Inefficiencies in Task Allocation for Multiagent Planning with Bilateral Deals

Mathijs de Weerdt  
Delft University of Technology  
m.m.deweerdt@tudelft.nl

Roman van der Krogt  
Cork Constraint Computation Centre, UCC  
roman@4c.ucc.ie

Abstract
Distributed planning in a multiagent environment may give rise to inefficiencies. We study this effect focussing on the task allocation problem. We show that in the worst case, the result of a multiagent approach can be arbitrarily bad in theory when recontracting and multilateral deals are not allowed. This is a more precise result than was previously known, which was that we are not guaranteed to find the optimal solution. We show that the sources of this disappointing result are the impossibility to come back on (bad) contracts in combination with either selfish agents, or agents that have incomplete information on potential costs. Furthermore, we show some preliminary experimental results of the effect of these causes on the optimality of a solution for multiagent task allocation. Interestingly, none of the experiments exhibit the very negative outcomes that are predicted by the theory. Although it is too early to draw conclusions, this might indicate that in practical situations, the circumstances that lead to the theoretical results are very unlikely.

Introduction
Multiagent planning is planning for a group of agents. This can be done either centrally or distributedly. When the agents in question are working closely together, a central approach is often the most preferred as it allows the most efficient solution to be reached. An example of such an approach is proposed by (Kalofonos & Norman 2004). However, in some cases the agents in question are not prepared to exchange their private information or inform a third party, and thus a distributed planning algorithm has to be employed to find a solution. This is the approach that, for example, (Durfee 1999) proposes. Unfortunately, there are several factors that make it harder to reach the optimal solution in this case. For example, because the agents do not wish to disclose all their information, other agents need to base their decisions on incomplete information. Also, agents are inclined to make selfish decisions, rather than doing what is best for the group. To investigate this phenomenon, this paper takes a closer look at these situations, focussing on the task allocation part of the multiagent planning problem.

The well-known task allocation problem asks to assign tasks requiring certain capabilities to agents (Shehory & Kraus 1998). It is equivalent to the multidimensional 0-1 knapsack problem, which is NP-complete (Kellerer, Pfler schy, & Pisinger 2004).\(^1\) In the case of a single auctioneer, this problem can be solved optimally (although in the worst case taking exponential time) by a centralised Vickrey-Clarke-Groves (VCG) mechanism (Krishna 2002). In a multiagent setting, however, we cannot always assume that all tasks originate from one auctioneer. Therefore, we consider the more natural multiagent task allocation problem where the tasks are initially owned by some of the agents. Not all of these tasks can be completed fully by the owner agents (managers). By making contracts with other agents (contractors), the initial (infeasible) allocation can be changed into an allocation where as many tasks can be executed as possible. Recent studies have shown that under certain assumptions, any sequence of multilateral deals that are mutually beneficial (with side payments) will eventually reach an optimal solution (Sandholm 1998). Moreover, even without side payments, such a sequence can reach a Pareto-optimal solution (Endriss et al. 2003). Simply allowing recontracting can lead to repeating cycles of making and breaking contracts. However, there are protocols that prevent such deadlock situations. For example, the levelled commitment protocol introduces penalties for breaking contracts (Sandholm & Lesser 2001).

These results are interesting. However, making a sequence of deals would mean that contractors are allowed to find other agents to do parts of the job, who in turn also may try to find other agents, etc. This is called recontracting (Sandholm 1996).\(^2\) In many situations the owner of a task would not like this to happen. In practice, it is often part of contract between two agents that one of them really is going to do the job himself. (Sandholm 1998) proves that

\(^1\)The multidimensional integer knapsack problem admits a polynomial time approximation scheme (PTAS) (Chandra, Hirschberg, & Wong 1976), but no fully polynomial time approximation scheme (FPTAS).

\(^2\)A note on terminology used in task allocation research. Suppose that agent \(a_1\) agrees to undertake (commits to) a task \(T\) for another agent \(a_2\).

- **decommitment** is the act of \(a_1\) informing \(a_2\) that it can no longer uphold its part of the agreement
- **recontracting** is the act of \(a_1\) trying to find other agents to undertake (part of) task \(T\). It is not used for the activity \(a_2\) has to undertake after \(a_1\) decommits.
Definition 1. Let \( A \) be a set of agents, and let \( O \) be the set of all operators that can be executed by one or more of these agents. For each agent \( a \in A \) we have a multiset of operators \( O_a \subseteq O \) that it is able to execute. The execution of an operator \( o \in O_a \) infers some cost, given by a function \( c : A \times O \rightarrow \mathbb{R}^+ \).

