Simultaneous Planning: A Real-Time Planning Method

Ramon F. Brena, Emmanuel Martinez

Center for Intelligent Systems
Monterrey Institute of Technology
ramon.brena@itesm.mx, A00588294@itesm.mx

Abstract. We present a novel real-time planning method as we have applied it to the Robocup competition. The basic idea in this method is to simultaneously keep in consideration more than one plan by evaluating and maximizing an expected utility and feasibility of the current and the future possible actions. This could be better than maximizing the expected utility of just one plan or the current possible actions, because in a highly dynamic environment like Robocup, plans get frequently stuck due to unexpected events, like an unforseen blocking, etc. In these cases, a "second-best" plan could rescue the situation. We propose a specific architecture for implementing this method in the context of a player agent in the Robocup competition. In this paper, after motivating and presenting the method, we propose a specific architecture for implementing this method in the context of a player agent in the Robocup competition and show how we have applied it to the "Borregos" Robocup team, that participated at the Robocup 2004 tournament at Lisbon. Some preliminary figures about performance and experimental evidence showing the potential of our method are presented as well.

1 Introduction

Real-time soccer is a complex game, involving a great deal of coordination between team members, and the development of strategies leading to marking as many goals as possible and receiving as few goals as possible.

Robocup simulated soccer represents that complexity to a certain degree, meaning that many interesting soccer complexities are present in the simulated soccer as provided by the soccerserver system [3], like limited vision and hearing, noise, etc.

In simulated Robocup each player is a decision-maker under uncertainty, because its vision is limited to a "cone", so the part of the field not seen is basically unknown, and players try to overcome this limitation by using their memory, extrapolating the previous situation, using communication to exploit information available to their partners, etc., but a great amount of uncertainty remains anyway.

Thus, simulated soccer is a challenging problem from the standpoint of coordination, communication, uncertainty handling, learning, planning, and so on [8].

© A. Gelbukh, H. Calvo (Eds.)
Avances en la ciencia de la computación en México
Research on Computing Science 13, 2005, pp. 3-11

Many Robocup teams have applied techniques issued from the Artificial Intelligence (AI) field [14] to develop sophisticated skills. Among those AI-related techniques we find Neural Nets, Reinforcement Learning and Probabilistic Reasoning for low-level skills [15], and decision trees [16], Reinforcement Learning [17] and Multiagent Coordination methods like Coordination Graphs [11], for high-level skills.

One area that perhaps has not reached a high degree of development in Robocup is planning [1]. Due to the very dynamic nature of simulated soccer, long-term planning is pointless, as almost any long plan will fail when facing unexpected conditions. Thus, most teams are highly reactive and rely more an very polished low-level skills like ball interception and shooting than on clever playing ideas. Very few teams have applied a planning method to Robocup; in [13], a planning method using opponent modelling is applied for the Coach league team from Carnegie Mellon University. Using bayesian networks, the coach models the opponent's behavior and generates a plan. Finally, the coach communicates the plan to the players. In [5] players have different plans in their memory and they search for one plan to arrive to the opponent goal; their plans consist of dribbles and passes. The cited works use traditional planning methods where the starting point and the ending point of the plan are defined before its execution starts.

In most planning methods [18], when a plan is interrupted at execution time due to unexpected events, players have to start over from scratch a new plan -which will be most probably interrupted as well.

