Decision-Theoretic Planning for Playing Table Soccer

Moritz Tacke, Thilo Weigel and Bernhard Nebel
Institut für Informatik
Universität Freiburg
79110 Freiburg, Germany
***take,weigel,nebel*@informatik.uni-freiburg.de**

Abstract. Table soccer (also called “foosball”) is much simpler than real soccer. Nevertheless, one faces the same challenges as in all other robotics domains. Sensors are noisy, actions must be selected under time pressure and the execution of actions is often less than perfect. One approach to solve the action selection problem in such a context is decision-theoretic planning, i.e., identifying the action that gives the maximum expected utility. In this paper we present a decision-theoretic planning system suited for controlling the behavior of a table soccer robot. The system employs forward-simulation for estimating the expected utility of alternative action sequences. As demonstrated in experiments, this system outperforms a purely reactive approach in simulation. However, this superiority of the approach did not extend to the the real soccer table.

1 Introduction

Playing *table soccer* (also called “foosball”) is a task that is much simpler than playing real soccer. Nevertheless, one is faced with all the challenges one usually has to deal with in robotics. One has to interpret sensor signals and to select actions based on this interpretation. All this has to be done while keeping in mind that the sensor signals are noisy and the actuators are less than perfect.

One approach to solve the *action selection* problem is to use purely *reactive* methods. These are methods that select actions based on the current sensor input with only a minimum amount of computation. The selection of actions can be based on layered finite state automatons [2] or even simpler by using a simple decision tree. However, these purely reactive approaches have the disadvantage that they cannot anticipate changes in the environment caused by its own actions or by exogenous actions and for this reason might act sub-optimally.

Approaches such *behaviour networks* [3, 6] address this problem by modeling all possible actions, their consequences, and something similar to success likelihood. Based on this model, actions that promises to achieve the goals best are selected. As demonstrated by different robotic soccer teams [4, 7], this approach can be quite successful. However, it lacks theoretical foundation and, in fact, it is not clear under what circumstances the approach provably achieves its goals.

Decision-theoretic planning in contrast addresses the action selection problem by explicit deliberation about possible actions and aims at generating plans which promise to yield the *maximum expected utility* for an agent. This is achieved by explicitly considering the uncertain effects of the actions, the incomplete knowledge about the world and the possibly limited resources for carrying out a plan.

Decision theoretic planning can be implemented in various fashions. Using a classical refinement planner, it is possible to calculate the plan with the maximum expected utility by keeping ranges of possible utility values for partial plans [5]. A very popular way to realize decision-theoretic planning is the modeling of the planning problem as a Markov decision process [1].

The main challenge in using such an approach is to simplify the model of the domain such that the computational costs are not prohibitive. For this reason, we do not consider e.g. all possible ways an action can fail, but distinguish only between successful execution and failure. Furthermore, we do not consider responses to successful ball interceptions but consider the plan as failed once the opponent has intercepted the ball. Finally, planning is carried out only to a limited depth and the utility of the resulting state is assessed using a heuristic measure.

The rest of the paper is structured as follows. In Section 2, the KiRo system is presented. Section 3 describes the implementation of a decision theoretic planning algorithm for KiRo. Experimental results are presented in Section 4 and a short conclusion and outlook is given in Section 5.

2 KiRo

KiRo is a table soccer robot, i.e., an automated table soccer table [8]. Its hardware consists of the following components (see Figure 1):

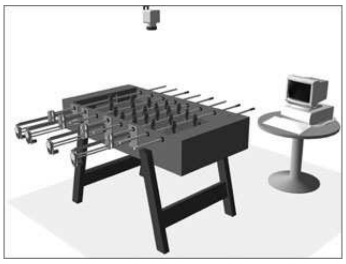
Figure 1. The hardware setup

- a standard table soccer table, where all rods of one player are equipped with electro motors strong enough to shift and turn the

rods fast,
- an overhead camera, and
- a standard PC, on which the control software runs.

