Intuitive Plan Construction and Adaptive Plan Selection

Kai Stoye and Carsten Elfers

Universität Bremen, Germany
{kstoye,celfers}@informatik.uni-bremen.de

Abstract. Typical tasks of multi agent systems are effective coordination of single agents and their cooperation. Especially in dynamic environments, like the RoboCup soccer domain, the uncertainty of an opponent's team behavior complicates coordinated team action. This paper presents a novel approach for intuitive multi agent plan construction and adaptive plan selection to attempt these tasks. We introduce a tool designed to represent plans like in tactical playbooks in human soccer which allows easy plan construction, editing and managing. Further we introduce a technique that provides adaptive plan selection in offensive situations by evaluating effectiveness of plans and their actions with statistically interpreted results to improve a team's style of play. Using experts as a concept for abstracting information about a team's interaction with another, makes fast accommodated plan selection possible. We briefly describe our software components, examine the performance of our implementation and give an example for rational plan selection in the RoboCup Small Size League.

1 Motivation and Related Work

A common hypothesis is, that the more a team's behavior is adapted to the opponent, the more effective it is. There are two major ways for adaption: One is to analyse the opponent and adapt the own team behavior to it (e.g. [2,8]). The other way, introduced in this work, is to analyse the own team effectiveness corresponding to the opponent and adapt team behavior thereby. The idea behind our approach is to enhance the B-Smart software[1], developed for the *RoboCup Small Size League*, which uses more or less only reactive behavior selection, with a deliberative component providing strategic moves and multi agent coordination. Deliberative principles become more and more one of the most challenging aspects within the focal point of artificial intelligence in RoboCup (e.g. see team descriptions to appear in [7]). This approach is restricted to offensive game situations and uses predefined stepwise action sequences, like tactical playbooks in human soccer (cf. [9]), further called plans. The approach is based on the work of Bowling et al. [3,4] who are using a play (multiagent plan)-based coordination and opponent-adaption for the CMDragons Small Size Team. However, our approach differs in the way of plan/play construction and adaptive selection. Our

[1] For further information see: http://www.b-smart.de/

U. Visser et al. (Eds.): RoboCup 2007, LNAI 5001, pp. 278–285, 2008.

implementation is divided into two parts. One separate component for creating and managing plans, the *Strategist*, and one for plan selection, execution and assessment, the *Coach*.

2 Plan Structure

In this section we briefly describe the structure of a plan. The highest level in our hierarchy is the *planbase*, which is a container for multiple *plans*. The planbase is used as a superior structure for classification and to provide a possibility for a (topical) subsumption of plans to improve clarity. Plans consist of *variants* and *plan steps*. A single step is defined as a tuple of *conditions* and *actions*. Conditions are restrictions which must be satisfied to enter the corresponding plan step. The variants are lists of conditions containing at least one element to describe the entry condition for a specific plan, s. Fig. 1. In order to allow different entry conditions for the same plan, different variants can be specified. Unlike all other steps, the first step only has a list of actions without any conditions, because these were already defined in the variant(s). These actions should be performed, if one of the variants matches with the actual game situation. Single conditions and actions, in contrast, must all be fulfilled. After the first plan step an arbitrary number of steps can follow. In our approach actions and conditions are predefined constructs. Conditions can be separated into different classes. *Assignment Conditions* are restrictions, which can change the world representation[2]. *Dependent Conditions* are restrictions, which are dependent on Assignment Conditions in order to determine their validity. The third class are *Independent Conditions*. In contrast to the other classes, they are independent to prior assignments and to the evaluation of other conditions. Also other constructs are needed for a complete plan step, the *actions*; they trigger agents' behaviors during plan execution. Typically a plan always ends with a goalshot, because it is the main aim in offensive situations to score for your team. Possibly it can be imagined that there also exists plans for gaining space or other intentions.

