Multilevel Approach to Agent-Based Task Allocation in Transportation

Martin Rehák, Přemysl Volf and Michal Pěchouček

Department of Cybernetics and Center for Applied Cybernetics
Czech Technical University, Technická 2, Prague, 166 27, Czech Republic
{mrehak,volf,pechouc}@labe.felk.cvut.cz

Abstract. We present a hybrid algorithm for distributed task allocation problem in a cooperative logistics domain. Our approach aims to achieve superior computational performance by combining the classic negotiation techniques and acquaintance models from agent technology field with methods from the operation research and AI planning. The algorithm is multi-stage and makes a clear separation between discreet planning that defines the tasks and allocation of resources to available tasks. Task allocation starts with centralized planning based on acquaintance model information that prepares a framework for efficient distributed negotiation. The subsequent distributed part of the task allocation process is parallel for all tasks and allows the agents to optimally allocate their resources to proposed tasks and to further optimize the allocation by negotiation with other agents. Parallel execution of the task allocation mechanism allows the algorithm to answer the planning request in predictable time, albeit at expense of possible non-optimality. In the experiments, we evaluate the relative importance of OR and negotiation parts of the task allocation process.

1 Introduction

In this contribution, we present a hybrid approach to distributed planning and task allocation problem in the domain of cooperative logistics. The key use case of the suggested planning algorithm is to organize the transport of large quantities of humanitarian aid sent to the disaster area, using the available *resources* – vehicles. The vehicles are operated by several self-interested *transporter agents* who are reluctant to provide *complete* information about their capabilities, services and current status. On the other hand, the transporters are ready to share the necessary amount of information (referred to as *semiprivate knowledge*) and negotiate with others about available delivery services and contracts' acquisition. The effort is planned and organized by *requestors* (humanitarian agents) who decompose the aid distribution into appropriate tasks and allocate these tasks among the transporters in an efficient manner. This planning and task allocation problem (formally specified in Section 2) is obviously computationally very complex and there is no easy way how to identify an optimal solution, even with complete and centralized knowledge.

Existing multi-agent approaches tackle the problem using the negotiation and auction based approaches [1, 2], where each agent retains its own private information and task definition and allocation is negotiation-based. Even though the *Contract-Net-Protocol* [3] (CNP) based solutions are widely used in industrial environment and provide rather efficient mechanisms for distributed task allocation they suffer from important limitations in our application domain:

- undesirable need to implicitly communicate substantial amount of private knowledge (when initiating a contract-net-protocol and when replying with a proposal),
- they found locally optimal solution for a single specific contract while being inefficient for a batches of contracts and
- the need to backtrack and re-negotiate if the problem features a set of interdependent actions

The problems with undesirable knowledge disclosure can be partially addressed by forming *alliances* and other organizational structures that structure the information dispersion in the community. Semiprivate information (see 2.1) shared within the alliance in its effect reduces future requirements for sharing private information outside the alliance [4]. The problems related to optimization of higher number of requests were addressed in parts by extension of the original CNP to $OSCM$-CNP that can always find a globally optimal solution in a finite number of steps [5]. In our work we have combined the concept of semi-private knowledge sharing (as defined in [4]) and Extended Contract-Net-Protocol (ECNP) as described in [1] and further extended towards practical application by [2]. This protocol achieves the result by using the negotiation between the requestor and perspective providers.

When the perspective requestor wishes to solve the task by ECNP, it asks other agents to cover the task completely or at least partially. Agents submit their bids, the best ones are selected and provisionally granted the task. The rest of the task is auctioned again and new auctions are organized until the whole task is covered. If the remaining task can not be covered, the algorithm must achieve consistency by backtracking – revocations of provisionally granted tasks and auctioning new ones, as the tasks may be mutually dependent. For example, in case of transport from A to C via B, we have to allocate either the complete trajectory, or BOTH $A \rightarrow B$ and $B \rightarrow C$. If we provisionally grant the $A \rightarrow B$ and later fail to allocate $B \rightarrow C$, $A \rightarrow B$ must be revoked. Problem gets more complex when we consider the consistency of quantities between dependent actions, parallelism of actions and limitations regarding the use of single resource contributing to several subtasks. Even with a unique central requestor, the planning problem is completely decentralized and requires intensive communication. Consequently, this approach introduces computational performance problems when it plans in large state spaces. Such planners outperform manual planning [6], but mathematical programming techniques will typically offer better performance. On the other hand, mathematical programming-based solutions require centralized knowledge and precise problem formulation. The

agent-based approach brings more flexibility than classical approaches as the agents may combine many sources and types of knowledge to prepare the plan, each agent contributing its knowledge, reasoning and resources. Agents don't need to be aware of each other's resource availability, provided that they are syntactically and semantically interoperable. This allows the agents to avoid explicit disclosure of their private information, while not completely solving the implicit disclosure problem stated above.

