

An Approach to Plan Recognition and Retrieval for Multi-agent Systems¹

Zhanxiang Huang, Yang Yang, Xiaoping Chen

Department of Computer Science and Technology
University of Science and Technology of China
POBOX4, Hefei, Anhui, 23007, China
huangzx@mail.ustc.edu.cn

Abstract. Planning plays an important role in the domain of multi-agent systems. Recognizing opponent's plans and taking appropriate countermeasures will improve a team's adaptability and cooperation in dynamic, adversarial environments. This paper describes our work on establishing a mechanism to recognize and retrieve team plans based on observations of agents' coordinative behaviors.

1. Introduction

Many rational agents in multi-agent systems depend on their plan library while reasoning, making decision and executing coordinate behaviors. It is certain that a team whose member agents could analyze their opponents' actions and take appropriate countermeasures will have advantages in competitions. Moreover, recognizing and learning from other teams' plan will help to enhance a team's cooperation and to construct its own plan library. Hence it is significant to establish a mechanism analyzing opponents' behaviors to recognize and retrieve their plans automatically. For this purpose, many previous researches have been conducted. As for plan representation, there are the syntactic and semantic representation [9], temporal constraint network [6,8], and hybrid representation [1], etc. On plan recognition, logical reasoning and statistical methods are both used [4]. On the basis of these researches, we develop a technique for auto-plan-recognition and implement it in RoboCup simulation robot soccer domain. Our approach, combining plan representation, plan recognition and retrieval techniques, translates the multi-variable information stream obtained from the observation of the dynamic and adversarial world into streams of agents' behaviors using prediction and backfill techniques. After that, interesting and frequent behavior sequences, patterns, in those streams are found by statistical dependency test. Then relevant plans are retrieved from the patterns and converted into certain forms of the

¹ This work is supported by NSFC (60275024) and High-Tech Project (2001AA422200).

formalized plan representation. Experiment results indicate that our techniques can extract interesting patterns and turn them into a team's own plans.

The rest of the paper is organized as follows: In section 2 we present a sort of formalized multi-agent plan representation. In section 3 the critical techniques of translation from observation to agents' behaviors are depicted. Sequential behavior pattern recognition method we adapted is described in section 4 and our plan retrieve algorithm in section 5. Experiments and results are given in Section 6 and conclusion and future work in section 7.

2. Team plan representation

A team plan should contain the list of agents it involves, its starting condition, its goal state, and the body, which is composed of several agents' behaviors that have temporal constraints on each other. Unlike temporal constraint network [6,8] which has complex asynchronous restrictions between agents' behaviors, our representation is simplified to synchronous, orderly expression that consists of multiple steps, each of which contains the co-occur behaviors of agents. Then it can meet the request of auto plan retrieval. Furthermore this expression describes the condition to execute this plan and the goal state after the plan is finished. The plan representation is defined by BNF as follows:

```
plan ::= agent-list start-condition goal-state body;  
agent-list ::= agent+;  
body ::= step+;  
step ::= leading-act associate-act*;
```

This plan representation can be used to describe plans in multi-agent system other than robot soccer teams, like simulation battle field etc, and only few changes need making. In the specific field of robot soccer game, this plan representation is embodied below:

```
start-condition ::= (agent region)+;  
goal-state ::= (BALL region);  
leading-act ::= leading-type region-info role-info;  
associate-act ::= associate-type region-info role-info;  
leading-type ::= INTERCEPT | PASS | DRIB | SHOT;  
associate-type ::= POSITION  
region-info ::= source-region goal-region;  
role-info ::= leading-agent associate-agent;  
source-region ::= region;  
goal-region ::= region;  
leading-agent ::= agent;  
associate-agent ::= agent;  
agent ::= agent-ID;  
region ::= region-ID;
```

The region-ID corresponds to one area in the soccer field which could take any specific shape, and the agent-ID is mapped to one of the player in the soccer team we observed. The definition indicates that each step in a plan must include one and only one leading action, but any number of associate actions could co-occur with the leading action in one step. The first step in the plan can be carried out only when the start condition is satisfied and other steps must wait until the steps before them are executed and finished. One sample of a one-two plan, which is frequently used in soccer games, demonstrates the plan representation definition below.

```

plan one-two{
  agent-list: 1,2
  start-condition: (1 region1) (2 region2)
  goal-state: (BALL region4)
  body: step1: leading-act: (PASS 1 2 region1 region3)
        step2: leading-act: (INTERCEPT 2 2 region2 region3)
              associate-act: (POSITION 1 1 region1 region4)
        step3: leading-act: (PASS 2 1 region3 region4)
};

```

This formalized plan presentation is the foundation of our whole work. And the rest process has several stages: First the observations are translated into agents' behavior queues. Second the frequent and interesting sequences are picked out from the agents' behavior queues. Third those sequences are retrieved and transform into plans as the form of our plan presentation. Finally these plans are refined while multi-agent teams adopt them. Work mentioned above will be discussed orderly in the following sections.

