Task Coordination and Decomposition in Multi-Actor Planning Systems

A.W. ter Mors[a,b], J.M. Valk[b] and C. Witteveen[a]

[a] Faculty EEMCS, Delft University, P.O. Box 5031, NL-2600 GA, Delft
{a.w.termors, c.witteveen}@tudelft.nl

[b] Almende BV, Westerstraat 50, NL-3016 DJ, Rotterdam
{adriaan, jeroen}@almende.com

Abstract

We discuss a framework for coordinating self-interested agents that can be used to decompose a multi-agent task based planning problem into independent subproblems. This problem decomposition can be achieved by a simple protocol and allows the agents to solve their part of the problem without the need to interact with other agents and in such a way that the resulting plans can be seamlessly integrated into a joint plan without the need for revising individual plans. We illustrate the application of the framework to the logistic planning problems used in the AIPS planning competition. After a thorough analysis of the problem we show how existing planners can benefit from such a decomposition technique.

1. Introduction and motivation

In this paper we deal with methods to solve multi-actor task-based planning problems such as occurring in e.g. manufacturing, supply-chain management and air traffic control. A common characteristic of such planning problems is that a set of autonomous planning agents, using their individual tools and capabilities, must come up with a joint solution to a problem consisting of a set of interdependent tasks. Typically, none of the agents is capable to solve all tasks and each agent is assigned a (disjoint) subset of tasks to perform. To complete its part of the job, each agent has to come up with a plan to perform the tasks assigned to it. These individual plans have to be integrated to create an executable joint plan. In general, it is assumed that the participating agents are self-interested and non-cooperative.

At least three major subproblems can be distinguished in such a multi-actor planning problem: a *task allocation problem* (which agent performs which subtasks), an *individual planning problem* for each of the agents involved (how to ensure that the tasks allocated to me can be performed) and a *plan coordination problem* (how to ensure that the individual planning processes can be integrated such that a solution to the overall problem can be achieved).

In this paper we concentrate on the plan coordination problem and we will restrict the discussion of this plan coordination problem to multi-agent systems where the agents are self-interested and non-cooperative planners requiring complete *planning autonomy*. That is, the actors do not want to be interfered during their planning and do not want to revise their plans afterwards. To select a suitable plan coordination method for these planning problems, we will first briefly review the multi-agent planning literature on plan coordination methods.

In the multi-agent planning literature, one can distinguish three main approaches to plan coordination. In the first approach (cf. [2,7,14]) coordination between the agents is established *after* the completion of the individual planning processes. It is assumed that agents independently work on their own part of the planning problem and achieve a solution for it. Then, in an after-planning coordination phase, possible conflicts between independently generated individual plans are resolved and positive interactions between them are exploited by exchanging and revising parts of the individual plans.

In the second approach (cf. [3–5,12]) coordination and planning are treated as *intertwined* processes where the agents continuously exchange planning information to arrive at a joint solution. From a coordination perspective, the main difference with the first approach is that positive (negative) interactions between *partial* individual plans are exploited (resolved) *before* each of the agents comes up with a completely developed plan. It therefore should be considered as a coordination *during* planning approach where the agents are cooperative in the sense that they are willing to exchange planning information with other agents and change their current plans if necessary.

The third approach is to perform coordination *prior to* the planning process. In this *pre-planning coordination* process some or all of the dependencies between the agents are resolved before any planning takes place[1]. The most influential pre-planning approaches in the literature are *social laws* and *cooperation protocols.* Social laws (cf. [15,16]) are general rules that govern agent behavior; if a collection of agents abide by these rules, then their behavior will be coordinated without the need for any problem-specific information exchange between the agents. In many situations, however, coordination cannot be achieved (or not efficiently) through general, problem-independent rules alone. In such cases, *cooperation protocols* ([9]) can be applied. Such protocols require simple forms of problem-specific information exchange before the agents can start planning and they guarantee that if the agents adhere to the protocol, then the individual plans can easily be assembled into a joint plan for the overall task.

Given these three main approaches, it might be clear that the first two approaches are not suitable for coordinating self-interested, non-cooperative planners that require complete planning autonomy. In this paper we therefore concentrate on *pre-planning coordination methods* that can be used to implement cooperation protocols. As a consequence, before the agents start to plan, there is a single coordination phase ensuring that the agents can plan independently and concurrently and in such a way that they do not need to revise their plan afterwards.

Elsewhere (see [10]), we have presented a complete overview of such a task-based pre-planning coordination method together with a complexity analysis of the planning problems associated with this approach. In this paper we will only give a brief overview of this framework and then point out how it can be applied to a well-known set of *logistic* planning problems as used in the AIPS 2000 planning competition. Considering the logistic planning problem as a multi-agent planning problem, we will show that the application of this pre-planning coordination framework enables the design of a very simple pre-planning coordination method. This coordination enables a decomposition of the original planning problem into a set of subproblems that can be solved by the agents independently from the others, and in such a way that the resulting plans (solutions to the subproblems) can be simply joined to constitute a joint plan for the total planning problem.[2]

Each subproblem to be solved by the individual agents constitutes a restriction of the original logistic planning problem and cannot be solved optimally (in reasonable time). Moreover we will show that it is highly unlikely that there exist a δ-approximation algorithm with $\delta < 1.2$ for such a subproblem. Then we show that, quite surprisingly, the coordination method might be used to obtain a polynomial distributed 1.25-approximation algorithm for solving the complete multi-actor logistic planning problem.

Finally, we present some results obtained by applying our coordination approach to the logistic benchmark problems used in the AIPS2000 planning competition, showing that also existing single-agent planning methods can benefit from this simple coordination method.