The main contribution of this paper is the integration of classical AI and operational research 'heavy-duty' solvers with multi-agent techniques to efficiently address the existing limitations of multi-agent approach, while not compromising the solution flexibility. We argue that the abstract models of collaboration in agent systems as they are now used within the multi-agent system community have severe drawbacks – they are well suited for simple reasoning and limited amount of knowledge, while little scalable. Their performance tends to degrade with increasing problem complexity and shift of the focus from qualitative to quantitative reasoning. Therefore, we suppose that the AI/OR techniques are a very good fit for agent reasoning due to their high performance and little or no scalability problems. The traditional problems related to their application – restrictive applicability conditions (e.g. linearity and certainty) are solved by modern methods [7] and on the other side, acquaintance models [8] provide the necessary knowledge inputs for the model, as well as an efficient mechanisms for its maintenance.

Specifically, the main difference when comparing our approach with ECNP is the separation of planning and task allocation. In the planning phase, we elaborate and merge alternative plans how to accomplish the task (e.g roads to take and intermediary storage locations), so that we obtain a directed bipartite graph with all the alternatives included and merged – an abstract plan as formally defined in section 2. AI planners perform very well in this task. Then, in the task allocation phase, we alternate the use of OR techniques and negotiation to perform the task allocation, as described in Section 3. In Section 4 we empirically evaluate the algorithm by measuring the influence of negotiation step on the quality of the solution.

2 Problem Statement

In the logistics planning problem we consider (see Fig 1), we address the transport of goods from single start location to terminal location[1] using the resources belonging to self-interested agents. Therefore, we must (i) prepare a sequence/graph of actions to perform and (ii) allocate resources to these actions in order to maximize the expected amount of delivered goods. In the formal problem presentation below, we present the problem from the perspective of the

[1] This formal simplification doesn't reduce the generality of our approach - in case of need, we may define formal zero-cost actions between the initial/terminal objective and the real terminal objective for each part of the cargo, provided that we impose appropriate restrictions on these actions.

requestor – the agent denoted A_0 that leads the collaborative planning and task allocation process.

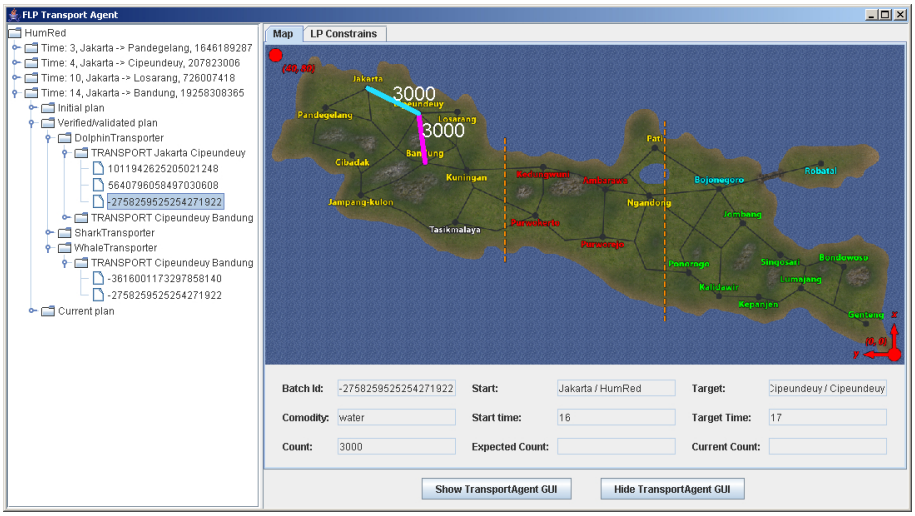

Fig. 1. Domain Map With Plan Example

To describe the plan we follow the approach proposed by [9] and instead of decomposing the plan into the action-state graph, we will describe it using actions and objectives (called objects in [9]). In this representation, the global state is defined as a combination of local state of all objectives.

