Reusing Past Plans in Distributed Planning

Toshiharu Sugawara
NTT Basic Research Laboratories
3-1 Wakamiya, Morinosato
Atsugi, Kanagawa 243-01, Japan
sugawara@ntt-20.ntt.jp

Abstract

This paper describes plan reuse in multiagent domains. In distributed planning, a plan is created by distributed centers of planner agents that have their own viewpoints. Plan reuse where a past plan result is reused for the new problem was proposed for single-agent planning and can achieve efficient planning. A special issue for applying it to distributed planning is that, even if the local agent thinks that the new problem is identical to a past problem, other agents may have quite different goals. Another issue is to realize efficient distributed planning, like in a single-agent case. This paper shows that the past plan can be reused regardless of other agents' goals under the assumption that the initial state has only "in-facts." A generated plan and related information are stored as a plan template so that an agent can reuse it in future planning. This information includes generated plans, subgoals, non-local effects that may affect or be affected by other agents' plans, and their conflict resolution methods that were actually used. An agent can create a plan efficiently using a template, because it can skip a part of planning actions, detect conflicts in an early stage, and reduce communication costs. First, this paper presents the planning-with-reuse framework. Then how plan templates are created and reused is also illustrated using some block world examples. Finally, we experimentally show that efficient distributed planning can be achieved.

Introduction

In cooperative distributed problem solving, distributed planning is important to generate cooperative activities. It is, however, a sophisticated task because of not only the computational cost of the fundamental planning algorithm but also costs of communications among agents, detections of task relations and generations of coordinated actions. This often results in difficulty in developing practical systems. "More efficient distributed planning" is a very important research issue.

In single-agent planning, *plan reuse* has been proposed by Kambhampati(Kambhampati 1993) to achieve efficient planning by avoiding repetitive planning activities. When a new problem is given, the system retrieves the plan that was created for a past identical or similar problem then modifies it so that it can be applied to the new one. This paper addresses the issue of whether or not, in *hierarchical* distributed planning, an agent can reuse past plans for new incoming problems in an environment where a number of identical or similar problems recur. There are a number of issues in realizing plan reuse in distributed planning for multiple agent actions.

The first issue is the similarity of new and old problems. Even if an agent thinks that the new problem is similar to a past one, other agents may have quite different goals. Is the plan created for the past problem useful for planning for the new one? Another important issue is performance, that is, can plan reuse really achieve efficient planning? Especially, it is important to reduce the costs of communication, conflict detection, and conflict resolution (negotiation) because they are relatively costly. The underlying ideas to these issues are that it is possible in such an environment that

(1) other agents' goals may affect the process of specialization of a high-level plan such as how to minimize and resolve conflicts, but the high level plan is usually similar or identical.

(2) even if there is a conflict, it can be detected in an earlier stage of planning and can be resolved by one of the methods used in the past.

(3) planning activities of a peer agent that does not reuse a past plan may also become simple by sharing a common subgoal.

We propose the plan-reuse framework that is applicable to the new problem regardless of other agents' goals. In this framework, a generated plan is stored as a plan template. A template has a number of plans to achieve the goal or subgoals (intermediary goals). Possible conflicts and resolution methods that were actually detected and used in the past planning activities are also stored. Unlike plan reuse in single-agent planning, a template is incrementally generated because a