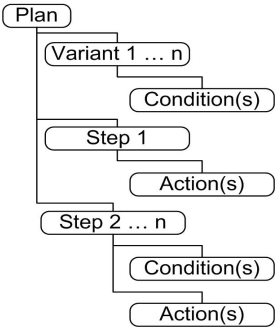

Fig. 1. Elementary plan structure

3 Plan Construction

In this section the software component for constructing plans will be briefly introduced, the *Strategist*. It is a stand-alone program designed for: Creation of plans based on the already mentioned plan elements, editing plans and managing plans and the associated planbases. One design principle for the Strategist is to

[2] World representation means a representation of the "world", which holds beside the world model information about internally used assignments.

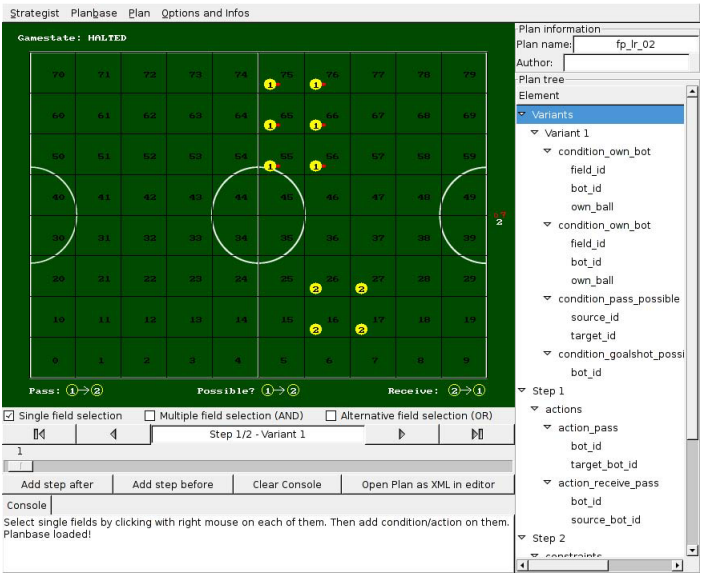

Fig. 2. B-Smart Strategist

provide an easy and intuitive useable interface for these tasks. Therefore, it uses an abstract visualization of the playing field and the plan elements, where these elements can be added or removed simply by using the mouse. This approach was inspired by, as it is common in some "real" sports, blackboard applications for explaining tactical moves. An impression of the software layout is illustrated in Fig. 2. Inspired by this, plans can be designed by human users and must not be generated automatically, like in other approaches, e.g. systems using a planner to generate possible strategies (e.g. [6]). We decided to use a qualitative world model with grid rastering. That means, the playing field is divided into equal rectangular fields, in our implementation $8 \cdot 10$ regions, where every field is numbered consecutively, so that it is explicit. This abstraction level seems to be a good trade-off between search complexity and accuracy. This concept of world representation is inspired by Schiffler et al. [11].

4 Plan Selection and Execution

4.1 Identification of Applicable Plans

An important step towards the execution of a plan is the determination of the subset of plans having a consistent variant regarding the current world model. An effective way to find this subset is to refine the search space by the position of the ball carrier. As previously described a wholly offensive plan execution has been implemented. An important property that this assumption provides, is that every variant has exactly one condition, specifying the ball carrier's position.

This feature is used for a fast refinement of the search space of plans. While loading the planbase, variants will be checked by the position of the ball carrier in the variants. This positional information is used as a key to address a list of possibly applicable plans. Doing this, the set of plans will be reduced by the plans with a ball carrier on an inconsistent position considering the current world model. In other words, we define a subset of plans, the set of possibly applicable plans with all plans containing a consistent position of the ball carrier to the current world model that can be obtained by a simple lookup.

$$Plans_{applicable} \subseteq \text{Plans}_{possiblyApplicable} \subseteq \text{Plans} \tag{1}$$