The software executes a control cycle, consisting of the following four steps:

1. During the *vision analysis phase*, the positions of the various items on the field are estimated (see Figure 2(a)).
2. These positions are combined with knowledge about former ones in order to build the new *world model*. The world model, shown in Figure 2(b), contains information about the positions and movements of all items on field. The field is represented by a coordinate system where the origin is in the middle of the field and the x-axis connects both goals.
3. Based on this world model, the best actions are chosen in the *action selection* phase.
4. *Action execution* translates the chosen actions into steering commands. These commands are sent to the actuators.

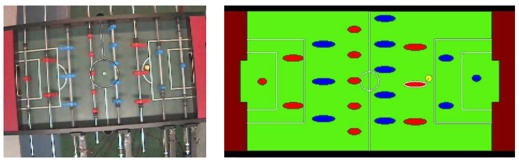

Figure 2. (a) The camera picture and (b) the generated world state

Since table soccer is a fast-paced game, the cycle duration has to be as short as possible in order to be able to react in time. KiRo works with a cycle time of 20 msec, which leads to strict bounds for the time available to select appropriate actions.

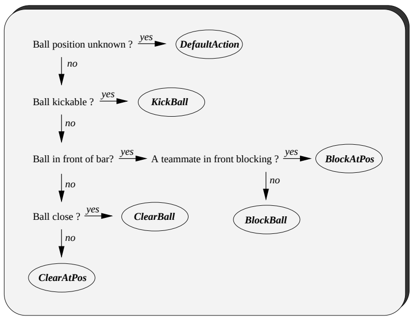

Figure 3. The original action selection procedure

The first approach to action selection has been a purely reactive decision tree. This essence of this approach is depicted in Figure 3. Although very crude, this approach was able to beat 75% of a random sample of human opponents [8].

In contrast, the system presented in this paper selects the action that promises the best consequences. To identify this action, it is necessary to plan ahead and to simulate the change in the world state caused by the different actions. As this approach needed a different kind of action model, the action control had to be completely rewritten.

3 Decision-Theoretic Planning for Table Soccer Playing

Table soccer does not seem to be well suited for decision-theoretic planning because it involves an opponent, which apparently means that we have to use game-solving methods, e.g., minimax search, instead of planning. However, the game is highly asymmetric. Only the player in possession of the ball, the *attacking player*, is able to decide on the future development of the game. The other player, the *defending player*, has to wait until the ball can be intercepted. For this reason, we can focus on a sub-game that is much easier to tackle.

We will consider the sub-game where the attacking player has continuous control of the ball. That means that this sub-game ends when the attacking player either scores a goal or looses control of the ball. This implies that the game tree can be pruned at nodes where the defending player intercepts the ball. Furthermore, all opponent actions that are unsuccessful in intercepting the ball do not influence the game at all. In summary, the defending player never needs to look ahead and just tries to intercept the current ball while the attacking player considers different possibilities of shooting the ball and takes the opponent into account only as a threat to the next action. So, we can indeed use decision-theoretic planning techniques to address the table soccer playing problem.

As a general strategy, KiRo uses a reactive positioning scheme when it is the defending player and employs decision-theoretic planning in the role of the attacking player. In what follows, we only consider the situation when KiRo is the attacking player.

3.1 Action and Forward Simulation

When planning, the attacking player can act all the time and the state changes continuously, because the ball is in motion most of time. Planning in such a setting is, of course, computationally infeasible. However, we do not have to consider all possible actions and all movements of the ball:

1. Most of the time, the movement of the ball and the rods is predictable. The ball moves according to its inertia, the rods move in the way which is specified by the employed actions.
2. The movement of the ball and of the rods are fully independent apart from one case: One figure touching the ball. In this situation, the movement of the ball is influenced.
3. Whenever the movement of the ball changes, the actions of the rods are likely to change as both players react on the new situation.

