Gliders2d: Source Code Base for RoboCup 2D Soccer Simulation League

Mikhail Prokopenko[1,2]([✉]) and Peter Wang[2]

[1] Complex Systems Research Group, Faculty of Engineering and IT,
The University of Sydney, Sydney, NSW 2006, Australia
mikhail.prokopenko@sydney.edu.au
[2] Data Mining, CSIRO Data61, PO Box 76, Epping, NSW 1710, Australia

Abstract. We describe Gliders2d, a base code release for Gliders, a soccer simulation team which won the RoboCup Soccer 2D Simulation League in 2016. We trace six evolutionary steps, each of which is encapsulated in a sequential change of the released code, from v1.1 to v1.6, starting from agent2d-3.1.1 (set as the baseline v1.0). These changes improve performance by adjusting the agents' stamina management, their pressing behaviour and the action-selection mechanism, as well as their positional choice in both attack and defense, and enabling riskier passes. The resultant behaviour, which is sufficiently generic to be applicable to physical robot teams, increases the players' mobility and achieves a better control of the field. The last presented version, Gliders2d-v1.6, approaches the strength of Gliders2013, and outperforms agent2d-3.1.1 by four goals per game on average. The sequential improvements demonstrate how the methodology of human-based evolutionary computation can markedly boost the overall performance with even a small number of controlled steps.

1 Introduction

The RoboCup Soccer 2D Simulation League contributes to the overall RoboCup initiative, sharing its inspirational Millennium challenge: producing a team of fully autonomous humanoid soccer players capable of winning a soccer game against the 2050 FIFA World Cup holder, while complying with the official FIFA rules [6]. Over the years, the 2D Simulation League made several important advances in autonomous decision making under constraints, flexible tactical planning, collective behaviour and teamwork, communication and coordination, as well as opponent modelling and adaptation [7,20,28–30,33,34,38,41]. These advances are to a large extent underpinned by the standardisation of many low-level behaviours, world model updates and debugging tools, captured by several notable base code releases, offered by "CMUnited" team from Carnegie Mellon University (USA) [35,37], "UvA Trilearn" team from University of Amsterdam (The Netherlands) [15], "MarliK" team from University of Guilan (Iran) [40], and "HELIOS" team from AIST Information Technology Research Institute (Japan) [1]. As a result, almost 80% of the League's teams eventually switched their code

base to agent2d over the next few years [30]. The 2016 champion team, Gliders2016 [25,30], was also based on the well-developed code base of *agent2d-3.1.1* [1], and fragments of MarliK source code [40], all written in C++.

The winning approach developed by Gliders combined human innovation and artificial evolution, following the methodologies of *guided self-organisation* [19,21,22,26] and *human-based evolutionary computation* (HBEC). The latter comprises a set of evolutionary computation techniques that incorporate human innovation [8,16]. This fusion allowed us to optimise several components, including an action-dependent evaluation function proposed in Gliders2012 [27], a particle-swarm based self-localisation method and tactical interaction networks introduced in Gliders2013 [4,11,12,17,24], a new communication scheme and dynamic tactics with Voronoi diagrams utilised by Gliders2014 [23], bio-inspired flocking behaviour incorporated within Gliders2015 [32], and opponent modelling diversified in Gliders2016 [25]. The framework achieved a high level of tactical proficiency ensuring players' mobility and the overall control over the field.

We describe a base code release for Gliders, called *Gliders2d*, version v1, with 6 sequential changes which correspond to 6 evolutionary HBEC steps, from v1.1 to v1.6. Since Gliders2d release is based on agent2d, the version Gliders2d-v1.0 is identical to agent2d-3.1.1 (apart from the team name), but every next step includes a new release. It is important to point out that Gliders2d is an evolutionary branch separate from the (Gliders2012—Gliders2016) branch, and both branches evolved independently. The final version of the presented release, Gliders2d-v1.6, is neither a subset not superset of any of Gliders2012—Gliders2016 teams. However, as a point of reference, we note that Gliders2d-v1.6 has a strength approaching that of Gliders2013 [24], and future releases will improve the performance further.