An ***abstract plan*** (e.g. route plan), typically prepared by AI techniques in the first phase of the planning, is a directed bipartite graph, where one side is composed of ***objectives*** (corresponding to locations in our case), defined by the set $O = \{o_0(start), o_1, o_n(terminal)\}$, with each member defined as $o_i = (prer_{o_i}, allows_{o_i})$. Both the $prer_{o_i}$ and $allows_{o_i}$ are subsets of the action set $Ac = \{a_1, a_2, ...a_m\}$ that forms the other side of the bipartite graph. Ac contains ***actions*** a_i (transports) linking the objectives, where each action is defined as $a_i = (prer_{a_i}, allows_{a_i})$. The sets $prer_{a_i}$ and $allows_{a_i}$ are subsets of O. By definition, we always start from a single *start objective* o_0 (with no prerequisites: $prer_{o_0} = \emptyset$) and terminate in a *terminal objective* that corresponds to the achieved goal state: $allows_{o_n} = \emptyset$. [2]

Batches constitute the cargo that is transported. Each batch p_i from the set P is defined by its size $size(p_i)$ and *type* (liquid, bulk, etc...) that defines the resources (e.g. type of the vehicle) that may carry it. We assume that all batches

[2] Therefore, in our graph, the nodes are defined as $Ac \cup O$, while the directed edges describe the relations expressed in *allows* and *prer* sets of each action or objective. We may also note that the global state of the system is defined by the state of all objectives.

can be split during transport; we denote $p_i^{a_j}$ the part of the batch allocated to action a_j.

The transport problem is being solved by ***agents*** from the set $Ag = \{A_0 ... A_k\}$, including the requestor A_0. Each agent is modelled by other agents as a tuple of its resources $A_i = (res_{A_0}(A_i))$ – in our case, aggregate information about its vehicles. All ***resources***, regardless of their owner agent form a set $R_{A_0} = \{r_1^{A_i}, r_2^{A_j}, r_l^{A_j}\}$, where the super index of each resource denotes the agent to which this specific resource belongs. Each resource is described by a tuple $r_i^{A_j} = (A_j, allowed_{r_i}, cap_{r_i})$, where the A_j denotes the owner agent of the resource, $allowed_{r_i}$ is a set of actions (transports) to which the resource can be assigned, and cap_{r_i} defines its capacity.

Tasks are a result of the planning process. They form a set $T = \{t_{a_1} ... t_{a_m}\}$, and each task corresponds to a single action a_i of the abstract plan. It is defined as $t_{a_i} = (batch_{t_{a_i}}, com_{t_{a_i}})$, where $batch_{t_{a_i}}$ is a set of batches (or their fractions) transported in the scope of the task and $com_{t_{a_i}}$ is a set of ***commitments*** – each commitment[3] $c = (a_i, A_j, r_k^{A_j}, p_l^{a_i}, cap)$ is an assignment of a specific resource r_k (and consecutively its owner A_j) to one partial batch $p_l^{a_i}$ from the set $batch_{t_{a_i}}$ and cap determines the assigned capacity. If the resource capacity allows it, one resource r_k can be committed to more than one batch/action and a single partial batch $p_l^{a_i}$ can be covered by several commitments – in such case, we denote $cap(r_k^{a_i})$ the aggregate size of all commitments from the task t_{a_i} to which the resource r_k is committed. Commitments of agents' resources relative to a single task define a ***team*** working on the task as a set $E = \{e_{a_1}, e_{a_2}, ..., e_{a_m}\}$. Each such team $e_{a_i} \subset Ag$ contains all the agents contributing their resources to the task t_{a_i}. A union of all teams from the set E contains all agents participating at project solution. According to [4], it is equivalent to the coalition.

2.1 Public, Semi-Private and Private Information

In order to plan efficiently, the agents must share appropriate information. The amount of shared knowledge must be carefully sized with respect to self-interestedness of the agents. They are not ready to provide their competitors with more information than necessary. This principle leads us towards definition of several knowledge sharing levels defined in [4]:

- *Public knowledge* is accessible to any agent in the system.
- *Semi-private knowledge* is mutually shared within groups of trusted agents.
- *Private knowledge* is accessible only to the owner agent and never shared with anyone else

While public knowledge includes information about agent identity, existence, location and basic annotation of provided services – *type* of the resources $res_{A_0}(A_i)$ it offers, but without any information concerning their capacity, number or

[3] Formally, until being evaluated and updated by bidding agents, commitments must be regarded to as mere *commitment opportunities*.

restrictions, private knowledge contains the detailed information about its resources, including their individual capacity, restrictions, locations and other information. Semi-private knowledge collects information that facilitates the planning process and enables collaborators to prepare the plans easier than by negotiating all possible options. For each agent A_i, it includes the information about its resources *aggregated* by type and including the restrictions regarding their use on the set Ac – typically, we include a set of roads that this specific vehicle/group can cover. Such compromise provides enough knowledge for the first stage of the planning process, and detailed task allocation is then finalized in the course of negotiation without exposing more data than necessary[4].

