Conflict Estimation of Abstract Plans for Multi-Agent Systems

Toshiharu Sugawara[1] Satoshi Kurihara[2] Toshio Hirotsu[3]
Kensuke Fukuda[4] and Toshihiro Takada[5]

[1]Waseda University
Shinjuku, Tokyo 169-8555, Japan
sugawara@entia.org

[2]Osaka University
Ibaraki, Osaka 567-0047, Japan
kurihara@ist.osaka-u.ac.jp

[4]National Institute of Informatics
Chiyoda, Tokyo 101-8430
kensuke@nii.ac.jp

[3]Toyohashi University of Technology
Toyohashi, Aichi 441-8580, Japan
hirotsu@entia.org

[5]NTT Communication Science Laboratories
Atsugi, Kanagawa 243-0198, Japan
takada@brl.ntt.co.jp

ABSTRACT

In hierarchical planning, selecting a plan at an abstract level affects planning performance because an abstract plan restricts the scope of primitive plans. However, if all primitive plans under the selected abstract plan have difficult-to-resolve conflicts with the plans of other agents, the final plan after conflict resolution will be inefficient or of low quality. In this paper, we propose a conflict estimation method to generate quality plans efficiently for multi-agent systems by appropriately selecting abstract plans in hierarchical planning. This method enables agents to learn which abstract plans are less likely to cause conflicts or which conflicts will be easy to resolve.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Distributed artificial intelligence

General Terms

Theory

Keywords

Planning, Conflict Resolution, Coordination

1. INTRODUCTION

The objective of our research is, in hierarchical planning[2, 3], to predict which tasks in an abstract plan will conflict with other agents' plans at a lower level with higher probability and either involve a costly conflict resolution process and/or result in a low-quality plan after it has been resolved. To isolate this kind of situation, Sugawara *et al.* [4] introduced *conflict patterns (CP)* at a certain abstract level called the *screening level (SL)*. In addition, they introduced a negative utility, called *conflict discount*, which cumulatively predicts the probability of conflicts in the subsequent refinement process, the cost of resolutions, and the quality/performance of the resulting plans on the basis of CPs in the SL plans and past experience. The conflict discount is calculated and updated by using statistically learned expected values or by reinforcement learning, so that the agents select a more appropriate refinement at the SL. We assume that the initial utility is good for selecting appropriate plans only for single-agent cases. Thus, agents learn the conflict discount appropriate for the MAS environment in order to select better SL plans. While reference [4] illustrated the way in which the conflict discount is adjusted, it did not experimentally tackle the question of whether or not the final plans were actually efficient.

In this paper, we formally define conflict patterns and discuss the estimation of their conflict discounts. A notion of sub-conflict patterns for avoiding redundant calculations conflict discounts and reducing memory space is introduced. An experimental evaluation of the efficiency of plans generated by our method for a simulated laboratory room will be shown in out presentation. We will omit the background to our issue because of limited page numbers; see [4] for a detailed discussion.

AAMAS'07, May 14–18, 2007, Honolulu, Hawai'i, USA.

2. SCREENING LEVEL

We only briefly describe the background to the issue and our planning architecture. See [4] for a more detailed discussion.

Our planning architecture in the current version is centralized but agents share only the plans described in a certain abstract-level model; the *manager agent* (or *m-agent*) holds the plans that are being scheduled or executed at this level. This level is called the *screening level (SL)*. We assume that the plans at this level are simpler than ones at the primitive level but are detailed enough to be analyzed for conflict estimation. Hence, while the m-agent does not have all the detailed plans, it does maintain all SL plans presently scheduled or being executed (These SL plans are called *SL-valid* plans). This centralized architecture may cause the concentration of jobs. However, to reduce the overload by this concentration, the m-agent only checks SL-valid plans, which we assume that is simpler and usually a restrictive number.