2. Coordination by Pre-Planning

We consider task-based planning problems where a set T of tasks has to be solved by several

[1]In the remainder of this paper we will focus on pre-planning coordination in which *all* inter-agent dependencies are coordinated prior to planning.

[2]Stated this way, the pre-planning coordination method comes down to a generalization of classical problem decomposition methods like, e.g. *divide-and-conquer* or *partitioning.*

autonomous agents, each having their own capabilities and using their own (planning) tools. This set T consists of elementary tasks where each elementary task t, or simply *task*, is a unit of work that can be performed by a single agent. These elementary tasks are interrelated by a partially ordered *precedence relation* $\prec$: a task t_1 is said to precede a task t_1, denoted by $t_1 \prec t_2$ if the execution of t_2 may not start until t_1 has been completed; this occurs if, for example, achieving t_1 results in creating resources needed to perform t_2. The tuple $\mathcal{T} = (T, \prec)$ is called a *complex task*. Examples of such complex tasks are easy to find: manufacturing, airport planning and supply-chain management can be considered as consisting of (sets of) interrelated elementary tasks.

Such a complex task $\mathcal{T} = (\mathcal{T}, \prec)$ is given to a set $\mathcal{A} = \{A_1, \dots A_n\}$ of autonomous planning agents. We assume that the tasks in T have to be be assigned to the agents in $\mathcal{A}$ by some task assignment $f : T \rightarrow \mathcal{A}$ thereby inducing a *partitioning* $\mathbf{T} = \{T_1, \dots, T_n\}$ of T, where $T_i = \{t_j \in T \mid f(t_j) = A_i\}$ denotes the set of tasks allocated to agent A_i. As a result of this task assignment, A_i also inherits the precedence constraints that apply to T_i, i.e., the set $\prec_i = \prec \cap (T_i \times T_i)$. These sets $\prec_i$ together constitute the set $\prec_{intra} = \bigcup_{i=1}^{n} \prec_i$ of *intra-agent* constraints, while the remaining set of constraints $\prec_{inter} = \prec \setminus \prec_{intra}$ constitutes the set of *inter-agent* constraints. So each agent A_i now is responsible for achieving the (complex) subtask $(T_i, \prec_i)$ while the agents are dependent upon each other via the inter-agent constraints $\prec_{inter}$.

We will not deal with the problem how the task assignment has been achieved. In fact, this problem can be discussed separately from the coordination problem and is trivial to solve in our application case.

In this task-based framework we do not make any assumptions about the *planning tools* used by the agents. Whatever plan/schedule representation the agents (internally) employ, we assume that a plan P_i developed by agent A_i for its set of tasks T_i can always be specified by a structure $P_i = (T_i, \pi_i)$ that refines the structure $(T_i, \prec_i)$, i.e. $\prec_i \subseteq \pi_i$, which means that an agent's plan must respect the original precedence relation $\prec_i$, but his plan may induce additional constraints.

Since the agents are assumed to be competitive, the following restrictions have to be obeyed in coordinating their planning activities:

1. during the planning process, the agents are completely autonomous, i.e., they choose a plan for their part of the task *independently from the other agents*;

2. after planning, the individually constructed plans can be joined into a joint plan without the need to revise any individual agent plan.

The following example shows that these requirements are not trivially met; here, the agents do not exchange information about their intended plans, and the result is that their plans are in conflict and cannot be combined without revision.

Example 1 *Consider the following simple case: (see Figure 1) we have two agents A_1 and A_2 and four tasks $T = \{t_1, t_2, t_3, t_4\}$. The precedence relation $\prec$ is given as $\prec = \{(t_1, t_3), (t_4, t_2)\}$. Suppose that t_1, t_2 are assigned to A_1 and t_3, t_4 to A_2. Then A_1 has to solve the subtask $(\{t_1, t_2\}, \emptyset)$, while A_2 has to solve $(\{t_3, t_4\}, \emptyset)$. If A_1 chooses a plan where t_2 is completed before t_1 and A_2 chooses a plan where t_3 is completed before t_4, there exists no feasible joint plan preserving $\prec$ and the individual plans, as the combination of their plans with the inter-agent constraints constitutes a cycle: $t_1 \prec t_3 \prec t_4 \prec t_2 \prec t_1$.*

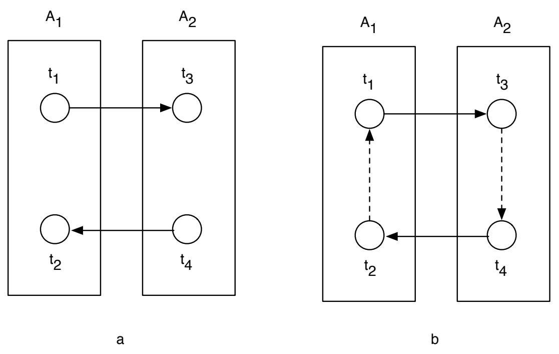

Figure 1. A set of interdependent tasks $T = \{t_1, t_2, t_3, t_4\}$ and two agents $A1$ and $A2$ each assigned to a part of T (see a). If agent $A1$ decides to make a plan where t_2 precedes t_1 and $A2$ makes a plan where t_3 precedes t_4 (see b), these plans cannot be combined.

In the above example, the solution is to add an additional constraint prior to planning, for instance, $t_1 \prec t_2$ to the set of intra-agents constraints of agent A_1. Then, whatever plans the agents come up with (respecting their intra-agent constraints, of course), the results can be combined into an acyclic joint plan.