3 Algorithm Presentation

This section provides an overview of the planning algorithm we suggest, combining the social model and linear programming planner with bounded and well-focused negotiations in the later stages of the process. The planning process proceeds as follows (see also Fig 2):

1. **Initial Planning**: Requestor uses its social knowledge and planning capabilities in order to prepare the initial plan. This happens in two phases:

(i) abstract plan construction and
(ii) task allocation to the agents.

2. **Local Plan Evaluation**: Initial plan is evaluated by the respective agents:

(i) the members evaluate the plan and match the proposed commitments with their available resources and
(ii) make an attempt to trade the proposed commitments within teams working on the same tasks to optimize the allocation of their resources.

3. **Coherence & Verification**: The requestor incorporates the proposals, including the traded tasks information, into the task allocation problem from the initial planning phase and solves the problem again.

4. **Plan Execution**: Final commitments are received by members, may be swapped and the plan is executed.

3.1 Initial Planning

In the first phase of the plan, we assume that the requestor (denoted A_0 and materialized as a humanitarian agent in our scenario) has a goal to accomplish and is obliged to cooperate with other agents. It uses its social knowledge to draft a preliminary plan in the following steps.

[4] Note that the sets R as perceived by various agents are not identical due to the fact that they don't have the access to the same information.

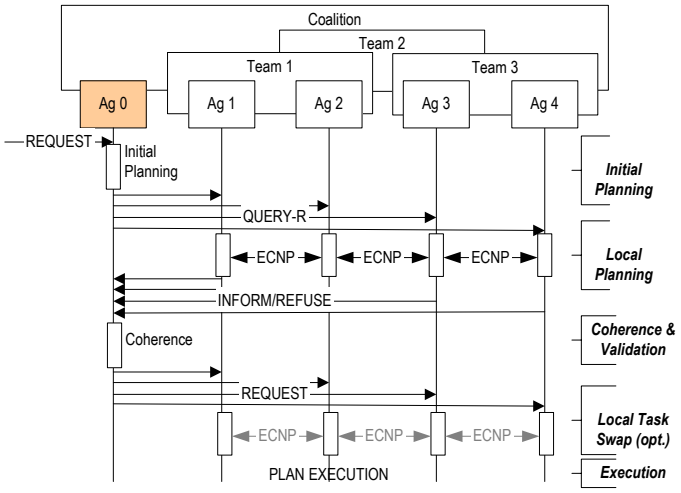

Fig. 2. Overview of the protocol phases: Agent A_0 is a requestor and has decomposed the global task into three tasks.

Constructing the Abstract Plan. The first step is a preparation of the abstract plan – an action-objective bipartite graph capturing the relationship between initial and terminal objectives (states). This graph typically describes and integrates several alternative solutions of the abstract planning problem. Shared objectives of these solutions allow flexible task assignments to actions from different original sub-plans and their seamless combination. Abstract plan must contain at least one path connecting the initial and terminal objective – if no such path can be identified, agent A_0 is unable to solve the planning problem.

Constructing the abstract plan is a computationally exponential problem in complex domains. Recent advancements in the field of AI planning provided very efficient techniques for constructing the plans in reasonable amount of time such as GraphPlan [10], SAT-Plan or their variants. These techniques implement a sophisticated breadth-first search based on expansion of the bipartite graph or iterative propositionalization of the planning problem. In the experiments presented in Section 4, we have used a pre-prepared static plans in order to achieve repeatability between various configurations tested. In the general case, the choice of the appropriate algorithm for this stage of planning depends on the complexity of the domain and specific requirements regarding computational efficiency and desired solution quality.

Task Allocation. Once an acceptable abstract plan is established, requestor proceeds with the allocation of batches and resources to individual actions in the plan, while respecting the constraints defined in the objectives. Note that for sake of computational efficiency, some actions and objectives from the abstract plan can be removed during this phase if there are no resources or batches to allocate to them. Then, we use a linear programming (or its variants [7]) that either

provides an acceptable initial task allocation T, or identifies the constraints that prevent the agent from finding the solution.

The constraints we define for the problem are the following. The first equation expresses the node equilibria - conservation of goods in each node.

$$\forall o_i \in O \setminus \{o_0, o_n\}, \forall p_j \in P : \sum_{a_k \in prer(o_i)} size(p_j^{a_k}) = \sum_{a_l \in allows(o_i)} size(p_j^{a_l}) \quad (1)$$

The initial node has a simpler relation, declaring that we can't take away more cargo than available:

$$\forall p_j \in P : size(p_j) \geq \sum_{a_l \in allows(o_0)} size(p_j^{a_l}) \quad (2)$$

while the terminal node doesn't introduce any constraint.