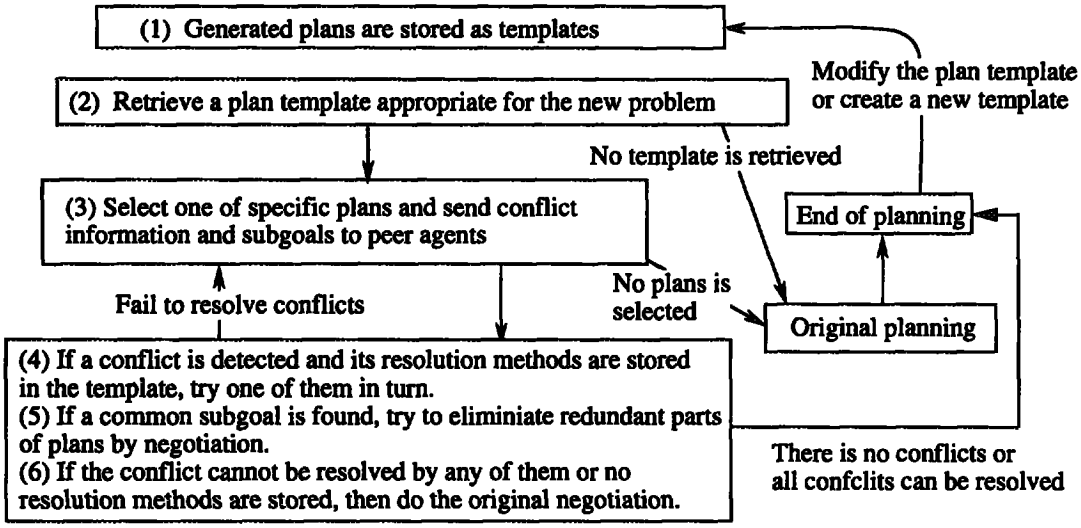


Figure 1: Planning with Reuse

planning result highly depends on the other agents' goals; in some problems, all agent goals may be independent thus no coordination can be made from the plan derived from this type of problem.

This paper explains plan reuse based on Corkill's distributed hierarchical planning algorithm and using block world examples. First, the planning-with-reuse framework for distributed planning is proposed in the next section. Reference (Sugawara 1994), which is a preliminary report, also proposed the plan reuse in distributed planning, but it does not clearly discuss the planning framework. Furthermore, the method proposed there has the drawback that plan reuse does not work well for positive relations(Martial 1992) among plans. In this paper, the plan template is extended to cope with this drawback. Then how a plan template is generated and reused is described using examples. We experimentally show that plan reuse can considerably reduce planning time then discuss why efficient planning can be realized. Finally, further research issues are discussed.

Plan Reuse Framework

Plan Template

When a plan for a given problem is generated and actually leads to the required goal or subgoal, the plan is stored as a *plan template*. A plan template has the following information:

Initial State and Goal: The initial state and its goal are stored. A plan template is a collection of operation sequences that can achieve this goal from the initial state. This pair of the initial state and goal are used to retrieve the template appropriate for a new problem.

Generated Plans in Each Level: In hierarchical planning, a plan is gradually specialized level by level. A higher-level plan is replaced by a sequence of lower-level plans (this lower-level plan is called a subplan in this paper) and this specialization process is iterated until all elements of the plan are replaced by operations that are executable by the agent. These generated (sub)plans in all levels are stored.

Subgoals of Plans: Any (sub)plan has a corresponding (sub)goal. Subgoals are also stored in a plan template to eliminate redundant actions.

Conflict Information: Actions in a plan may affect or be affected by other agents' activities and induce some conflicts. A detected conflict and which subplan causes or is affected by it are stored.

Resolution Methods: The resolution method for a detected conflict which actually led to a compromise is stored. An appropriate conflict resolution method depends on other agents' plans (because two compromises may result in another conflict), thus the stored methods cannot usually be utilized. We think, however, that they are helpful for planner agents working in an environment where a number of problems recur.

Templates are created based on planning activities for incoming problems. Generated (sub)plans and (sub)goals are stored hierarchically. Non-local effects and detected conflicts are also recorded and linked with plans that are related to those effects and conflicts. Actions for resolving conflicts that were actually used are also added. Examples of plan templates are presented in Section 3.