For this reason, we can interleave actions and physical simulations in a regular manner. Given a world state, an action as well as a reaction of the opponent, we simulate the evolution of the world until the point of time at which one of both players can manipulate the ball again. At this moment is it necessary to stop the simulation and to evaluate the reactions of both players in the new situation.

3.2 One Iteration of Planning

For a given world state s, in which KiRo is playing the ball, all possible consequences of KiRo's actions as well as all possible reactions of the opponent are considered. For each combination among these, a new world state is constructed which is used as a base for further planning – provided that the opponent is not playing the ball in this state.

The search tree consists of three different kind of nodes which alternate in a given sequence. Each of theses types corresponds to a different planning step. The starting point of plan iteration is always a *state node* $\mathbf{s}$ corresponding to a game state s. The first step is to select a set of applicable actions $(a_1, \ldots, a_n)$. For each $a_i \in (a_1, \ldots, a_n)$ one *action node* $\mathbf{a}_i$ is created. As these nodes represent the different choices in state s, the utility value of the state node $\mathbf{s}$ is the maximum utility among its successors (once these are known):

$$utility(\mathbf{s}) = \max_{0<i\leq n} utility(\mathbf{a}_i).$$

Figure 4 illustrates the generation of the action nodes.

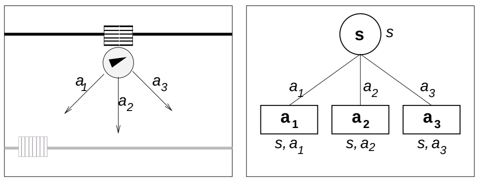

Figure 4. The action choice and its representation within the tree

The next step is the estimation of the opponent's reactions. Based on the world state s a set $\{(o_1, p(o_1)), \ldots, (o_m, p(o_m))\}$ of hypotheses is created where the o_j classify the reactions and the $p(o_j)$ the associated probabilities. Note that we assume that these reactions depend only on s and are independent from the chosen action $\mathbf{a}_i$, which in fact is true in table soccer. There is usually no way an opponent can react to an the action of an attacking player.

For each action node $\mathbf{a}_i$, a set of successor *opponent nodes* $\{\mathbf{o}_{i1}, \ldots, \mathbf{o}_{im}\}$ is created. The value of $\mathbf{a}_i$ is the expected value over its successors:

$$utility(\mathbf{a}_i) = \sum_{j=1}^{m} p(o_j) \cdot utility(\mathbf{o}_{ij}).$$

In Figure 5 two possible reactions of the opponent and the formalization of this fact in the tree is depicted.

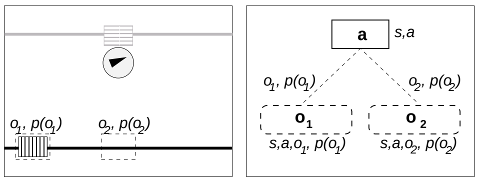

Figure 5. The formalization of the opponent

Every opponent node $\mathbf{o}_{ij}$ contains informations about the world state s as well as one specific action a_i and reaction $(o_j, p(o_j))$. To finish the planning step, the consequences $\{c_1, \ldots, c_q\}$ of a_i are estimated along with their probabilities $p(c_1), \ldots, p(c_q)$. The now gathered information $\langle s, a_i, o_j, c_k \rangle$ is used to estimate new world states s_{ijk} by means of a simulator. These states are used to build a new layer of *state nodes* $\mathbf{s}_{ijk}$.

The utility value of the precedent opponent node $\mathbf{o}_{ij}$ is calculated by

$$utility(\mathbf{o}_{ij}) = \sum_{k=1}^{q} p(c_k) \cdot utility(\mathbf{s}_{ijk}).$$

Figure 6 shows two different outcomes of an action and the use of the simulator to create a new world state based on the collected data.