Agents may own some tasks. A task requires the execution of some operators to be completed successfully.

\[ \text{Agents may own some tasks. A task requires the execution of some operators to be completed successfully.} \]

Definition 2. Each agent \( a \in A \) owns a set of tasks \( T_a \). Each task \( t \) out of a set of tasks \( T = \bigcup_{a \in A} T_a \) is defined by a multiset of operators \( O_t \subseteq O \) required to fulfill \( t \). Furthermore, each task has a reward for completing this task: \( v : T \rightarrow \mathbb{R}^+ \).

Each task requires the operators of one or more agents, such that assigned operators fulfill all requirements for the task. The exact assignment of subtasks to agents is defined by a task allocation.

Definition 3. A task allocation defines for each task \( t \) and for each agent which operators of this agent are used for \( t \). A task allocation is thus defined by the following map \( TA : T \times A \rightarrow 2^O \).

For a task allocation we define two properties:

- A task allocation is correct if each agent does not allocate more operators than it has: for each \( a \in A \) the following multiset relation holds: \( \bigcup_{t \in T} TA(t, a) \subseteq O_a \), and
- a task allocation is complete for a certain task \( t \) if all allocated operators together are sufficient: \( \bigcup_{a \in A} TA(t, a) \supseteq O_t \). We denote the set of all completed tasks by \( T' \).

We can now define the utility of a task allocation.

Definition 4. The social welfare of a task allocation \( TA \) is defined by the reward of all completed tasks \( T' \) minus the costs of all used operators:

\[ \sum_{t \in T'} (v(t) - \sum_{a \in A} \sum_{o \in TA(t, a)} c(a, o)). \]

The task allocation problem is the problem of finding a correct task allocation \( TA \), preferably such that the social welfare is maximised.

In general, in a multiagent system, we assume agents make individually rational (selfish) decisions. Moreover, in many real-life settings they do not give information to other agents unless strictly necessary. For such applications where agents are not allowed to change contracts they have previously agreed to, no mechanism exists that can guarantee an approximation of the efficient solution, as we show in the next section.

Mechanisms without recontracting

When we do not allow recontracting, there are a number of reasons for the eventual solution to get sub-optimal. Here, we focus on two that may lead to the solution becoming arbitrarily bad. Firstly, the result of a multiagent task allocation mechanism can be very inefficient, because an alternative for a task is chosen by an agent without considering the quality of all alternatives, simply because the quality of these alternatives is not known.

Theorem 1. In the task allocation problem without recontracting, when agents have no precise information on the costs of operators of other agents required for their tasks, the result can be arbitrarily bad in the worst case.

Proof. Let a factor \( \alpha > 1 \) be given. We prove this theorem by constructing an example where a wrong decision can lead to costs that are \( \alpha \) times the optimal solution. Consider the following problem (cf. Table 1). Agent \( a_1 \) has two equally
rewarding tasks $t_1$ and $t_2$ that can be realised by $\{o_1, o_2\}$, and $\{o_1, o_3\}$, respectively, and it has operators $O_1 = \{o_1\}$. Agent $a_0$ on the other hand has no tasks, but $O_2 = \{o_2, o_3\}$. Assume that $c(o_2, o_2) = 1$, and $c(o_2, o_3) = \alpha$ and that agent $a_1$ does not know this difference. It is now straightforward to see that if agent $a_1$ decides to auction $t_2$ (first), then $t_1$ can not be fulfilled anymore, and the resulting costs will be $\alpha$. This is $\alpha$ times as bad as the optimal solution. 

Secondly, inefficient results may occur, because agents that get the task initially, make locally optimal (egoistic) decisions when selecting which task to fulfill first.

**Theorem 2.** In the task allocation problem without recontracting where agents take locally optimal (egoistic) decisions, the global result can be arbitrarily bad in the worst case (even if the agents know the costs of the alternatives of a task in advance).

**Proof.** Let a factor $\alpha > 1$ be given. We prove this theorem by constructing an example where an egoistic choice for an auction can lead to a solution that is more than $\alpha$ times worse than the optimal solution. Consider the following problem (cf. Table 2). Agent $a_1$ has two tasks: $t_1$ can be realised by $o_1$ and $t_2$ can be realised by $o_2$, both with reward $2$. Suppose agent $a_1$ knows that agent $a_2$ has tasks $t_3$ and $t_4$ with reward $2 + 2\alpha$ that can be realised by $\{o_1\}$, and $\{o_2, o_3\}$, respectively. Agent $a_1$ has actions $O_1 = \{o_2\}$, and a third agent ($a_3$) has actions $O_3 = \{o_1, o_3\}$. Assume that $c(o_3, o_1) = 1$, $c(o_1, o_2) = 1$, and $c(o_3, o_3) = 2\alpha$. Assume that agent $a_1$ first gets a chance to auction its task. There are two scenarios:

1. If agent $a_1$ chooses to fulfill $t_1$ (by $o_1$), agent $a_3$ will participate. Then agent $a_2$ can only deal with $t_4$ by hiring both $o_2$ (from agent $a_1$) and $o_3$ (from agent $a_3$). In this case the total profit is $2 + 2 + 2\alpha - 1 - 1 - 2\alpha = 2$ and tasks $t_2$ and $t_3$ are not fulfilled.