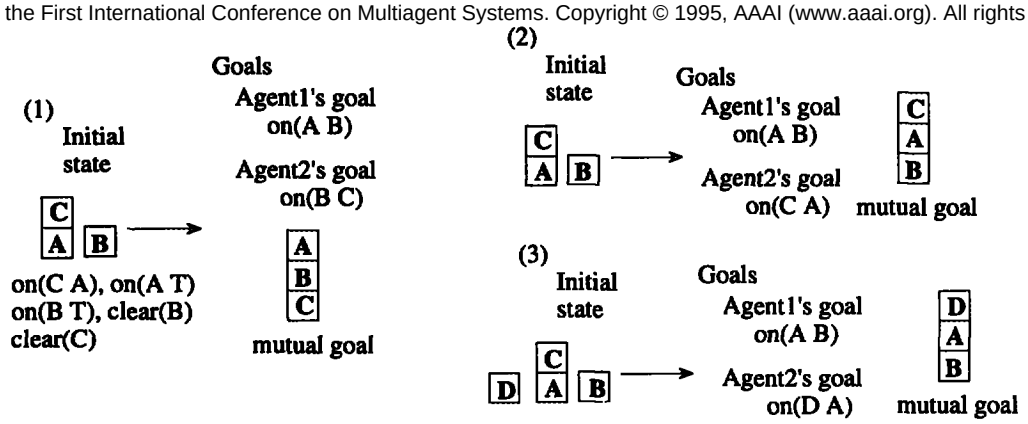


Figure 2: Block World Examples

Plan Reuse Framework

Suppose that a new problem and a past problem whose plan is stored as a template are given. An *in-fact* is the fact in the initial state of the new problem but not in that of the past problem. An *out-fact* is the fact that is not in the initial state of the new problem but is in that of the past problem. An *extra-goal* is the local goal of the new problem but not the goal of the past problem (Kambhampati 1993). We discuss the case where there are no extra-goals and no out-facts. This assumption may appear to restrict applicability, but from the global view point, extra-goals are always included and applicable problems are not so restricted. This discussion is presented in Section 5.

The proposed framework assumes that an agent has the original planning and negotiation processes. Our plan reuse is added to these processes. When a new problem is given, an agent retrieves an appropriate plan template whose initial state and goal are identical or similar (that is, they have only in-facts) to those of the new problem (See Fig. 1). If no plan template can be retrieved, the original planning process is invoked. The template has a collection of plans to achieve the goal. The agent selects one of them and broadcasts conflict information and subgoals related to the selected plan to the peer agents. If the agent or a peer detects a known conflict, the agent tries to resolve the conflict using recorded methods in turn. If a common subgoal is found, only one of the agents selects it and others eliminate this subgoal. If the local agent eliminates this part, it redistributes conflict and subgoal information, because the elimination of higher-level plans may also eliminate other nonlocal effects and subgoals in lower-level plans. If all known conflicts cannot be resolved or unknown conflicts are detected, the original negotiation process is invoked. If all conflicts are resolved, then this planning is done, otherwise the agent selects another plan from the template and iterates the above steps. When no more plans can be selected, the original planning process is invoked. After the execution of the plan (and when it actually leads to the desired result), the plan template is modified based on the new plan or the plan is stored as a new plan template.

Completeness of Planning

It is obvious that the proposed planning framework can always generate a plan for a problem if the original planning and negotiation can produce a plan for the problem, since the original planning and negotiation is invoked when it fails to reuse the past plan. However, the generated plan may not be optimal, while the original planning can generate the optimal plan.

Planning with Reusing Old Plans

This section describes how plan templates are created and how plan reuse proceeds using block world examples.

Distributed Planning

Corkill's hierarchical (non-liner) planner (Corkill 1979) is based on an analysis of the distributed NOAH system. First, we describe his distributed planning using the block world example in Fig. 2(1).

Given the goal on(A B), agent1 creates the first level of the plan as shown in Fig. 3(1b), then it sends an effect that may affect other agents' plans. A similar plan is also generated in agent2 (Fig. 3(2b)). Agent2 then finds that the agent1's plan denies clear(B), which may cause a conflict. Agent2 requests synchronization as the first resolution method, that is, it sends the request "wait deny(clear(B))" until put(B C) is completed. Agent1 receives and accepts this request. Since clear(A) in agent1 is not the phantom goal, this goal is further expanded as shown in Fig. 3(1c). At this level, however, agent1 finds the deny(clear(C)) from agent2 causes a conflict, thus it sends the request "wait deny(clear(C))" until put(C,Table) is completed. This message changes agent2's plan as shown in Fig. 3(2d). In addition, this plan indicates a constraint on the

