Gliders2012: Tactics with Action-dependent Evaluation Functions

Mikhail Prokopenko[1], Oliver Obst[1], Peter Wang[1], and Jason Held[2]

[1] CSIRO Information and Communication Technologies Centre, PO Box 76, Epping, NSW 1710, Australia
[2] Saber Astronautics Australia, 53 Balfour St, Chippendale, NSW 2008, Australia

1 Introduction

The RoboCup Simulation League [1] incorporates several challenging features, setting a benchmark for Artificial Intelligence (AI). The following list includes some of the most prominent characteristics of the RoboCup 2D Simulation League:

- distributed client/server system running on a network, leading to fragmented, localized and imprecise (noisy and latent) information about the environment (field) [2];
- concurrent communication with a medium-sized number of agents [3];
- heterogeneous sensory data (visual, auditory, kinetic) and limited range of basic commands/effectors (turn, kick, dash, . . .) [4];
- asynchronous perception-action activity and limited window of opportunity to perform an action [5];
- autonomous decision-making under constraints enforced by teamwork (collaboration) and opponent (competition) [6];
- conflicts between reactivity and deliberation [7];
- no centralized controllers and centralized world model (no global vision, etc.) [8, 9].

From the onset of the RoboCup effort it was recognized that, as a benchmark, RoboCup is fairly different from another classical AI problem — chess. As pointed out by Asada et al. [10], chess and RoboCup differ in a few key elements: environment (static vs dynamic), state change (turn-taking vs real-time), information accessibility (complete vs incomplete), sensor readings (symbolic vs non-symbolic), and control (central vs distributed). This difference has been well understood over the last decade. Nevertheless, there are some similarities, for example, efficient evaluation functions used by the RoboCup agents are conceptually similar to evaluation functions used by chess computers: in either case the agent is attempting to consider multiple future states, assign some values to the alternative outcomes, and choose an action optimizing the evaluations. One may argue that superior performance of recent world champions in the RoboCup 2D Simulation League [11, 12] may be attributed, at least partially, to sophisticated evaluation functions employed by these teams. In this short paper we describe a novel mechanism utilizing action-dependent evaluation functions, comparing it to some well known constructive models used by belief revision and belief update [13].

The experiments are carried out using a new simulated soccer team for the RoboCup soccer 2D simulator [14], Gliders2012. The team code is written by C++ using agent2d: the well-known base code developed by Akiyama et al. [15]. Other software packages are used as well:

- librcsc: a base library for RCSS with various utilities describing relevant geometrical constructs, world model, etc.;

- soccerwindow2: a viewer program for RCSS, working as a monitor client, a log player and a visual debugger;
- fedit2: a formation editor for agent2d, allowing to design a team formation.

2 Motivation and approach

2.1 Chess analogy

As argued by Laramée, in chess "the evaluation function, is unique in a very real sense: while search techniques are pretty much universal and move generation can be deducted from a game's rules and no more, evaluation requires a deep and thorough analysis of strategy" [16]. He lists several main board evaluation metrics: material balance (an account of which pieces are on the board for each side), mobility (a measure of how many move options are available, especially for powerful chess pieces), board control (a side controls a square if it has more pieces attacking it than the opponent), development (minor pieces should be brought into the game as quickly as possible), pawn formations, king safety and tropism (a measure of how easy it is for a piece to attack the opposing king; usually measured in terms of distance).

One may draw some parallels with RoboCup Simulation. For example, the goal safety and distances to the opposing goal are analogous to king safety and tropism, pawn formations may give some hints to team formations, development is somewhat similar to developing an attack from within your own half, board control is akin to blocking and marking opponent players (i.e., field control), mobility is achieved by either positioning teammates to receive a pass, or creating multiple directions for dribble, and material balance can be computed by accounting for heterogeneous player types and remaining stamina values. All these analogies are, of course, not direct — nevertheless, they may be provide some inspiration for an evaluation function relevant for RoboCup Simulation.

2.2 Basic evaluation

The evaluation function of agent2D is, however, quite simple. Using the chess analogy, it implements *tropism* only, and is intended to make the basic client play in a goal-oriented fashion. For a player controlling the ball, it considers two features of each possible resultant state s: its X-coordinate (the larger the better) and the distance from it to the opponent's goal (the smaller the better). That is, the opponent's goal is the ultimate desirable resultant state S, and each action a is rated in terms of a single distance metric D

$$r(a) = D(s = result(a), S) \; . \tag{1}$$

The action that is selected is simply the one that minimizes the distance between resultant and desirable states, i.e., minimizes this metric:

$$a^* = \arg\min_a r(a) \; . \tag{2}$$

For the players who are not controlling the ball and are engaged in intercept behavior, the evaluation function is not specified explicitly. These players select positions on the field according to their roles in the team formation.

