristic on T
  forall t : T begin
    allocateCNP(t,R)
    if allocation not successful then
      exit with failure or
      mark t as not allocated and continue
    end
    forall a : R begin
      apply dynamic improvement strategy
    end
  end
  forall a : R begin
    apply final improvement strategy
  end
end

function allocateCNP(t, R) begin
  forall a : R begin
    find winner with the lowest insertion
     estimation of t
  end
  if winner is found then
    assign t to the winner
  else
    allocation not successful
  end
end
```

where v is the number of resource agents, $\mathcal{T}_i$ is a subset of tasks allocated to the resource agent a_i, cost($\mathcal{T}_i$) is the cost of the overall plan of agent a_i performing $\mathcal{T}_i$ defined by (3), and

$$\mathcal{T}_i \subseteq \mathcal{T}, \bigcup_{i=1}^{v} \mathcal{T}_i = \mathcal{T} \tag{9}$$

$$\forall i, j : \mathcal{T}_i \cap \mathcal{T}_j = \emptyset \text{ iff } i \neq j. \tag{10}$$

B. Abstract Algorithm

The abstract algorithm representing the presented multiagent solver attempting to minimize objective function defined by (7) is captured by Algorithm 1. According to the abstract architecture depicted in Fig. 1, it contains three phases as following.

1) The first phase of the function `solve` is task preprocessing provided by the task agent. The ordering heuristic represents case-specific sorting of the tasks to increase the solver's efficiency in the particular domain. In some cases, the ordering has no influence, but in others, it may provide significant improvement especially in domains with stronger task dependencies.
2) The second phase is iteration over all tasks and allocation performed by the allocation agent minimizing the insertion cost computed by resource agents (the `allocateCNP` function). As part of this iteration, the dynamic improvement based on cooperation of allocation agent and all resource agents takes place—the improvement strategy is applied to every resource agent after allocation of each task (see following for the description of improvement strategies).
3) The third phase of the `solve` function is the final improvement of the solution. After allocation of all tasks the improvement strategy is executed by all resource agents.

The algorithm is based on local optimization of a single-task insertion and subsequent improvement. Each iteration of the algorithm provides a greedy (order-dependent) task allocation supported by locally optimized solution of resources utilization (which can be seen from global point of view as hill-climbing search). The algorithm does not use any backtracking mechanism or exhaustive search of the state space. It has a significant impact on the algorithm's computational complexity (see Section II-C), but it is susceptible to finding locally efficient solution only. The global solution quality is improved by execution of improvement strategies.

The resource agent uses a case-dependent resource-planning heuristic for these computations. The functions for allocation are as follows.

1) *Insertion estimation* $cost^{estI}(t, a)$*:* The estimation of the cost of the task insertion. It represents the increase of the agent's a cost function caused by undertaking the task t.
2) *Insertion* $cost^{insert}(t, a)$*:* The real cost of the task insertion. This value is determined by adding a new task t to the plan of the agent a in the current state. It is the result of the particular resource-planning algorithm of the resource agent.

The opposite functions used by improvement strategies are as follows.

1) *Removal estimation* $cost^{estR}(t, a)$*:* The estimation of the cost of the task removal. It represents the decrease of the agent's a cost function caused by dropping the task t.
2) *Removal* $cost^{remove}(t, a)$: The real cost of the task removal. This value is determined by removing the task t from the plan of agent a in the current state. It is the result of the particular resource-planning algorithm of the resource agent.

The allocation in the CNP part of the Algorithm 1 is based on the determination of the winner agent. The winner of task t is a resource agent a with the lowest insertion cost; therefore,

$$\text{winner} = \underset{a \in \mathcal{R}}{\text{argmin}} \ \text{cost}^{\text{estI}}(t, a). \tag{11}$$

The allocation agent *allocates* an unallocated task $t \in \mathcal{T}$, where $\forall a_i \in \mathcal{R} : t \notin \mathcal{T}_{a_i}$ to a winner agent a

$$\text{allocate}(\mathcal{T}_a, t) \Rightarrow t \in \mathcal{T}_a \tag{12}$$

provided that local plan of agent a exists and the agent a is able to fulfill this task using the plan for the cost estimation $\text{cost}^{\text{estI}}(t, a)$ used in (11).

Algorithm 2 The abstract algorithm improvement strategies.

```
function improveDW(a,R) begin
   t_w = the worst task of agent a
   forall a' : R \ a begin
      find winner with the lowest cost^estI of t_t
   end
   if cost^estI of winner is lower then cost^estR of a then
      delegate t_w from a to winner
   end
end

function improveDA(a,R) begin
   forall t : tasks of agent a begin
      forall a' : R \ a begin
         find winner with the lowest cost^estI of t
      end
      if cost^estI of winner is lower then cost^estR of a then
         delegate t from a to winner
      end
   end
end

function improveRA(a,R) begin
   forall t : tasks of agent a begin
      remove t from agent a
      allocateCNP(t,R)
   end
end
```

One of the improvement strategies (see following) is based on identification of the most resource consuming task of resource agents. We define *worst task* t_w of agent a as follows:

$$t_w = \underset{t \in \mathcal{T}_a}{\operatorname{argmax}} \ \operatorname{cost}^{\mathrm{estR}}(t, a) \tag{13}$$

i.e., the savings (difference of the total plan cost with and without this task) are maximized.