This addition of constraints constitutes the essence of our *pre-planning* coordination method: prior to planning, we seek a *minimal set* of additional, intra-agent (precedence) constraints, such that any combination of agent plans respecting these constraints can subsequently be combined into a feasible joint plan. It can be easily shown that such a minimal set $\Delta \subseteq \bigcup_{i=1}^{n} T_i \times T_i$ of intra-agent constraints, called a *coordination set*, always exist. In general, however, such sets are extremely difficult to compute (cf. [10]) and we have to rely on approximation algorithms to find non-minimal coordination sets that guarantee revision-free independent planning.[3]

3. A distributed coordination algorithm to achieve plan coordination

Since finding and deciding a minimal coordination set Δ is too complex to solve in reasonable time (unless P=NP), in this section we propose a distributed approximation algorithm for finding not necessarily minimal coordination sets with low-polynomial time complexity.

The algorithm is based on the following idea: we assume that each agent A_i knows whether or not some task t it has to execute is dependent upon another task t' (or that another task t' depends upon t). If t' does not belong to its own set of tasks, both the identity of t' and the executing agent A_j are unknown. In constructing a plan for $(T_i, \prec_i)$, each agent A_i can safely start to make a plan for the subset $T_i^1 \subseteq T_i$ of tasks that are not dependent (via inter-agent constraints $\prec_{inter}$ or via A_i's own constraints $\prec_i$) upon other tasks. Such tasks are called *prerequisite-free.* To determine whether a given task t in T_i is prerequisite-free or not, we let the agents use a third party in the form of a common *blackboard.* This blackboard stores all dependency relations between tasks belonging to different agents and, given a query t, the blackboard can answer whether or not a task t depends on another task t' belonging to another agent. Once each agent A_i has selected such a prerequisite-free subset T_i^1 (which may be empty), the agent informs the blackboard that the tasks t in T_i^1 can be removed from the set T_i, together with all precedence constraints in $\prec$ that involve task t. Then the agents start a new round where each agent A_i again selects a subset T_i^2 of the set of remaining tasks not dependent on other tasks. These rounds continue until after, say, $k \leq |T|$ rounds the set of remaining tasks is empty. As a result, the original task T_i of agent A_i is partitioned into an ordered set of k (possibly empty) subsets $T_i^1, \ldots, T_i^k$.

Hereafter, the coordination set Δ_i for agent A_i is constructed as follows: first, all empty subsets T_i^j in $\{T_i^j\}_{j=1}^k$ are removed. Next, for every $j = 1, \ldots, k-1$ and for every pair of tasks t, t': if $t \in T_i^j$ and $t' \in T_i^{j+1}$, the precedence constraint (t, t') is added to Δ_i.

To obtain these sets T_i^j, each agent A_i executes the following algorithmic scheme: In each round

Algorithm 1 Coordination by partitioning

```
Require: a task $(T_i, \prec_i)$ for agent $A_i$
Ensure: a linearly ordered partition $(T_i^1, \ldots, T_i^k)$ of $T_i$
 1: let $k := 0$
 2: while $T_i \neq \emptyset$ do
 3:    $k := k + 1$
 4:    ask the blackboard for the subset $T_i^k \subseteq T_i$
       of tasks that are prerequisite-free,
 5:    if $T_i^k \neq \emptyset$ then
 6:       Let $T_i = T_i - T_i^k$
 7:       Send the set $T_i^k$ to the blackboard
 8:    else
 9:       $k := k - 1$
10:    end if
11: end while
12: return $(T_i^1, \ldots, T_i^k)$
```

of the algorithm, agent A_i splits off a (possibly empty) set of tasks T_i^k from T_i. In Step 4, agent A_i asks the blackboard to compute the set T_i^k of tasks in T_i that are free of prerequisites.

The worst-case performance of the coordination-by-partitioning algorithm, mea-

[3]In general, the problem to find such a set is Σ_2^p-hard, even if the agents have a modest number of tasks (8 or more) to perform.

sured in terms of the cost of the resulting plan, is bounded by the maximum number of rounds the agents need to complete the coordination algorithm. For, suppose an agent A_i splits his set of tasks T_i into k segments $T_i^1, \ldots, T_i^k$, then the cost of the combined plan for all k segments can be at most k times as high as the cost of a plan for his unconstrained set of tasks T_i. As an example, consider the case where an agent only has two tasks t_1 and t_2. If there exists no precedence constraint between t_1 and t_2, then we might imagine that an agent can execute both tasks in parallel. By introducing the constraint $t_1 \prec t_2$, the tasks can no longer be executed in parallel, so the cost of the plan may increase — but with no more than the cost of the original plan, since at most all work done on t_1 must be redone for t_2. It is easy to see that the maximum value of k (the number of rounds) for any agent is bounded above by the depth d (i.e., the length of the longest chain) of the composite task $\mathcal{T} = (T, \prec)$. Therefore, preferably, this algorithm might be used in those cases where the depth of the complex task is relatively small compared to the number of tasks, such as e.g. in multimodal transportation.

Considering the time complexity of the algorithm, for every value of k an agent does not need to spend more than $O(|T_i|)$-time to find a prerequisite-free subset and to update the set of current tasks. Hence, since $k \leq |T|$, the total time each agent needs to find its partitioning is $O(|T|^2)$.

4. Application to logistic planning

To illustrate the practical use of our approach, we apply the coordination algorithm as a preplanning coordination approach to a logistic planning problem used in the AIPS2000 planning competition. Before we discuss the results, we will define this logistic planning problem and analyse its complexity into some detail.