The m-agent detects possible conflicts, according to resource and task information at the SL, by identifying the possibility of whether multiple plans will use the same resources, such as locations (e.g., squares in Fig. 1). An example is illustrated in Fig. 2, for which the SL is level 2 in Fig. 1; a square at this SL (specified by a

978-81-904262-7-5 (RPS)

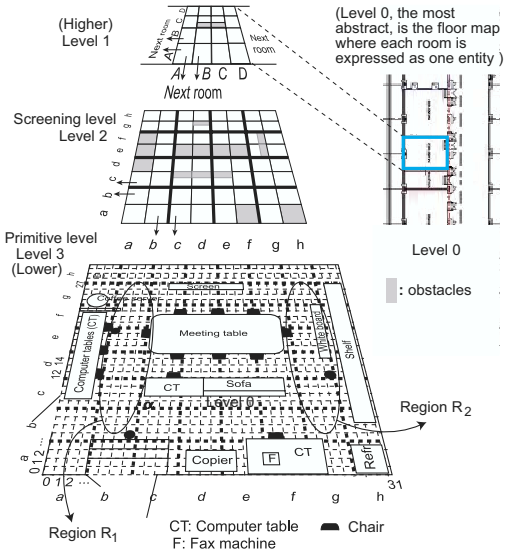

Figure 1: Example of hierarchical description.

pair of lower-case bold letters) corresponds to 4x4 squares in the primitive model (a square in the primitive level is specified by a pair of positive integers). In Fig. 2, the m-agent can suggest that task t_l = *move(**cd**→ **dd**)* in the new plan may conflict with task t'_n =*move(**cd**→ **bd**)* in the SL-valid plan, where *move(**cd**→ **dd**)* is the SL plan expressing the agent's movement from somewhere in area ***(c,d)*** to area ***(d,d)***. This conflict can be expected if some squares in area ***(c,d)*** can be simultaneously occupied by two agents during a certain time interval.

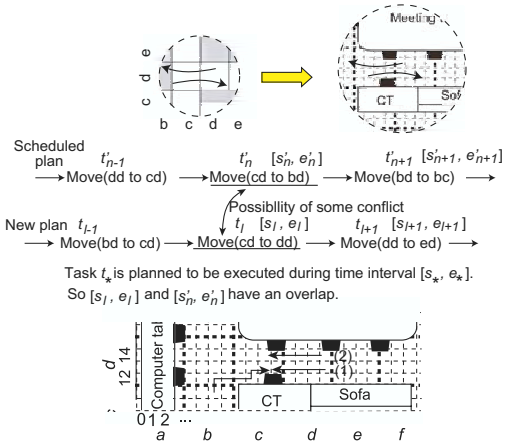

Figure 2: Example of a detected conflict.

The questions of when and where conflicts likely occur and whether their resolutions are difficult depend upon the system's environment. Suppose that three agents want to pass through area ***(b,d)***. In the SL model, this area (place is a resource) is expressed as a single entity, so conflicts can be expected. However, this area has enough room for three agents if each agent occupies a small square at the primitive level; hence, the conflicts might not actually occur or might be easily resolved. However, in ***(c,d)*** where agents move only left or right, there is not enough room for three agents. Thus, it seems probable that the agents' plans will have conflicts there. Of course, this probability is influenced by the temporal relationships of the agents entering area ***(c,d)***. If a conflict is detected, one of the agents must step out of the other agent's way and wait for it to pass by before resuming its movement.

3. CONFLICT ESTIMATION

In this section, we explain how utilities with conflict discounts are learned through experience.

3.1 Conflict pattern

We introduce the concept of the *conflict pattern (CP)* of to express the conflicts of a plan with the plans of other agents. First, we focus on an SL task identified as having a conflict. Let t be an SL task in a new SL plan p, denoted by $t \in p$. Suppose that SL plans $p_1, \ldots, p_k$ of other agents are SL-valid. Then the CP $\mathcal{P}$ is expressed as

$$\mathcal{P}(t) = (t, (t'_1, o_1), \ldots, (t'_h, o_h))$$

where $t'_i \in p_j$ $(1 \leq \exists j \leq h)$ and o_i is optional data. This CP describes the situation where t is expected to conflict with $t'_1, \ldots, t'_h$ in SL-valid plans.

