Plan Merging: Experimental results

Mathijs de Weerdt, Roman van der Krogt and Jonne Zutt

Delft University of Technology,
PO Box 5031, 2600 GA Delft, The Netherlands
{M.M.deWeerdt | R.P.J.vanderKrogt | J.Zutt}@ewi.tudelft.nl

Abstract

In this paper we discuss the results of a *plan merging* algorithm. This algorithm coordinates the plans of multiple, autonomous agents, each able to independently find a plan. This algorithm is evaluated using realistic data from a taxi company. We show that when we allow passengers to be a few minutes later at their destination and share rides, we can obtain more than 5% reduction of the taxi driving distance. When we allow for a delay of 15 minutes (a common amount of time in subsidized transport) we can gain up to 30%.

1 Introduction

Many planning and coordination problems can be modeled as a multi-agent planning problem, i.e., a (distributed) set of related AI-planning problems. For example, the coordination of transport organizations, armies, production processes, etc., to name just a few. One of the techniques to solve such multi-agent planning problems is *plan merging*. Using this approach, we assume that each planner is capable of finding a solution for its own planning problem, for example using a refinement planning method [5]. When all agents have constructed their plans, these plans are compared to see whether cooperation is beneficial. For example, if during a plan two taxis bring someone from the train station to the hospital at about the same time, these passengers can share a taxi.

This paper discusses the results of a plan merging approach using so-called *resource facts* (part of our Action Resource Formalism [9, 10]). Such a resource fact aggregates the attributes of exchangeable objects into a single predicate, facilitating the search for opportunities to cooperate.

In the next section we take a closer look at the plan merging problem and its solution. Then, we present experimental results obtained from using this plan merging algorithm on data on the operations of taxi company. We finish by discussing possible extensions to the plan merging algorithm and related work.

2 Plan merging

The principle idea behind plan merging is that agents can refrain from work (i.e. they can remove actions from their plans) for which the results (i.e. the resources deliverd by these actions) are readily available at (several) other agents. Given

some plan for an agent that satisfies its goals, this agent can often reduce more costs by cooperating with other agents than by trying to optimize locally. In case multiple agents have similar parts in their plans, this can be quite advantageous. In this section we describe how this can be done by exchanging resource facts using *plan merging* [10].

To facilitate the exchange of resource facts, we assume one of the agents, or a trusted third party, acts as the *auctioneer*. The auctioneer announces when the agents can start the plan merging, thereby announcing some minimum allowed cost reduction value. All agents deposit requests with this auctioneer. Each *request* corresponds with the removal of an action from an agent's plan and contains a set of resource facts the agent needs to remove the particular action. Further, the request contains a cost reduction value defined by the difference in costs between the old plan and the resulting plan if the exchange succeeds.

The (greedy) auctioneer deals with the request with the highest potential cost reduction first. We assume all the agents honestly announce their cost reduction values. Right before each auction round starts, the requesting agent (a_i) is asked for the specific set of resource facts that has to be replaced by resource facts of other agents – this set is called the *RequestSet*. This set is not necessarily equal to the set in the initial request, since other exchanges influence the availability of resource facts for the agents.

To put up an auction for a request of an agent a_i, the set of requested resource facts is sent to each agent, except to a_i. The agents return all their free resource facts for which there is an equivalent one in the request set *RequestSet*, and include the price of each of their offered resource facts. When all bids (collected in R') are collected by the auctioneer, it selects for each requested resource fact the cheapest bid.

If for each resource fact in *RequestSet* a replacement can be found, the auctioneer tells the requesting agent a_i that it may discard the corresponding action(s). The replacing resource facts R'' are marked as goals for the providing agents, and become additional 'initial' resource facts for agent a_i. Furthermore, we have to add *dependencies* between these goals of the providing agents and the initial resource facts for the requesting agent.

If an agent's request succeeds, it discards the corresponding action in its plan, together with a possible set of actions in front of this particular action that also becomes obsolete. At the end of each successful exchange each involved agent has to update the cost reduction values of all of their requests, because this value can change as the agent can now have more or less resource facts available. One could repeat this process until none of the auctions has been successful.

The plan-merging algorithm is an *any-time* algorithm, because it can be stopped at any moment. If the algorithm is stopped, it still returns an improved set of agent plans, because this algorithm used a greedy policy, i.e., dealing with the requests with the largest potential cost reduction first. Algorithm 1 can be shown to have a worst-case time complexity of $O(n^2)$ where n is the number of actions of the plans of all agents involved in plan merging [10]. In this paper, we focus on the experimental results of the plan merging algorithm and give only a short introduction to the formalism and the plan merging algorithm itself. For a more elaborate

discussion of both the formal and practical analysis of this algorithm we refer to the thesis by De Weerdt [9].