4.1. Logistic planning problems

The logistic planning problem we have in mind consists of a set *Loc* of locations $loc_{i,j}$ where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots n$. A city c_i is a subset $c_i = \{loc_{i,j} \mid j = 1, \ldots, n\}$ of locations, where each location can be visited by a truck. In each city c_i we distinguish a special location $loc_{i,1}$ as the airport of city c_i. All airports are connected by directed flights and are visited by airplanes. There is also a set of orders $o = (l, l')$ for picking up and delivering packages, consisting of a pick-up location $l \in Loc$ and a delivery location $l' \in Loc$. Let O denote the set of orders given. We distinguish *intra-city* orders and *intercity orders*. An intra-city order requires a *load action* at the pick-up location, a *move action* and an *unload action* at the destination location. So the cost of an intra-city order is minimally 3 actions. For an inter-city order, we distinguish a *pre-order* phase, a *plane-phase* and a *post-order* phase. In the pre-order phase, an intra-city order is carried out by transporting the package to the airport of the pick-up city, during the plane-phase, the package is transported to the destination airport and finally in the post-order phase the package is transported from the airport to the final destination. So an inter-city transportation might require at least 6 load/unload actions and at least 3 move actions. We use a simple uniform cost model where every load, unload and move action has unit cost. Hence, the cost of a plan simply equals the number of actions it contains.

Given an instance (Loc, O) of this logistic planning problem we are looking for a plan P that carries out all orders in O. Such a plan is a sequence of load/unload and move actions completing all the orders in O. The cost of P, denoted by $|P|$, is the sum of the cost of all actions occurring in P and of course, we would like to obtain an optimal plan P^*, i.e. a plan with minimum cost[4].

4.2. A preplanning coordination approach to the logistic planning problem

Note that an instance (Loc, O) of the logistic planning problem can be easily translated to an instance of a multi-agent planning problem: Every inter-city order o_j consists of a linearly ordered sequence of (at most) three elementary tasks $t_{j1} \prec t_{j2} \prec t_{j3}$, where t_{j1} is the task of transporting the package from its pickup location to the airport by the truck agent in the pick-up city, t_{j2} is the plane task of transporting the package to the airport of the destination city and finally t_{j3} is a truck task consisting in transporting the package to its destination location. Note that the task assignment is trivial here: intra-city orders are assigned to the truck agents, inter-city

[4]Note that we are not looking for a minimum time plan.

orders to the plane agents. Also note that there are no precedence constraints within the set of orders assigned to a particular agent, so all precedence constraints are inter-agent constraints.

It is easy to see that a feasible joint solution is not guaranteed if the agents plan completely independently from each other. Therefore, we apply our approximation algorithm to ensure independent planning. Note that here the depth of the partial order is 3 (a pre-transportation order followed by a plane order followed by a post-transportation order). Hence the number of rounds equals at most 3. Applying the distributed approximation algorithm 1, it is easy to see that:

1. in the first round, all truck agent select their local transportation orders together with all their pre-transportation orders;

2. in the second round, only the plane agents select all their (plane) orders;

3. in the third round, only truck agents select all their post-transportation orders.

This implies that only the truck agents will receive additional constraints: per city, the local truck agents have to make a plan for the local, the pre-transportation orders and the post-transportation orders in their city. Their transportation plan, however has to satisfy the additional constraints that every local and pre-transportation order should precede any post-transportation order in the plan (Equivalently, they can be asked to make two plans: one for the combined local and pre-transportation orders and one for the post-transportation orders). The planes have to make a plane plan for the plane orders.

Now it is easy to see that the plans of the plane agent and the truck agents can be easily combined into a joint plan, in such a way that (i) all pre- and local transportation plans are combined; (ii) all actions in the local plans for pre- and local transportation precede the actions in the plane plan, (iii) all post-transportation plans are combined and (iv) all actions in the plane plan precede the actions in the combined post-transportation plans. It is easy to verify that the plan thus composed indeed is a solution to the original problem.

Surprisingly, this simple coordination approach, theoretically as well as experimentally, is one of the best coordination methods we can find for this logistic planning problem. In order to evaluate the overall performance of the method, we first concentrate on the single agent (truck and plane agent) planning problems and their solution.

4.3. Finding solutions to the local planning problems

Let $Loc = \{loc_1, loc_2, \ldots, loc_n\}$ be a set of locations in a given city (or a set of airports) and let $O \subseteq Loc \times Loc$ a set of pickup-delivery orders over Loc. We assume that all locations are directly connected and the cost to travel from c_i to c_j is the same (equal to 1) for all $i \neq j$. We would like to construct a sequence S over Loc such that all pickup-delivery orders can be carried out by visiting the locations in the order indicated by S. Such a *visiting sequence* S is a sequence over Loc with possible repetitions. An order $(l, l') \in O$ can be fulfilled by S if there exists sequences α, β, γ over Loc such that $S = \alpha l \beta l' \gamma$, i.e. S contains an occurrence of l before an occurrence of l'. A visiting sequence S over Loc is a *solution* to the instance (Loc, O) if every order in O can be fulfilled by S.

To find an optimal solution requires us to find a solution of minimal length. This *minimal visiting sequence* problem is an NP-hard problem (See Appendix, subsection A.1).

Consider now the following simple approximation algorithm to construct a visiting sequence satisfying the orders O, given the locations Loc and the set O:

Stage 0
Let $O = \{o_i\}_{i=1}^m$ and let S be empty;

Stage i ($i \geq 1$)
Let $o_i = (l, l')$ and let the current visiting sequence be S;
if both l an l' do not occur in S, add (l, l') to the end of S; if only l occurs in S and l' does not occur in S, add l' to the end of S; if only l' does occur in S, add l to the front of S; if both l and l' occur in S then add l' to the end of S only if ll' does not occur as a (scattered) subsequence of S;
If $i < n$ goto stage $i + 1$, else stop.