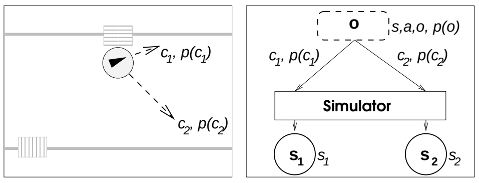

Figure 6. The possible outcomes of an action

After one iteration of planning, there exists a number of new state nodes containing the world state resulting from every possible action a_i, every possible reaction o_j and every possible consequence c_k of the action a_i. After the search tree has been built up to the leafs, those get evaluated using the utility function. These evaluations propagate backward through the tree until the root state node $\mathbf{s}$ is reached. The utility of $\mathbf{s}$ is the maximum expected utility among all selectable actions in world state s; the action a_i yielding this value is the one to be selected in s.

3.3 Choosing an Applicable Action

Currently, KiRo's capabilities comprise the following actions for operating each of the four rods under his control:

- *KickBall*: Rotate the rod by 90° in order to kick the ball forward or diagonally to the left or right.
- *BlockBall*: Move the rod so that a figure intercepts the ball.
- *ClearBall*: Move to the same position as *BlockBall* but turn the rod to let the ball pass from behind.
- *StopBall*: Pen the ball in between figure and field.

Altogether, it is possible to assign a single rod one out of 6 different actions (three of them being kicks in different directions). Taking all four rods into account is it possible to create 24 different assignments of actions. In other words, our search tree would have a branching factor of 24. Fortunately, one can easily reduce this factor because only the rod close the ball can kick and the others have a choice between the remaining three actions. We decided to assign statically *BlockBall* to all rods between the ball and the own goal in order to have a defence even when the ball is accidentally lost or reflected. Rods between the opponent's goal and the ball should not handicap the chances to score a goal. For that reason, they are assigned *ClearBall*.

With these static assignments, the branching factor is reduced to six while KiRo is playing the ball. If the opponent possesses the ball, a static defensive allocation of actions without any planning is employed. This reflects the already mentioned observation that planning in these situations is useless.

In some situations, performing a certain action might be useless (e.g. trying to stop an already stopped ball) or the chance of a successful carrying out of an action might become too low. In these cases, such actions are not evaluated in order to reduce the complexity further.

As the means the opponent is going to use in order to protect his goal are unknown, it is necessary to guess what his reactions will be. Each of these guesses is weighted by the probability that this reaction will occur. Currently, the opponent is always expected to either let his rods unmoved or to protect his goal according to the scheme applied in the *BlockBall*-action. Each of these alternatives is weighted with a probability of 0.5. These probabilities are, of course, only a crude approximation and it is planned to replace this simple opponent model by a more sophisticated, experience-based one.

3.4 Calculating the Successor State

The last – and most important – step is to estimate the probability of a success of KiRo's actions and to create a new world state based on the data collected yet. The actions of KiRo can have two possible outcomes, success and failure. The probabilities of these depend on the encountered world state, e.g. a quickly moving ball is more difficult to kick than a static one. Based on the world state and on data about the effectiveness of the actions collected during earlier games, the success probability is estimated.

The means to create a new world state based on the informations about KiRo's actions, their success or failure and the opponent's reactions is a simple simulator. This simulator models the world based on the following principles:

- The angle of incidence and the angle of reflection are equal.
- Friction is ignored.

Additionally, the simulator has a model that allows to interprete the steering commands issued by both players. Of course, the real world is only very coarsely modeled by a simulator based on these principles. Further, since the input data is imperfect, the simulation is accumulating errors. The simulated span of time, however, is very short, so that the errors are still acceptable.

Simulation is performed in two steps: In the first one, the game is simulated until the point of time in which KiRo's action takes place. Afterwards, two successor states are generated: One of the resulting world states is a state according to the known consequences of the success of KiRo's action. The other state reflects the failure of the action. In this case, we have a new problem: The consequences of a successful action are known – in case of failure, anything can happen. The most frequent kind of failure is the inability of KiRo to hit the ball. For this reason, the failing case is simulated by letting the ball pass the failing rod without changing its movement parameters. The resulting world state is not used as a starting point for a new planning step; it is directly evaluated using the heuristic utility function (see Subsection 3.6).