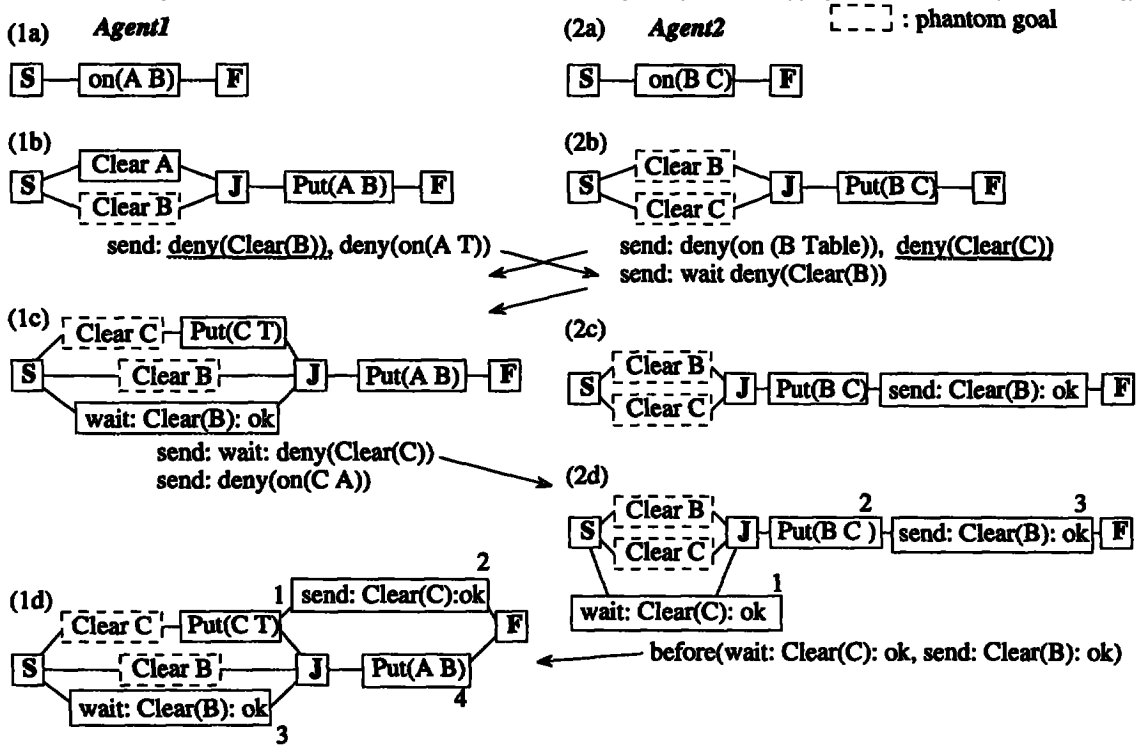


Figure 3: An Example of Distributed Hierarchical Planning

order of synchronization and agent2 sends this constraint to agent1. For agent1, there are no constraints on send:clear(C) and wait:clear(B), thus it can finally create the plan as shown in Fig. 3(1d).

Creating Plan Templates

Let us focus only on agent1's plan. This plan includes a number of results. It has clear(A) as a subgoal. It causes non-local effects deny(on(A Table)) and deny(clear(B)). Deny(clear(B)) actually affected agent2's plan and the conflict could be resolved by synchronization. Furthermore, the subgoal clear(A) was affected by deny(clear(C)) and this conflict was also resolved by synchronization. With the initial state and the goal of the problem, the final plan, subgoals, conflicts, and conflict resolution results are stored as a plan template, an example of which is illustrated in Fig. 4(1a) (the initial state is not included in the figures for brevity). Here 'deny' terms connected with actions, such as deny(on(C A)), are non-local effects and the underlined ones are 'resolved' by the method indicated by the lines (in this case, 'by synchronization'). Clear(A) and on(A B) are subgoals and nodes inside the corresponding rectangles are their subplans and operations. Collectively, these descriptions of effects and resolution methods are called *annotations*.

Plan Reuse (Detecting Conflicts)

The initial states and agent1's goals are identical but agent2's goals are not in the first and second problems (see Fig. 2 (1) and (2)). Given the initial state and goal of the second example, agent1 retrieves the template (Fig. 4(1a)), then sends the related subgoals and non-local effects described in the template. For example, the message, tellme:deny(clear(C)) indicates the request that, if agent2's plan denies clear(C), the effect should immediately be sent since agent1's plan is affected by it.

On(C A), which agent2 thinks is a phantom goal, is denied by agent1. By additional communications, agent2 finds it is no longer a phantom goal and it has to be re-achieved after it is denied (Fig. 4(2b))[1]. The node on(C A) is expanded (Fig. 4(2c)), then the effects caused by the plan are sent. The effect deny(clear(A)) may cause a conflict and should be resolved (in this case by synchronization). Also agent1 can conclude that agent2's plan does not cause deny(clear(C)) and is not affected by deny(clear(B)). The actions for resolving these conflicts are, thus, eliminated. The resulting plans are illustrated in Fig. 4(1b) and (2d).