It is not difficult to show that this (polynomial) algorithm achieves an approximation bound $\epsilon = 2$. The remarkable news is that, for the time being, *we cannot do better than this*: It is known [6] that the Minimum Directed Feedback Vertex Set (MDFVS) problem is an APX-hard problem[5] and until now, the best known approximation algorithm guarantees an $O(\log n \log \log n)$-factor approximation, that is, until now it has not been shown that the problem is in APX. The following proposition shows that finding a polynomial $(2 - \epsilon)$-approximation algorithm for finding minimal visiting sequences would immediately imply that the MDFVS-problem is in APX:

Proposition 1 *Let $\delta > 1$ be an arbitrary constant. The MDFVS-problem is δ-approximable iff the minimum visiting sequence problem is $(2 - \frac{2}{\delta+1})$-approximable.*

Proof See Appendix. ■

Note that whenever $\delta = \delta(n)$ is *not* bounded above by some constant, it follows that $\sup\{2 - \frac{2}{\delta(n)+1}\} = 2$. Hence, we have:

Corollary 1 *The MDFVS-problem is in APX iff the local planning problem is ϵ-approximable for some $\epsilon < 2$.*

Since, as we remarked, the best know approximation algorithm for directed minimum vertex feedback set achieves an approximation ratio $O(\log n \log \log n)$, the best known approximation bound for finding a minimal visiting sequence therefore is $\epsilon = 2$.

4.4. An approximation based on the coordination approach

To make a comparison with the MDFVS-problem, we used a visiting sequence S as a solution to the local planning problem (Loc, O). Such a visiting sequence, of course, can be easily translated into a plan P for the local planning problem: If S is the visiting sequence solution, the cost $|P|$ of the corresponding transportation plan P is (taking into account one additional load and unload action per action, each of unit cost): $|P| = 2.|O| + |S|$. Let $|Loc| = n$ and $|O| = m$ and assume that every location $l \in Loc$ occurs in at least one order $o \in O$. Therefore, $m \geq n$. Since the length of S is at most $2n$ and at most 2 times the length of the optimal visiting sequence S^*, with $|S^*| \geq n$, it follows that the performance ratio, i.e. the cost of the local plan thus determined versus the cost of the optimal local plan P^*, is bounded above by

$$\frac{|P|}{|P^*|} = \frac{2.|O| + |S|}{2.|O| + |S^*|} \leq \frac{2m + 2n}{2m + n} \leq \frac{2m + 2m}{2m + m} = 4/3.$$

We can slightly improve this upperbound by observing that $m \geq |S|$ should hold since, according to the approximation algorithm for constructing visiting sequences, every location l that occurs twice in S is the result of adding at least two orders, one in which l occurs as source and one as a destination. Hence, there must be at least two orders for each such an occurrence. Therefore, using the inequalities $m \geq |S|$ and $|S| \leq 2n$, we achieve

$$\frac{|P|}{|P^*|} \leq \frac{2m + |S|}{2m + n} \leq \frac{2|S| + |S|}{2|S| + 0.5|S|} = 1.2$$

Note that local planning problems constitute special cases (restrictions) of the complete multi-actor logistic planning problem we want to solve. Therefore, the following result is an immediate consequence of the preceding observations and shows that currently the best we can hope for is an approximation algorithm for the total logistic planning problem that achieves $\epsilon = 1.2$:

Theorem 1 *There exists no polynomial ϵ-approximation algorithm with $\epsilon < 1.2$ for the multi-actor logistic planning problem as defined unless the MDFVS-problem is in APX.*

Now we show that our coordination approach performs remarkably well with respect to this "lowerbound": if we use the coordination approach and the approximation algorithm for finding minimal visiting sequences as discussed above to solve the local planning problems, we obtain an approximation algorithm for the total planning problem with a performance ratio that is only slightly worse: $\epsilon = 1.25$ instead of the lowerbound 1.2:

Proposition 2 *The multi-actor logistic planning problem is 1.25-approximable.*

[5]An optimization problem is an APX-problem if it has a polynomial c-approximation algorithm for some constant $c \geq 1$.

Proof See Appendix. ■

Example 2 *To show that this upper bound can be achieved, we present a set of tight examples. Consider problem instances with $n+1$ cities and $n+1$ locations per city. The set of orders is*

$$O = \{(loc_{1,1}, loc_{1,j}), (loc_{1,j}, loc_{1,1})|\ j = 2, \dots, n+1\}$$
$$\cup\{(loc_{j,1}, loc_{1,k})|\ j = 2, \dots, n+1,\ k = 2, \dots, n+1\}$$
$$\cup\{(loc_{1,1}, loc_{j,1})|\ j = 2, \dots, n+1,\ k = 2, \dots, n+1\}$$

The total number of load/unload actions equals $4n+4n+2n$, while the number of moves in a plan is $2n+1+2n+1+n$ and the optimal number of moves equals $2n+2$. Therefore, the performance ratio equals $\lim_{n\to\infty} \frac{15n+2}{12n+2} = 1.25$.

Note that the drop in approximation quality between solving only the local problems and solving the total planning problem has to be attributed to the additional overhead caused by coordinating the local plans.

5. Experimental results

The performance results obtained in the previous sections are theoretical and worst-case results. In order to test and to compare our coordination approach with other planning approaches to the logistic problem, in this section we will show some results using a benchmark set for general planners. With this comparison we want to show two things: First of all, the average performance of our coordination approach is much better than should be expected, if the worst-case performance ratio is taken as the norm. Secondly, the coordination approach can be used to enhance the planning power of existing planners significantly, thereby showing that it enables single agent planning technology to be used for multi-agent problems.

