Decomposition in Task Based Multi-Agent Planning Systems
with an application to logistic problems*

A.W. ter Mors J.M. Valk C. Witteveen†

Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands,
{a.w.termors, j.m.valk, c.witteveen}@ewi.tudelft.nl

Abstract

We discuss a general framework for coordinating self-interested agents in the pre-planning phase that can be used to decompose multi-agent task based planning problems into independent subproblems. The decomposition allows the agents to solve their part of the problem without the need to interact with other agents and such that the resulting plans can be easily combined into a joint plan. 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 *task planning* problems such as occur in e.g. manufacturing, supply-chain management and air traffic control. Often such problems can be modelled as multi-agent planning problems: using their individual tools and capabilities, a set of planning agents 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. Since interdependent tasks may be assigned to different agents, in general, the agents are not completely free to construct their own plan; therefore some form of *coordination* is needed between them to come up with a joint plan to complete the planning problem.

The currently dominant approaches towards coordination in multi-agent systems (cf. [4, 2, 9]) come down to either *post-planning coordination*, where coordination between the agents is established *after* the completion of the individual planning processes, or *coordination during planning*, where the agents continuously exchange planning information and adapt their plans to arrive at a joint solution. In both coordination approaches it is more or less taken for granted that agents are prepared to exchange information about their (partial) plans and to revise their plans in order to coordinate their solutions. This assumption makes them less suitable to solve planning problems with self-interested, competitive agents that do not want to be interfered during their planning activities and also are not prepared to revise their plan once it has to be combined with other plans into a joint solution.

In this paper we want to concentrate on planning methods for such non-cooperative agent systems. In such environments a *pre-planning* coordination approach seems to be a better alternative: before the agents start to plan, there is a single coordination phase (often part also part of the task assignment process) ensuring that the agents can plan independently and do not need

*This research is supported by the Dutch Ministry of Economic affairs under the SENTER TSIT program Cybernetic Incident Management (TSIT2021).

†Also at the Centre for Mathematics and Computer Science, P.O. Box 94079, NL-1090 GB Amsterdam, The Netherlands, email: C.Witteveen@cwi.nl

to revise their plan afterwards. Stated this way, the pre-planning coordination comes down to a generalization of a classical problem decomposition method like, e.g. *divide-and-conquer*.

Elsewhere (see [7]), we have presented a complete overview of this pre-planning coordination method together with a complexity analysis of the planning problems associated with this approach. In this paper we will give a brief overview of the framework and then show 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 protocol*. This protocol 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. We show that even these subproblems cannot be solved optimally (in reasonable time) and, moreover, that it is unlikely that there exist δ-approximation algorithms with $\delta < 1.2$ for them. We present a very simple and efficient 1.2-approximation algorithm to solve the subproblems (i.e., local planning problems). Combining the protocol with these polynomial approximation algorithms, we achieve a polynomial 1.25-approximation algorithm for solving the complete logistic planning problems.

Finally, in Section 4, we present some results obtained by applying our approximation algorithm to the logistic benchmark problems used in the AIPS2000 planning competition, showing that also existing planners can benefit from this simple coordination protocol.

2 The Coordination by Pre-Planning Framework

We consider task planning problems where a set T of interrelated tasks has to be solved by several autonomous agents, each having their own (task) capabilities and using their own (planning) tools. The set T consists of elementary tasks and is assumed to be partially ordered. Each elementary task, or simply *task*, is a unit of work that can be performed by a single agent. The partial order is induced by a *dependency relation* between tasks. Without loss of generality, we will assume that this dependency relation can be specified as a *precedence relation*: A task t_1 depends on another task t_2 if there exists a *precedence constraint* between them and if a task t_2 is preceded by task t_1, denoted by $t_1 \prec t_2$, then execution of t_2 may not start until t_1 has been completed; for example, achieving t_1 results in creating resources needed to perform t_2. Such a set T of interdependent tasks 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.

More precisely, a complex task $\mathcal{T} = (T, \prec)$ is a non-empty set of tasks $T = \{t_1, \ldots t_m\}$, partially ordered by a set of precedence constraints $\prec \subseteq T \times T$. This complex task is given to a set $\mathcal{A} = \{A_1, \ldots A_n\}$ 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 $f : T \to \mathcal{A}$ thereby inducing a *partitioning* $\mathbf{T} = \{T_1, \ldots, T_n\}$ 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 $\prec_i = \prec \cap (T_i \times T_i)$ constitutes the set of constraints for A_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 $(T_i, \prec_i)$ and the agents are dependent upon each other via the inter-agent constraints $\prec_{inter}$.

In this paper, 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.