3. Element behavior recognition

Agents' information are achieved through observation, and in specific real-time, multi-agent environment, the incoming observations are consecutive or discrete multi-variable streams. However, the plan recognition is based on sequential behaviors. Hence the observations must be translated into agents' sequential element behaviors. Here element behavior means the minimal cooperative behaviors between agents. In other words, an agent's element behaviors are those minimal actions with coordinate meanings to other agents.

In the field of RoboCup simulation soccer games, the incoming information from observation are discrete temporal snapshots of the soccer field, and each snapshot includes the position, velocity, body heading, and head angle information of every player in the field, info of the ball, and the game state info at a particular moment. There is equal interval between each two snapshots. The mission of this stage is to turn the incoming information into players' element behavior sequences sorted by temporal order. The element behaviors in soccer games are pass, dribble, shoot, interception, and positioning and we use a structure to describe a single element behavior:

```

basic-behavior {

```

```

    type;
    leading-agent; associate-agent;
    source-region; goal-region;
}

```

The leading-agent denotes this behavior's executor, the associate-agent denotes this behavior's object, and the source-region and goal-region denote the region the ball locates when this behavior start and ends respectively. One hypothesis we make here is that a plan of rational and organized multi-agent team has one and only one leading action at a time slice. As for soccer, pass, dribble, shoot, and interception are leading actions, and positioning is associate action. To detect actions in the observations we define several events to activate our recognition function. They are Control-Ball(agent, t), which means agent start controlling ball at time t; With-Ball(agent, t), agent continue controlling ball at time t; Release-Ball(agent, t), agent release ball at time t; and Moved(agent, t1, t2), agent's position changes between t1 and t2. These events can be easily told from the multi-variable information stream. When Release-Ball(agent, t) occurs, agent's behavior at time slice t-1 may be pass, dribble or shoot. We utilize backfill technique, to delay the confirmation of one behavior's parameters until it finishes, which let the time complexity of the recognition be only $O(1)$, and make our approach also suit online analysis. Though the backfill technique is able to tell most of the behaviors, agents' actions in noisy and adversarial setting are possibly interrupted by other agents before they finish. For example, a player's shoot action may be break down by the opponent team's goalie, and the backfill technique can do nothing about this situation. To assure the integrality of behavior sequences, we employ the prediction results under environment that has no adversary and noise, to estimate a player's intention and the behavior's parameters at the beginning of the behavior with the aid of interception model. For instance, when the event of Release-Ball(agent, t) happens, the interception result of each teammate of the agent is calculated basing on the interception model. If there is any teammate can intercept the ball before it goes outside the field, then if the one who can intercept the ball first is agent then let this behavior's type be dribble else pass, else if the point the ball goes outside is close enough to goal then fill the behavior's type with shoot else this action is regarded failed. At the same time, let the region where ball locates when this behavior completes in prediction be the goal-region, and if the behavior's type equals pass, fill this action's goal-agent with the player who can intercept ball fastest in prediction. Then a 2-dimension vector of agents' behaviors is obtained, which is illustrated in the figure 1 below.