[1]Negotiation is out of the scope of this paper, thus how to obtain a compromise is not discussed here.

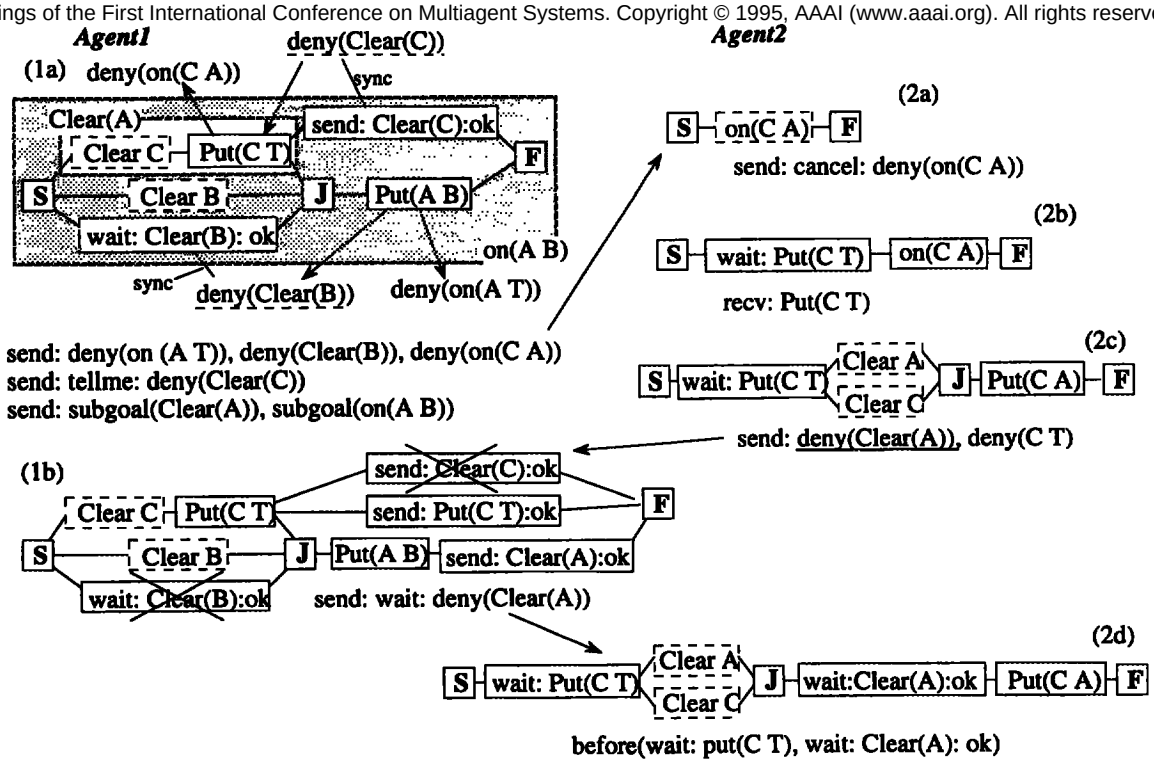


Figure 4: An Example of Plan Template and Plan Reuse

Modifying Plan Templates

New conflict information and resolution methods should be added into the plan template. The new template must be composed of the new plan and the original template, and it can be easily obtained by the following method. When a plan is reused, some nodes in the template, such as the 'wait:clear(B):ok' node, may be omitted. These nodes are not actually deleted in the memory but just have 'deleted' flags. The final templates can be acquired by eliminating the 'deleted' flags. The plan template is shown in Fig. 5(1a).

Plan Reuse (Sharing a Common Subgoal)

The example problem in Fig. 2(3) has a different initial state but only in-facts, thus agent1 can reuse the same template. In this problem, both agents have a common subgoal, clear(A). Agent2 plans this problem from scratch, but can take advantage of reusing a template in agent1.