The improvement strategies basically swap tasks between agents—we say an agent a_i *delegates* task $t \in \mathcal{T}_i$ to an agent a_j

$$\text{delegate}(\mathcal{T}_i, \mathcal{T}_j, t) \Rightarrow t \notin \mathcal{T}_i \wedge t \in \mathcal{T}_j. \tag{14}$$

The *admissible delegation* of task t from agent a_i to agent a_j, where $i \neq j$ is a delegation that satisfies the *improvement condition*

$$\operatorname{cost}^{\mathrm{estR}}(t, a_i) - \operatorname{cost}^{\mathrm{estI}}(t, a_j) > 0. \tag{15}$$

The improvement strategies used by abstract algorithm are as follows (see Algorithm 2 for more details).

1) *Delegate worst* (*DW*)*:* Each resource agent a_i identifies its worst task t_w according to (13) and tries to delegate it to another agent a_j if the improvement condition defined by (15) holds and a_j is the allocation winner according to (11).
2) *Delegate all* (*DA*)*:* Each resource agent a_i delegates all its tasks $\mathcal{T}_i$ to the winner of each task if the improvement condition is satisfied.
3) *Reallocate all* (*RA*)*:* Each resource agent successively removes all its tasks from the plan and allocates them again using the CNP strategy. The result of the allocation can be the same as before task removing, or a change of the position of the task in the current agent plan, or delegation to another agent. To ensure proper function of RA improvement strategy, we require for successive removing and inserting of task t from/to agent a that

$$\operatorname{cost}^{\mathrm{insert}}(t, a) \leq \operatorname{cost}^{\mathrm{remove}}(t, a) \tag{16}$$

i.e., when removing and reinserting task $t \in \mathcal{T}_a$, the $\operatorname{cost}(\mathcal{T}_a)$ does not increase.

C. Complexity Analysis

The general computational complexity of the multiagent solver is introduced in [5]. Using transformation of the multiagent-planning problem to the distributed constraint satisfaction problem, the worst case time complexity of the multiagent planning is upper bounded by

$$f(\mathcal{I}) \times \exp(\text{comm}) + \exp(\text{int}) \tag{17}$$

where $f(\cdot)$ is the factor inducted by requesting each agent to plan, while committing to a certain sequence of actions, $\mathcal{I}$ is the complexity of an individual agent's planning, $\exp(\text{comm})$ represents a factor exponential in min–max number of per-agent commitments, and an additive factor $\exp(\text{int})$ represents the interactions of agents.

The consequences of (17) lead to interesting features of the multiagent solver, such as that there is 1) no direct exponential dependence on the number of agents; 2) no direct exponential dependence on the size of the planning problem or size of the joint plan; and 3) no direct exponential dependence on the length of individual agent plans [5].

In our case, the feature 2) resulting from (17) does not have a strong impact if the decomposition algorithm of the allocation agent is exponential because the size of its problem is the same as the size of the overall problem, but for the resource agents (and other subordinate agents in the case of a complex hierarchical structure) this feature holds. However, the exponential factors are usually reduced by the polynomial heuristics—decomposition, allocation, optimization, and resource-planning strategies implemented in real applications. The ordering strategy of the task agent does not have a strong influence on the worst case complexity because of its additive nature and low complexity (provided that tasks can be compared in constant time). The multiagent solver benefits in the domains, where problem can be easily decomposed to independent tasks, and/or where the polynomial heuristics for the resource planning exist.

For the abstract algorithm (see Algorithm 1), (17) can be represented as follows:

$$n \times \log(n) + n \times O^{\mathrm{alloc}} + n \times m \times O^{\mathrm{impr}} \tag{18}$$

where n denotes the number of tasks and m is the number of resource agents. The first part represents the ordering heuristic and its complexity corresponds to the complexity of standard sorting algorithms. The middle part is the complexity of the allo-

cation part of the algorithm and the last part is the improvement part of the algorithm.

The allocation and improvement time complexities of the algorithm are defined as follows:

$$O^{\text{alloc}} = m \times O(\text{cost}^{\text{estI}}(t,a)) + O(\text{cost}^{\text{insert}}(t,a))$$
$$O^{\text{impr}} = fi(n')$$

where $n' = n/m$ is the average number of tasks allocated to a particular resource agent and $fi(n')$ is the factor representing the complexity of the implemented agents' improvement strategy. We assume

$$O(\text{cost}^{\text{estI}}(t,a)) = O(\text{cost}^{\text{insert}}(t,a)) = fr(n')$$
$$O(\text{cost}^{\text{estR}}(t,a)) = O(\text{cost}^{\text{removal}}(t,a)) = fr'(n')$$

where $fr(n')$ is the factor representing the complexity of the implemented agents' resource-planning strategy for the task insertion and $fr'(n')$ is the factor representing the complexity of the implemented agents' resource-planning strategy for the task removal. Therefore, the general worst case time complexity of the presented abstract algorithm is as follows:

$$n \times (\log(n) + m \times (fr(n') + fi(n'))). \tag{19}$$

The complexity of improvement strategies defined in Section II-B are as follows:

$$fi^{\text{DW}}(n') = O^{\text{worst}} + fr'(n') + m \times fr(n')$$
$$fi^{\text{DA}}(n') = fi^{\text{RA}}(n') = n' \times (fr'(n') + m \times fr(n'))$$

where O^{worst} is the complexity of identification of the worst task in the plan, i.e., finding the task with greater $\text{cost}^{\text{estR}}(t,a)$. It can be upper bounded by the iteration through all n' tasks and computation of removal cost of each one, so its worst case complexity is upper bounded by $O^{\text{worst}} = O(n' \times fr'(n'))$. For combination of all described improvement strategies, the improvement complexity is upper bounded by

$$fi(n') = n' \times fr'(n') + m \times n' \times fr(n'). \tag{20}$$

The factor $m \times n' = m \times (n/m) = n$, so the improvement complexity has no relation to the number of agents. Combining (19) and (20) and taking $n' = n$, the *worst case time complexity* of the abstract algorithm is as follows:

$$n \times \log(n) + n^2 \times fr'(n) + m \times n^2 \times fr(n). \tag{21}$$

The presented complexity analysis shows the polynomial impact of the decomposition and delegation principles used by the multiagent abstract solver. The impact of the two factors introduced by (17) is the following:

1) the complexity of the operations of resource agent are multiplied by n^2;
2) the influence of the number of resource agents is linear.