Our objectives in making this first release are threefold: (a) it includes several important code components which explain and exemplify various approaches taken and integrated within the champion team Gliders2016; (b) it illustrates the HBEC methodology by showing some of the utilised primitives, while explicitly tracing the resultant performance (i.e., the fitness) for each sequential step from v1.1 to v1.6; (c) it demonstrates how one can make substantial advances, starting with the standard agent2d code, with only a small number of controlled steps. It may help new teams in making the first steps within the league, using the available base code.

2   Methodology and Results

The HBEC approach evolves performance across an artificial "generation", using an automated evaluation of the fitness landscape, while the team developers innovate and recombine various behaviours. The mutations are partially automated. On the one hand, the development effort translates human expertise into novel behaviours and tactics. On the other hand, the automated evaluation platform, utilised during the development of Gliders, and Gliders2d in particular, leverages the power of modern supercomputing in exploring the search-space.

Each solution, represented as the team source code, can be interpreted as a "genotype", encoding the entire team behaviour in a set of "design points". A design point, in the context of a data-farming experiment, describes a specific combination of input parameters [10], defining either a single parameter (e.g., pressing level), complex multi-agent tactics (e.g., a set of conditional statements shaping a positioning scheme for several players), or multi-agent communication protocols [14,30,41].

While some design points are easy to vary, others may be harder to mutate and/or recombine due to their internal structure. For example, a specific tactic (design point), created by a team developer, may be implemented via several conditional statements each of which comprises a condition and an action, involving multiple parameters and primitives (see next subsections for examples). These components can then be mutated and recombined as part of the genotype.

The solutions are evaluated against a specific opponent, over thousands of games played for each generation. In order to maintain coherence of the resultant code, which evolves against different opponents in parallel, auxiliary conditions switch the corresponding parts of design points on and off for specific opponents [30], in an analogy to epigenetic programming [39]. The fitness function is primarily based on the average goal difference, with the average points as a tie-breaker, followed by the preference for a lower standard error.

The main thread in the evolutionary branch described in this release aims to ensure a better control of the soccer field, by different means: (i) stamina management with higher dash power rates; (ii) more intense pressing of the ball possessing opponent; (iii) actions' evaluation aimed at delivering the ball to points stretching the opposition most; (iv) attacking players positioning to maximise their ball reachability potential; (v) defending players positioning to minimise the ball reachability potential of the opponents; (vi) risky passes. These improvements may in general be applied to robotic teams in physical RoboCup leagues.

In tracing the relative performance of Gliders2d from v1.1 to v1.6 we used three benchmark teams: agent2d-3.1.1 itself [1], Gliders2013 [24], and the current world champion team, HELIOS2018 [18]. For each sequential step, 1000 games were played against the benchmarks. Against agent2d, the goal difference achieved by Gliders2d-v1.6 improves from zero to 4.2. Against HELIOS2018, the goal difference improves from -12.73 to -4.34. Finally, against Gliders2013, the goal difference improves from -5.483 to -0.212, achieving near-parity. Tables 1, 2, and 3 summarise the performance dynamics, including the overall points for and against, goals scored and conceded, the goal difference, and the standard error of the mean.

2.1   Gliders2d v1.1: Stamina Management

The first step in improving upon agent2d performance, along the released evolutionary branch, is adding adjustments to the agents' stamina management (confined to a single source file `strategy.cpp`). Specifically, there are four additional assignments of the maximal dash power in certain situations, for example:

```
// G2d: run in opp penalty area
       else if ( wm.ball().pos().x > 36.0 && wm.self().pos().x > 36.0 && mate_min <
            opp_min - 4 )
                  dash_power = ServerParam::i().maxDashPower();
```

This fragment of the source code demonstrates how these specific situations are described through conditions constraining the ball position, the agent position and its role, the offside line, and the minimal intercept cycles for the Gliders2d team (`mate_min`) and the opponent team (`opp_min`). Such constraints can be evolved by mutation or recombination of primitives (`argument (op) X`), where `X` is a constraint, the argument is a state of relevant variable, e.g. `wm.ball().pos().x`, and `(op)` is a relational operator, e.g., $<$, $>$, $==$, and so on. The action form may vary from a simple single assignment (the maximal dash power in this case), to a block of code.