Agent1 selects the same template and broadcast conflict and subgoal information. Agent2 concurrently starts its planning and sends conflict information in the first level to agent1 (Fig. 5(2a) and (2b)). The effect deny(clear(A)) is a known conflict and its resolution is attempted by synchronization that is recorded in the plan template (Fig. 5(1b)). In the second level of planning, agent2 finds the common goal clear(A) and commits this subgoal to agent1. Agent1 accepts this commitment. Then both agents can obtain the planning results in Fig. 5(1c) and (2c).

Another example in which plans in a template cannot be reused and thus the final plan is generated from scratch is described in (Sugawara 1994). In this case, the new generated plan and its conflict information are added into the template as alternative plan.

Evaluations

The experimental results are shown in Table. 6. The examples in Fig. 2 are too simple to evaluate plan reuse, so more complex examples were used in the experiments[2]. In E1 and E2, agent1 achieved approximately 7 times faster run time when it planned with reusing the template than when it planned from scratch. The elapsed time of both agent1 and agent2 was almost reduced by half. The run time of agent2 planning was also slightly improved. In E1 and E2, this plan reuse enabled the agents to quickly find conflicts and reduce the cost of communications. On the other hand, agent1 and agent2 drastically reduced their planning run time in E3. E3 had a number of common subgoals (positive relations), and planning from scratch was rather costly. Plan reuse could reduce planning time not only in agent1 but also in agent2 because

[2]Some results of planning time are shorter than those in (Sugawara 1994) because the author improved the planning program.

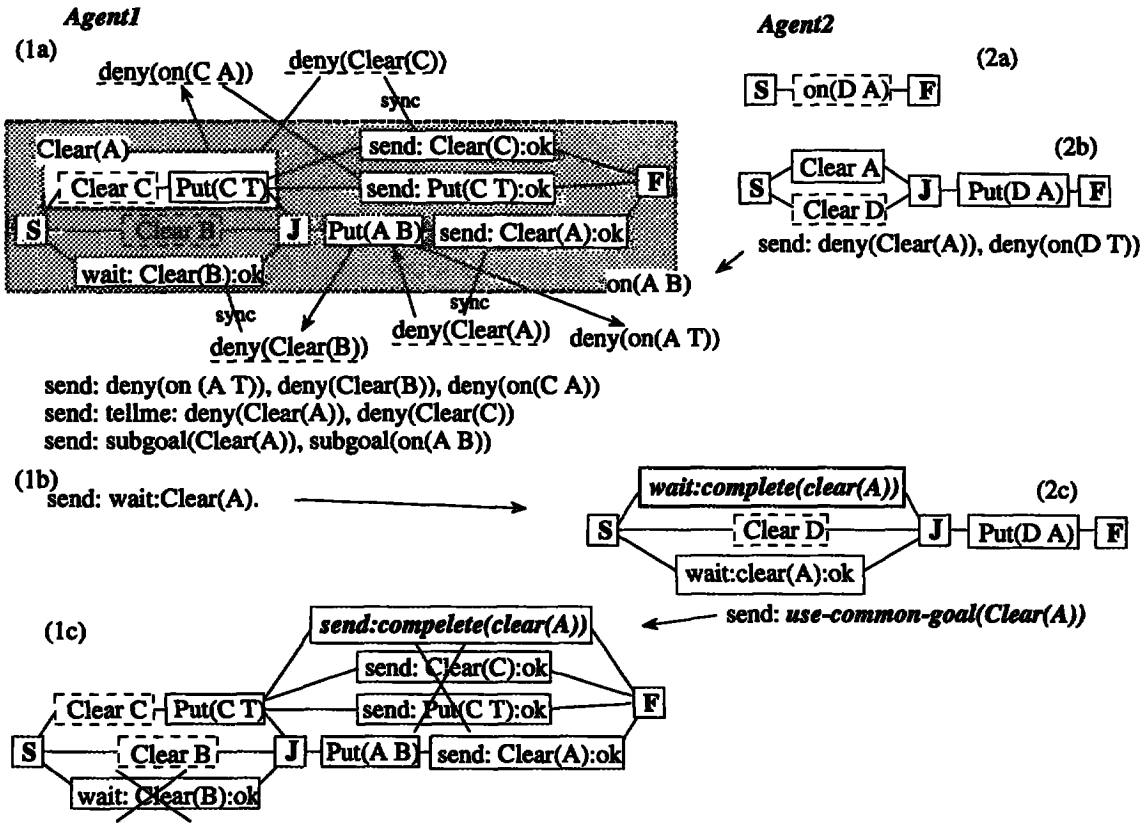
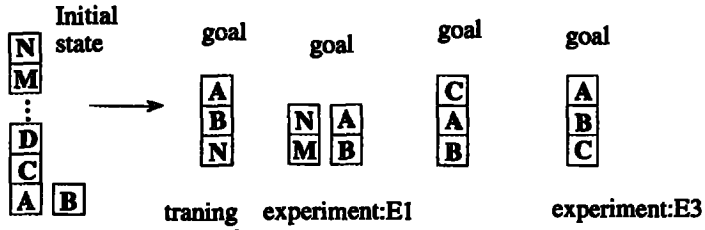