As previously described, some conditions depend on others. This makes it impossible to prove the consistency of a condition set in a commutative way. However, within one Condition Class, the consistency check is commutative. Assignment Conditions do not depend on any other conditions, Dependent Conditions only depend on Assignment Conditions and Independent Conditions do not depend on any other conditions either. To prove a set of conditions of different condition classes correctly by consistency, methods to prove commutative sets with constraints (like CSPs, s. [10, ch. 5]) are not appropriate. It has to be ensured that all Assignment Conditions are proven before Dependent Conditions. To do so, all condition sets are ordered while loading the planbase by their condition class. The order of conditions allows the reduction of the branching factor for the consistency check to prove the condition set of satisfiability from $n! \cdot d^n$ to d^n (cf. [10, ch. 5]), with n conditions and d possible assignments. In order to prove a given set of conditions by consistency a modified backtracking algorithm is used to check the constraints between Dependent Conditions and Assignment Conditions. This algorithm differs from regular backtracking (cf. [10, ch. 5]) by using the given order of conditions to check consistency, assign values and solve conflicts. For every possible assignment to a condition a new node will be created and recursively checked until a consistent assignment will be found or all paths have been visited. The need for backtracking to check Assignment Conditions is exemplified in Fig. 3 with two players, player one on position 1 and 2 and player two only on position 2 on the left side of the figure. On the right side the resulting tree for the backtracking is illustrated. At first position 1 will be assigned to player one, but this assignment leads to a conflict, because no position could be assigned to player two. The conflict is solved by assigning position 1 to player two and assigning position 2 to player one.

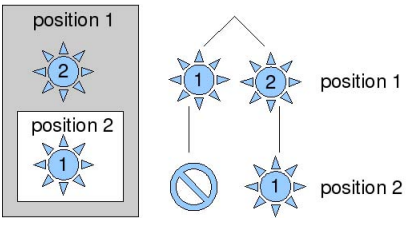

Fig. 3. Example for backtracking

To reduce the set of possibly applicable plans to the set of applicable plans, all variants of all possibly applicable plans are checked by the modified backtracking algorithm of consistency. Only plans with at least one consistent variant are in the set of applicable plans $Plans_{applicable}$.

4.2 Plan Assessment

Since this approach only allows one plan to be executed at a time, one plan has to be selected from the set of applicable plans. Assessing applicable plans is the first step that permits a reasonable decision. In this approach the assessment of applicable plans will be accomplished by experts. Experts are optional modules, which gather relevant information during plan execution to assess plans or aspects of plans. Most experts are able to abstract from the gathered information to evaluate similar plans. Four basic experts have been implemented, the Dribble Expert, the Goalshot Expert, the Standard Expert and the Pass Expert. Each expert evaluates one aspect of each plan[3], except the Standard Expert which evaluates the wholly plan without regarding one special aspect of the plan additionally to the individual experts. Each expert holds a table with the amount of successful executions and failures of plans or an aspect of plans.

If an expert is called to assess a given plan, it has to return a probability that contains the expert's assumption about the observed actions in the plan being successfully executed. In order to interpret a given set of samples statistically, experts need a function, which relates the given samples to a probability. The Beta distribution provides useful properties to get these probabilities to assess plans, like the mode and the probability density defined by two hyperparameters specifying the shape of the distribution (s. [10, ch. 19]). In this case the two free variables/hyperparameters are specified by the number of successes and the number of failures defining the shape of the distribution. With these two variables the distribution is fully defined and we remember that these two variables are gathered by the experts. For a low amount of samples the Beta distribution mode could result in extreme probabilities like 0 or 1. This behavior is not preferable, because it leads to excluding plans and avoids reinvestigating plans that have been failed in the beginning. Therefore we define the Beta 2-2 distribution similar to the Beta distribution by adding a fair sample, 2 successes and 2 failures. This avoids interpreting a low amount of samples inadequate funded, but still converges with a higher amount of samples to the expected value. Another feature is that the mode can be defined for unavailable a-priori evidences easily by the amount of successes $succ$ and the amount of failures $fail$ as $mode_{beta22[succ,fail]} = \frac{succ+1}{succ+fail+2}$ (cf. [1]).

In order to provide an assessment of successful executions more or less decisive than unsuccessful executions, the number of successes and failures can be factorized (cf. [3]). This allows the configuration of more optimistic or more pessimistic evaluations. The overall assessment for an expert of a given plan is defined by the product of the single assessments of each action examined by the expert, because each action is viewed as an independent event. The overall evaluation of all experts is computed in the same way, by multiplying all experts' evaluations, giving the probability about all aspects of the plan will be successful.