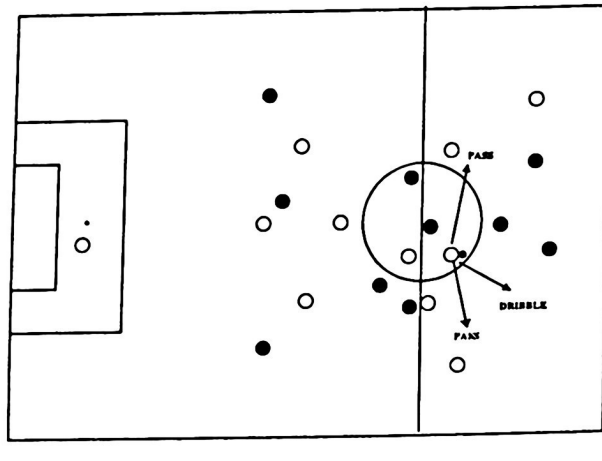

Fig. 1. Example of the problem of decision making in Robocup

Let us examine an example. Fig. 1 shows a game situation where the agent with the ball must take a decision between giving the ball to a well-positioned, but lonely partner, or filtering a long pass to the right, or even to continue dribbling, as indicated by the arrows in the figure.

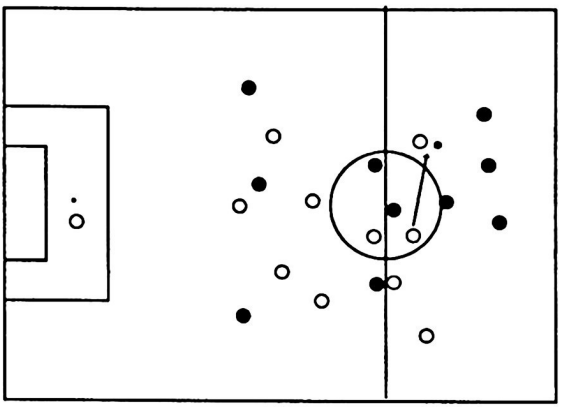

Fig. 2. Wasted action

Fig.2 shows that the after taking the decision of passing to the left, the lonely player gets stuck by enemies. Of course things could go bad as well if another action, like dribbling, is chosen, but the difference is that if the player continues dribbling, it still has the other options. The point here is not that dribbling is better that passing, but that under uncertainty is good to have options for cases where what seemed to be the best option just fails and has to be dropped. In our method we stress the importance of having as many *options* as possible.

The basic idea in our planning method is to take into consideration, when evaluating a particular possible move, how many and how good *options* the action keeps open after executing it. The advantage of doing this, instead of just selecting the best possible plan or action at a given moment according to some metric, is that in a highly dynamic environment like Robocup, plans often get stuck due to unexpected events, so the agent has to replan from scratch. In our system, for the contrary, when a plan gets stuck, it is more probable that there is another remaining option—precisely because the planning method has been fostering good options all the time.

Our planning method belongs to the class of on-line planning [7]. On-line planning interleave planning and execution whereas in traditional planning a plan is made and then executed. Agent-centered search [9] is a technique that implements on-line planning, restricting planning to the part of the domain where the current state of the agent is found. Agent-centered search decide on the local search space, search an action, and execute it. This process is repeated until the goal state is reached.

In the next section we will detail our planning method In Section 3 we will describe the Robocup application of our method in our "Borregos" team, and finally in Section 4 we will discuss pros and cons of our planning method, list some results of the team, and draw some future work lines.

2 Method Description

Our planning method considers a collection of current possible actions and the next cycle future actions.[1] The collection of current possible actions (CPA) is taken from a "playbook"[2]. CPAs are specified at the *role* level, involving abstract roles like the "passer", the "receiver", and so on. But to be evaluated, specific players are bound to the current possible actions roles present in the current situation, so CPAs are *instantiated* to fit the specific players and positions.

The playbook have the following plays: Pass, Filtered Pass, Dribble, Outplay (a very long dribble), Clearball (a kick to certain position), and Shoot to goal. We developed some classes to manage the actions but at the end, the actions are parsed to high level actions of the UvA Trilearn code.