Algorithm 1 PLAN_MERGING(A)

1. *auctioneer broadcasts minimum allowed cost reduction.*
2. *auctioneer retrieves requests with their cost reduction from all agents A.*
3. **while** *some requests left* **do**
   - 3.1. *get the request with the highest cost reduction.*
   - 3.2. *ask the requesting agent a_i for the required resource facts RequestSet.*
   - 3.3. **for each** *agent $a_j \in A \setminus \{a_i\}$* **do**
     - 3.3.1. *ask a_j for free res. facts equivalent to RequestSet.*
     - 3.3.2. *add these resource facts to R'.*
   - 3.4. **if** *$R' \supseteq RequestSet$* **then**
     - 3.4.1. *let $R'' \subseteq R'$ be the cheapest set that satisfies RequestSet.*
     - 3.4.2. *add for each $r \in RequestSet$ the corresponding dependency to R''.*
     - 3.4.3. *remove as much actions as possible from a_i.*
     - 3.4.4. *for each involved agent, update the cost reduction of all requests.*

3 Experimental results

3.1 Research questions and set-up

After a formal analysis of the plan-merging algorithm, two questions remained which we tried to answer by doing experiments.

- We know that the worst-case time complexity is $O(n^2)$. We are also interested in the constant term of the average time complexity, e.g., how long does it take to merge the plans of a whole day for a moderately sized taxi company.
- In the previous section, we claimed that single-agent plans can be further optimized by cooperating with other agents using this algorithm. We would like to verify this claim experimentally.

The tests are performed by merging the plans of taxis in a taxi company. From taxi company Zeevang in Purmerend we received information on rides of about 35 taxis exactly as they were planned in January and February 2002. This information includes a taxi number, (start and end) data and time, number of passengers and the origin and the destination location of each order. From this information we reconstruct the plan for each of the taxis, and try to find improvements over these plans as created by the dispatcher using the plan merging algorithm.

We assume that picking up a passenger along an already planned route has no long-term effects on the plan of a taxi. Furthermore, we assume that in this domain the costs of a plan equals the distance driven by the taxis. Consequently,

Table 1: The standard deviation of the fits in Figure 1 and 2.

Δt	run-time fit stdd.	reduction fit stdd.
3	0.310	12.6
6	0.468	21.0
15	0.517	30.9

our goal is to reduce this distance by exchanging orders among taxis (agents). Therefore, one of the resource facts describes a passenger and its attributes (the destination, their preferred pickup time, etc). The most important resource fact, however, describes the ride of a taxi and models the posibility for passengers to travel from one location to another. This is the only type of resource fact that is exchanged between taxis.

When a customer is transported by taxi A instead of by taxi B the additional distance and time needed by taxi A is estimated using the Euclidean distance and an analysis of all drives as extracted from the data set. An exchange is only allowed if the passenger's estimated arrival time is not increased by more than Δt minutes and the detour length of taxi A is less than the distance reduction of the other. Therefore, in this domain we define resource facts equivalent when their time attributes differ at most Δt minutes.

3.2 Results

First we analyze the running time of plan merging. We expect the worst-case running time to be $O(n^2)$ where n is the number of actions. To test this hypothesis, we run the algorithm with a fixed Δt of 3, 6, and 15 minutes and a fixed day on a number of plans varying from 2 to 35. Each test is performed 20 times on a randomly selected set of taxis. For each run we store the total number of actions of all involved plans and the run time on a 1GHz Pentium processor. The results of these runs can easily be fitted with a quadratic function (about $4 \cdot 10^{-5} n^2$), as can be seen in Figure 1. The standard error of these fits is very small, as can be seen in the first column of Table 1.

Next we are interested in the improvement of the plans. For each run we also store the total distance driven by the taxis, before and after the plan merging algorithm. In Figure 2 the difference between these values is plotted against the number of actions. As can be seen, more relaxed time constraints on the arrival time of the passengers lead to more improvement. Furthermore, the total improvement seems linear in the number of actions. The standard deviation of this fit is shown in the last column of Table 1. The *relative* improvement in drive distance (in percentage) is given in Table 2. The main disadvantage of reducing the distance driven by the taxis is that the passengers need more time to get to their destinations. Table 2 also shows the *increase* in passenger travel time (in percentage).