Furthermore, for each action a_i (elementary transport) and each batch p_j, we must ensure that the commitments cover the whole partial batch $p_j^{a_i}$ ($size(p_j^{a_i} \leq p_j)$ due to the possible parallelism):

$$\forall a_i \in Ac, \forall p_j \in P : p_l^{a_i} = \sum_{c \in com_{t_{a_i}} : batch(c) = p_l^{a_i}} cap(c) \quad (3)$$

then, we must also make sure that no resource is used beyond its capacity. To do so, we must determine the sets of actions that may share a resource in the scope of the plan. Therefore, we introduce an ordering on the set of actions Ac in the abstract plan. This partial ordering relationship is defined by causality in the plan: we say that $a_i < a_j$ iff a_j belongs to the transitive closure of the set $allows_{a_i}$, meaning that a_i shall be completed before a_j starts. Partiality of the ordering relation defines the relation of resource allocation *compatibility* between two actions. Actions a_i and a_j are compatible, iff $a_i < a_j$ or $a_j < a_i$. Otherwise, they are incompatible and can't share single resource. The set $incompatible(a_j)$ used below includes all actions from Ac incompatible with the action a_j.

$$\forall r_i \in R \forall a_j \in Ac : cap(r_i) \geq \sum_{a_k \in a_j \cup incompatible(a_j)} cap(r_i^{a_k}) \quad (4)$$

In practice, the above relationships generated for different actions is often identical and duplicate restrictions are removed from the problem specification for sake of performance. Alternatively, we may decide that the resources can be used only for single action within the plan, and then the set $a_k \in a_j \cup incompatible(a_j)$ encompasses the whole Ac. This restriction can reflect maintenance and required downtime requirements.

Besides the restrictions, we need to set-up the **utility function** for which we optimize:

$$U_m = \sum_{p_i \in P} size(p_i^{o_n}) \quad (5)$$

where $p_i^{o_n}$ denotes the part of the batch p_i delivered to the terminal objective o_n and $ag(c)$ the agent committing to c. This simple function maximizes the amount of the cargo delivered to the objective. Extension of this function to more specific forms that may include transport prices and/or trustfulness of individual agents is straightforward, even if the formulation must be kept strictly linear. Linearity limitations can be partially addressed by iterative application of the solver and the use of FLP methods [7] that allow us to handle the uncertainty in social knowledge. These methods are especially relevant when we also consider the trustfulness of individual agents.

Once the solution of the above problem is identified, requestor determines all perspective participants (owners of resources assigned to various tasks) and queries each perspective member whether it is capable and willing to participate. Therefore, each perspective participant A_i is sent a structure: $cma_{A_i} = (coalmem, assign)$, where the set $coalmem$ lists all coalition members and the set $assign$ lists the relevant information about tasks the agent's resources are assigned to, defined as $(e_{a_j}, com_{t_{a_j}}(A_i))$, where j is an action (task) index and $com_{t_{a_j}}(A_i)$ are commitments suggested to agent A_i on task t_{a_j}.

3.2 Local Plan Evaluation

When the agents A_i (selected by the requestor in the previous step) receive the proposals from the requestor, they must use their private knowledge to create the bid reflecting their preferences and local situation. At this level, we handle several issues that are ignored by the requestor's first-level planning – resource granularity (unknown to the planning agent due to the privacy issues) and relations between the resources assigned to different tasks. In the first round, each agent assigns its resources to the commitments that are the best fit for available resources, trying to cover all commitments. Then, it will offer the excess capacity of the resources assigned to the task t_{a_j} to all members of the team e_{a_j} using the multi-phase auction mechanism described in [1]. This step is designed to eliminate the resource allocation inefficiencies that are due to the possible requestor's lack of knowledge about actual resources or a side effect of selected planning method. More formally (see also Fig. 3), to start the the negotiations, each agent A_i working on task t_{a_j} broadcasts a CFP message containing its free capacity to all team $e_{t_{a_j}}$. If the other team members are interested in using this capacity for the task they were allocated, they submit their bid. Agent A_i selects one or more bids and answers them with a temporary grant, making them binding *for the bidders*; other are refused. When the agent A_i participates in several teams, it can now reshuffle its resources between the tasks to use them in an optimal manner. Once the resource reallocation is terminated, all compatible temporary grants are confirmed, while the others may be refused (In case the agent has replaced the original resource with a lower-capacity one.). If appropriate, agent can now offer the new free capacity for trading using the same protocol.