Adding the stamina management conditions increased the goal difference against HELIOS2018 from -12.729 to -6.868, and against Gliders2013 from -5.483 to -2.684.

2.2   Gliders2d v1.2: Pressing

The second step along this evolutionary branch is adding adjustments to the agents' pressing behaviour (confined to a single source file `bhv_basic_move.cpp`). The `pressing` level, more precisely, level of *pressure*, is expressed as the number of cycles which separate the minimal intercept cycles by the agent (`self_min`) and the fastest opponent (`opp_min`). The intercept behaviour forcing the agent to press the opponent with the ball is triggered when `self_min < opp_min + pressing`. In agent2d the pressing level is not distinguished as a variable, being hard-coded as 3 cycles, and making it an evolvable variable is an example of a simple innovation. Specifically, there are several assignments of the pressing level, tailored to different opponent teams, agent roles and their positions on the field, as well as the ball location.

Again, adding the evolved conditions for pressing increased the goal difference against agent2d from near-zero to 1.288, against HELIOS2018 from -6.868 to -6.476 (this increase is within the standard error of the mean), and against Gliders2013 from -2.684 to -1.147.

2.3   Gliders2d v1.3: Evaluator

The third step modifies the action evaluator, following the approach introduced in Gliders2012 [27], which diversified the single evaluation metric of agent2d by considering multiple points as desirable states. The action-dependent evaluation mechanism is described in detail in [25,27], and the presented release includes its implementation (source files `sample_field_evaluator.cpp` and `action_chain_graph`).

In particular, a new variable, `opp_forward`, is introduced, counting the number of non-goalie opponents in a sector centred on the agent and extending

to the points near the opponent's goal posts. The single evaluation metric of agent2d is invoked when there are no opponents in this sector, or when the ball is located within (or close to) the own half. Otherwise, the logic enters into a sequence of conditions (marked in the released code), identifying the "best" point out of several possible candidates offered by Voronoi diagrams. A Voronoi diagram is defined as the partitioning of a plane with n points into n convex polygons, so that each polygon contains exactly one point, while every point in the given polygon is closer to its central point than any other [13]. The best point is selected to be relatively close to the teammates' positions, and far from the opponents' positions. The distance between the identified best point and the future ball location, attainable by the action under consideration, is chosen as the evaluation result.

The action-dependent evaluation mechanism increased the goal difference against agent2d from 1.288 to 1.616, while not providing a notable improvements against the two other benchmarks, as it is applicable in attacking situations which are rare in these match-ups at this stage.

2.4   Gliders2d v1.4: Positioning

To make a better use of the new field evaluator, the positioning scheme of the players is adjusted by selecting the points according to suitably constructed Voronoi diagrams. For example, a Voronoi diagram may partition the field according to the positions of the opponent players; the candidate location points can be chosen among Voronoi vertices, as well as among the points located at intersections between Voronoi segments and specific lines, e.g., offside line, as illustrated in [23]. All the constrained conditions are evolvable.

The positioning based on Voronoi diagrams increased the goal difference against agent2d from 1.616 to 2.387, again maintaining the performance against the two other benchmarks.

2.5   Gliders2d v1.5: Formations

This step did not change any of the source code files—instead the formation files, specified in configurations such as `defense-formation.conf`, etc. were modified with fedit2. This approach, pioneered in the Simulation League by [2,3], is based on Constrained Delaunay Triangulation (CDT) [9]. For a set of points in a plane, a Delaunay triangulation achieves an outcome such that no point from the set is inside the circumcircle of any triangle. Essentially, CDT divides the soccer field into a set of triangles, based on the set of predefined ball locations, each of which is mapped to the positions of each player. Moreover, when the ball takes any position *within* a triangle, each player's position is dynamically adjusted during the runtime in a congruent way [2,3,30]. Overall, a formation defined via CDT is an ordered list of coordinates, and so, in terms of evolutionary computation, mutating and recombining such a list can be relatively easily automated and evaluated.