The complexity analysis shows us an important feature of agent-base solver. When using polynomial heuristics for task insertion and removal, the implemented multiagent solver provides polynomial worst case complexity. Together with linear computational scaling with the number of agents makes the presented abstract architecture suitable for many application areas.

Algorithm 3 The incremental improvement algorithm.

```
Input: Set of Resource Agents R
Output: Improved solution

function improve(R) begin
  improvement := true
  repeat until improvement is false
    improvement := false
    forall a : R begin
      apply dynamic improvement strategy
      if solution has been improved then
        improvement := true
      end
    end
  end
end
```

In real-application areas, ordering heuristics can be found that result in allocation with no need of using improvement strategies (e.g., production-planning system described in Section III-D). In other cases, the planning of resource agents is implemented with low complexity (e.g., linear) and the improvement strategy has greater importance. In Section III, we present several applications developed using the described abstract multiagent solver.

D. Incremental Improvement Strategy

The basic multiagent solver can be enhanced by the incremental improvement strategy. Algorithm 1 allocates the tasks and runs improvement strategies (both dynamic and final), keeping the computational complexity low (i.e., guaranteeing a good response time of the algorithm). In many cases, it is beneficial to perform incremental improvement of the solution when we have enough time to wait for better results or the environment changes dynamically. The changes of the task constraints, resources availability, and execution uncertainty affects the efficiency of the static plan during execution. Algorithm 1 optimizes (7) in a single run upon conditions that are valid at the time of algorithm execution. When some conditions change, the solution quality diverts. In such a case, the incremental improvement strategy should be used to keep the solution up to date with the dynamic environment changes and/or improve the solution over the time.

The nature of the task allocation process of the multiagent solver enables us to run the improvement strategies continuously and interrupt its improvement during any run without loosing the correctness of the solution (assuming the delegation as atomic process). The incremental improvement strategy described in Algorithm 3 is an anytime algorithm monotonically improving the quality of the solution. The improvement strategies (see Algorithm 2) are executed until the solution is no longer being improved (i.e., the overall cost of the solution defined by (7) does not change between iterations).

The inner loop iterates over all resource agents and provides the same features (complexity and convergence) as the improvement part of Algorithm 1. The outer loop terminates when the quality of two consequent solutions is the same.

E. Resource Agent Strategy Admissibility

The presented multiagent solver features strongly depends on the functions provided by the resource agents. We investigate the allocation process correctness and delegation stability, and define the constraints to functions provided by admissible resource agent strategy.

When using standard CNP, we require the estimation functions of resource agent a to provide accurate estimations of inserting/removal of task t, e.g.,

$$\text{cost}^{\text{estI}}(a,t) = \text{cost}^{\text{insert}}(a,t) \tag{22}$$

$$\text{cost}^{\text{estR}}(a,t) = \text{cost}^{\text{remove}}(a,t). \tag{23}$$

In the case of more advanced allocation protocols with backtracking (e.g., extended CNP [8]), this accuracy can be relaxed. In all cases, these constraints defined on the resource agent functions ensure that Algorithm 1 is able to find a local minimum of the cost function (7) and to guarantee the proper behavior of improvement strategies.

The estimation functions provided by resource agent strategy are *admissible* only if

$$\text{cost}^{\text{estI}}(a,t) \leq \text{cost}^{\text{insert}}(a,t) \tag{24}$$

$$\text{cost}^{\text{estR}}(a,t) \geq \text{cost}^{\text{remove}}(a,t) \tag{25}$$

where for the upper bound of the estimation error holds

$$|\text{cost}^{\text{insert}}(a,t) - \text{cost}^{\text{estI}}(a,t)| < \varepsilon_I \tag{26}$$

$$|\text{cost}^{\text{remove}}(a,t) - \text{cost}^{\text{estR}}(a,t)| < \varepsilon_R \tag{27}$$

and delegation improvement condition according to (14) and (15) is modified to

$$\text{cost}^{\text{estR}}(t,a_i) - \text{cost}^{\text{estI}}(t,a_j) > \varepsilon_I + \varepsilon_R. \tag{28}$$

Keeping the defined constraints (improvement condition, estimation error, and admissibility of estimation functions), the execution of any of the improvement strategies results in reduction of the solution cost or the solution is unaffected (i.e., the new solution is not worse than the previous one) and the incremental improvement strategy termination is ensured after a finite number of steps (if the environment does not change during the execution of the improvement algorithm).