Now we describe the basic steps involved in our system's operation:

- First we construct a two-level *play tree*, which is a subtree of a search tree, prunning off some not plausible plays, based on heuristics, like avoiding passes to players not in the passer neighborhood. In particular we check the compatibility between the plays in the first and second levels. These future actions options have to be "compatible" in the sense that the future actions can be performed after doing a current possible action. [2]
- The best evaluated current possible action is chosen from the CPA with its list of future actions. This seems entirely like conventional planning, but here the trick is that we evaluate first-level plays not entirely based on the maximum-value child, but mainly based on the accumulated value of its children, giving thus higher evaluations to plays followed by many options. The exact formula for play evaluation is presented below.
- The selected first-level action is refined through an *optimization* process. Here several variations of the action are generated, for instance changing slightly the direction, speed, etc., in order to choose the exact point yielding a maximum utility with respect to its list of future actions. This is the final decision for the agent.

Our evaluation of possible plays is based on their *expected utilities*, i.e. the product of their benefit (in case they are successful) by their probability of success; this is why we call it *combined evaluation*:

$$e(a) = bf(a) * pf(a) \tag{1}$$

where bf is the *evaluation of benefit* function (basically a heuristic taking into account the position of ball and players, see [12]), and pf the *feasibility* function,

[1] We decided not to consider more than one future cycle, so our planning trees are just two levels deep.

[2] For instance, after a "clear ball" you cannot make a pass, so these moves are not compatible.

which returns a number between 0 and 1. The feasibility function is supposed to correspond to the fraction of times a given play could be successful in the given situation.

The formula used to calculate the combined evaluation $E(a)$ of a first-level action a is as follows:

$$E(a) = k_1 e(a) + k_2 \max\{e(s_i(a))\} + k_3 \sum\{e(s_i(a))\} \tag{2}$$

where $s_i(a)$ the i-th successor of action a in the tree, $e(a)$ is the combined evaluation of current possible action a with respect to its feasibility and utility, $e(s_i(a))$ are the evaluations with respect to the feasibility and utility of the future actions $s_i(a)$ of the current possible action a.

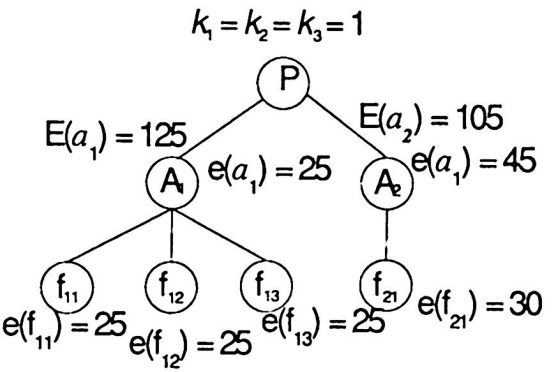

Fig. 3. A simple two-level tree

We illustrate the application of this formula to a simple tree presented in Figure 3. In conventional maximum-driven tree evaluation, the branch to the right would be selected, because this one gives a maximum value. But applying our formula, the evaluations are different. According to our formula the evaluation of node A_1 is:

$$E(A_1) = e(A_1) + \max\{e(f_{11}), e(f_{12}), e(f_{13})\} + \sum\{e(f_{11}), e(f_{12}), e(f_{13})\}$$
$$E(A_1) = 25 + \max\{25, 25, 25\} + \sum\{25, 25, 25\}$$
$$E(A_1) = 25 + 25 + 75 = 125$$

Whereas the evaluation of the node A_2 is:

$$E(A_2 = e(A_2) + \max\{e(f_{21})\} + \sum\{e(f_{21}\}$$

$$E(A_2) = 45 + \max\{30\} + \sum\{30\}$$
$$E(A_2) = 45 + 30 + 30 = 105$$

So, applying our method, we select the branch to the left, where the action A_1 is chosen to be optimized and performed.

3 Implementation

The Borregos team is based in the UvA Trilearn 2003 source code [4]. We have implemented some specific skills, like goal-shooting, considering some ideas presented in [10].

In the current prototype, our planning method is applied to the agent with possession of the ball; teammates just apply reactive heuristics aiming to help the player with the ball, like to stay far from opponents while attacking, etc. Of course, in principle the planning method could be applied to every single player, and most probably we will do it in future versions (see section 5).