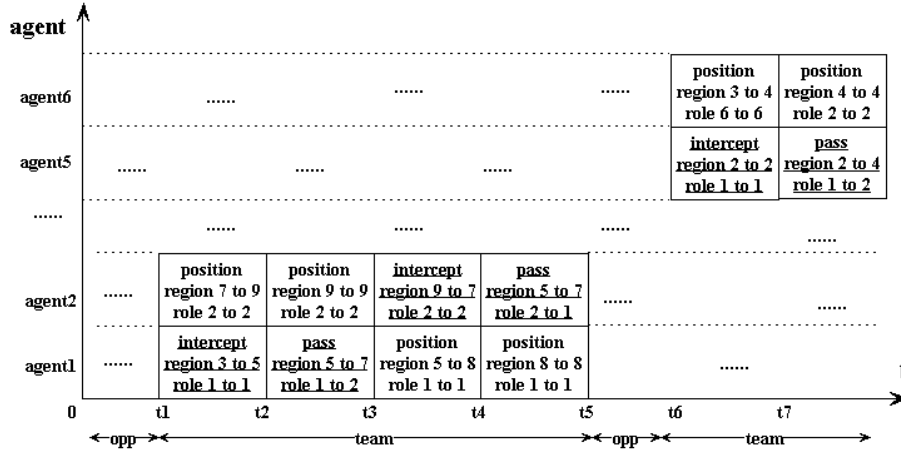


fig.1 Co-occur behavior sequence

The x-axis stands for time, and the y-axis stands for agent identities. Each intersected cell denotes one agent's behavior at a time slice, and those behaviors underlined are leading behaviors. As showed in the figure above, the multi-sequences from the moment the team observed starting control ball to losing ball is called a multi-sequences fragment. The multi-sequences from t1 to t5 and that from t6 are multi-sequences fragments. The following work will base on the outcome of this stage.

4. Plan recognition

At this stage we will analyze the sequences attained above and pick out those interesting and frequent behavior sequences as the team's plans on the assumption that the team's plan is embedded in those sequences. Two primary methods to discovery of significance sequence patterns have been discussed in literatures. One determines the significance of sequences and subsequence based on the support-confident framework [5]. The other uses statistical dependency testing instead [7]. We used statistical dependency test, because it may be more suitable to discovering coordinated sequential behavior, since it is able to filter frequently observed sequences whose constituent behaviors have co-occurred due to chance. Works by Gal A. Kaminka on comparing these two methods draw the same conclusion [3].

In order to select interesting sequences, we adapt a data structure, trie [2]. A trie is a multi-way tree structure useful for storing strings over an alphabet. It has been used to store large dictionaries of English words in spelling-checking programs and in natural-language understanding programs. We make each node except root represent a step, including leading action and associate actions, in a plan, and then a sequence from root to a node represents multi-agent behaviors sequence. Sequences have the same leading behavior, not necessary having the same associate behaviors, in each step are only stored once in the trie. An additional attribution of count is added to

Contingency Table

	n	not n	total
prefix	cell 11	cell 12	prefix margin
not prefix	cell 21	cell 22	not prefix margin
total	n margin	not n margin	all

The test for independence is carried out as following steps. Step 1: make the hypothesis that two variables are independent. Step 2: find the critical value from chi-square distribution table. Step 3: create the contingency table to derive the expected values. Here, n is the node for which this contingency table is constructed, p is the parent node of n, and the length of prefix equals l-1.

Cell 11 = count of n;

Prefix margin = count of p;

Cell 12 = prefix margin – cell 11;

N margin = end-length (n, l);

Cell 21= n margin – cell 11;

All = end-length (total, l);

Not n margin = all – n margin;

Not prefix margin = all – prefix margin;

Cell 22 = not n margin – cell 12;

E11 = (prefix margin)*(n margin)/ all;

E12 = (prefix margin)*(not n margin)/ all;

E21 = (not prefix margin)*(n margin)/ all;

E22 = (not prefix margin)*(not n margin)/ all;

And calculate test value from:

$$C^2 = \sum_{ij} \frac{(\text{Cell}_{ij} - E_{ij})^2}{E_{ij}}, \text{ Cell}_{ij} \text{ is the observed frequency, } E_{ij} \text{ is the expected frequency;}$$

Step 4: making decision, if chi-square > critical value then reject hypothesis, else cut this branch away from the node's parent. Then we trim the trie using DFS algorithm according to the compare result of a node's chi-square value and the critical chi-square value. The trie after pruning only contains those interesting behavior sequences, but some longer sequences will contain some other shorter sequence also in the trie, and we performed a further trimming to remove those shorter ones.

5. Plan retrieval

This stage's task is to convert those sequences stored in the trie after priming to the form of our formalized plan representation. For each leaf node, the path from the root to the leaf suggests a multi-agent's plan. The players participating in this plan would be picked out from all the leading behaviors' executors. The start condition would be set as the players' position info before the first step of the plan and each node on that

path will be orderly refined and turned into a step of the plan, deciding the associate players' behaviors. The goal state will be formed according to the ball's location and the leading player of the last step. We use a recursive algorithm which finds every leaf node and retrieve the plans from every path from the root to a particular leaf node, including filling the agent-list, start-condition, goal-state, and the concurrent leading action and associate behaviors in each step in a plan. While deciding the associate action of a relevant player in a single step, one key technique here is to unite the secondary parameters of associate behaviors, such as the positioning action's source-region because the secondary parameters are not critical while the plan is performed. And if there are still conflict associate behaviors in a step then select the one whose frequency count is maximal. This technique is elaborated in the following algorithm shown in figure 3 that is called with the parameter of root node.