Using features discussed in previous sections, we can define the *admissible resource agent strategy* (its internal heuristics and estimation, and allocation functions) has to:

1) use admissible estimation functions according to (24) and (25);
2) bound the estimation error according to (26) and (27);
3) fulfill the improvement condition defined by (28) (the delegation is admissible if the solution cost improves);
4) when using the reallocate-all improvement strategy, (16) must hold (i.e., when reallocating, the new solution has to be better or at least the same as the previous one);
5) the algorithm execution time must be low compared to the frequency of the environment changes (the environment is considered static during the execution of the algorithm).

III. Applications

Due to high flexibility, robustness, and scalability, the agent technologies promise a wide industrial applicability [9] even in the real-time environments with a need of dynamic reconfiguration and replanning [10]. This section presents four examples of the multiagent systems implemented using the presented abstract architecture. It demonstrates the applicability of the concept in a wide variety of real-problem domains—vehicle routing problems (VRPs), strategic mission planning, multirobot frontiers exploration, and production planning. All presented multiagent systems share the same abstract architecture, algorithm, and improvement strategies. For each application domain, a particular resource optimization heuristic has been designed and implemented by the resource agents. All the presented systems' implementations have low-computational complexity and provide high-quality solutions.

A. VRP Solver

The VRP is a well-known optimization problem. The problem is NP-hard and is defined as routing of a fleet of gasoline delivery trucks between a terminal and a number of service stations. The trucks have load capacity limitations and deliveries have to be accomplished at minimum total cost (distance traveled).

The agent-based approach to a variant of the VRP solver has been presented, for example, in [11]. Zeddini *et al.* use three types of agents—client, bidder, and vehicle agents. The approach is based on the CNP allocation and the optimization is based on exchange of tasks between the vehicle agents. The vehicle agents use an insertion heuristic and improvement strategy for task swapping between them. The error of the solution (compared to the optimal solution) presented in the paper is 4%–29% for standard benchmark problems.

A similar approach has been used in [12] for a dynamic variant of *k*-VRP (new tasks are added during the execution), where the initial allocation is generated using a centralized algorithm. The dynamic task allocation is made by the CNP protocol. Then, two improvement phases are applied. The *intraroute optimization* is applied to each agent route and *interroute optimization* is performed between vehicle agents.

The VRP family solver built upon an abstract multiagent solver, which is introduced by this paper is presented in [13]. The solver uses three ordering strategies, the combination of all improvement strategies with dynamic improvement enabled or disabled, and the resource agent algorithm implemented using a standard cheapest insertion heuristic for traveling salesman problem [14].

The objective function is based on the distance traveled by vehicles (represented by resource agents) and fully corresponds to (7). The implemented RA strategy fulfills the improvement conditions defined by (15) and (16) and admissibility conditions defined by (22) and (23).

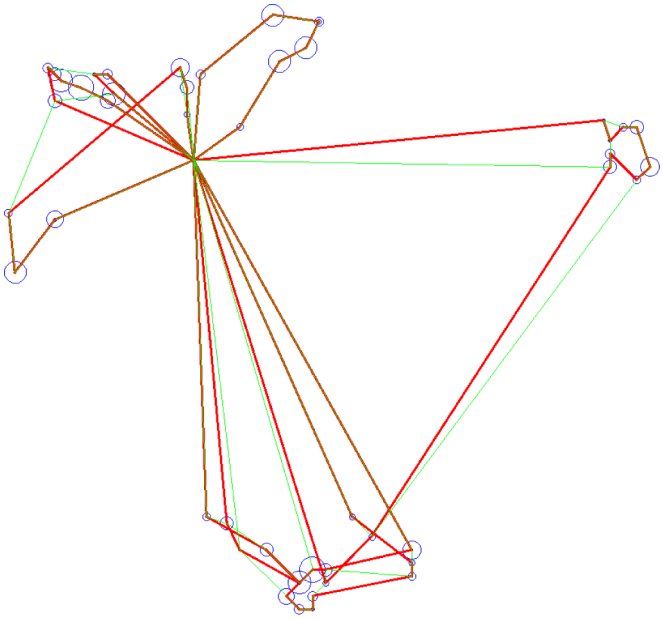

Fig. 2. Example of the VRP solver result.

The complexity of the resource agent cost computation functions is kept low, i.e., $fr(n') = O(n')$ and $fr'(n') = O(1)$. The worst case time complexity defined by (21) (using $n' = n/m$) is $O(n^3)$, which has been also proved experimentally on benchmark instances.

The example of the solution provided by the solver is in Fig. 2. There are 52 nodes served by seven vehicles (an optimal number of vehicles has been obtained). The circles in the nodes represent the size of transportation demand. The quality of the solution to the problem presented in Fig. 2 is 93.4% compared to the optimal path length. We measure the quality as $\frac{\text{cost}_{\text{solver}}}{\text{cost}_{\text{optim}}} \times 100[\%]$, where $\text{cost}_{\text{solver}}$ is the solution cost provided by the solver and $\text{cost}_{\text{optim}}$ is the cost of the best known solution. The optimal vehicle's path is depicted by thin green lines and the solution produced by the multiagent solver is represented by thick red lines.