To implement our strategy, we developed data structures for CPAs and instantiated data, then we implemented evaluation functions (both for benefit heuristic evaluation and for feasibility evaluation), action optimization, and parsing functions.

Our evaluation and feasibility functions make geometrical calculations and consider variables like traveled distance, opponent goal proximity, opponents density, teammates proximity, etc., that are reported elsewhere [12]. The feasibility function has been gradually adjusted in such a way that it corresponds to the average success rate of plays; this could be considered as a form of learning.

Another component links our decision mechanisms to a soccerserver command through the use of the UvA Trilearn code.

Figure 4 illustrates the way the planning method is implemented. First, according to the current situation of the game, the system checks the current possible actions that can be executed and make the instantiation, listing the CPA and the future actions of each member of the CPA. Next, current possible actions and its future actions are evaluated with respect to its feasibility and utility. After that, the best current possible action is optimized and finally executed. In the next cycle, when an agent has the ball, the process start again.

4 Experiments

We have been validating our strategy by playing games against our team disabling the use of future plays consideration. With the use of the proxyserver [6] we generated statistics to compare the performance of the team with the simultaneous tactics.

As a preliminary series of experiments, we ran several games between our team with the planning method ("Planning") and the same team without the planning method, and running a maximum-value method that chooses the action

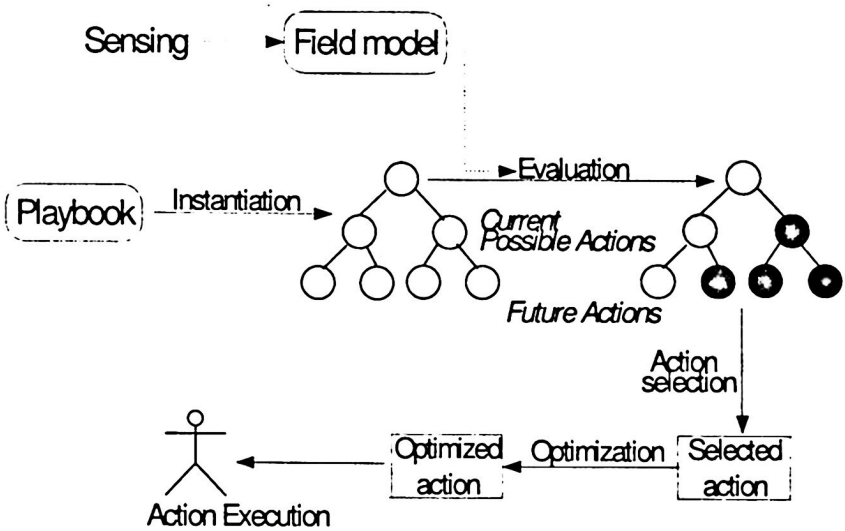

Fig. 4. Robocup prototype architecture

with the greater evaluation ("Nemesis"). Accumulated results are presented in table 1. The statistical parameters are the following:

- Total goals scored (Goal Scored)
- Average territorrial advantage - ball in opposition half (territ)
- Average pass success rate (psucc)

As we can see in the table, the result was a slight, but clear advantage of the "Planning" team, both in terms of goals scored (26 scored, 22 received) and in terms of other statistical mesures, like territorial dominance.

Table 1. Statistical parameters

Team	Goal Scored	territ	psucc
Planning	26	50.815	74.583
Nemesis	22	49.183	57.225

As we can see in Table 1, the goals scored by "Planning" team was clearly greater than the received ones (scored by "Nemesis"). Other parameters show a very slight advantage in territorial dominance, and also a great advantage in passing success rate (that is, number of passes that succeeded divided by the total number of passes). This last parameter can explain the global better behavior of the "Planning" team with respect to the reference "Nemesis" team.

