UvA Trilearn 2003 Team Description

Jelle R. Kok, Nikos Vlassis, and Frans Groen

Faculty of Science, University of Amsterdam
{jellekok,vlassis,groen}@science.uva.nl

Abstract. This paper describes the main features of the *UvA Trilearn* soccer simulation team, which participated for the first time at the *RoboCup-2001* competition. The main concepts of the previous teams will be addressed, followed by the improvements introduced in *UvA Trilearn 2003*. These include an extension of the intercept skill, improved passing behavior and especially the usage of coordination graphs to specify the coordination requirements between the different agents. Finally, we will give some conclusions and describe future research directions.

1 Introduction

The *UvA Trilearn 2001* [1] soccer simulation team was built by two masters' students for their graduation project. Much of the effort in this team had gone into getting the lower levels to work, since we felt that these would be the most crucial for the success of the team. This has among other things led to a multi-threaded three-layer architecture with an advanced synchronization method, a probabilistic world model from which several high-level conclusions could be derived and a layered skills hierarchy.

UvA Trilearn 2002 [3] contained several improvements which included improved localization methods using particle filters, behavior modeling of teammates, and an action selection method based on a priority-confidence model.

During these projects much attention was paid to software engineering issues to facilitate future use. This led to highly modular object oriented code and to a multi-level log system for quick debugging (similar to [6]). We have released large parts of our source code[1], which several teams use as a basis. The released source code contains our lower levels (synchronization, world model, basic agent skills) together with a simple high-level strategy, similar to the one released by FC Portugal [5] after *RoboCup-2000* to make a working team.

In our current work, *UvA Trilearn 2003*, we have made three extensions to our team. First of all, we have improved the intercept skill such that it also takes opponents into account to determine the best interception point. Secondly, we improved the passing behavior of the players by taking much more passing options in consideration. This is possible since we use an efficient algorithm [7] to calculate the interception time of an opponent. Finally, we make use of coordination graphs [2] to specify the coordination between the agents.

[1] Available from `http://www.science.uva.nl/~jellekok/robocup/`.

2 Intercept skill

In *UvA Trilearn 2003* we have improved our interception skill. Intercepting the ball is one of the most important player skills that is frequently used by every type of player. In the previous teams, a player would always intercept the ball at the first future position of the ball it was able to reach in the same number of cycles. For this an iterative scheme was used to compute the future positions of the ball. A loop was executed in which a prediction method was used to predict the position q_{t+i} of the ball a number of cycles i into the future and to predict the number of cycles n that the agent will need to reach this position. This was repeated for increasing values of i until $n < i$ in which case it was assumed that the agent should be able to reach the point q_{t+i} before the ball. The agent would then move to this position q_{t+i}.

Obviously, this is not the optimal behavior in all situations. Because of the stochastic nature of the simulator, it was possible that the agent just missed the ball and had to follow the ball until the ball lost almost all its velocity. Furthermore, this method does not take opponents into account that are able to intercept the ball at an earlier position than the intercepting player.

In our current implementation, we use an efficient numerical algorithm [7] to compute the interception time of an agent to the ball. This method first abstracts away from the discrete nature of the simulator by assuming that the the ball moves in its current direction with instantaneous velocity ($V = V_0 e^{-t/\tau}, \tau = 2s$) and the player moves with a fixed (maximum) velocity. Several simple heuristics can be used to compensate for the turning and acceleration of the agent to reach the used velocity. Using these simplifications it is possible to calculate the first possible interception time efficiently using an adaption of Newton's method. In most cases the interception time can now be determined in less than five iterations. See the appendix of [7] for the complete procedure.

Using this method, we can calculate the first position q_{t+k} along the ball trajectory at which an opponent can intercept the ball. Our previous intercept method is now adapted such that the intercepting player will move to the ball position q_{t+k} in case $k < n$. When $k > n$ the intercepting player will not move to the first possible interception point, but to the point q_{t+j} such that $n \leq j \leq k$ and the calculated distance between q_{t+j} and the point the intercepting player is able to reach in j cycles is minimized. The agent thus performs a 'safer' intercept and is less dependent on the stochasticity of the simulator.

3 Passing options

In our previous teams, we only looked to one passing option when considering to play the ball to a teammate. The direction of the pass was determined based on the widest angle between the different opponents and the speed was based on the desired (fixed) end speed of the ball when it reached the teammate. The successfulness of this pass was determined based on the interception time of the fastest opponent to this ball trajectory. We only considered one passing option since our method to calculate the specific interception time for the opponents was too expensive to be applied many times during one cycle.