Remark. When making a plan for a set of tasks T_i, the simplest case is when A_i exactly knows how to perform a task t and the actions to perform it bear no relation to the actions performed for another task t'. In that case, a plan for T_i can be easily composed from the set of actions needed to perform each individual task. Sometimes, however, there is no such compositionality: the plan

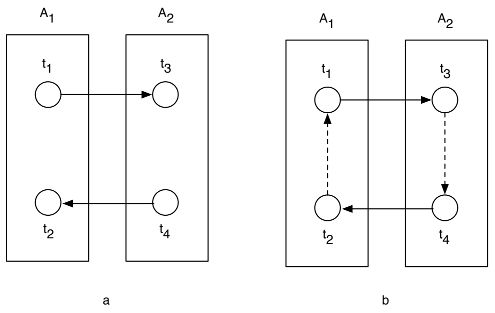

Figure 1: A set of interdependent tasks $T = \{t_1, t_2, t_3, t_4\}$ and two agents A1 and A2 each assigned to a part of T (a). If agent A1 decides to make a plan where t_2 precedes t_1 and A2 makes a plan where t_3 precedes t_4 (see b), these plans cannot be combined.

to perform two tasks differs from a simple ordering of the actions performed for each of the tasks separately. For example, if A_i has to perform the set of tasks $T_i = \{\text{buy milk}, \text{buy cookies}\}$ then his plan might be something along the lines of: go to supermarket ; get milk ; get cookies ; pay ; go home — the agent will not first go to the supermarket for the milk, and later return to the supermarket for the cookies. Thus, in the latter case, the agent has to solve a real planning problem to complete T_i. ■

In this framework we do not make any assumptions about the planning tools used by the agents, or about the particular planning problems the agents have to solve. 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 specify additional ordering constraints.

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

1. during the planning process, the agents are completely autonomous, i.e., can 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 an 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 $(\{t_3, t_4\}, \emptyset)$. Suppose now that 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. Then 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: $t_1 \prec t_3 \prec t_4 \prec t_2 \prec t_1$.*

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 constitutes the essence of our *pre-planning* coordination approach: 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 coordination sets, always exist. In general, however, such sets are extremely difficult to find (cf. [7]) and we have to relay on approximation algorithms to find non-minimal coordination sets that guarantee uncomplicated independent planning.

To illustrate the practical use of our approach, we apply this pre-planning coordination approach to a logistic planning problem used in the AIPS2000 planning competition. First, however, we will define the logistic planning problem and analyse its complexity into some detail.

3 Logistic planning problems

The logistic planning problem we will discuss 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 packages and for each package an order o consisting of a pick-up location $l \in Loc$ and a different 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 pickup 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[1].

3.1 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} the plane task of transporting the package to the airport of the destination city and finally t_{j3} 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 orders to the plane agents. 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 solution is not guaranteed if the agents plan completely independently from each other. So we will apply the idea of adding additional constraints in order to ensure independent planning. The method we use is using a protocol that implicitly adds some additional intra-agent constraints, by *scheduling* parts of the tasks the agents have to solve.

The protocol we will use to solve the logistic planning problem (the so-called *arbiter-0 protocol*) is surprisingly simple:

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

The arbiter-0 protocol

1. The total set of orders O orders is decomposed into a set of local transportation, pre-transportation, plane orders and post-transportation orders.
2. 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. This 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).
3. The planes have to make a plane plan for the plane orders.
4. All plans developed are assembled and 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.

A simple observation tells us that the plan thus composed indeed is a solution to the original problem.

Observation 1 *Let P_i be the plan of the truck agent i in city c_i for the local and pre-transportation, let Q be the flightplan of the planes and let R_i be the plan of the truck agents i for the post-transportation phase. Then the simple composition $P_{i_1} \circ \ldots P_{i_n} \circ Q \circ R_{i_1} \circ \ldots \circ R_{i_n}$ is a feasible transportation plan for all the orders given.*

The correctness of this plan can be easily verified: all local and pre- transportation plans can be completed independently; at the end of an arbitrary sequence of these plans, all intercity packages have arrived at the airport. Then the planes will deliver these packages at their destination airports and if this plane plan is completed all packages have arrived at the destination airports. Now the post transportation plans can be completed independently.

Surprisingly, this simple protocol, theoretically as well as experimentally, is one of the best coordination protocols we can find for this special logistic problem. In order to evaluate the performance of the protocol, we first concentrate on the single agent planning problems and their solution.