(a) If agent $a_1$ decides to deal with $t_2$ (by $o_2$), it can do this itself, and then $t_3$ (with $o_1$) can be done by agent $a_2$ with help of $o_3$, resulting in a total net profit of $2 + 2 + 2\alpha - 1 - 1 = 2 + 2\alpha$. In this case task $t_1$ and $t_4$ are not fulfilled.

Agent $a_1$ prefers scenario 1, because it will receive not only part of the profit for $t_1$, but also part of the profit of the more interesting task $t_4$. (For example, if we assume that the division of profit per task is done according to the Shapley value (Shapley 1953), agent $a_1$ will receive $1 + \frac{1}{2}$ instead of just 1 in scenario 2.) This is in spite of the fact that scenario 1 is $\frac{2 + 2\alpha}{2} = 1 + \alpha > \alpha$ times as bad as scenario 2 for the group as a whole.

To summarise, because agents cannot decommit or recontract their contracts (cf. the footnote on page 1), contracts are bilateral, and

1. they do not have complete information, or
2. make selfish decisions,

any mechanism for multiagent task allocation under these assumptions can end up with a very bad allocation, in theory. The question now is: how strong are the consequences of these results in practical cases?

**Experiments**

To know whether the worst cases mentioned in the proofs above really occur in practice, we would need to study as many (ideally all) realistic problem settings as possible. Obtaining such instances is, however, still future work. In this paper we study the effect of the reasons for bad results by generating random instances of the multiagent task allocation problem.

The random problem generator takes as input the following parameters:

- the number of agents, operators and tasks, as well as
- probability distribution functions (pdfs) of operators and tasks over the agents,
- a pdf to determine the operators required for the tasks, and
- pdfs for the operator costs and task rewards.

Given these pdfs, a multiset of (ground) actions $O$ is generated, and each agent $a \in A$ is assigned some of these ground actions $O_a \subseteq O$ (a multiset, i.e., with overlap). Furthermore, the costs are generated by the pdf for operator costs for each action $o$ an agent combination: $c(o, a)$. Finally, the random problem generator creates a set of tasks $T_a$ for each agent $a \in A$, and selects for each task $t \in T = \bigcup_{a \in A} T_a$ the set of ground actions required to fulfill this task, and a reward $r(t)$.

To be able to study the effect of the reasons given in the previous section, we constructed a specialised search algorithm that can find centralised, optimal solutions for the gen-
generated problems. This gives us a figure to compare the multiagent solution methods to. To be as fair as possible, we did not implement separate solvers that focus on one of the possible reasons. Instead, we opted for a method with which we can put restrictions on the search that our search algorithm performs. These restrictions are carefully crafted to simulate the effects of a distributed search. By enabling or disabling restrictions, we can study each of the reasons in isolation. In the remainder of this section, we present some of our findings. In particular, we focus on two different cases (in addition to the optimal, central case):

1. the case where agents make selfish decisions. The restriction in this case is that we do not optimise the maximum profit of the whole group of agents, but the maximum profit for the auctioning agents. This means that an agent will choose the task to allocate that is the most profitable for him.

2. the case where agents have no information on the costs of operators. The restriction here is that agents are not given the precise costs of the operators of fellow agents.

Instead, it is given an approximation that it believes to be true. Each agent bases its decision which task to auction first upon this incomplete information.

In both cases, we assume individually rational agents. That is, agents accept to take part in a task only if the reward they are getting offsets their costs. To ensure this property, we use the Shapley value (Shapley 1953) to fairly allocate the total rewards of the tasks over the agents that cooperate to undertake it.