[3] E.g. the Dribble Expert evaluates dribbling actions.

4.3 Plan Selection and Execution

In order to decide which plan should be executed if more than one plan is applicable, the evaluation of each applicable plan will be normalized to 1. The normalization gives a probability distribution for the plan selection (cf. [4]), used to decide on a plan randomly, with a higher probability of choosing a high assessed plan. With a lower probability, weaker plans will be executed, which allows a reassessment of these plans. This is a helpful method to avoid an opponent predicting the plan our system is selecting and to reassess the effectiveness of our team's plans in case of the opponent team is changing their behavior or the bad luck the own team had during these plan's executions.

After selecting a plan, it will be executed. Execution means to perform the action sequences defined by the plan in the order of the steps. A transition from one step to the following is accomplished, if all conditions of the following step that are related to the ball carrier, are satisfied. If the following step contains unsatisfied conditions, but the conditions related to the ball carrier are satisfied, the plan execution will be aborted. This prevents the ball carrier from waiting for other involved players to perform their actions completely and avoids impeding the flow of the game. If a plan execution is aborted or a plan/step has been completely performed, the experts will be notified and extract relevant information. If a goal opportunity exists the plan will be stopped and the chance to score will be used. The selection, adaption and (high-level) plan execution is implemented as an optional module in the B-Smart Agent Software, called Coach. The low-level execution of plans' actions is realized by the Agent Software.

5 Preliminary Evaluation

In this section we consider performance measurements of plan selection and examine a sequence of plan executions and their resulting assessments. First we start with the performance measurements. Therefore, a series of 100 plan selections for each test with up to 100 practical plans have been performed on an AMD Athlon XP 2000+ with 1666 MHz and 1280 Mb RAM (s. Fig. 4). Each test differs in the number of loaded plans, listed in the first column. The second column, *appl. plans*, lists the amount of applicable plans found by one selection request. The last column, *required time*, shows the time used for one plan selection phase. Clearly the required time depends on more factors than the amount of plans and the amount of applicable plans found, but the given criteria are of capital importance. To examine an example of a sequence of plan executions with the resulting assessments, we partly specify two plans by their executed actions in Fig. 5 and the execution sequence with the resulting plan assessments in Fig. 6

plans	appl. plans			required time		
	min	max	Ø	min	max	Ø
10	1	1	1	0 ms	6 ms	1.2 ms
20	1	9	3.49	0 ms	19 ms	6.26 ms
100	1	22	4.3	1 ms	22 ms	9.52 ms

Fig. 4. Performance test for plan selection

delivered by the Beta 2-2 mode. We assume that successes and failures are equally factorized by 1. Further, we assume that the three specified plans have the same variants, so they will be applicable at the same game situations.

First have a look at the a-priori assessments in column one in Fig. 6. It can be seen that plan B will be executed at 67%, this seems to be rational by the assumption that short plans have a higher success-rate than longer plans. We assume that plan B will be chosen and aborted in the first step, while performing a short pass. We notice that thereby plan A gets a higher assessment, but is assumed to be still less effective than plan B. This is explained by the aborted pass which exists in plan A too. Now we assume that plan A is chosen and that this plan will be executed successfully. Now we see that we cannot assume that a pass is more or less effective, because one time it was successful and the other time not. But we know that plan B failed and plan A was successful, so it is rational that our concept assumes a higher success probability for plan A than for plan B and will next choose plan A by 58%. It can be noticed, that the more promising plans become higher assessed at an accommodate speed and in a rational way. As we see, an essential feature this approach provides is to assess plans appropriately even before they have been executed the first time by using abstracted evaluations from experts, in our example plan A gets an higher assessment even before plan A have been tried the first time.