In our current team, we use the same procedure as described in Section 2 to determine the interception time of the opponents to a possible future ball trajectory. Using this method, we can generate many different ball trajectories with different speeds and efficiently calculate the interception time of our teammate and that of the opponents to this ball trajectory. We then take that ball trajectory in which our teammate intercepts the ball just in front of its current position and furthermore maximizes the time difference between the interception time of our teammate and the fastest opponent.

This has a big advantage over the previous method, as it results in a much wider variety of passing options.

4 Coordination

Depending on the current situation, certain agents on the field have to coordinate their actions, i.e. the agent with the ball must decide to which nearby agent to pass, the receivers must anticipate a future pass and the defenders must coordinate to organize the defense. In our current team, we incorporate the coordination requirements between the agents using coordination graphs [2].

A coordination graph (CG) represents the coordination requirements of a system. A node in the graph represents an agent, while an edge in the graph defines a (possible directed) dependency between two agents. After the topology of the graph is dynamically updated based on the current context, only the interconnected agents have to coordinate their actions at any particular instance to find the joint optimal action. This can be done using an efficient variable elimination algorithm. Via a context-specific decomposition of the problem into smaller subproblems, CGs thus offer scalable solutions to the problem of multi-agent decision making.

In order to apply CGs to the continuous domain we assign roles to the agents and then coordinate the different roles [4]. Such an assignment provides a natural way to parametrize a coordination structure over a continuous domain. Furthermore, this can be regarded as an abstraction of a continuous state to a discrete context, allowing the application of existing techniques for discrete-state CGs. Finally, roles can reduce the action space of the agents by 'locking out' specific actions. For example, the role of the goalkeeper does not include the action 'score', and in a 'passive' role the action 'shoot' is deactivated. Such a reduction of the action space can offer computational savings, but more importantly it can facilitate the solution of a local coordination game by restricting the joint action space to a subspace that contains only one Nash equilibrium.

In our current team, we use this framework to improve upon the passing between the teammates. Instead of being reactive (a player only starts intercepting after it observes a change in the ball velocity), the coordination framework makes sure that the agents already move to the direction the pass will be going to before the pass is actually given. An example of a rule to specify such a coordination pass can be defined as follows:

$$
\begin{aligned}
\langle p_1^{passer} \quad ; \quad & \text{has-role-receiver}(j) \quad \wedge \\
& \neg\text{isPassBlocked}(i, j, dir) \quad \wedge \\
& a_i = \text{passTo}(j, dir) \quad \wedge \\
& a_j = \text{moveTo}(dir) : u(j, dir)\rangle \ \forall j \neq i
\end{aligned}
$$

See [4] for details of the algorithm.

5 Conclusion and future directions

In this paper we have quickly addressed some improvement in our new soccer simulation team *UvA Trilearn 2003*. We haven't provided experiments to test the results of the improvements, but the outcome of the regional tournaments in which we participated[2] does show that the improvements have a positive influence on the team as a whole.

For future directions, we are interested in applying reinforcement learning techniques to a continuous-domain CG in order to learn the payoff functions in an automatic way. Finally, from an application point of view we want to apply the CG model further to the simulation RoboCup, such that the agents also coordinate during other actions than passing, like organizing the defense or obstructing opponent passes.

References

1. R. de Boer and J. R. Kok. The Incremental Development of a Synthetic Multi-Agent System: The UvA Trilearn 2001 Robotic Soccer Simulation Team. Master's thesis, University of Amsterdam, The Netherlands, Feb. 2002.
2. C. Guestrin, S. Venkataraman, and D. Koller. Context specific multiagent coordination and planning with factored MDPs. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, Edmonton, Canada, July 2002.
3. J. R. Kok, R. de Boer, N. Vlassis, and F. Groen. UvA Trilearn 2002 team description. In G. Kaminka, P. Lima, and R. Rojas, editors, *RoboCup 2002: Robot Soccer World Cup VI*, page 549, Fukuoka, Japan, 2002. Springer-Verlag.
4. J. R. Kok, M. T. J. Spaan, and N. Vlassis. Multi-robot decision making using coordination graphs. In *Proceedings of the 11th International Conference on Advanced Robotics*, Coimbra, Portugal, 2003.
5. L. P. Reis and J. N. Lau. FC Portugal Team Description: RoboCup-2000 Simulation League Champion. In P. Stone, T. Balch, and G. Kraetszchmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, pages 29–40. Springer Verlag, Berlin, 2001.
6. P. Riley, P. Stone, and M. Veloso. Layered Disclosure: Revealing Agents' Internals. In *Proceedings of the Seventh International Workshop on Agent Theories, Architectures and Languages (ATAL-2000)*, 2000.
7. P. Stone and D. McAllester. An Architecture for Action Selection in Robotic Soccer. In *Fifth International Conference on Autonomous Agents*, 2001.

[2] Champion *German Open 2003* with goal difference 136-0, and Champion *American Open* with goal difference of 100-0.