Figure 1 shows a CDT fragment; for example, the point 110, where the ball is located, defines the following intended positions for the players:

```
Ball -48.66 22.71
1 -50.72 6.07
2 -46.08 3.12
3 -47.6 10.53
4 -43.58 -3.75
5 -48.49 18.65
6 -44.3 13.29
7 -41.17 5.8
8 -40.32 17.03
9 -21.01 -17.44
10 -19.94 26.01
11 -22.62 5.8
```

The released changes in Gliders2d-1.5 formations are aimed at improving the defensive performance, placing the defenders and midfielders closer to the own goal. A notable performance gain was observed against all three benchmarks. The goal difference against agent2d increased from 2.387 to 3.210; against HELIOS2018: from -6.422 to -4.383; and against Gliders2013: from -1.039 to -0.344.
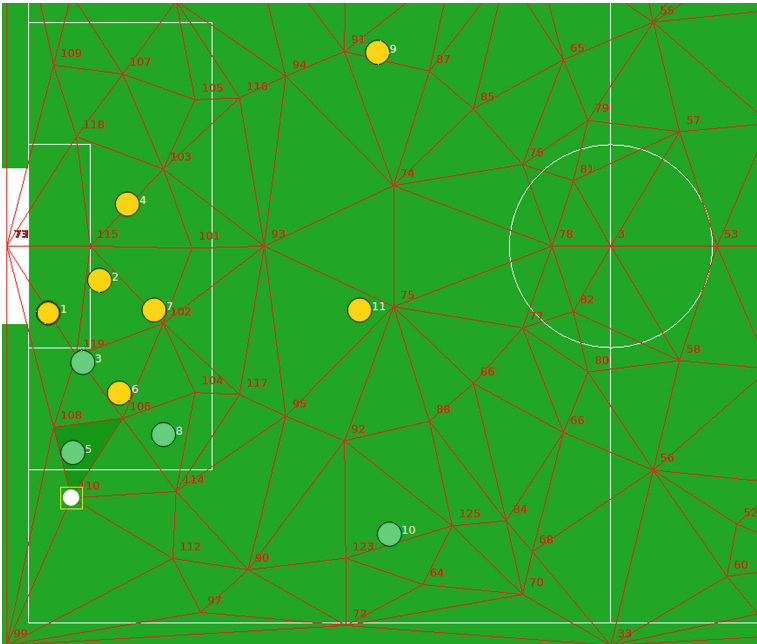
Fig. 1. Example of a Delaunay triangulation, used by `defense-formation.conf`, produced by fedit2. The triangle formed by points 106, 108 and 110 is highlighted. When the ball is located at 110, the players are supposed to be located in the shown positions.

Table 1. Performance evaluation for Gliders2d against agent2d, over ∼1000 games carried out for each version of Gliders2d against the opponent. The goal difference improves from zero to 4.2, while the average game score improves from (2.29:2.29) to (5.21:1.01).

Gliders2d	Points for	Points against	Goals scored	Goals conceded	Goal diff.	Std. error
v0.0 (agent2d)	1.384	1.414	2.287	2.289	−0.002	0.040
v1.1 (stamina)	1.345	1.468	2.254	2.290	−0.036	0.049
v1.2 (pressing)	2.161	0.691	2.642	1.355	1.288	0.051
v1.3 (evaluator)	2.252	0.607	2.997	1.381	1.616	0.063
v1.4 (positioning)	2.515	0.367	3.849	1.461	2.387	0.086
v1.5 (formations)	2.785	0.154	3.995	0.785	3.210	0.181
v1.6 (risky passes)	2.840	0.116	5.214	1.014	4.200	0.172

Table 2. Performance evaluation for Gliders2d against HELIOS2018, over ∼1000 games carried out for each version of Gliders2d against the opponent. The goal difference improves from −12.73 to −4.34, while the average game score improves from (0.12:12.85) to (0.26:4.60).