Note that the auctioning and negotiation takes place only within the single task team, therefore minimizing the knowledge dispersion and communication

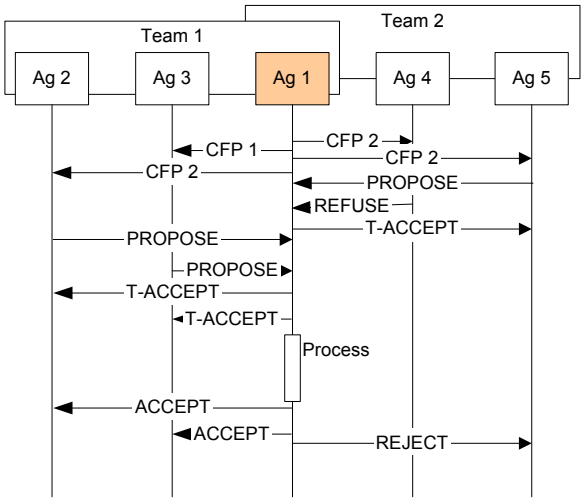

Fig. 3. Use of the ECNP to allocate agent's resources across two different teams. Agent A_1 first temporarily accepts the offer from A_5, but later on finds a better resource allocation and prefers to commit larger resource to *team 1*. Therefore, it rejects the bid from A_5.

load. On the other hand, agents may therefore miss a better task allocation. Once the negotiation is finished, all team members send their answers to the requestor. The answer is a list of commitments that are actually *binding* for each agent, but may differ from those originally assigned to the agent as: **(i)** the agent is not always able to cover the whole assigned commitment and commits only to a part of the original commitment or **(ii)** it notifies the requestor about the transfer of the whole commitment or its part to other team member (this member lists this commitment in its turn as covered). When the agents submit their binding commitments to the requestor, they have an alternative to offer the free capacity of the resources they've allocated to the task to the requestor - the requestor may include use it to cover other batches from the same task, as specified by relation 6. While this remains an attractive optimization feature, this approach has two major drawbacks – the requestor can easily guess the capacity of agent's resources and the free resources can not be used on another task. Positive influence of the negotiation in this step is clearly visible while analyzing the experimental results presented in Section 4.

3.3 Coherence & Verification Phase

In this phase, requestor receives the answers from the participants and must re-combine them into a globally coherent plan. As the initial planning has produced a coherent plan, the plan is coherent when all proposed commitments

were covered by members. If not, the requestor must add all updated commitments/refusals from the agents to the initial plan and perform the new calculation to make sure that the condition 1 is valid for the final plan.

Updated commitments are included as follows (refusals or previously unassigned commitments are considered as commitments with 0 capacity):

$$\forall a_i \in Ac, \forall r_j \in R : cap(r_j^{a_i})^{prop} \geq cap(r_j^{a_i})^{final} \tag{6}$$

It is at this stage of planning process when we also detect the failure to execute the plan altogether – the proposals (or actually refusals) from the members may be mutually incompatible. If the requestor manages to find an acceptable planning outcome, it prepares the final commitments (with the quantities assigned that are less or equal to the binding ones proposed by members) and re-submits them to the participants.

3.4 Plan Execution

As the proposals by the agents were binding, participating agents shall be all able to start performing the assigned tasks immediately. Alternatively, when the final commitments are lower than the ones they have proposed, they may change their resource allocation or trade the assignments with their peers in the team in the same way as in the Local Plan Evaluation phase, provided that they manage to honor their commitments.

4 Experiments

In order to evaluate the significance of team-wide negotiation phase (e.g during Local Plan Evaluation), we have compared the performance of the algorithm with and without this feature. The experiment provides a comparison of our distributed hybrid approach with centralized planner working with the same information, while the role of the participating agents is reduced to the allocation of tasks to their specific resources (vehicles).

To perform the experiment, we have used the ACROSS scenario [11] based on $\mathcal{A}$-**globe** multi-agent platform. In this scenario, we simulate a population dispersed in communities on an island. These communities need to trade their production to satisfy their needs, creating the demand for transport. This demand is covered by **Transporter agents**, representing transport companies owning one or more vehicles, each company with its own private preferences regarding the cooperation rules, transport actions it can perform and vehicles of various type and capacity. Into this setting, we introduce a humanitarian catastrophe that partially disables a local production in a predetermined area. **Humanitarian Agent** then tries to address the emergency using its own stock of goods, that must be transported to the affected area. Transport is organized by the Humanitarian Agent as a requestor (with centralized planning capability) and Transporters, who provide their resources.