3.5 Estimating the Success Probabilities

In order to estimate the success probability for a given action on a certain rod in a given world state, a Bayesian approach is employed. The first step is to classify the ball movements by a 4-tuple $\langle d_x, d_y, v_x, v_y \rangle$, where

- d_x is the distance in x-direction to the rod.
- d_y is the minimal distance in y-direction to a figure on this rod .
- v_x is the relative velocity in x-direction. "Relative" means in this context e.g. "approaching" or "departing"
- v_y is the relative velocity in y-direction.

Each of these values is discretized according to a seven step scale.

The task is now to calculate the success probability $P(S|d_x, d_y, v_x, v_y)$ of the action given this tuple. Using Bayes' rule yields

$$P(S|d_x, d_y, v_x, v_y) = \frac{P(d_x, d_y, v_x, v_y|S) \cdot P(S)}{P(d_x, d_y, v_x, v_y)}$$

In order to simplify the computation of the conditional probability, two independence assumptions are made:

1. $P(d_x, d_y, v_x, v_y) = P(d_x) \cdot P(d_y) \cdot P(v_x) \cdot P(v_y)$
2. $P(d_x, d_y, v_x, v_y|S) = P(d_x|S) \cdot P(d_y|S) \cdot P(v_x|S) \cdot P(v_y|S)$.

This "naive Bayes" assumption is clearly not met; this, however, is usually the case in naive Bayes approaches (otherwise they would not be called "naive"). Using this assumption, we are able to give an easy to compute estimate for the success probability:

$$\begin{aligned} &P_{NB}(S|d_x, d_y, v_x, v_y) \\ &\hat{=} \quad \frac{P(d_x|S) \cdot P(d_y|S) \cdot P(v_x|S) \cdot P(v_y|S) \cdot P(S)}{P(d_x) \cdot P(d_y) \cdot P(v_x) \cdot P(v_y)} \end{aligned}$$

3.6 The Utility Function

Table soccer poses a highly dynamic environment where only little time is available for selecting the most appropriate action. Due to the high uncertainties in sensing and acting, it is infeasible to create and carry out a complete plan for reaching the final aim of scoring a goal. It is necessary to plan with a limited horizon and to use an *utility function* for evaluating world states. Planning in this fashion is similar to depth-limited minimax search with a heuristic evaluation of the leafs of the game tree. The utility of inner nodes is estimated using a *rollback procedure* [1].

The heuristic utility function is used to estimate the world states contained in the leafs of the search tree – provided one has not reached a scored goal yet. The principles underlying this function are:

- If a goal is either scored or going to be scored (i.e. ball behind the keeper, moving towards the goal), a value of 100 is returned if it is the opponent's goal, otherwise 0.
- The closer the ball is to the opponent's goal wall, the better.
- If the distance between the ball and both front walls is equal, it is neither important who controls the ball nor whether the ball is on the left or right of the field. The closer it gets to one of the walls, the bigger the importance of these facts.

4 Results

The decision-theoretic planner has been fully implemented in the KiRo system and tested using a simulator and the real table soccer system.

4.1 Computational Costs

The implemented system runs on a 1.7 GHz AMD processor. On this processor, the vision analysis phase, the world modeling step and action execution (see Section 2) require together roughly 5 msec. With a cycle time of 20 msec, this gives us approximately 15 msec per cycle for planning.

Table 1 shows the worst case runtimes for different search depths. The search depth is in this case defined as the number of plan iterations. One planning step constructs a subtree of the depth 3; a search depth of x therefore corresponds to a tree depth of $3x$. The table shows that search depth values over 3 are clearly not feasible for KiRo with the current processor speeds. However, with newer, faster CPUs we might even be able to go to search depth of 3.

Search depth	Runtime
1	0.3 msec
2	10 msec
3	23 msec
4	45 msec
5	80 msec

Table 1. Worst case runtimes

4.2 Performance Experiments

Two kinds of performance experiments have been conducted. The simulator has been used to compare the reactive and the decision-theoretic planning action selection directly by playing against each other. On the real table, games against human adversaries were performed to test both approaches indirectly.