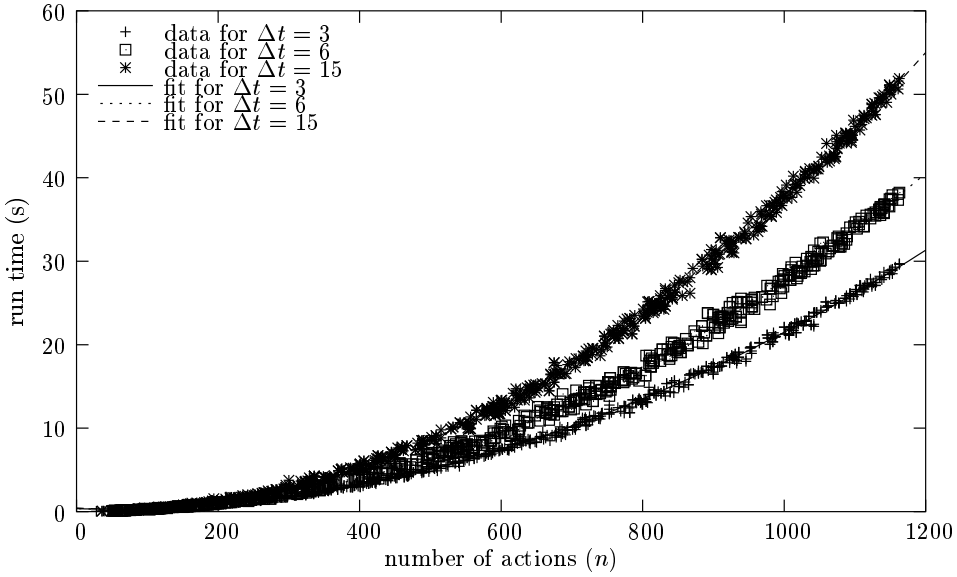

Figure 1: The run time versus the number of actions for three different values for Δt.

Table 2: The standard deviation of the decrease of the drive distance and the increase of the passenger travel time.

Δt	reduction distance (%)	stdd.	increase time (%)	stdd.
3	3.32	2.06	2.60	1.85
6	8.41	4.12	7.05	3.94
15	18.0	7.62	16.4	7.26

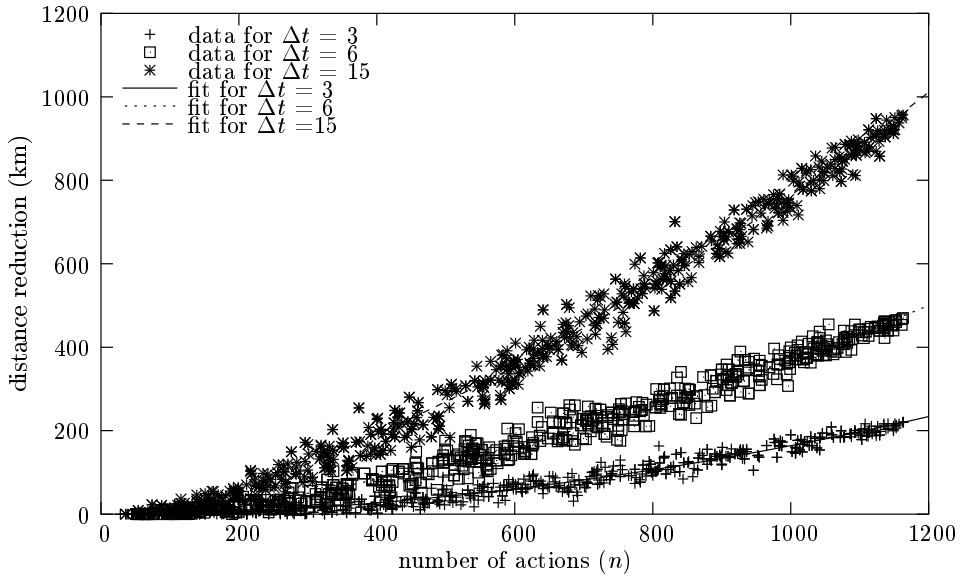

Figure 2: The improvement in drive distance versus the number of actions for three different values of Δt.

4 Discussion and related work

Plan merging is a method for coordinating plans of autonomous agents. For this algorithm we assume that each agent is able to construct a plan for its own problem and we try to obtain a more efficient solution by merging these plans. Experimental evidence is gained by running this algorithm on data from a taxi-company. The results show that this technique is both efficient and effective. The run time is quadratic in the size of all input plans. It takes less than a minute to improve the plan of a whole day of a taxi company consisting of 35 taxis (1200 actions). Within this time the distance driven by the taxis is reduced significantly. When passengers are allowed to arrive within an interval of 3 minutes from their desired arrival time the total distance can be about 5 percent improved over the schedule created by the dispatcher of the taxi company. Moreover, when passengers are allowed to arrive within an interval of 15 minutes, as is the current agreement for most Dutch low-budget elderly transportation services (Vervoer op Maat), the improvement can be up to 30 percent.