In the Artificial Intelligence Planning and Scheduling (AIPS) competition of the year 2000, several general-purpose planning systems competed in a number of planning domains. The logistic planning problem as described in Section 4.1 was one of the domains featured. We have used the AIPS logistics dataset in our experiments because of its status as benchmark problem set, and also because it allows us to compare our decomposition-by-coordination approach to a selection of centralized planning systems.

In Table 1, we compare plan cost (in terms of the number of moves in a plan) for four planners. In the second column the costs of the optimal plans are given as calculated by encoding the complete instance as an ILP-problem and solving it exactly (of course not taking into account the time needed to find a solution); the third column represents the cost of the plans produced using the coordination approach; the fourth, fifth, and sixth columns represent a selection of the planning systems competing in the AIPS: the competition-winning TALplanner [11], and the above-average performers STAN [13] and HSP2 [1]. Each row in Table 1 represents an instance in the dataset, characterized by the number of packages that have to be transported for that instance. Of the roughly 200 instances in the dataset, we have made a random selection of 12.

It will come as no surprise that the results produced using the coordination approach, especially since the local plans in fact were solved exactly, deviate little from the optimal plans — on average less than 5%. These results are significantly better than what would be expected (25%) based on the worst-case performance ratio of 1.25.

The plans produced by the coordination approach are comparable in quality with the plans produced by STAN (only 2% deviating from the optimum) and TALplanner (about 7%). To illustrate that for some solvers the problems in the AIPS dataset are far from trivial, HSP2 does not manage to solve (within reasonable time and memory constraints) any instances where more than 40 packages have to be transported and produces significantly worse plans (about 44% deviating from the optimum). The cpu-times needed to produce the plans by the coordination approach were a few seconds for each of the planning instances occurring in Table 1.

As we remarked before, the coordination approach cannot only be used to solve multi-agent planning problems using simpler single-agent planning tools; we can also apply it as a *pre-processing* step for a given planning system that has trouble solving such multi-agent planning problems. The idea is that the problem instance then can be decomposed into smaller subproblems that can be solved independently by the planning system, whereafter the solutions to the subproblems can be simply combined into a solution to the whole problem instance.

nr packages	min nr moves	Coordination	TALplanner	STAN	HSP2
20	107	113	111	110	145
25	143	150	152	149	206
30	175	182	183	177	250
35	177	181	186	182	264
40	228	239	239	232	337
45	269	284	285	276	–
50	286	299	306	293	–
55	319	327	338	326	–
60	369	391	398	376	–
65	371	387	397	382	–
70	405	426	437	416	–
75	438	458	471	448	–

Table 1
Results for 12 randomly chosen instances from the AIPS00 logistics dataset. For each instance the minimum number of moves is determined and for each planner the number of moves produced is given

Specifically, what we propose is the following method: using the coordination approach, decompose a multi-agent logistics instance into a set of single-agent planning problems. Then, feed each of the single-agent planning subproblems to the planning system, and combine the results (plans) according to the protocol into an overall plan, thereby solving the complete multi-actor instance. What we would like to see using this method is significant savings in computation time without significant loss in plan quality compared to the use of the planner solving the complete instance. For this experiment we chose STAN (since it produced almost optimal plans for the complete instance) and HSP-2 (since it consumed a lot of cpu time).

To test these expectations, we randomly selected 51 problems from the AIPS00 planning dataset and observed both the reduction in cpu-time and the reduction in plan cost comparing a solution produced by using only the planner with a solution by using the planner in combination with the coordination approach. The results are given in Figure 2.

It can be observed that both STAN and HSP2 definitely benefit from pre-processing by the coordination approach: both planning systems regularly achieve savings in computation time of over 80%. In addition, we can see that HSP2 produces plans that are on average 20% cheaper, i.e., requiring 20% less actions. Also note that the plan cost for STAN does not increase significantly when using the coordination approach. Finally, it can be observed that even after a decomposition into smaller subproblems for quite some instances HSP-2 again was not able to produce a solution within reasonable time. This means that even for the local planning problems HSP-2 still has considerable difficulty in solving them.

6. Conclusion

We discussed a task-based planning framework and its application to a logistic planning problem. We illustrated the framework with an application to a logistic planning problem used in the AIPS00-planning competition and we showed that there exists a very simple coordination protocol derived from the general framework for multi-agent planning that enables us to *decompose* the planning problem into simpler subproblems that can be solved independently. This coordination by decomposition approach does not result in inefficient plans when compared to a theoretically optimal centralized planning system. Rather, a centralized planning system will typically have trouble finding a solution for the entire multi-agent instance at once. Consequently, our approach, where each agent can make a plan independently of the other agents, will result in more efficient plans.

Our current framework does not address issues of temporal constraints in multi-agent planning. Future work will concentrate on coordination methods for temporal planning systems,

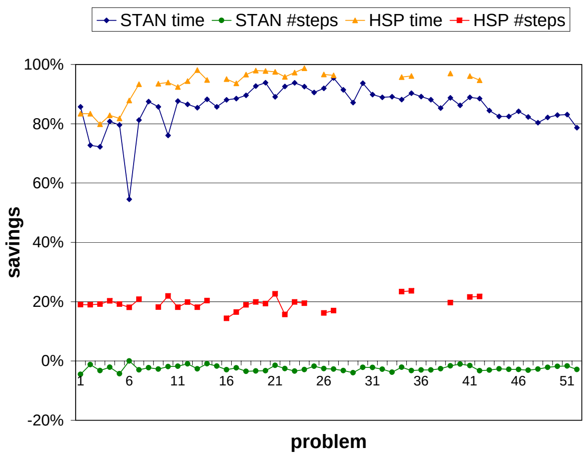

Figure 2. Savings in CPU times and plan cost (#steps) when STAN and HSP2 make use of the coordination approach as a pre-processing step.

building upon approaches like [8] to decouple temporal planning problems.