3.2 Finding solutions to the local 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.

The problem to find an arbitrary solution S is an easy problem: Let S' be an arbitrary sequence where every city in Loc occurs exactly once and consider the concatenation $S = S' \circ S'$. Since for every $(l, l') \in O$ an occurrence of l in the first subsequence and an occurrence of l' in the second subsequence can be selected, clearly S is a solution. To find an optimal solution requires to find a solution of minimal length. The problem, however, to find such a minimum solution is an NP-hard problem, since the NP-hard *minimum directed feedback vertex set* (MDFVS) [6] is easily reducible to it:

Proposition 1 *Let $G = (V, A)$ be an instance of the MDFVS-problem. Then S is a minimum visiting sequence solving the instance (Loc, O), where $Loc = V$ and $O = A$, if $F = \{v \in V \mid v \text{ occurs twice in } S\}$ is a minimum feedback vertex set of G.*

Proof See Appendix[2]. ∎

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

```
Stage 0
  Let $O = \{o_i\}$ and let $S$ be empty;

Stage i
  Let $o_i = (l, l')$ and let the current 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 algorithm achieves the approximation bound $\epsilon = 2$. The remarkable news is that, for the time being, *we cannot do better than this*: It is well-known [3] that the MDFVS-problem is an APX-hard problem[3] 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 the local planning problem would immediately imply that the MDFVS-problem is in APX:

Proposition 2 *Let $\delta > 1$ be an arbitrary constant. The MDFVS-problem is δ-approximable iff the local planning 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 minimum feedback vertex set 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 ratio for the order routing problem therefore is $\epsilon = 2$.

3.3 An approximation based on the arbiter-0 protocol

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: if S is the visiting sequence solution, the cost of the corresponding transportation plan P is (taking into account the additional load/unload actions): $|P| = 2|O| + |S|$. Let $|Loc| = n$ and $|O| = m$. Since the length of S is at most $2n$ and at most 2 times the length of the optimal

[2]Proofs are given to help the referee

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

visiting sequence, it follows that the performance ratio, i.e. the length of the local plan thus determined versus the length of the optimal local plan, is bounded above by

$$\frac{|P|}{|P^*|} \leq \frac{2m+2n}{2m+n}$$

where P^* is the optimal plan for the local planning problem. Assuming every location to occur in at least one order, we have $m \geq n$. In fact, it is easy to see that $m \geq |S|$ since according to the approximation algorithm we use, every location l that occurs twice in S is the result of adding at least two orders in which l occurs as source and/or destination. Hence, there must be at least two orders for each such an occurrence. Therefore,

$$\frac{|P|}{|P^*|} \leq \frac{4n+2n}{4n+n} = \frac{6}{5} = 1.2$$

Note that local planning problems constitute special cases (restrictions) of the complete 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 logistic planning problem as defined unless the minimum directed feedback vertex set is in APX.*

We will now show that our arbiter-0 protocol performs remarkably well with respect to this "lowerbound": if we use the arbiter-0 protocol and the approximation algorithm discussed above for solving the individual 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 3 *The logistic planning problem is* 1.25*-approximable.*

PROOF See Appendix. ■

Example 2 *We present a tight example. Consider a problem instance with $n+1$ cities and $n+1$ locations per city. The set of orders is*

$$\begin{array}{rcl} O & = & \{(loc_{1,1}, loc_{1,j}), (loc_{1,j}, loc_{1,1})|\ j = 2, \ldots, n+1\} \\ & \cup & \{(loc_{j,1}, loc_{1,k})|\ j = 2, \ldots, n+1,\ k = 2, \ldots, n+1\} \\ & \cup & \{(loc_{1,1}, loc_{j,1})|\ j = 2, \ldots, n+1,\ k = 2, \ldots, n+1\} \end{array}$$

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

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

4 Experimental results

The performance results obtained in the previous sections are theoretical and worst-case results. In order to test and to compare our arbiter-0 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 arbiter-0 approach is much better than should be expected, if the worst-case performance ratio is taken as the norm. Secondly, the arbiter-0 protocol 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 3 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.

packages	min nr moves	Arbiter-0	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

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 arbiter-0 protocol; the fourth, fifth, and sixth columns represent a selection of the planning systems competing in the AIPS: the competition-winning TALplanner [8], and the above-average performers STAN [10] 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 arbiter-0, 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.