For the 115 evaluated benchmark instances (standard VRP benchmark problems), the multiagent solver provides solutions with the quality of at least 81% compared to the optimal solution with average quality better than 91% (corresponds to solution error of 19%, respectively, 9%). The self-organizing capability of the system successfully minimizes the number of vehicles used. The results show that the application of dynamic improvement strategy provides better results than batch processing with final improvement strategy only. The best performance has been reached using DA and RA improvement strategies. The implemented solver demonstrates very good applicability of the abstract multiagent solver to the family of routing problems and easy adaptation to problem variants.

B. Strategic Mission Planning Using Social Commitments

The multiagent abstract solver presented in this paper has been used in a system for distributed planning and coordination in dynamic nondeterministic multiactor mixed-initiative environment [15]. The system provides flexible planning, replanning, and task allocation. The system addresses several issues that have to be solved in order to fulfill the requirements on a system planning in dynamic nondeterministic environments. An overview of the problems is as follows.

1) *Distributed planning:* Planning in such an environment is practically realizable only as a distributed process.
2) *Distributed resource allocation:* An integral part of the planning process is resource allocation both of the acting entities in the world and of the static resources, and as the planning process is distributed, the resource allocation has to be distributed as well.
3) *Distributed plan execution and synchronization:* Constituted distributed plan consisting of several personal plans has to be executed by the entities. The personal plans need to be coordinated in distributed manner, as the entities do not know each other's plans.

From the perspective of allocation agent, the planning process can be divided into three phases: 1) the hierarchical task network I-Plan planner [16] is used for creating an abstract plan for a long-time horizon; 2) the plan instantiating process uses a distributed resource allocation based on the CNP [6]; and 3) with the help of this protocol, the appropriate subordinate agents (resource agents) are found and the responsibilities for the plan actions are fixed. The internal resource agents optimization uses the as-early-as-possible scheduling heuristic. The heuristic causes the earliest possible execution of the plan actions, which affects the length of the whole plan in the nondeterministic environment. The effect is directly proportional to the amount of nondeterminism in the world (which has been simulated as random prolonging of the actions durations). The agents' hierarchy in the system is captured by Fig. 4, there is a commander creating tasks for builders, builders implementing the abstract task agent and allocation agent roles in parallel, and finally, we have a set of trucks implementing the abstract resource agent roles.

All plans are described in the form of social commitments substituting plan actions. The commitment is a knowledge-base structure describing agent's obligation to change the world-state and a set of rules defining what the agent should do if the obligation is not satisfiable [17]. The commitments enable expressive description of the decommitment rules, and thus, the replanning process, it captures the improvement strategies executed on the moment when the environment changes in the way that collide with the plan.

In other words, the replanning process (i.e., the plan improvement process) by means of social commitments can be described as successive recommitting [17]. For the decommitting purposes, three basic decommitment rules were used: full decommitment, delegation, and relaxation [18].

The deployment scenario of the system is a disaster relief operation, where the resources have to be transported to the impacted zone [19]. There are material resources, emergency units, and transport units in the scenario. The emergency units create plans and request the transportation for itself and for required material resources. The problem is to find such a global

Fig. 3. Strategic mission planning system—scenario island screenshot.

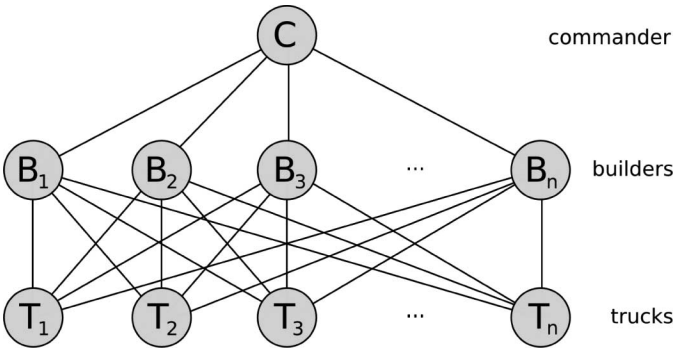

Fig. 4. Strategic mission planning system—hierarchy of agents.

plan that all units and material are transported to the demanded area as soon as possible using limited transportation resources (see a screenshot of the scenario island in Fig. 3).

The entire system provides fast convergence to the efficient solution with time complexity of $O(n^3)$. The heuristic of the resource agent strategy is admissible in terms defined in Section II-B and provides $fr(n') = O(1)$ and $fr'(n') = O(n')$. The plan execution stability in the dynamic environment is enhanced using improvement strategies captured by decommitment rules, and thus, incrementally invoked abstract algorithm presented in Section III.

C. Cooperative Frontiers Exploration

The presented multiagent solver has been also utilized for cooperative frontiers exploration problem [20]. The problem is to find the shortest path for a convoy of vehicles through a partially known urban area. The street map of the area is *a priori* known, but the actual condition of the routes is not. There is the convoy moving through the city using a path-planning algorithm that incorporates the information obtained by a set of small autonomous vehicles. These vehicles explore the area ahead of the convoy (see the scenario screenshot in Fig. 5).