References

1. B. Bonet and H. Geffner. Heuristic search planner 2.0. *AI Magazine*, 22(3):77–80, Fall 2001.
2. J.S. Cox and E. H. Durfee. Discovering and exploiting synergy between hierarchical planning agents. In *Second International Joint Conference On Autonomous Agents and Multiagent Systems (AAMAS '03)*, 2003.
3. K. S. Decker and V. R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence (DAI-94)*, pages 65–84, 1994.
4. E. H. Durfee and V. R. Lesser. Partial global planning: a coordination framework for distributed hypothesis formation. *IEEE Transactions on systems, Man, and Cybernetics*, 21(5):1167–1183, 1991.
5. E. Ephrati and J. S. Rosenschein. Multiagent planning as the process of merging distributed sub-plans. In *Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence (DAI-93)*, pages 115–129, 1993.
6. P. Festa, P. Pardalos, and M. Resende. Feedback set problems, 1999.
7. D.E. Foulser, Ming Li, and Qiang Yang. Theory and algorithms for plan merging. *Arificial Intelligence*, 57(2–3):143–182, 1992.
8. Luke Hunsberger. Algorithms for a temporal decoupling problem in multi-agent planning. In *Proceedings AAAI/IAAI*, pages 468–475, 2002.
9. N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.
10. M.M. de Weerdt J.M.Valk and C. Witteveen. Algorithms for coordination in multi-agent planning. *I. Vlahavas and D. Vrakas (ed.), Intelligent Techniques for Planning (to appear)*, 2004.
11. J. Kvarnström and P. Doherty. Talplanner: A temporal logic based forward chaining planner. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):119–169, 2000.
12. V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. NagendraPrasad, A. Raja, R. Vincent, P. Xuan, and X.Q. Zhang. Evolution

of the GPGP/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, July 2004.

13. D. Long and M. Fox. Efficient implementation of the plan graph in stan. *Journal of AI Research*, 10:87–115, 1999.
14. F. Von Martial. *Coordinating Plans of Autonomous Agents*, volume 610 of *Lecture Notes on Artificial Intelligence*. Springer Verlag, Berlin, 1992.
15. Y. Moses and M. Tennenholtz. On computational aspects of artificial social systems. In *Proceedings of DAI-92*, 1992.
16. Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1–2):231–252, 1995.

A. Appendix

A.1. The minimal visiting sequence problem is NP-hard

Proposition 3 *Let $G = (V, A)$ be an instance of the* minimum directed feedback vertex set *(MDFVS) problem. Then S is a minimum visiting sequence solving the instance (Loc, O), where $Loc = V$ and $O = A$, iff $F = \{v \in V \mid v$ occurs twice in $S\}$ is a minimum feedback vertex set of G.*

Proof Without loss of generality we may assume that G does not contain isolated nodes. Let $S \in V^*$ be a minimum solution to $(Loc, O) = (V, A)$ and consider $F = \{v \in S \mid v \text{ occurs twice in } S\}$. Observe that, by assumption, every node of V occurs in at least one pair (v, v') of A, every location v occurs at least once in S.

1. *F is a feedback vertex set of G.* Let $C = (v_0, v_1, \ldots v_k, v_0)$ be a directed cycle in $G = (V, A)$. We show that for at least one $v_i \in C$, $v_i \in F$. On the contrary, assume that for no $i \in \{0, 1, \ldots, k\}$, v_i occurs twice in S. For $i = 0, 1, \ldots k$, let j_i be the (unique) index of v_i in S. Hence, there exists a total ordering $j_{i_0} < j_{i_1} < \ldots < j_{i_k}$ of these index positions in S. But this implies that the order $(v_k, v_0) \in A$, requiring that S should contain an occurrence of v_k occurring before an occurrence of v_0; contradiction.
2. *F is a minimum feedback vertex set.* If F is not a minimum feedback vertex set, there exists another subset F' of V with $|F'| < |F|$ such that F' intersects every cycle of G. But then it is easy to see that S is not a minimum solution: Let $A' = \{(v, v') \in A \mid v' \notin F'\}$ and take the instance (V, A'). Clearly, since $G' = (V, A')$ is acyclic, there exists a topological ordering T of nodes in V respecting A'. Now let $S' = T.F'$. It is easy to show that S' is a solution to (V, A): take an order $(v, v') \in A$. If $(v, v') \in A$, T and therefore S' is a sequence satisfying (v, v'). If $(v, v') \notin A$, v occurs in T and v' occurs in F, hence S' satisfies (V, A). Therefore, S' is a smaller solution than S; contradiction. Hence, F is a minimum feedback vertex set.