We consider these as preliminary experiments, and currently we continue to refine the method implementation, in particular the evaluation functions, that are so critical to the planning method. More extensive experimentation is of course needed. But we consider an initial point is made about the possibilities of our planning method.

5 Conclusions and Future Work

We have presented a novel real-time planning method for highly dynamic domains with uncertainty, where possible actions evaluation is based not on the maximum in the search tree, but on a metric considering the quantity and quality of the options left available by the actions being considered.

An implementation of our method in the domain of the Robocup competition has been presented, which is at the heart of our "Borregos" team. Preliminary results, in the context of the SoccerServer Simulation league, are presented.

One critical aspect of our method is its high computational cost, because we need to perform many evaluations and then one optimization –involving even more evaluations. We relied on heuristics to reduce the search space for making the complexity manageable.

We have shown that our planning method has had promising results. We still have to work in the tunning of the evaluation, utility, and feasibility functions to increase the number of shoots to goal plays and the opportunities to score a goal.

Current research is focused on carefully evaluating our method's performance, for rigurous comparison agains other planning methods, especially maximum-value based. Also, in the future, we want to apply machine learning techniques for parameters learning.

Acknowledgements. This work was supported by the Monterrey Tech's Research Grant CAT011.

References

1. James Allen, James Hendler, and Austin Tate. *Readings in Planning.* Representation and Reasoning Series. Morgan Kaufmann, San Mateo, California, 1990.
2. Michael Bowling, Brett Browning, and Manuela Veloso. Plays as effective multi-agent plans enabling opponent-adaptive play selection. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS'04)*, 2004.
3. Mao Chen, Klaus Dorer, Ehsan Foroughi, Fredrik Heintz, ZhanXiang Huang, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Jan Murray, Itsuki Noda, Oliver Obst, Patrick Riley, Timo Steffens, Yi Wang, and Xiang Yin. Users manual: Robocup soccer server (for soccerserver version 7.07 and later).
4. R. de Boer and J. Kok. The incremental development of a synthetic multi-agent system: The uva trilearn, 2001.
5. Ahmad Farahany, Mostafa Rokooey, Mohammad Salehe, and Meisam Vosoughpour. Mersad 2004 team description, 2004.

6. Ian Frank, Kumiko Tanaka-Ishii, Katsuto Arai, , and Hitoshi Matsubara. The statistics proxy server. In Peter Stone, Tucker Balch, and Gerhard Kraetszchmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, pages 303–308. Springer Verlag, Berlin, 2001.
7. Lise Getoor, Greger Ottosson, Markus Fromherz, and Bjoern Carlson. Effective redundant constraints for online scheduling. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, pages 302–307, Providence, Rhode Island, July 1997. AAAI Press / MIT Press.
8. Frans Groen, Matthijs Spaan, and Nikos Vlassis. Robot soccer: Game or science.
9. Sven Koenig. Agent-centered search. *AI Magazine*, 22(4):109–131, 2002.
10. J. Kok, R. de Boer, and N. Vlassis. Towards an optimal scoring policy for simulated soccer agents, 2001.
11. J. Kok, M. Spaan, and N. Vlassis. Multi-robot decision making using coordination graphs.
12. Emmanuel Martinez. A real-time planning method for Robocup (in spanish). Master's thesis, Tecnologico de Monterrey, Mexico, 2005.
13. Patrick Riley and Manuela Veloso. Planning for distributed execution through use of probabilistic opponent models. In *IJCAI-2001 Workshop PRO-2: Planning under Uncertainty and Incomplete Information*, 2001.
14. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.
15. Peter Stone. Layered learning in multiagent systems. In *AAAI/IAAI*, page 819, 1997.
16. Peter Stone and Manuela Veloso. Using decision tree confidence factors for multiagent control. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 86–91, New York, 9–13, 1998. ACM Press.
17. Peter Stone and Manuela M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.
18. M. Zweben and M. S. Fox. *Intelligent Scheduling*. Morgan Kaufmann, 1994.