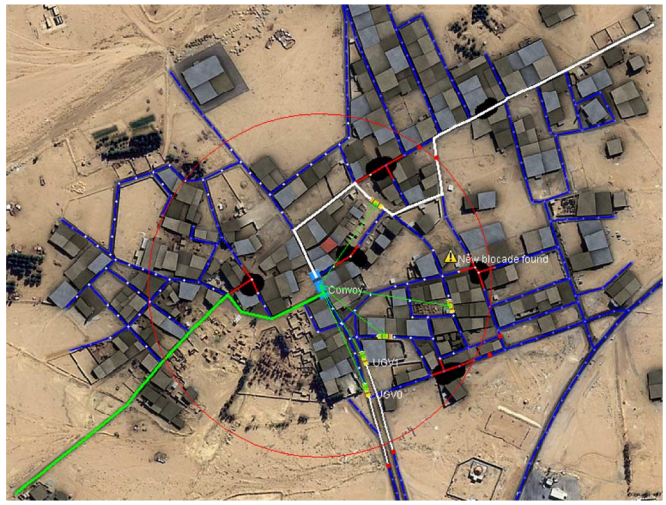

Fig. 5. Cooperative frontiers exploration—scenario screenshot.

The goal of the multiagent system is to find the shortest path through the area ensuring that 1) the convoy will not stop or u-turn because of a street blockade and 2) the total traveled path of all vehicles is minimized. The multiagent system is composed of one convoy agent (task agent and allocation agent role) and a set of unmanned ground vehicles (UGV) agents (resource agent role). Besides path planning and navigating through the city, the convoy agent generates a set of frontiers for exploration [21]. In this case, it is not a classical mapping task in an unknown area, but the frontiers represent points of interests in the known street map that should be investigated to ensure the convoy may freely pass through the desired route. A similar robotic problem has been also solved using a multiagent system for distributed robotic planning, e.g., in [22]. The first goal (convoy nonstoping movement toward a target) is secured using the incremental replanning algorithm for convoy path planning based on anytime planning algorithm D-star [23]. The second goal (minimization of the traveled path) is handled by a multiagent solver using the presented architecture and algorithms. It is shown [20] that the multiagent solver provides almost real-time response and significantly reduces the convoy traveled path even with small number of UGVs keeping the overall traveled distance overhead low.

The convoy agent dynamically creates points of interests (exploration frontiers) and allocates them to the UGV agents using Algorithm 1. In the case of any new information is discovered, Algorithm 3 is used and also new frontiers are generated and allocated (or some frontiers can be removed). The UGV agents use a route optimization heuristic that attempts to minimize the traveled distance similar to the VRP described earlier. It fulfills the improvement condition defined by (15) and admissibility conditions defined by (24) and (25). The computational complexities of the strategy are $fr(n') = O(n')$ and $fr'(n') = O(1)$. The experimentally evaluated worst case time complexity of the multiagent solver has been upper bounded by $O(n^3)$.

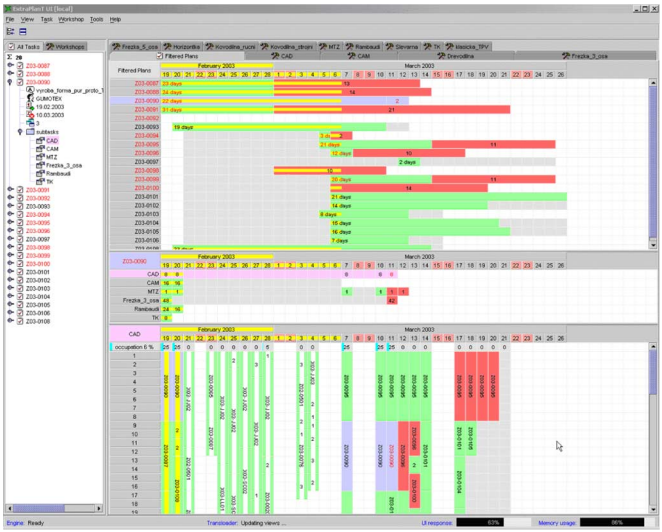

Fig. 6. Screenshot of multiagent production-planning system.

D. Production Planning

Classical-planning systems (using scheduling algorithms with various heuristics, constraint logic programming, genetic algorithms, and simulated annealing [24], [25]) work centrally and allocate resources usually in one run for every product (order) presented in the system. These methods use mostly stochastic algorithms and generate near-optimal solutions for minimization of defined criteria (e.g., a sum of weighted tardiness and inventory costs). Such a solution is fully sufficient, while the required replanning and rescheduling affects the entire plan. The plan is usually completely rebuilt and a random aspect of the algorithms can cause major (unwanted) changes in plans after replanning. This might not be suitable for many manufacturing areas. With physical distribution of the production units, it is advantageous to decompose and distribute the planning problem [1].

The multiagent technology addresses all the phases of the manufacturing decision-making support, while there are few implementations of multi-agent systems that cover more than a single stage. There are solutions for low-level scheduling or control systems, the product configuration and quotation phases to short-term and long-term production planning and supply-chain management [26].