algorithm Retrieve(NODE n)

if n is a leaf node

then

 create an empty plan p , and let $path$ represent the path from root to n ;

 add all executors of leading actions in $path$ to $pagent - list$;

 for each node x in path do

 add a step s to p ;

$sleading - act = xleading - act$;

 for each agent a other than $sleading - agent$ in $pagent - list$ do

 unite a 's associate acts and their $counts$ with the same $goal - region$;

 let aa be the associate act maximizing the $count$;

 insert aa into s ;

 distill region info of agents in $pagent - list$ to form $pstart - condition$;

 distill the ball's region in the last step to form $pgoal - state$;

else

 for all children c of n do Retrieve(c);

fig. 3 Retrieval Algorithm

6. Experiments and results

In order to evaluate our approach, we fully implemented a mechanism for recognizing and retrieving plans basing on observation in RoboCup simulation soccer game domain, and conducted a series of experiments.

First we tested veracity of the technique for element behavior recognition, which is also used by our online coach and monitor tools. On one hand, we developed our own simulation team and let every player in the team orderly record the actions it intended to execute, such as pass, dribble, shoot, etc. On the other hand, our program observed and recognized the agents' element actions and kept the recognition results in the other record. Then we compare these two records, and use the ratio of correct times to whole times, correct ratio, to evaluate our technique. In the experiments we got an average correct ratio of 96 percents, and this technique has been successfully applied in our monitor to display the players' behaviors.

Second, we tested the sequence filtering method, in which some parameters will have a strong impact on the experiment result, such as the definition of region and the selection of critical chi-square value. We defined the region in three different scales in the shape of rectangle and set the critical chi-square value with different values. After that plan set 1, consisting of several simple plans, were designed in our own simulation team and an inactive team was employed as the opponent team. Sample plans we designed are listed as follows.

Sample plan in plan set 1:

step1: pass (role 6 to 7, region 6 to 11)

step2: intercept (role 7, region 10 to 11) && position (role 10, region 12)

step3: pass (role 7 to 10, region 11 to 12)

step4: intercept (role 10, region 11 to 12) && position (role 9, region 8)

step5: pass (role 10 to 9, region 12 to 8)

For each pair of region scale and critical chi-square value, we extend the time of a soccer game to 4 times the default 6000 cycles, to make sure that our team would have chance to perform its plans, and let our team play the opponent team. Then our recognition and retrieval program analyzed our players' behaviors and output a plan set, set 2, in which the number of steps in each plan is greater or equal second. And then every item in set 2 was examined. If an item in set 2 is also an item or a subsequence of an item in set 1, it's marked with correct; else it's mark incorrect. With that the ratio of the correct ones to incorrect could be calculated. We repeated this process 4 times, and use the average correct ratio to evaluate the recognition and retrieval results. Figure 4 shows the test results at different region scale and critical value.