The evaluation function (2) has reached a significant aim: all types of actions (dribbles, passes, etc.) can be directly compared to each other in terms of a single metric. At the same time, the simple

computation is not adequate to support a very sophisticated tactical play in mid-field, or even near opponent's penalty area. Another drawback is that all the actions are judged in relation to a single point: the opponent goal.

2.3 Multiple desirable states: tactics

Our main objective in this study is to retain the advantage of a single metric, but diversify the evaluation by considering multiple points as desirable states. Moreover, we suggest not only that the most desirable state can change from one cycle to another, but also that a player may entertain multiple desirable states at any given time (cycle). This diversity is brought about by different tactics. For example, a player may consider one desirable state S_1 if passing to the left (action a_1) pursuing one tactic, another resultant state S_2 if passing to the right (action a_2) guided by another tactic, and yet another desirable state S_3 if dribbling to the center (action a_3) suggested by a third tactic. Each of the considered actions is evaluated with respect to the corresponding desirable state that represents one of possible tactical ways to develop the play.

In other words, at any given time, there is a number m of tactics represented by a set of desirable states: $\{S_1, \ldots, S_m\}$, and the feasible actions are partitioned into m sets: $A_1, \ldots, A_m$, so that for every action a_i, there is a set A_j such that $a_i \in A_j$. We denote the function mapping an action to its tactical state by

$$tactics : a \rightarrow S \; . \tag{3}$$

Then each feasible action is rated with respect to the corresponding desirable state:

$$r(a) = D(s = result(a), S = tactics(a)) \tag{4}$$

followed by selection according to the optimization (2). This approach does not impose tactics in a top-down fashion, selecting one tactic and the sub-selecting the best action for the chosen tactic. Rather, all feasible actions are considered, and tactics contribute to the evaluation via the desirable states suggested by the tactics. In certain cases the opponent's goal becomes one of possible desirable states (one of the tactics), keeping the goal-oriented behavior of agents.

The difference between definitions (1) and (4) is simply that the desirable states that the player is trying to reach are not independent of actions, but rather *are* action-dependent, and this dependence is tactical. To re-iterate, the comparison between two actions a_1 and a_2 according to the first definition (1) always assumes the same action-independent state S that is evaluated against, while the proposed definition (4) allows for different desirable states $S_1 = tactics(a_1)$ and $S_2 = tactics(a_2)$. The metric D is the same for all actions, retaining the advantage of a uniform comparison across different action types.

We would like to point out at this stage a difference between the proposed action-dependent evaluation function and other action-dependent formalisms, e.g., with action-dependent features generalizing state space proposed by Stone and Veloso [17]. The latter study described a multi-agent learning paradigm called team-partitioned, opaque-transition reinforcement learning (TPOT-RL). TPOT-RL introduced the concept of using action-dependent features to generalize the state space. However, regardless of action-dependent features, each possible action a is evaluated by TPOT-RL based on the current state of the world using a *fixed* function $e : (S, A) \rightarrow U$. That is, the function e is the same for all actions in TPOT-RL.

Another interesting point is the analogy between multiple desirable states unified by the proposed evaluation function and the constructive model for belief update and belief revision [13]. Belief

revision is the process by which a rational agent changes their beliefs about a static world in the light of new data. Belief update on the other hand is the process by which an agent maintains their beliefs up to date with an evolving world. The constructive model for belief revision includes a single similarity structure centered on all possible worlds consistent with current beliefs (a single system of nested spheres), and identifies the nearest sphere which is consistent with the new data. Peppas et al. [13] have shown that the model for belief update uses multiple systems of spheres (one for each possible world), finds in parallel the spheres consistent with the new data that are nearest to their respective central possible worlds, and collects possible worlds within these spheres. Arguably, the action-dependent evaluation proposed here is akin to the constructive model of belief update.

2.4 Mobility and field control

The function $tactics$ implements the *mobility* aspect of evaluation, by diversifying options of the player controlling the ball in continuing the game. The other teammates can also use this function in selecting a desirable state for their positioning. That is, a player choosing a position on the field does not have to have a single best point, given the current state. It may consider multiple points, each of which is again dependent on the action. For example, the player may consider state (point) S_1 when moving to the left wing with action a_1, and state (point) S_2 when blocking a nearest opponent with action a_2. Each of the resultant states $s_1 = result(a_1)$ and $s_2 = result(a_2)$ are compared with the corresponding desirable states suggested by the tactics S_1 and S_2, and the action achieving the best proximity in terms of the metric D is selected. The diversification in positioning achieves both *mobility* (by enabling better passes to these teammates) and *field control* — by taking key points and blocking key directions.