4.3 Results on the Simulator

Two kinds of experiments were conducted on the simulator: In a first run, the decision-theoretic planning procedure had a fixed search depth of two. Using this setting, a number of games has been played. While the program using the reactive action selection scheme shot in average one goal in 10 minutes, the decision-theoretic approach scored once in 1.5 minutes.

As many of these goals were own goals by the reactive control system, a second criteria was employed: Field superiority. In this context, a team is called field superior if it is capable of keeping the ball in the opponent's half most of the time. The field superiority value of a team is therefore the percentage of time during which the ball was in the opponent's half of the field. Table 2 shows the results for the decision-theoretic planning approach, ordered by the employed search depth.

Search depth	Field superiority value
1	64%
2	72%
3	74%
4	57%

Table 2. Field superiority values for the decision-theoretic planning approach

The decision theoretic planning approach is field superior. The field superiority increases with the search depth until the efficiency gets decreased due to the excessive time consumption.

4.4 Results on Physical Table Soccer System

The good results from the simulation experiments could not be replicated on the real table soccer system. Since we do not have a table with robot control for both sides, we had to conduct the tests indirectly by playing against humans. The setting of these experiments was as follows. Every opponent team consisted of two players. The teams were not allowed to switch their positions, and every team had to play four matches against the robot. During two of these, the reactive action selection was controlling the robot; the other two matches were performed by the planning approach. The order of the matches was randomly drawn and the human opponents did not know against which action selection they were playing.

In 56 Matches, the reactive approach was on average able to shoot a goal in 0.6 minutes, while it took the human opponents 1.56 minutes to score. The planning approach hit the goal once in 0.84 minutes and admitted goals by the human teams on average every 1.19 minutes.

5 Conclusion

We presented a decision-theoretic planning approach to play table soccer. The presented approach used a forward-simulation scheme as well as an opponent model and a naive Bayesian approach to estimate the success probabilities of its own actions.

This approach was able to dominate a reactive action selection mechanism in direct matches performed on a simulator, but proved to be inferior in indirect comparisons playing on the real table soccer table against human teams. While the result on the real table soccer table is disappointing, the simulation results have shown that the approach has potential. In particular, there are a number of parameters that appear to be worthwhile to be experimented with. The opponent model, for example, is currently very simple and could, e.g., be trained by recording real games. Furthermore, the success probability should also be adapted to the real table. Finally, the execution of actions themselves might be able to be enhanced. Summarizing, the decision-theoretic planning approach has shown promise but still has to live up to its expectations.

REFERENCES

[1] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[2] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1), 1986.

[3] K. Dorer. Behavior networks for continuous domains using situation-dependent motivations. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1233–1238, Stockholm, Sweden, 1999.

[4] K. Dorer. The magmaFreiburg Soccer Team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, Lecture Notes in Artifi cial Intelligence, pages 600–603. Springer-Verlag, Berlin, Heidelberg, New York, 2000.

[5] P. Haddawy and M. Suwandi. Decision-theoretic refi nement planning using inheritance abstraction. In K. Hammond, editor, *Proc. Second International Conference on Artificial Intelligence*. University of Chicago, Illinois, AAAI Press, 1994.

[6] P. Maes. Situated agents can have goals. In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 49–70. MIT Press, Cambridge, MA, 1990.

[7] T. Weigel, J.-S. Gutmann, A. Kleiner, M. Dietl, and B. Nebel. CS-Freiburg: Coordinating Robots for Successful Soccer Playing. *IEEE Transactions on Robotics and Automation*, 18(5):685–699, October 2002.

[8] T. Weigel and B. Nebel. KiRo – An Autonomous Table Soccer Player. In *Proc. Int. RoboCup Symposium '02*, pages 119 – 127. Springer-Verlag, Fukuoka, Japan, 2002.