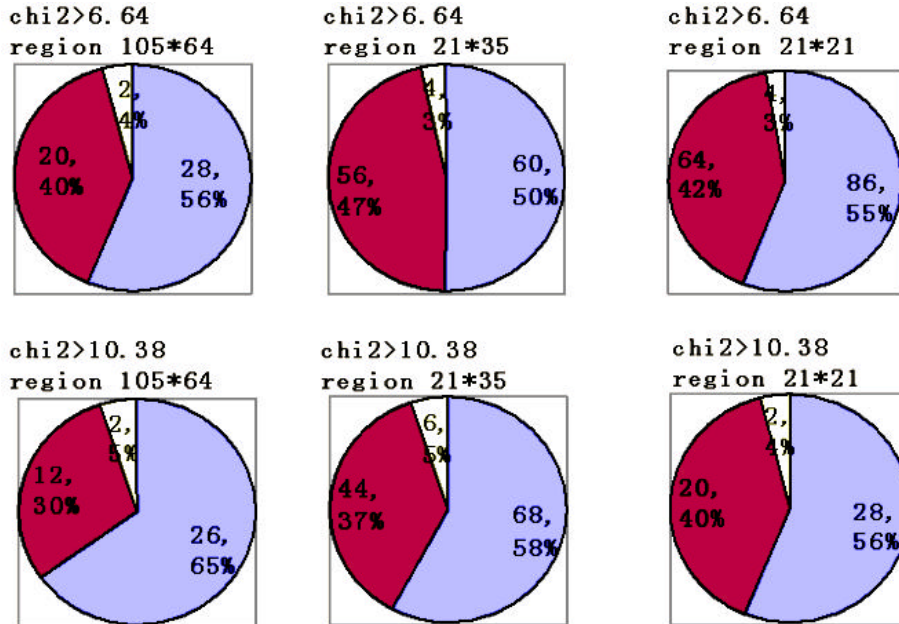


fig. 4 ■ Incorrect ■ correct □ hit

The result seemed not quite satisfying, but we soon found out that a few sequences in set 2 have only minor appearances. When we inspect these incorrect cases, we realized that our team's plan execution sometimes has redundant steps or omit several steps, which is the result of failure of players' element behaviors caused by noise or inexact information. However these rarely happen sequences imply the agents' comeback actions to their plans when plan execution failures occur, which is the true situation in the real world. And these sequences are consequential, for example, if player6 pass ball to player2 by mistake, the next behavior must be player2 pass ball to player6 that is player2 comes back to the plan from the failure of player6. Moreover, our team's plan execution is often interrupted by the opponent team that led to many plans in set 2 are fragments of the integrated plans in set 1.

The results of our experiment indicate that our approach is fruitful.

7. Conclusion and future work

In this paper we provided an approach to recognize and retrieve multi-agent teams' plans basing on observation, and completely realized an auto-learning mechanism in RoboCup simulation soccer game field. We presented a synchronous plan representation, adapted the trie structure, and gave an algorithm for recognizing and retrieving team plan in the given plan representation.

Some work has been done in the area of plan recognition through observation [3,6,8]. However, since our goal is to make agents learn team plans from other teams,

even human teams, through observation automatically, techniques in exist are difficult to be utilized in team plan retrieval because of depending on a predefined plan library or losing information of co-occur behaviors and behaviors' parameters. Perhaps the most similar work is by Gal A. Kaminka, but Gal's work ignored the co-occur behaviors and behaviors' parameters, which makes it infeasible to retrieve plan from the interesting sequences.

This work is the first step of our plan recognition and retrieval research. Now we are investigating methods to evaluate the plan achieved from observation, and trying to add variables into plan's representation, which will enhance the description ability of the formalized plan representation. Another direction of our future work is to substitute the single stream sequential behavior patterns discovery technique with multi-stream dependency detection (MSDD) [10] measures.

Reference:

- [1]. A. Osherenko. Plan Representation and Plan Execution in Multi-agent Systems for Robot control. In workshop of KI 2001 Joint German/Austrian Conference on Artificial Intelligence.
- [2]. D.E. Knuth, *Sorting and Searching*. Volume 3 of *The Art of Computer Programming*. Addison Wesley, 1973.
- [3]. G. A. Kaminka, M. Fidanboylu, A. Chang, and M. Veloso, Learning the Sequential Behavior of Teams from Observations. In *Proceedings of the 2002 RoboCup Symposium*
- [4]. H. A. Kautz, A Formal Theory of Plan Recognition and its Implementation. In *Reasoning About Plans*, by J.F. Allen, H.A. Kautz, R.N. Pelavin, and J.D. Tenenber, Morgan Kaufmann Publishers, San Mateo, CA, 1991, chapter 2:69-126.
- [5]. J. Han, M. Kanmber, *Data Mining Concepts and Techniques*, volume 6 of *Mining Association Rules in Large Databases*, Morgan Kaufmann, 2000.
- [6]. P. Riley and M. Veloso. Planning for distributed execution through use of probabilistic opponent models. In *IJCAI-2001 Workshop PRO-2: Planning under Uncertainty and Incomplete Information*, 2001.
- [7]. R. R. Sokal and F. J. Rohlf. *Biometry: The Principles and Practice of Statistics in Biological Research*. W.H. Freeman and Co., New York, second edition, 1981.
- [8]. R. Weida and D. Litman. Terminological Reasoning with Constraint Networks and an Application to Plan Recognition. In *Principles of Knowledge Representation and Reasoning: Proc. of the Third Intl. Conf.* 1992.
- [9]. S. Kambhampati and B. Srivastava, Universal Classical Planner: An algorithm for unifying State-space and Plan-space planning. In *Current Trends in AI Planning: EWSP '95*, IOS Press, 1995.
- [10]. T. Oates and P.R. Cohen. Searching for structure in multiple streams of data. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*. Morgan Kaufmann, 1996.