The scenarios we compare differ by the abilities of the plain transporter agents - in one case, they only use their own resources to cover the assigned transport requests. In the second case, they perform the peer-to-peer ECNP-based negotiation as described in Section 3.2. In the result of our experiments, we will see to which extent this extension increases the performance, while keeping the explicit (shared social knowledge) or implicit (disclosed as a negotiation side effect) private information disclosure low.

As a reasonably domain-independent measure of planning quality, we use the resource utilization efficiency: the ratio of the actual cargo loaded on the individual vehicles compared to their capacity. Such metrics evaluates the whole task allocation part of the planning process, from the initial planning down to the assignment of batches to individual vehicles within transporter fleet. On the other hand, it doesn't reflect the abstract planning phase quality. We consider this as a correct decision, because we concentrate on task allocation and the abstract planning is (i) highly domain dependent and (ii) easy to validate independently of the rest of the planning process. The experiments were performed in the situations differing by the requested quantity of the cargo to transport and the results were aggregated. However, in all cases, the demand exceeds the supply of available resources given their limitations.

As we can see in Fig 4, the overall behavior of the system is stable (except for a brief startup period not represented in figure) and the differences between the quality of the solution are relatively minor, with the use of the ECNP negotiation phase as a biggest differentiator. Even if the difference due to the ECNP use is only about 3 percentage points, this modest improvement is caused by the fact that in most cases, there is only a limited potential for improvement. Most contracts require simply the use of the full available capacity, and the ratio of plans where the negotiation can improve the results is relatively small. In the similar manner, we can see that the difference due to the introduction of more or less restrictive approach to resource planning is negligible, less than a fraction of percentage point. However, we feel that even these improvements are crucial. Besides their clear economic interest, they improve the planning process by pushing the solution closer to Pareto-optimality and while the heavy duty methods still do most of the planning work (as we can clearly see from the results), the negotiation process reaches the locally optimum solution that can not be further easily improved. This feature is important for industrial applications, as the clients are often not satisfied by solutions that are not locally optimal, even if the overall result approximates the global optimum close enough.

Please note that with respect to the framework described in this paper, we have omitted the last phase of the ECNP negotiation and used a fixed base of abstract plans to achieve repeatability between experimental runs.

5 Conclusions and Future Work

In this paper, we have presented an algorithm for cooperative task allocation in an environment with self-interested agents. While our algorithm remains a

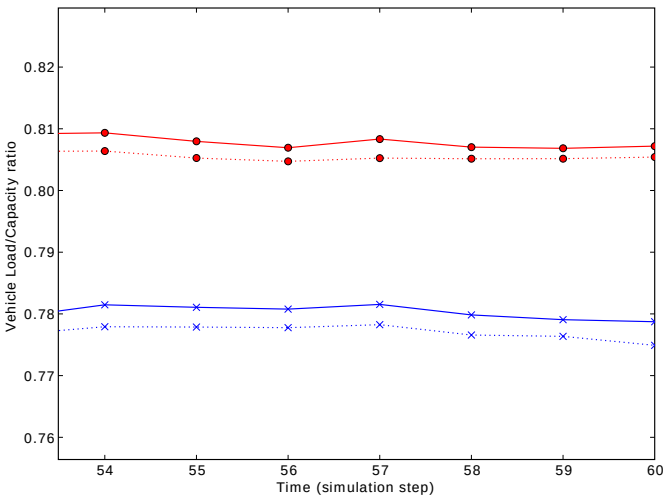

Fig. 4. Performance of the various variants of the algorithm evaluated by resource usage. Full lines represent the solution with Formula 4 in its nominal form, while the dotted lines present the planner performance when all actions are considered incompatible for resource sharing aspects. Values with circular points were obtained with ECNP, crossed points without.

distributed solution, it uses the concept of social knowledge to delegate important part of the planning to single agent in the community. In the same time, the individual agents still retain control of their resources and protect their private information.

The algorithm presented in this paper has several properties that make it interesting for industrial applications:

Reduced communication is a result of the use of the social knowledge in the *initial planning* step of the algorithm. Instead of several rounds of auctions, action decomposition and backtracking, requestor uses its social knowledge to compose balanced task teams and pre-assign commitments to each potential member.

Increased parallelism is also a consequence of the initial planning introduction and separate abstract planning step. Pre-assigned teams of agents negotiate in parallel on the problem that was already decomposed and tentatively allocated, instead of sequential negotiation in the ECNP, where the tasks are allocated successively, one after another, and frequent backtracking is required.