Figure 5: An Example of Plan Reuse (Sharing a Common Subgoal)



Agent1's goal	on(A B)	on(A B)	on(A B)	on(A B)
Agent2's goal	on(B N)	on(N M)	on(C A)	on(B C)
Elapsed time of Agent1		171ms (367ms) 2.15	156ms (337ms) 2.16	125ms (498ms) 3.98
Run time of Agent1		31ms (243ms) 7.84	35ms (227ms) 6.49	29ms (288ms) 9.93
Elapsed time of Agent2		169ms (369ms) 2.18	158ms (333ms) 2.11	127ms (663ms) 5.22
Run time of Agent2		60ms (71ms) 1.18	54ms (73ms) 1.35	58ms (317ms) 5.47

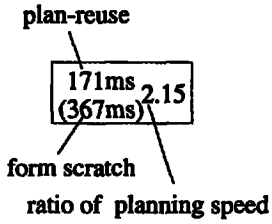


Figure 6: Experimental Results

agent2 was able to utilize the common goal clear(A) thus taking advantage of agent1's plan template.

From our experiments, we can conclude that plan reuse can achieve efficient planning overall. There are a number of reasons for efficient planning:

Skipping planning activities: An agent that reuses a template can skip a part of the planning process, as well as plan resuse in a single agent case. In the examples described in the previous sections, agent1's planning actions were only to retrieve the template, to detect unknown conflicts, and to resolve these conflicts.

Reducing communications: Plan reuse can be expected to reduce cost of communications. An agent that reuses the template can send recorded possible conflicts to its peers when it selects a specific plan among the template. Communications use little of the CPU resource but is time-consuming. Less communications can drastically shorten planning time in both the agent and its peers.

Detecting conflicts in earlier stages: Conflicts can be detected in an earlier stage, so efficient distributed planning can be achieved. An agent reusing a template can quickly find affected conflicts and then quickly invoke the negotiation process. A peer agent can also reduce planning time. This is because

(1) based on received conflict information, it can generate a plan that has less conflicts.

(2) it can find conflicts in an earlier step. For example, in the second example, if agent1 did not use the template, agent2 would have thought that on(C A) was a phantom goal, so it would have done nothing at first. However, after agent1 generated its plan, agent2 would have finally found that the goal was not a phantom anymore then would have invoked the planning process.

Sharing a common subgoal: By sharing common goals, an agent can indirectly take advantage of plan reuse in a peer agent because it can eliminate a number of planning steps for the common goals. The third example problem illustrates that agent2 could commit a subgoal to agent1 and skipped this planning part. The experiment E3 clearly shows that plan reuse in agent1 reduces planning time in agent2.

We can conclude, in general, that plan reuse in distributed planning is potentially more effective than in planning by a single agent, in the sense of planning time. Trade-off between the amount of memory to store plan templates, required planning time to create a plan and how frequently used each of templates should be considered in the future research (see the next section).

Discussion and Further Research Issues

Our plan reuse assumes that a new problem has only in-facts. This assumption may seem to be a strong limitation but, in fact, it is not. From the global view point, our plan reuse can be applied to problems that have extra-goals, because only the goal of an agent is identical and goals in other agents are thought to be extra-goals. It is also pointed out that planning from scratch is often faster than plan reuse. Costs for plan reuse is usually originate from the retrieval and modification of a past plan. In our framework, (1) a plan template which has only in-facts is retrieved (thus fast retrieval can be achieved), and (2) the modification is only done for conflict resolutions, which is also required to planning from scratch. Moreover, it is possible to reduce the number of communications which dominate planning time. The author thinks that, after a number of training instances, planning from a template is more efficient than planning from scratch, in multiagent domains.