■

A.2. Proof of Proposition 2

($\Rightarrow$) Assume that the MDFVS-problem is δ-approximable for some $\delta > 1$. We derive a $2 - \frac{2}{\delta+1}$-approximation algorithm for the minimum visiting sequence problem as follows: *(i)* Given an instance (Loc, O) of the minimum visiting sequence problem, find an approximable minimum feedback vertex set F for the directed graph $G = (Loc, O)$. *(ii)* Determine in time $O(|Loc| + |O'|)$ a topological ordering S' of Loc compatible with the partial order $O' = \{(l, l') \in O \mid l' \notin F\}$ and let $S = S'. <F>$ where $<F>$ is an arbitrary ordering of the locations l occurring F. Clearly, S is a solution of (Loc, O) and the performance ratio of this approximation algorithm equals:

$$\frac{|S|}{|S^*|} = \frac{|Loc| + |F|}{|Loc| + |F^*|} \leq \frac{|Loc| + \delta \times |F^*|}{|Loc| + |F^*|}$$

where F^* denotes the optimal solution, i.e. a minimum feedback vertex set of $G = (Loc, O)$. Note that $|F^*| \leq |Loc|$. Hence, this ratio is bounded above by

$$\frac{|Loc| + |Loc|}{|Loc| + \frac{|Loc|}{\delta}} = 2 \times \frac{\delta}{\delta + 1} = 2 - \frac{2}{\delta + 1}$$

($\Leftarrow$) Conversely, assume that the order routing problem is $\epsilon = 2 - \frac{2}{\delta+1}$ approximable for some $\delta > 1$. We prove that MDFVS has a δ-approximation algorithm.

Let $G = (V, A)$ be an instance of the MDFVS-problem. Consider the instance (Loc, O) with

$Loc = V$ and $O = A$ of the order routing problem. Let S be a solution found by the approximation algorithm for the order routing problem and let S^* be an optimal solution to the instance. Without loss of generality we can assume that no vertex l occurs more than twice in S. Let $F^* = \{v | v \text{ occurs twice in } S^* \}$ and $F = \{v | v \text{ occurs twice in } S\}$. We know that both F and F^* are feedback vertex sets and that F^* is a minimum feedback vertex set of G.

Note that for all instances[6] I, $\frac{|S_I|}{|S_I^*|} = \frac{|V|+|F|}{|V|+|F^*|} \leq 2 - \frac{2}{\delta+1}$. Let $f : N \to N$ be a bounding function such that for all instances such that $|F^*| = n$, $|F| \leq f(n) \times n$. We will show that $f(n) \leq \delta$.

Let $|V| = m$ and $|F^*| = n$. Then we have:

$$\frac{m+|F|}{m+n} \leq \frac{m+|F|}{m+\frac{|F|}{f(n)}} \leq 2 - \frac{2}{\delta+1}$$

where $|F| \leq m$. The left-hand side is maximized choosing $|F| = m$. Hence,

$$\frac{2m}{m+\frac{m}{f(n)}} = \frac{2}{1+\frac{1}{\delta}} \leq 2 - \frac{2}{\delta+1}$$

implying that $f(n) \leq \delta$.

A.3. Proof of Proposition 3

To determine this performance ratio, let $\hat{\mathrm{m}}^0_{pre}$, $\hat{\mathrm{m}}^0_{plane}$, $\hat{\mathrm{m}}^0_{post}$ denote the number of moves found using the local approximation algorithms for the pre-transportation and the plane transportation phase and the post transport phase, respectively. Note that the number of moves equal the length of the visiting sequence found by the algorithm. We note that $\hat{\mathrm{m}}^0_{post} = m^0_{post}$ is optimal, since this number of moves can be determined exactly (the local problem to be solved is the problem of finding a visiting sequence in an acyclic order graph, and this problem is polynomially solvable). For the remaining number of moves we have $\hat{\mathrm{m}}^0_{pre} \leq 2m^0_{pre}$ and $\hat{\mathrm{m}}^0_{plane} \leq 2m^0_{plane}$. Here, the right- hand side variables denote the optimal number of moves in the pre-transportation, the plane and the post-transportation plan, respectively. The same holds for the remaining variables. We will use these numbers to relate them to the number of load/unload actions that have been executed.

[6]Without loss of generality we may assume that G does not contain isolated points.

Take, for example, $\hat{\mathrm{m}}^0_{pre}$ (the case for $\hat{\mathrm{m}}^0_{plane}$ is analogous). Let $\hat{\mathrm{m}}^0_{pre} = m + 2k$ where $m + k = n$, n is the total number of locations and k the number of locations occurring twice in the visiting sequence found. This immediately implies that the number of orders to be executed in the local plan must have been at least $(m-1)+2k$. Therefore, the number of load/unload actions must be $l^0_{pre} = 2(m-1+2k)$ and the total cost of the pre-transportation plans must equal $l^0_{pre} + \hat{\mathrm{m}}^0_{pre} = 2(m-1+2k)+m+2k = 3m+6k-2$ Analogously, we have the following relationships: the number of load /unload actions for the plane equals $l^0_{plane} = 2(m'-1+2k')$ if $\hat{\mathrm{m}}^0_{plane} = m'+2k'$. Hence, $l^0_{plane} + \hat{\mathrm{m}}^0_{plane} = 3m' + 6k' - 2$.

Hence, denoting the plan as computed by $\hat{\mathrm{p}}^0$, we derive

$$\begin{aligned}
\frac{|\hat{\mathrm{p}}^0|}{|p^*|} &= \frac{l^0 + \hat{\mathrm{m}}^0}{p^*} \\
&= \frac{l^0 + \hat{\mathrm{m}}^0_{pre} + \hat{\mathrm{m}}^0_{plane} + m^0_{post}}{p^*} \\
&\leq \frac{3m + 6k - 2 + 3m' + 6k' - 2 + l^0_{post} + m^0_{post}}{3m + 5k - 2 + 3m' + 5k' - 2 + l^0_{post}}
\end{aligned}$$

Note that m^0_{post} equals the number of (destination) *locations* mentioned in the post planning part. The right-hand side is maximized if $m^0_{post} = m + k = k = n$ and $m' + k' = k' = n$. Since in that case we also have $l^0_{post} = 2n$, we derive

$$\frac{|\hat{\mathrm{p}}^0|}{|p^*|} \leq \frac{15n-4}{12n-4} \leq \frac{5}{4} = 1.25.$$