Iterative reduction of the solution space is another key feature – each step of the planning, centralized or distributed, reduces the solution space. Initial planning performs the greatest reduction, as the actions/tasks are selected, resources pre-allocated and agent teams created. Local planning phase then further clarifies resource allocation and team composition and the results of this phase are incorporated as additional restrictions for the planning problem solved in the

coherence and validation phase – we ensure that any overall solution will respect the commitments received from participating agents and can be executed. The final solution is restricted by the boundaries of the initial planning with additional restrictions. If the plan can not be implemented due to the member refusal or resource incompatibility, the situation is detected in the coherence planning step. In the algorithm as suggested, we don't allow any backtracking (except the team-scale negotiation), increasing the outcome predictability. On the other hand, the algorithm as presented doesn't guarantee that the result it returns will be the optimal plan. We don't consider this as a serious drawback, because none of the comparably efficient algorithms currently in use can guarantee such result.

Thanks to the above features, our solution offers **fast response times**. We assume that the centralized planning steps (e.g. initial planning and coherence phase) can be performed rapidly in most practical setups, as the separate tasks of route finding (abstract planning) and linearized task allocation can be solved in polynomial time. During the negotiation and local plan evaluation phase, the size of the problem is already significantly reduced, and the agents only allocate the assigned (or traded) tasks to available resources. While this problem is NP complete in general case of indivisible batches, its relatively small size and batch divisibility make this problem easily solvable by application of appropriate heuristics (as in our implementation) or standard search algorithms[5]. In practice, as we always restrict the time to answer in the each stage of interaction protocol (as a protection against communication failures), the time to return the solution is determined by twice the negotiation timeout plus the time required to solve the centralized planning.

In our future work, we plan to benchmark the results against an implementation of the ECNP-based planning and task allocation mechanism. However, such benchmark presents several important challenges to tackle – we must make sure that the implementation of the ECNP is good enough for a fair comparison and that the underlying planning problem (or a set of problems) will not disadvantage any of the evaluated solutions.

Acknowledgment

Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-04-1-3044. The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

[5] Required quality of this algorithm depends on the number of available resources and assigned batches (for each cargo type), but even relatively large problems can be solved using mixed-integer programming approaches like branch-and-bound algorithm [12].

References

1. Fischer, K., Muller, J.P., Pischel, M., Schier, D.: A model for cooperative transportation scheduling. In: Proceedings of the First International Conference on Multiagent Systems., Menlo park, California, AAAI Press / MIT Press (1995) 109–116
2. Perugini, D., Lambert, D., Sterling, L., Pearce, A.: A distributed agent approach to global transportation scheduling. In: The 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003), Halifax, Canada (2003) 18–24
3. Smith, R.G.: The contract net protocol: High level communication and control in a distributed problem solver. In IEEE Transactions on Computers **C-29**(12) (1980) 1104–1113
4. Pěchouček, M., Mařík, V., Bárta, J.: A knowledge-based approach to coalition formation. IEEE Intelligent Systems **17**(3) (2002) 17–25
5. Sandholm, T.: Contract types for satisficing task allocation: I theoretical results. In: Proceedings of the AAAI Spring Symposium. (1998)
6. Perugini, D., Lambert, D., Sterling, L., Pearce, A.: Agent-based global transportation scheduling in military logistics. In: AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, Washington, DC, USA, IEEE Computer Society (2004) 1278–1279
7. Carlsson, C., Fullér, R.: Fuzzy Reasoning in Decision Making and Optimization. Physica Verlag, Springer, Heidelberg (2002)
8. Pěchouček, M., Mařík, V., Štěpánková, O.: Role of acquaintance models in agent-based production planning systems. In Klusch, M., Kerschberg, L., eds.: Cooperative Infromation Agents IV - LNAI No. 1860, Heidelberg, Springer-Verlag, Heidelberg (2000) 179–190
9. Witteveen, C., Roos, N., van der Krogt, R., de Weerdt, M.: Diagnosis of single and multi-agent plans. In: AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, New York, NY, USA, ACM Press (2005) 805–812
10. Miguel, I., Jarvis, P., Shen, Q.: Flexible graphplan. In Horn, W., ed.: Proceedings of the Fourteenth European Conference on Artificial Intelligence. (2000) 506–510
11. Šišlák, D., Rehák, M., Pěchouček, M., Rollo, M., Pavlíček, D.: $\mathcal{A}$-**globe**: Agent development platform with inaccessibility and mobility support. In Unland, R., Klusch, M., Calisti, M., eds.: Software Agent-Based Applications, Platforms and Development Kits, Berlin, Birkhauser Verlag (2005) 21–46
12. Lawler, E.L., Wood, D.E.: Branch-and-bound methods: A survey. Operations Research **14**(4) (1966) 699–719