Gliders2d	Points for	Points against	Goals scored	Goals conceded	Goal diff.	Std. error
v0.0 (agent2d)	0.000	3.000	0.123	12.852	−12.729	0.514
v1.1 (stamina)	0.001	2.998	0.231	7.099	−6.868	0.276
v1.2 (pressing)	0.003	2.994	0.248	6.724	−6.476	0.140
v1.3 (evaluator)	0.004	2.992	0.269	6.821	−6.552	0.310
v1.4 (positioning)	0.002	2.996	0.298	6.720	−6.422	0.223
v1.5 (formations)	0.027	2.952	0.273	4.655	−4.383	0.197
v1.6 (risky passes)	0.024	2.961	0.260	4.600	−4.337	0.161

2.6   Gliders2d v1.6: Risky Passes

The final step of this release introduced `risk` level, expressed as the number of additional cycles "granted" to teammates receiving a pass, under a pressure from opponent players potentially intercepting the pass (`strict_check_pass_generator.cpp`). If `risk` level is set to zero, the default passing behaviour of agent2d is recovered. For positive values of `risk` the passes are considered as feasible even if an ideal opponent interceptor gets to the ball trajectory sooner than the intended recipient of the pass. The conditional statements include several new variables, used in mutating and recombining the conditions.

The addition of risky passes increased the goal difference against agent2d from 3.210 to 4.2; and against Gliders2013: from −0.344 to −0.212.

Table 3. Performance evaluation for Gliders2d against Gliders2013, over ∼1000 games carried out for each version of Gliders2d against the opponent. The goal difference improves from −5.48 to −0.21, while the average game score improves from (0.57:6.05) to (0.78:0.99).

Gliders2d	Points for	Points against	Goals scored	Goals conceded	Goal diff.	Std. error
v0.0 (agent2d)	0.022	2.968	0.569	6.052	−5.483	0.213
v1.1 (stamina)	0.183	2.730	0.596	3.280	−2.684	0.071
v1.2 (pressing)	0.539	2.230	0.613	1.760	−1.147	0.063
v1.3 (evaluator)	0.657	2.109	0.770	1.800	−1.030	0.067
v1.4 (positioning)	0.603	2.160	0.700	1.739	−1.039	0.077
v1.5 (formations)	1.039	1.607	0.700	1.044	−0.344	0.026
v1.6 (risky passes)	1.111	1.527	0.776	0.988	−0.212	0.038

3   Conclusions

In this paper, we described the first version of *Gliders2d*: a base code release for Gliders (based on agent2d-3.1.1). We trace six sequential changes aligned with six evolutionary steps. These steps improve the overall control of the pitch by increasing the players' mobility through several means: less conservative usage of the available stamina balance (v1.1); more intense pressing of opponents (v1.2); selecting more diversified actions (v1.3); positioning forwards in open areas (v1.4); positioning defenders closer to own goal (v1.5); and considering riskier passes (v1.6).

As has been argued in the past, the simulation leagues enable replicable and robust investigation of complex robotic systems [5,31]. We believe that the purpose of the RoboCup Soccer Simulation Leagues (both 2D and 3D) should be to simulate agents based on a futuristic robotic architecture which is not yet achievable in hardware. Aiming at such a general and abstract robot architecture may help to identify a standard for what humanoid robots may look like in 2050, the year of the RoboCup Millennium challenge. This is the reason for focussing, in this release, on the features which can also be used by simulated 3D, as well as robotic, teams competing in RoboCup, aiming at some of the most general questions: when to conserve energy (stamina), when to run (pressing), where to kick the ball (actions), where to be on the field (positioning in attack and defense), and when to take risks (passes). While the provided specific answers may or may not be widely acceptable, general reasoning along these lines may bring us closer to a new RoboCup Humanoid Simulation League (HSL). In HSL, the Simulated Humanoid should be defined in a standard and generalisable way, approaching human soccer-playing behavior [36], while the behavioural and tactical improvements can be evolved and/or adapted to this standardised architecture.

The released code: http://www.prokopenko.net/gliders2d.html.

The last presented version, Gliders2d-v1.6, is comparable to Gliders2013, achieving the average score of (0.78:0.99) against this benchmark, and outperforms agent2d-3.1.1 with the average score (5.21:1.01).

In tracing this evolutionary branch, we illustrated the methodology of human-based evolutionary computation, showing that even a small number of controlled steps can dramatically improve the overall team performance.