On the other hand, another agent may also use another template in multiagent distributed planning, so the first issue is to discuss, in detail, the negotiation strategy for plan reuse. This paper assumes one of the conventional negotiation algorithms for the block world domain is used. This might be inefficient however. The required negotiation algorithm should be taken into account templates each agent has. For example, in the third example, the agents selected agent1's plan to achieve the common subgoal since it was already generated in the template. This strategy sounds reasonable. So when both of agents use the templates, which plan should be selected? The plan that has less non-local effects may be better, but not always.

The second issue is the inconsistency of conflict resolutions. Suppose that agent1 uses a template and agent2 and agent3 are its peers. Agent1 and agent2 have a conflict and it can be resolved by the method in the template. Agent1 and Agent3 have another conflict and it can also be resolved by another method there. However, using both of these resolution methods simultaneously may cause another conflict. It may be better that information on inconsistency among resolutions be stored in a template. The third issue, which is related to the second, is whether or not more information to select a conflict resolution method should be stored. If a template has more than two methods, conditions when one of them is successful and when another is better should be identified and stored. The author thinks that this issue has a strong relation with learning for coordination actions (Sugawara 1993).

The fourth issue is not only to add new information to templates but also to cut off obsolete or hardly used information, because plan reuse is effective for recurring problems. If a rare problem occurs as the first problem, its template is generated but hardly used. If the environment of an agent changes, a number of conflicts in a template may never oc-

cur and thus conflict resolutions for these conflicts are also never used. These obsolete data cause inefficient planning or, more harmfully, confuse other agents' planning. The author thinks that statistics such as usage of a plan, occurrence of a conflict, usage of a resolution method should be calculated and then unnecessary data should be eliminated according to them. The final issue is adaptation of this plan reuse to other multiagent planning algorithms such as (generic) partial goal planning(Durfee & Lesser 1991; Decker & Lesser 1992). We think that important relations (not only logical relations but also quantitative relations) between tasks that frequently appear in different agents should be stored as conflict information. More research on this issue is also strongly needed.

Conclusion

Plan reuse for distributed planning was proposed. The idea behind this plan reuse is that other agents' goals may affect how high-level plans are specialized to minimize and avoid conflicts but higher-level plans are hardly affected. Furthermore, possible conflicts can be expected in an early stage of planning. A plan created for an actual incoming problem is stored as a plan template then it is reused for identical or similar problems (that have only in-facts from the local viewpoint). A more complete plan template is incrementally generated based on further incoming problems. A plan template provides an action sequence to reach the goal from the initial state. For multiagent planning, it also includes the known conflicts that critically affect the local and other agents' plans. Resolution methods actually used in past plans are also stored in the templates. By sharing a common subgoal, the peer agents can also eliminate a number of planning steps. Through a number of experiments, plan reuse was shown to achieve efficient planning overall. Plan reuse in distributed hierarchical planning is, at least, expected to be more effective than in single-agent planning.

References

Corkill, D. D. 1979. Hierarchical Planning in a Distributed Environment. in Proc. of IJCAI, 168-175.

Decker K.; and Lesser V. 1992. Generalizing the Partial Global Planning Algorithm, *International Journal on Intelligent Cooperative Information Systems*, 1(2):319-346.

Durfee E. H.; and Lesser V. R. 1991. Partial global planning: A coordination framework for distributed hypothesis formation, *IEEE Trans. on System, Man and Cybernetics*, 21(5):1167-1183.

Kambhampati S. 1993. Supporting Flexible Plan Reuse, in *Machine Learning Methods for Planning* ed. by Minton S., 397-434.

von Martial F. 1992. *Coordinating Plans of Autonomous Agents*, Lecture Notes in AI 610, Springer-Verlag, Berlin.

Sugawara T. and Lesser V. R. 1993. On-Line Learning of Coordination Plans, COINS Technical Report, 93-27, Dept. of Computer Science, Univ. of Massachusetts. (A shorter version of this paper is also published in Proceedings of the 12th Int. AAAI Workshop on Distributed AI, 1993)

Sugawara T. 1994. Plan Reuse in Cooperative Distributed Problem Solving, In Proceedings of the Seventh Australian Joint Conference on Artificial Intelligence (AI94).