The idea of field control can be traced to a generic framework describing abstract spatio-temporal relationships described by Dylla et al. [18]. The latter work did not menion field control explicitly but argued that a reachability relation is needed to express spatial relationships between the players and the ball. They suggested to use Voronoi diagrams: a Voronoi diagram is the partitioning of a plane with n points into n convex polygons such that each polygon contains exactly one point and every point in the given polygon is closer to its central point than any other [18]. This was further developed by Akiyama et al. who used a dual representation of Voronoi diagrams — the Delaunay triangulation [19, 11].

2.5 Example

Figure 1 illustrates the concept of action-dependent evaluation. The player controlling the ball (left team, number 11) has several options available: it can dribble in a general forward-left direction, pass to teammates 7 and 10 (in a number of ways, including direct and lead passes), etc. We consider three choices (shown by arrows): dribble forward-left, pass to the left to teammate 7, and pass to the right to teammate 10. The agent2d's evaluation function would most likely rated the dribble higher, as the resultant state (the arrow-head) has a larger X-coordinate and a smaller distance from the opponent's goal than the alternatives. The new evaluation function identifies two desirable states instead, shown by a small rectangle to the left of player 11, and a small circle to its right. The rectangle defines the tactic suggesting to develop an attack to the left and through the center, and the circle corresponds to the tactic preferring the right wing. The dribble and pass to teammate 7 are partitioned to the first tactic (rectangle), and the pass to teammate 10 belongs to the second tactic (circle). Each of these actions is rated by proximity of their resultant states (the arrow-heads) to the rectangle and circle respectively. The pass to number 10 has the smaller distance between the resultant and desirable states, and is then selected.

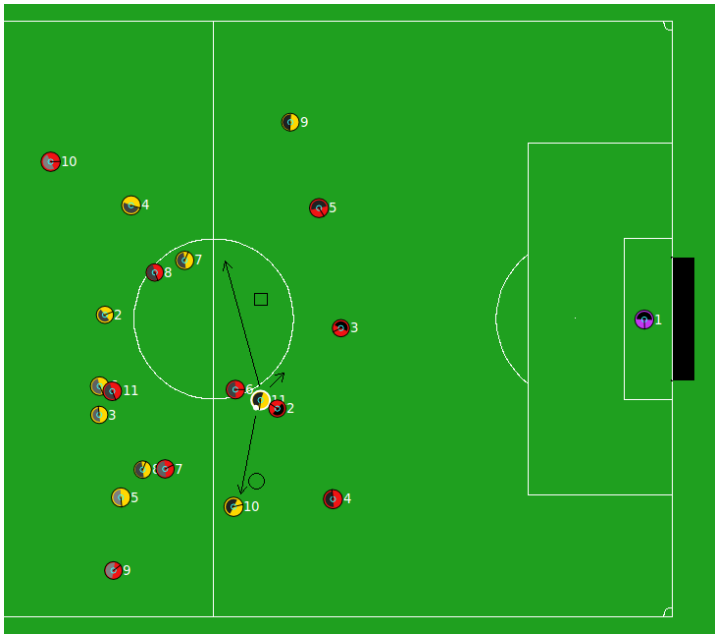

Fig. 1. Action-dependent evaluation. Small rectangle and circle show different desirable states. The arrows point to possible resultant states. The pass to player 10 is selected since the distance between its resultant and desirable states is the smallest.

3 Conclusion

We described a novel mechanism utilizing action-dependent evaluation functions, having applied it in the RoboCup Simulation 2D. The mechanism can be contrasted with some well known constructive models used by belief revision and belief update [13]. The approach also allowed us to draw parallels with evaluation functions employed by chess-playing computers, in terms of mobility, field control, tropism, etc. The evaluation function that varies desirable states dependent on contemplated actions is applicable in both ball-controlling and positioning scenarios. The tactics that correspond to multiple desirable states are not imposed in a top-down fashion, but rather contribute to the evaluation via these desirable states.

The proposed approach was implemented in Gliders2012 — a new team based on agent2d [15]. We carried out multiple iterative experiments, matching Gliders2012 up against the agent2d (HELIOS Base team), and achieving $\approx +4.0$ goal difference, typically averaged over 100 games.

Acknowledgments The Authors are thankful to Valentina Cupac, Andrew Curline, Tim D'Adam, Ivan Duong, Edward Moore, James Nugent, Tom Stewart for their contribution. Team logo was created by Matthew Chadwick. Some of the Authors have been involved with RoboCup Simulation 2D in the past, however the code of their previous teams (Cyberoos and RoboLog) is not used in Gliders2012.