References

1. Akiyama, H., Nakashima, T.: HELIOS base: an open source package for the RoboCup soccer 2D simulation. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013. LNCS (LNAI), vol. 8371, pp. 528–535. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44468-9_46

2. Akiyama, H., Noda, I.: Multi-agent positioning mechanism in the dynamic environment. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007. LNCS (LNAI), vol. 5001, pp. 377–384. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68847-1_38

3. Akiyama, H., Shimora, H.: HELIOS 2010 team description. In: Ruiz-del-Solar, J., Chown, E., Plöger, P.G. (eds.) RoboCup 2010: Robot Soccer World Cup XIV. LNCS, vol. 6556. Springer, Heidelberg (2011)

4. Budden, D., Prokopenko, M.: Improved particle filtering for pseudo-uniform belief distributions in robot localisation. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013. LNCS (LNAI), vol. 8371, pp. 385–395. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44468-9_34

5. Budden, D.M., Wang, P., Obst, O., Prokopenko, M.: RoboCup simulation leagues: enabling replicable and robust investigation of complex robotic systems. IEEE Robot. Autom. Mag. **22**(3), 140–146 (2015)

6. Burkhard, H.D., Duhaut, D., Fujita, M., Lima, P., Murphy, R., Rojas, R.: The road to RoboCup 2050. IEEE Robot. Autom. Mag. **9**(2), 31–38 (2002)

7. Butler, M., Prokopenko, M., Howard, T.: Flexible synchronisation within RoboCup environment: a comparative analysis. In: Stone, P., Balch, T., Kraetzschmar, G. (eds.) RoboCup 2000. LNCS (LNAI), vol. 2019, pp. 119–128. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45324-5_10

8. Cheng, C.D., Kosorukoff, A.: Interactive one-max problem allows to compare the performance of interactive and human-based genetic algorithms. In: Deb, K. (ed.) GECCO 2004. LNCS, vol. 3102, pp. 983–993. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24854-5_98

9. Chew, L.P.: Constrained Delaunay triangulations. Algorithmica **4**(1–4), 97–108 (1989)

10. Cioppa, T.M., Lucas, T.W.: Efficient nearly orthogonal and space-filling latin hypercubes. Technometrics **49**(1), 45–55 (2007)

11. Cliff, O.M., Lizier, J.T., Wang, X.R., Wang, P., Obst, O., Prokopenko, M.: Towards quantifying interaction networks in a football match. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013. LNCS (LNAI), vol. 8371, pp. 1–12. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44468-9_1

12. Cliff, O.M., Lizier, J.T., Wang, X.R., Wang, P., Obst, O., Prokopenko, M.: Quantifying long-range interactions and coherent structure in multi-agent dynamics. Artif. Life **23**(1), 34–57 (2017)
13. Dylla, F., et al.: Approaching a formal soccer theory from the behavior specification in robotic soccer. In: Dabnicki, P., Baca, A. (eds.) Computers in Sport, pp. 161–186. Bioengineering, WIT Press, London (2008)
14. Gabel, T., Klöppner, P., Godehardt, E., Tharwat, A.: Communication in soccer simulation: on the use of wiretapping opponent teams. In: Holz, D., Genter, K., Saad, M., von Stryk, O. (eds.) RoboCup 2018. LNCS (LNAI), vol. 11374, pp. 3–15. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-27544-0_1
15. Kok, J.R., Vlassis, N., Groen, F.: UvA Trilearn 2003 team description. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) Proceedings CD RoboCup 2003. Springer, Padua (2003)
16. Kosorukoff, A.: Human based genetic algorithm. In: 2001 IEEE International Conference on Systems, Man, and Cybernetics, vol. 5, pp. 3464–3469. IEEE (2001)
17. Lizier, J.T., Prokopenko, M., Zomaya, A.Y.: Coherent information structure in complex computation. Theory Biosci. **131**, 193–203 (2012)
18. Nakashima, T., Akiyama, H., Suzuki, Y., Ohori, A., Fukushima, T.: HELIOS 2018: team description paper. In: RoboCup 2018 Symposium and Competitions: Team Description Papers, Montreal, Canada, July 2018 (2018)
19. Nehaniv, C., Polani, D., Olsson, L., Klyubin, A.: Evolutionary information-theoretic foundations of sensory ecology: channels of organism-specific meaningful information. In: Modeling Biology: Structures, Behaviour, Evolution, pp. 9–11 (2005)
20. Noda, I., Stone, P.: The RoboCup soccer server and CMUnited clients: implemented infrastructure for MAS research. Auton. Agents Multi-agent Syst. **7**(1–2), 101–120 (2003)
21. Prokopenko, M., Gerasimov, V., Tanev, I.: Evolving spatiotemporal coordination in a modular robotic system. In: Nolfi, S., et al. (eds.) SAB 2006. LNCS (LNAI), vol. 4095, pp. 558–569. Springer, Heidelberg (2006). https://doi.org/10.1007/11840541_46
22. Prokopenko, M., Gerasimov, V., Tanev, I.: Measuring spatiotemporal coordination in a modular robotic system. In: Rocha, L., Yaeger, L., Bedau, M., Floreano, D., Goldstone, R., Vespignani, A. (eds.) Artificial Life X: Proceedings of the 10th International Conference on the Simulation and Synthesis of Living Systems, Bloomington, IN, USA, pp. 185–191 (2006)
23. Prokopenko, M., Obst, O., Wang, P.: Gliders 2014: dynamic tactics with Voronoi diagrams. In: RoboCup 2014 Symposium and Competitions: Team Description Papers, Joao Pessoa, Brazil, July 2014 (2014)
24. Prokopenko, M., Obst, O., Wang, P., Budden, D., Cliff, O.M.: Gliders 2013: tactical analysis with information dynamics. In: RoboCup 2013 Symposium and Competitions: Team Description Papers, Eindhoven, The Netherlands, June 2013 (2013)
25. Prokopenko, M., Wang, P., Obst, O., Jaurgeui, V.: Gliders 2016: integrating multi-agent approaches to tactical diversity. In: RoboCup 2016 Symposium and Competitions: Team Description Papers, Leipzig, Germany, July 2016 (2016)
26. Prokopenko, M.: Guided Self-organization: Inception. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-53734-9
27. Prokopenko, M., Obst, O., Wang, P., Held, J.: Gliders 2012: tactics with action-dependent evaluation functions. In: RoboCup 2012 Symposium and Competitions: Team Description Papers, Mexico City, Mexico, June 2012 (2012)