Although the plan merging algorithm and the used action resource formalism have already been published [8, 10], in this paper we (finally) show that they are powerful enough to describe and coordinate multi-agent plans. A disadvantage of plan merging is that it cannot be used in situations where an agent needs to cooperate with others to construct a plan. The plan merging scheme can be extended to a *multi-agent planning* method by (i) allowing agents to be able to *request* services from other agents and include the results in their plans, or (ii) by

allowing agents to be able to *offer* services to other agents and, upon a request, add these to their plans. Exchanging services enables agents to not only offer resource facts that are already in its plan (and unused), but also to adapt its plan to produce resource facts that are desired by other agents. Such extensions should lead to a distributed algorithm where self-interested agents create plans including coordinated (efficient and conflict-free) actions.

Most solutions to multi-agent planning problems *(i)* cooperatively create plans for all agents without dealing with the self-interestedness of agents, called cooperative distributed planning [2], such as PGP [1, 3], *(ii)* focus on task allocation [6] and conflict resolution *before* planning [4, 11], or *(iii)* conflict resolution *after* planning [7]. An extension of the plan merging algorithm should integrate coordination and conflict resolution in the planning phase, while maintaining the autonomous and self-interested aspects of agents.

We expect that such a coordinated planning algorithm yields even better results than the current plan merging algorithm, since opportunities to cooperate can be better utilized. Both the basic algorithm and the extensions use a resource fact oriented view on the world, and can be combined with most existing planning techniques. Such a general approach to coordinating plans of multiple agents can be used to solve many practical coordination problems, such as hospital scheduling, coordinating the transportation of goods or people, and managing the planning of joint forces on a mission of the UN.

To be able to use the proposed methods on integrating planning and coordination in these situations, still much work need to be done. Firstly, we need an adequate way to reward agents that offer services and share resource facts. Secondly, we need to know how to deal with agents that cannot or do not fulfill their contracts. Furthermore, we should test the developed algorithms in more realistic environments and improve them with (maybe even domain-dependent) heuristics. In addition, we need to look at a more dynamic (continual) version of the proposed algorithm where planning, replanning and execution are integrated. Finally, such approaches cannot be used in open multi-agent environments (e.g., the Internet) before a way is devised to deal with different ontologies (i.e., what are the resource facts in this domain), and a standard for agent communication and negotiation is chosen.

Acknowledgements

Pim van der Stoel from Tens B.V. and Ron Kooi from Taxi Zeevang made the data test set available.

Mathijs de Weerdt is supported by the Seamless Multimodal Mobility (SMM) research program and Roman van der Krogt and Jonne Zutt are supported by the Freight Transport Automation and Multimodality (FTAM) research program. Both research programs are carried out within the TRAIL research school for Transport, Infrastructure and Logistics.

References

[1] K. S. Decker and V. R. Lesser. Generalizing the partial global planning algorithm. *Int. J. of Intelligent and Cooperative Information Systems*, 1(2):319–346, June 1992.

[2] M. E. DesJardins, E. H. Durfee, C. L. Ortiz, and M. J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 4:13–22, 2000.

[3] E. H. Durfee. Distributed problem solving and planning. In G. Weiß, editor, *A Modern Approach to Distributed Art. Intel.*, chapter 3. The MIT Press, San Francisco, CA, 1999.

[4] D. Foulser, M. Li, and Q. Yang. Theory and algorithms for plan merging. *Art. Intel. J.*, 57(2–3):143–182, 1992.

[5] S. Kambhampati. Refinement planning as a unifying framework for plan synthesis. *AI Magazine*, 18(2):67–97, 1997.

[6] S. Kraus. *Strategic Negotiation in Multi-Agent Environments.* Intelligent Robots and Autonomous Agents. The MIT Press, San Francisco, CA, 2001.

[7] F. von Martial. *Coordinating Plans of Autonomous Agents*, volume 610 of *Lecture Notes on Art. Intel.* Springer Verlag, Berlin, 1992.

[8] J. Tonino, A. Bos, M. M. de Weerdt, and C. Witteveen. Plan coordination by revision in collective agent-based systems. *Art. Intel.*, 142(2):121–145, 2002.

[9] M. M. de Weerdt. *Plan Merging in Multi-Agent Systems.* PhD thesis, Delft Technical University, Delft, The Netherlands, 2003.

[10] M. M. de Weerdt, A. Bos, J. Tonino, and C. Witteveen. A resource logic for multi-agent plan merging. *Annals of Mathematics and Art. Intel., special issue on Computational Logic in Multi-Agent Systems*, 37(1–2):93–130, January 2003.

[11] Q. Yang, D. S. Nau, and J. Hendler. Merging separately generated plans with restricted interactions. *Computational Intel.*, 8(4):648–676, November 1992.