The plans produced by arbiter-0 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 arbiter-0 protocol were a few seconds for each of the planning instances occurring in Table 1.

As we remarked before, the arbiter-0 protocol 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

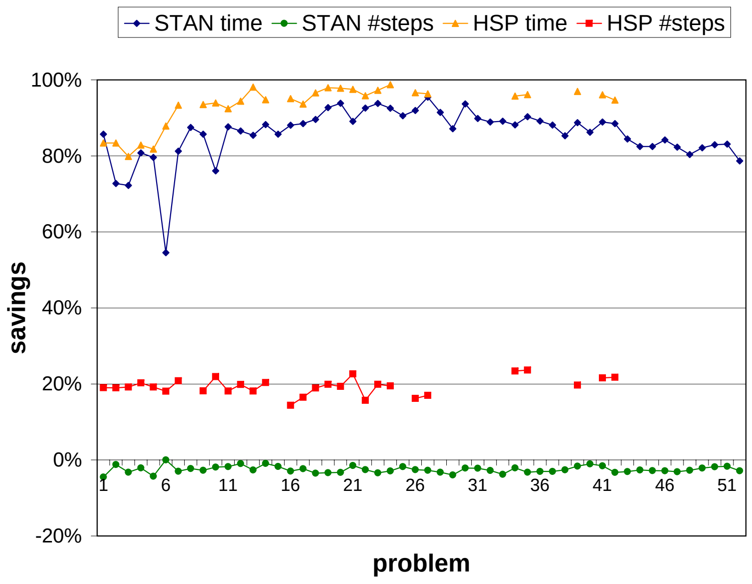

Figure 2: Savings in CPU times and plan cost (#steps) when STAN and HSP2 make use of the arbiter-0 protocol as a pre-processing step.

solved independently by the planning system, whereafter the solutions to the subproblems can be simply combined into a solution to the whole problem instance.

Specifically, what we propose is the following method: using the arbiter-0 protocol, decompose a multi-agent logistics instance into a set of single-agent planning problems. Then, we 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 multi-agent instance. What we would expect to see using this method is 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 arbiter-0 protocol. The results are given in Figure 2.

It can be observed that both STAN and HSP2 definitely benefit from pre-processing by the arbiter-0 protocol: 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 arbiter-0. 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.

5 Conclusion

We discussed a task-based planning framework and its application to a logistic planning problem. The framework allows us to identify constraints on subproblems to be solved by agents such that the final solution can be obtained by independent planning for these subproblems. 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.

It is interesting to note[4] that other approaches have also dealt with problem decomposition in planning. For instance, the planning systems STAN [5] and RealPlan [11] try to extract a number of particular subproblems, such as scheduling or transportation problems, as independent subproblems from the original problem. Thus, these systems decompose the original problem in a certain sense orthogonal to our decomposition method: we decompose on the basis of agents executing parts of the problem, the aforementioned planners decompose on the basis of subproblems. As our results suggest, combining both decomposition methods (e.g., running coordinated STAN) yields even more savings than using only one decomposition method, as Figure 2 shows.

References

[1] B. Bonet and H. Geffner. Heuristic search planner 2.0. *AI Magazine*, 22(3):77–80, Fall 2001.

[2] 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.

[3] P. Festa, P. Pardalos, and M. Resende. Feedback set problems, 1999.

[4] D.E. Foulser, Ming Li, and Qiang Yang. Theory and algorithms for plan merging. *Arificial Intelligence*, 57(2–3):143–182, 1992.

[5] M. Fox and D. Long. Hybrid STAN: Identifying and managing combinatorial optimisation sub- problems in planning. In *IJCAI*, pages 445–452, 2001.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness.* W.H.Freeman, 1979.

[7] 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.

[8] 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.

[9] 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.

[10] D. Long and M. Fox. Efficient implementation of the plan graph in stan. *Journal of AI Research*, 10:87–115, 1999.

[4]Thanks for the anonymous referees for pointing out this relationship.

[11] B. Srivastava. Realplan: Decoupling causal and resource reasoning in planning. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 812–818. AAAI Press / The MIT Press, 2000.

A Appendix

A.1 Proof of Proposition 1

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 order routing problem as follows: *(i)* Given an instance (Loc, O) of the order routing 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 (see ...).

Note that for all instances[5] 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 $f(n) = \delta$.

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 $\hat{\text{m}}^0_{pre}$, $\hat{\text{m}}^0_{plane}$, $\hat{\text{m}}^0_{post}$ 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 $\hat{\text{m}}^0_{post} = m^0_{post}$ 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 $\hat{\text{m}}^0_{pre} \leq 2m^0_{pre}$ and $\hat{\text{m}}^0_{plane} \leq 2m^0_{plane}$.(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.

Take, for example, $\hat{\text{m}}^0_{pre}$ (the case for $\hat{\text{m}}^0_{plane}$ is analogous). Let $\hat{\text{m}}^0_{pre} = m+2k$ 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 $l^0_{pre} + \hat{\text{m}}^0_{pre} = 2(m - 1 + 2k) + m + 2k = 3m + 6k - 2$ Analogously, we have the following relationships: the number of load /unload actions for the plane equals $l^0_{plane} = 2(m' - 1 + 2k')$ if $\hat{\text{m}}^0_{plane} = m' + 2k'$. Hence, $l^0_{plane} + \hat{\text{m}}^0_{plane} = 3m' + 6k' - 2$.

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

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

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

$$\frac{|\hat{\text{p}}^0|}{|p^*|} \leq \frac{15n - 4}{12n - 4} \leq \frac{5}{4}$$

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