References

1. Kitano, H., Tambe, M., Stone, P., Veloso, M.M., Coradeschi, S., Osawa, E., Matsubara, H., Noda, I., Asada, M.: The RoboCup Synthetic Agent Challenge 97. In: RoboCup-97: Robot Soccer World Cup I, London, UK, Springer (1998) 62–73
2. Noda, I., Stone, P.: The RoboCup Soccer Server and CMUnited Clients: Implemented Infrastructure for MAS Research. Autonomous Agents and Multi-Agent Systems **7**(1–2) (July–September 2003) 101–120
3. Stone, P., Veloso, M.: Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. Artificial Intelligence **110**(2) (June 1999) 241–273
4. Riley, P., Stone, P., Veloso, M.: Layered disclosure: Revealing agents' internals. In Castelfranchi, C., Lesperance, Y., eds.: Intelligent Agents VII. Agent Theories, Architectures, and Languages — 7th. International Workshop, ATAL-2000, Boston, MA, USA, July 7–9, 2000, Proceedings. Lecture Notes in Artificial Intelligence. Springer, Berlin, Berlin (2001)
5. Butler, M., Prokopenko, M., Howard, T.: Flexible synchronisation within RoboCup environment: A comparative analysis. In: RoboCup 2000: Robot Soccer World Cup IV, London, UK, Springer (2001) 119–128
6. Stone, P., Riley, P., Veloso, M.: Defining and using ideal teammate and opponent models. In: Proceedings of the Twelfth Annual Conference on Innovative Applications of Artificial Intelligence. (2000)
7. Reis, L.P., Lau, N., Oliveira, E.: Situation based strategic positioning for coordinating a team of homogeneous agents. In: Balancing Reactivity and Social Deliberation in Multi-Agent Systems, From RoboCup to Real-World Applications (selected papers from the ECAI 2000 Workshop and additional contributions), London, UK, Springer (2001) 175–197
8. Prokopenko, M., Wang, P.: Relating the entropy of joint beliefs to multi-agent coordination. In Kaminka, G.A., Lima, P.U., Rojas, R., eds.: RoboCup 2002: Robot Soccer World Cup VI. Volume 2752 of Lecture Notes in Computer Science., Springer (2003) 367–374
9. Prokopenko, M., Wang, P.: Evaluating team performance at the edge of chaos. In Polani, D., Browning, B., Bonarini, A., Yoshida, K., eds.: RoboCup 2003: Robot Soccer World Cup VII. Volume 3020 of Lecture Notes in Computer Science., Springer (2003) 89–101
10. Asada, M., Kitano, H., Noda, I., Veloso, M.: RoboCup: Today and tomorrow – What we have have learned. Artificial Intelligence **110** (1999) 193–214
11. HELIOS2010 Team Description. In: RoboCup 2010: Robot Soccer World Cup XIV. Volume 6556 of Lecture Notes in Computer Science., Springer (2011)
12. WrightEagle and UT Austin Villa: RoboCup 2011 Simulation League Champions. In: RoboCup 2011: Robot Soccer World Cup XV. Lecture Notes in Artificial Intelligence, Springer (2012)
13. Peppas, P., Nayak, A.C., Pagnucco, M., Foo, N.Y., Kwok, R.B.H., Prokopenko, M.: Revision vs. update: Taking a closer look. In Wahlster, W., ed.: 12th European Conference on Artificial Intelligence, Budapest, Hungary, August 11-16, 1996, Proceedings, John Wiley and Sons, Chichester (1996) 95–99
14. Chen, M., Dorer, K., Foroughi, E., Heintz, F., Huang, Z., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Murray, J., Noda, I., Obst, O., Riley, P., Steffens, T., Wang, Y., Yin, X.: Users Manual: RoboCup Soccer Server — for Soccer Server Version 7.07 and Later. The RoboCup Federation. (February 2003)
15. Akiyama, H.: Agent2D Base Code. http://www.rctools.sourceforge.jp (2010)
16. Laramée, F.D.: Chess Programming Part VI: Evaluation Functions. http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/chess-programming-part-vi-evaluation-functions-r1208 (2000)
17. Stone, P., Veloso, M.M.: Team-partitioned, opaque-transition reinforced learning. In: RoboCup-98: Robot Soccer World Cup II, London, UK, Springer (1999) 261–272
18. Dylla, F., Ferrein, A., Lakemeyer, G., Murray, J., Obst, O., Röfer, T., Schiffer, S., Stolzenburg, F., Visser, U., Wagner, T.: Approaching a Formal Soccer Theory from the Behavior Specification in Robotic Soccer. Bioengineering. In: Computers in Sport. WIT Press (2008) 161–186
19. Akiyama, H., Noda, I.: Multi-agent positioning mechanism in the dynamic environment. In Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F., eds.: RoboCup 2007: Robot Soccer World Cup XI. Springer, Berlin, Heidelberg (2008) 377–384