For the initial experiments done for this paper we generated 400 random problems with 1 up to 80 tasks with the following properties (uniformly random distributions unless noted otherwise): 8 agents, between 50 and 200 reward per task, between 3 and 6 operators required per task, between 5 and 15 operators available per agent, and a uniformly random initial allocation of operators and tasks to agents. The operator costs were drawn from the exponential distribution with $\lambda = 0.02$. Figure 1 shows the social welfare of the two methods mentioned above and the maximum social welfare obtained by a centralised complete search. Each point in this
From the graph we can see that making locally optimal decisions does not lead to a huge decrease of performance. The solutions achieved come within 87% of the optimal solution on average. Concerning the decisions that have been made without full information, however, we see that this significantly worsens the found solutions (on average around 67% of the optimal solution). Moreover, this increases for problems with more tasks (and more potential social welfare). This can be explained by the number of tasks per agent. If each agent has at most one task, local heuristics for task ordering do not have any effect, whereas in situations where agents have 10 tasks on average, it does clearly matter in which order these tasks are auctioned.

Remarkably, however, for none of the instances we see the extreme behavior predicted by the previous section. Although it is too soon to draw any conclusions from this observation, it is a hopeful indication that problem instances where results can get arbitrarily bad are very rare.

Discussion

We started the investigation described in this paper to come to understand the issue of inefficiencies for the multiagent planning problem. As a first step, we focussed our attention on the multiagent task allocation problem. This is that part of the solution to a multiagent planning problem that describes which agents will undertake (to plan for) which tasks. From the theoretical analysis we can conclude that no mechanism exists that can approximate the optimal solution for the task allocation problem with bilateral deals when recontracting and decommitment is not allowed. This is a bit unexpected, because we know that for the multi-dimensional knapsack problem a polynomial time approximation scheme (PTAS) exists (Kellerer, Pferschy, & Pisinger 2004). This can be entirely attributed to the fact that we are studying a multiagent variant of this problem. When agents are involved that are individually rational, the problem gets harder, because of lack of information, and a reduction of the possible decisions.

Let us look at this difference from the point of searching. First consider the situation where all information and control is available centrally, either because all agents are cooperative, or because of the use of an incentive compatible mechanism like VCG. In this case the problem is equivalent to the multi-dimensional knapsack problem, so we know that there exists a PTAS. Assuming P=NP, we also know that there are instances for which we need exponential time to find the optimal solution. For example, this problem can be optimally solved by an A*-search (Hart, Nilsson, & Raphael 1968) through the combinations of tasks to fulfill.

A mechanism to find a solution in a multiagent context without recontracting, can also be seen as a search. In this case, however, the search is restricted. Firstly, when a contract between agents is made, this cannot be changed anymore. In search this means that there are choices from which back-tracking is not allowed. Moreover, agents make selfish decisions, so certain choices in the search space are not explored. For example, if decisions have to be made by an agent that will be worse off for particular choices. Finally, when agents have incomplete information, the available information for making these choices is limited. Also in this light, it is no wonder that such a mechanism may turn up with arbitrarily bad results in some instances.

Therefore, when designing a system, we should try to prevent these situations as much as possible. One way is to introduce a mechanism such that agents can get back on made agreements, such as levelled commitment (Sandholm & Lesser 2002), or by introducing a pre-commitment phase (Aknine, Pinson, & Shakun 2004). When this is not possible, we should try to

- give agents at least a realistic estimate on the costs of actions and/or resources, and
- let agents behave a bit more socially, such that they refrain from making egoistic decisions that lead to a much lower social welfare.

We have seen that the results of multiagent task allocation can be arbitrarily bad in theory. However, in practical applications, it does not seem that bad. Why is that? We think this can be explained from two facts. Firstly, in reality, the costs of actions are usually approximately known, and secondly, for most situations, there is no strict limit on the available operators. When demand is high, price increases, but the examples in the proofs show that problems mainly arise when tasks cannot be fulfilled because of limited availability of operators. However, to go beyond such speculative arguments, and to be able to take advantage of our results, we need to thoroughly examine realistic cases.

Besides studying more realistic cases, we would like to find out under which conditions problems are such that a multiagent approach may lead to very inefficient solutions. Does a phase transition effect occur, similar the threshold phenomenon in satisfiability (Achlioptas 2001) and many other NP-hard problems? If so, we are very interested to discover such an effect.

Just as importantly, we have to investigate how these results translate back to the general distributed multiagent planning problem. We know that task allocation is a vital part of that problem so the negative theoretical results are just as applicable to planning as they are to task allocation. However, just as we rarely encounter instances corresponding to these negative results in task allocation, we seem to rarely encounter such instances in (distributed) multiagent planning. It would be interesting to understand how these two facts relate. Again, more realistic cases might shed a better light on this issue. This is therefore an avenue that we plan to pursue.

Acknowledgements

The research by Mathijs de Weerdt is supported by the Technology Foundation STW, applied science division of NWO, and the technology program of the Ministry of Economic Affairs. Roman van der Krogt is supported by an Irish Research Council for Science, Engineering and Technology (IRCSET) Postdoctoral Fellowship.
References