plan	step 1	step 2	step 3
A	*dribble*	*pass*	*goalshot*
B	*pass*	*goalshot*	

Fig. 5. Example plans

execution	plan	evl_{pass}	$evl_{dribble}$	$evl_{goalshot}$	$evl_{standard}$	evl_{all}	norm
a-priori values	A	0.5	0.5	0.5	0.5	0.06	0.33
	B	0.5	1	0.5	0.5	0.063	0.67
plan B failed	A	0.33	0.5	0.5	0.5	0.04	0.43
while passing	B	0.33	1	0.5	0.33	0.05	0.57
plan A was	A	0.5	0.67	0.67	0.67	0.15	0.58
successful	B	0.5	1	0.67	0.33	0.11	0.42

Fig. 6. Sequence of plan executions with assessments

6 Conclusions and Future Work

As shown in the evaluation, we implemented a fast and rational approach for adaptive plan selection for offensive soccer situations. Plans and actions are rated by their successes what indirectly models weakness of opponents' play and strengths of the own team abilities. This model is used to adapt the own team's play and allows to exploit the opponent's flaw. The fast adaption further allows to be prepared for unknown opponents. The intuitive plan construction software allows easy plan creation and provides useful managing features. All in all the whole system is a useful extension of the B-Smart team software and extends the

current system by a deliberative concept. However, this system is not limited to the Small Size League and could be integrated in other leagues as well.

For future work more properties of the beta distribution can be used, e.g. information about the certainty of the experts' assessments could be useful for selecting plans. Another rational feature would be to evaluate positions on the field depending on the current situation, e.g. this could be done by using potential fields like Vail and Veloso [12] did, or by Voronoi diagrams used by Dylla et al. [5]. Currently, each player tries to reach the middle of the target region defined in the actions, without considering the current game situation. Potential fields or Voronoi diagrams could help to choose a proper target for these actions. We introduced the concept of experts assessing plans and their actions. This is currently done in a rudimentary way and needs to be improved. Another useful feature could be to use previously recorded assessments by the experts to support the user while creating plans in the Strategist by estimating the success-probability of the edited plans and actions.

References

1. Balakrishnan, N., Galloway, V.B., Nevzorov, V.B.: A Primer on Statistical Distributions. Wiley-IEE (2003)
2. Ball, D., Wyeth, G.: Classifying an Opponent's Behaviour in Robot Soccer. In: Proceedings of the 2003 Australasian Conference on Robotics and Automation (ACRA) (2003)
3. Bowling, M., Browning, B., Veloso, M.: Plays as effective multiagent plans enabling opponent-adaptive play selection. In: Proceedings of International Conference on Automated Planning and Scheduling (ICAPS 2004) (2004)
4. Bowling, M., Browning, B., Chang, A., Veloso, M.: Plays as Team Plans for Coordination and Adaptation. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 686–693. Springer, Heidelberg (2004)
5. Dylla, F., Ferrein, A., Lakemeyer, G., Murray, J., Obst, O., Röfer, T., Stolzenburg, F., Visser, U., Wagner, T.: Towards a league-independent qualitative soccer theory for RoboCup. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 611–618. Springer, Heidelberg (2005)
6. Fraser, G., Wotawa, F.: Plan Description and Execution with Invariants. OGAI Journal 22(4), 2–7 (2003)
7. Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T.: RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434. Springer, Heidelberg (2007)
8. Lattner, A.D., Miene, A., Visser, U., Herzog, O.: Sequential Pattern Mining for Situation and Behavior Prediction in Simulated Robotic Soccer. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020. Springer, Heidelberg (2006)
9. Lucchesi, M.: Coaching the 3-4-1-2 and 4-2-3-1. Reedswain Publishing (2001)
10. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall (2002)
11. Schiffer, S., Ferrein, A., Lakemeyer, G.: Qualitative World Models for Soccer Robots. In: Conference Paper - KI 2006 WS on Qualitative Contraint Calculi. Knowledge-based Systems Group, RWTH Aachen (2006)
12. Vail, D., Veloso, M.: Multi-Robot Dynamic Role Assignment and Coordination Through Shared Potential Fields. In: Multi-Robot System, pp. 87–98. Kluwer (2003)