The optional data o_i can be any information that can be used to distinguish conflicting situations more accurately. For instance, it may be information about (relative) the time of execution and agents' names or types that suggest their ability/performance or physical size (when agents have physical bodies, such as robots and vehicles). In the example of Fig. 2, the CP is expressed as

$$\mathcal{P}_1(t_l) = (t_l, (t'_n, (\max(s'_n - s_l, 0), \min(e_l - s_l, e'_n - s_l)))),$$

where the optional data is the relative time interval during which the expected conflict may occur. To simplify the expression of this example, we describe the optional data in a more abstract form. For this purpose, we can use the expressions of time relativity; the duration of t'_n overlaps the anterior half[ah] or posterior half[ph] of the duration of t_l. Other cases of time relativity are expressed as 'overlap[ol].' Thus, $\mathcal{P}_1(t_l) = (t_l, (t'_n, r'_l))$, where r'_l = *ah, ph* or *ol*; this is the simplified relative time expression of [1].

The situation in Fig. 3 shows that t_l may conflict with t'_{n+1} and t''_{m-1}. The following CP corresponds to this situation:

$$\mathcal{P}_2(t_l) = (t_l, (t'_{n+1}, r'_l), (t''_{m-1}, r''_l))$$

where r'_l, r''_l = *ah, ph* or *ol*.

3.2 Concept of conflict discount

Let $U(p)$ (or $U(t)$) be the initial utility for a primitive plan p (or a primitive task t). $U(p)$ for a non-primitive plan (or task) is the range that cumulatively indicates possible lower-primitive plans/tasks. We introduce the *conflict discount* for a CP, $cd(\mathcal{P})$. The conflict discount is conceptually defined as

$$cd(\mathcal{P}) = U(pp) - U(pp_m) + CCR(\mathcal{P}) \qquad (1)$$

where pp is the primitive plan of the SL plan p before conflict resolution, and pp_m is the modified primitive plan for resolving conflict $\mathcal{P}$. The term *CCR* indicates the cost of conflict detection and resolution at the primitive level, which is calculated by combining the cost of requesting, receiving, and analyzing primitive plans from other agents and applying conflict resolution rules to modify the

new plan. So even if no conflict actually occurs at the primitive level ($U(pp) = U(pp_m)$), $cd(\mathcal{P}) \neq 0$. This is because, if a conflict is expected at SL, the cost of conflict detection will be incurred. Define $cd'(\mathcal{P}) = U(pp) - U(pp_m)$ as the difference of utility. The estimation of $cd(\mathcal{P})$ is described in the next section.

When an agent has a new SL plan p that is expected to have CPs $\mathcal{P}_1 \ldots \mathcal{P}_N$,

$$cd(p) = \sum_{i=1}^{N} cd(\mathcal{P}_i).$$

The agent uses the modified utility $U(p) - cd(p)$ instead of $U(p)$. When no conflicts are predicted, the agent uses $U(p)$ since $cd(p) = 0$. Our method statistically adjusts the conflict discounts for frequently appearing CPs. Because we focus on the efficiency of plans, we assume that $U(p)$ is the estimated execution time of the primitive plan in the example below.

3.3 Estimation of conflict discount

The conflict discount for a CP, $cd(\mathcal{P})$, is iteratively adjusted by the average function

$$cd_s(\mathcal{P}) = \sum_{i=1}^{s} d_i/s \qquad (2)$$

when CP is observed s times. If the environment can change, the following update function is more adaptive:

$$cd_s(\mathcal{P}) = \lambda * cd_{s-1}(\mathcal{P}) + (1 - \lambda) * d_s \quad (3)$$

where $0 < \lambda < 1$ and d_s indicates the s-th CCR_s plus the s-th observed differential utilities of the original primitive plan and the plan after the resolution of conflict corresponding to $\mathcal{P}$. Note that the conflict of $\mathcal{P}$ might not occur at the primitive level; if so, $d_s = 0 + CCR_s$. For example, if the partner agent takes route (1) in Fig. 2 and this conflict is resolved by a detour or by using "wait for two ticks" to wait until the partner agent passes by. In this case, $d_s = 2 + CCR_s$. However, if the partner agent takes route (2) in Fig. 2, no conflict actually occurs and $d_s = 0 + CCR_s$.

To acquire the CCR value for each plan, we assume that agents can monitor their planning activities by themselves. More precisely, CCR consists of the time for (1) requesting and receiving primitive plans from other agents that are suggested to have conflicts, (2) detecting actual conflicts between these plans and the local plan, and (3) modifying the local plan to resolve these conflicts.