The example of the system that uses the same conceptual fundamentals as multiagent abstract solver presented by this paper is presented in [27] and [28]. One of the goals of the multiagent system is to create a production plan for middle- and long-term horizon to give an overview of resource utilization (see Fig. 6 for a screenshot of a multiagent production-planning system). The production resources are represented by the resource agents maintaining the constrains and capabilities of individual production workshops. The task agent uses a classical task ordering heuristic based on weighted earliest deadline first, which significantly improves the solution and there is no need for improvement strategy execution after allocation phase of the batch of tasks. The production-order decomposition and planning is provided by the planning agent, which allocates the parts of the production order to the resource agents using the CNP according to Algorithm 1. In case of environment changes (e.g., an update of production times estimation, machines breakdowns, delays in production, etc.), Algorithm 3 is executed. The solution of the solver is optimal according to the cost computed as a weighted delay penalty.

Resource agents use an admissible strategy [according to (22), (23), and (15)] that minimizes the weighted average delay of the production orders (see [29] for more details). The computational complexities of resource agent algorithms are $fr(n') = fr'(n') = O(n')$; therefore, the overall solver complexity is upper bounded by $O(n^4)$.

IV. Conclusion

This paper describes an abstract multiagent solver architecture and an algorithm for implementing a wide variety of practical multiagent-planning and problem-solving systems. The algorithm maximizing social welfare of the cooperative agent community is introduced and analyzed. CNP-based task allocation and solution improvement using task delegation provides a powerful tool for problem solving when keeping the computational complexity within reasonable limits. We also discuss the limitations and admissibility constraints of the resource optimization heuristic that has to be designed to implement the multiagent solver for a particular problem domain and define the resource agent strategy admissibility.

The solver benefits in domains, where decomposition of a problem into independent tasks is possible. Finding independent subsets of tasks significantly reduces the need for interactions between agents. The tasks are planned and executed by individual agents independently, and the overall solution is then merged from the partial ones. The system is able to balance the allocation of the tasks to the agents and provide anytime monotonically improved solution.

The applicability of the presented abstract multiagent solver is demonstrated on several real systems operating in the domains of VRPs, strategic mission planning, multirobot frontiers exploration, and production planning. In all application areas, the implemented system provides low-computational complexity ($O(n^3)$ or $O(n^4)$) with good solution quality.

Acknowledgment

The authors' organizations and research sponsors are authorized to reproduce and distribute reprints and online copies for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of other parties.

References

[1] E. H. Durfee, "Distributed problem solving and planning," in *A Modern Approach to Distributed Artificial Intelligence*, G. Weiß, Ed. San Francisco, CA: The MIT Press, 1999, ch. 3.

[2] M. E. DesJardins and M. J. Wolverton, "Coordinating a distributed planning system," *AI Mag.*, vol. 20, no. 4, pp. 45–53, 1999.
[3] E. Ephrati and J. S. Rosenschein, "A heuristic technique for multiagent planning," *Ann. Mathematics Artificial Intell.*, vol. 20, no. 1–4, pp. 13–67, 1997.
[4] M. M. de Weerdt, A. Bos, J. Tonino, and C. Witteveen, "A resource logic for multi-agent plan merging," *Ann. Mathematics Artificial Intell., Special Issue Comput. Logic Multi-Agent Syst.*, vol. 37, no. 1–2, pp. 93–130, Jan 2003.
[5] R. I. Brafman and C. Domshlak, "From one to many: Planning for loosely coupled multi-agent systems," in *ICAPS*, 2008, pp. 28–35.
[6] R. G. Smith, "The contract net protocol: High level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. C-29, no. 12, pp. 1104–1113, Dec. 1980.
[7] K. J. Arrow, A. K. Sen, and K. Suzumura, Eds., *Handbook of Social Choice and Welfare (Handbooks in Economics)*. Amsterdam: North Holland, Aug. 2002.
[8] K. Fischer, J. P. Muller, M. Pischel, and D. Schier, "A model for cooperative transportation scheduling," in *Proc. First Int. Conf. Multiagent Syst.* Menlo park, California: AAAI Press/MIT Press, Jun. 1995, pp. 109–116.
[9] V. Marik and J. Lazansky, "Industrial applications of agent technologies," *Control Eng. Practice*, vol. 15, no. 11, pp. 1364–1380, 2007.
[10] P. Vrba and V. Marik, "Capabilities of dynamic reconfiguration of multiagent-based industrial control systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 40, no. 2, pp. 213–223, Mar. 2010.
[11] B. Zeddini, M. Temani, A. Yassine, and K. Ghedira, "An agent-oriented approach for the dynamic vehicle routing problem," in *2008 Int. Workshop Adv. Inf. Syst. Enterprises*, vol. 0, pp. 70–76, 2008.
[12] D. Barbucha and P. Jedrzejowicz, "Agent-based approach to the dynamic vehicle routing problem," in *7th Int. Conf. Pract. Appl. Agents Multi-Agent Syst,* ser. Advances in Soft Computing, vol. 55. Springer Berlin/Heidelberg, 2009, pp. 169–178.
[13] J. Vokrinek, A. Komenda, and M. Pechoucek, "Agents towards vehicle routing problems," in *Proc. Int. Joint Conf. Autonomous Agents Multiagent Syst*, pp. 773–780.
[14] D. J. Rosenkrantz, R. E. Stearns, and P. M. L. II, "An analysis of several heuristics for the traveling salesman problem," *SIAM J. Comput.*, vol. 6, no. 3, pp. 563–581, 1977.
[15] A. Komenda, J. Vokřínek, M. Pěchouček, G. Wickler, J. Dalton, and A. Tate, "I-globe: Distributed planning and coordination of mixed-initiative activities," in *Proc. Knowl. Syst. Coalition Operations 2009*, Chilworth Manor, Southampton, U.K., Mar.–Apr.