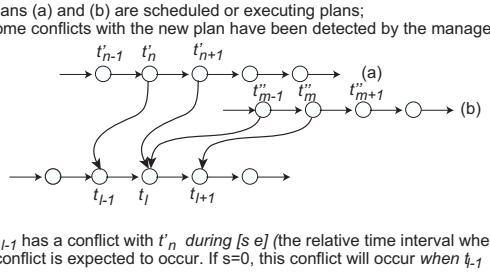

Figure 3: Example of conflicts between plans.

3.4 Sub-conflict patterns

It is probable that many CPs will be created, and storing many CPs would require a large amount of memory. This also incurs a large search cost, which degrades scalability. It also lowers the performance of conflict estimations of the CPs. Here, we can try to reduce the memory taken up by the CPs.

Suppose that $\mathcal{P}_1$ and $\mathcal{P}_2$ are CPs:

$$\mathcal{P}_1 = (t, (t_1, r_1), \ldots, (t_n, r_n))$$
$$\mathcal{P}_2 = (t', (t'_1, r'_1), \ldots, (t'_m, r'_m))$$

If $t = t'$ and $\{(t_1, r_1), \ldots, (t_n, r_n)\} \subseteq \{(t'_1, r'_1), \ldots, (t'_m, r'_m)\}$, then $\mathcal{P}_1$ is the sub-conflict pattern (sub-CP) of $\mathcal{P}_2$, denoted by $\mathcal{P}_1 \subseteq \mathcal{P}_2$. Now, we assume that $cd(\mathcal{P}_1) \leq cd(\mathcal{P}_2)$ if $\mathcal{P}_1 \subseteq \mathcal{P}_2$. This is a natural assumption because $\mathcal{P}_1$ is resolved if the conflict with $\mathcal{P}_2$ is resolved.

To save memory, the m-agent only stores CPs whose conflict discount values are near the turning point of the decision. For example, if $cd(\mathcal{P}_2)$ is sufficiently small, the cd value for $\mathcal{P}_1$ ($\subseteq \mathcal{P}_2$) will not necessarily be stored, so its cd estimation can be eliminated. Similarly, if $cd(\mathcal{P}_1)$ is large (so the agent will give up the current SL plan), the cd value for $\mathcal{P}_2$ ($\supseteq \mathcal{P}_1$) does not have to be stored.

4. EXPERIMENTAL RESULTS

Our experimental results showed that our method can reduce the length of plans about 7% in the case described in Fig. 2. The detailed graphs and data are shown in our poster presentation.

This improvement seems fairly small, but our simulated laboratory room is based on one of our actual rooms; we believe that our method is more significant in other situations. For example, if more robots moves right to left in the narrow area in Fig. 2 or the chair there is a bench (a longer chair), this improvement becomes larger so the resulting plans has relatively higher quality than the ones obtained by a conventional planning strategy.

5. CONCLUSION

This paper proposed a method to predict, at an abstract level called the screening level, the cost of possible conflict resolution, and the quality of the resulting plan, in order to generate better primitive (concrete) plans. In our framework, an agent called the manager agent maintains the plans that are scheduled or being executed at the screening level and predicts possible conflicts between these plans and the newly proposed plan. Then, if necessary, a detailed analysis of primitive plans is performed by individual agents. We conducted experiments to reveal the estimated additional cost (estimated cd and cd' values) of the plans after conflict resolution and the efficiency of plans derived from our method. Our method enables agents to decide whether the current plan should be refined or another plan should be created in an earlier stage, that is, before an agent creates its primitive plan; this decision makes agent's planning efficient.

Acknowledgement: This material is based on the work carried out when the first author was in NTT Communication Science Labs.

6. REFERENCES

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832 – 843, 1983.

[2] J. H. K. Erol and D. S. Nau. HTN planning: Complexity and expressivity. In *Proc. of AAAI-94*, pages 1123–1128, 1994.

[3] E. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2):115 – 135, 1974.

[4] T. Sugawara, S. Kurihara, T. Hirotsu, K. Fukuda, and T. Takada. Predicting Possible Conflicts in Hierarchical planning for Multi-Agent Systems. In *Proc. of AAMAS2005*, pages 813 – 820, 2005.