[16] A. Tate, "Intelligible ai planning," in *Proc. 20th ES Res. Develop. Intell. Syst. XVII*, M. Bramer, A. Preece, and F. Coenen, Eds. Springer, 2000, pp. 3–16.
[17] A. Komenda, M. Pěchouček, J. Bíba, and J. Vokřínek, " Planning and re-planning in multi-actors scenarios by means of social commitments," in *Proc. Int. Multiconf. Comput. Sci. Inf. Technol.*, vol. 3. IEEE, Oct. 2008, pp. 39–45.
[18] J. Vokřínek, A. Komenda, and M. Pěchouček, "Decommitting in multi-agent execution in non-deterministic environment: Experimental approach," in *Proc. 8th Int. Joint Conf. Autonomous Agents Multiagent Syst.*, 2009, pp. 977–984.
[19] A. Komenda, J. Vokřínek, M. Pěchouček, G. Wickler, J. Dalton, and A. Tate, "Distributed planning and coordination in non-deterministic environments (demo)," in *Proc. 8th Int. Joint Conf. Autonomous Agents Multiagent Syst*, 2009, pp. 1401–1402.
[20] J. Vokrinek, A. Komenda, and M. Pechoucek, "Cooperative agent navigation in partially unknown urban environments," in *Proc. 3rd Int. Symp. Practical Cognitive Agents Robots.*, 2010, pp. 46–53.
[21] A. Visser, Xingrui-Ji, M. van Ittersum, L. A. G. ?lez Jaime, and L. A. Stancu, "Beyond frontier exploration," in *RoboCup 2007: Robot Soccer World Cup XI*, ser. Lecture Notes in Computer Science, vol. 5001. Springer Berlin/Heidelberg, 2008, pp. 113–123.
[22] S. Thayer, B. Digney, M. B. Dias, A. T. Stentz, B. Nabbe, and M. Hebert, "Distributed robotic mapping of extreme environments," in *Proc. SPIE: Mobile Robots XV and Telemanipulator Telepresence Technologies VII*, vol. 4195, Nov. 2000, pp. 84–95.
[23] M. Likhachev, D. Ferguson, G. Gordon, A. T. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm," in *Proc. Int. Conf. Automated Planning Scheduling*, Jun. 2005, pp. 262–271.
[24] N. Sadeh and M. S. Fox, "Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem," *Artif. Intell.*, vol. 86, no. 1, pp. 1–41, 1996.
[25] J. K. k, L. Rothkrantz and J. L. 1/2, "Genetic algorithms with limited convergence," in *Information Processing with Evolutionary Algorithms*, ser. Advanced Information and Knowledge Processing. London UK: Springer, 2005, pp. 233–253.
[26] M. Pěchouček, J. Vokřínek, and P. Bečvář, "Explantech: Multiagent support for manufacturing decision making," *IEEE Intell. Syst.*, vol. 20, no. 1, pp. 67–74, Jan./Feb. 2005.
[27] J. Vokřínek, J. Hodík, M. Pěchouček, P. Bečvář, and J. Pospíšil, *ExtraPlanT as a Multi-Agent System for Extra-Enterprise Collaboration*. Information science Reference, 2008, ch. 593-600, pp. 593–600.
[28] J. Hodík, P. Bečvář, M. Pěchouček, J. Vokřínek, and J. Pospíšil, "Explantech and extraplant: multi-agent technology for production planning, simulation and extra-enterprise collaboration," *Int. J. Comput. Syst. Sci. Eng.*, vol. 20, no. 5, pp. 357–367, 2005.
[29] P. Bečvář, P. Charvát, M. P. J. Pospíšil, A. Říha, and J. Vokřínek, "Explantech/extraplant: Production planning and supply-chain management multi-agent solution," *EXP - in search innovation*, vol. 3, no. 3, pp. 116–125, 2003.

Jiří Vokřínek received the Master's degree in technical cybernetics at Czech Technical University in Prague, Prague, Czech Republic, he is working toward the Ph.D. degree at the Agent Technology Center, Department of Cybernetics at Czech Technical University in Prague.

He is a Research Scientist at the Agent Technology Center. His research interests include agent-based planning and simulation, artificial intelligence, multiagent systems, planning and replanning in manufacturing, virtual organizations, supply-chain management, and logistics.

Antonín Komenda received the Master's degree in Technical Cybernetics at Czech Technical University in Prague, Prague, Czech Republic. He is currently working toward the Ph.D. degree from Agent Technology Center of the Department of Cybernetics at Czech Technical University in Prague. His studies was focused on multiagent systems.

He is also a Researcher at the Agent Technology Center. His current research interests include multiagent systems, distributed planning and plan repairing.

Michal Pěchouček received the Master's degree in IT: Knowledge-based systems from the University of Edinburgh, Edinburgh, U.K., and the Ph.D. degree in artificial intelligence and biocybernetics from Czech Technical University in Prague, Prague, Czech Republic.

He is a Professor in artificial intelligence at the Department of Cybernetics, Czech Technical University in Prague. He is currently a Head of Agent Technology Center, Department of Cybernetics, CTU. His research interests include problems related to multi-agent systems, especially topics related to social knowledge, metareasoning, acting in communication inaccessibility, coalition formation, agent reflection, and multiagent planning.