28. 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. LNCS (LNAI), vol. 2752, pp. 367–374. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45135-8_32

29. Prokopenko, M., Wang, P.: Evaluating team performance at the edge of chaos. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 89–101. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-25940-4_8

30. Prokopenko, M., Wang, P.: Disruptive innovations in RoboCup 2D soccer simulation league: from Cyberoos'98 to Gliders2016. In: Behnke, S., Sheh, R., Sarıel, S., Lee, D.D. (eds.) RoboCup 2016. LNCS (LNAI), vol. 9776, pp. 529–541. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68792-6_44

31. Prokopenko, M., Wang, P., Marian, S., Bai, A., Li, X., Chen, X.: RoboCup 2D soccer simulation league: evaluation challenges. In: Akiyama, H., Obst, O., Sammut, C., Tonidandel, F. (eds.) RoboCup 2017. LNCS (LNAI), vol. 11175, pp. 325–337. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00308-1_27

32. Prokopenko, M., Wang, P., Obst, O.: Gliders 2015: opponent avoidance with bio-inspired flocking behaviour. In: RoboCup 2015 Symposium and Competitions: Team Description Papers, Hefei, China, July 2015 (2015)

33. Reis, L.P., Lau, N., Oliveira, E.C.: Situation based strategic positioning for coordinating a team of homogeneous agents. In: Hannebauer, M., Wendler, J., Pagello, E. (eds.) BRSDMAS 2000. LNCS (LNAI), vol. 2103, pp. 175–197. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44568-4_11

34. Riley, P., Stone, P., Veloso, M.: Layered disclosure: revealing agents' internals. In: Castelfranchi, C., Lespérance, Y. (eds.) ATAL 2000. LNCS (LNAI), vol. 1986, pp. 61–72. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44631-1_5

35. Stone, P., et al.: Overview of Robocup-2000. In: Stone, P., Balch, T., Kraetzschmar, G. (eds.) RoboCup 2000. LNCS (LNAI), vol. 2019, pp. 1–29. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45324-5_1

36. Stone, P., Quinlan, M., Hester, T.: Can robots play soccer? In: Richards, T. (ed.) Soccer and Philosophy: Beautiful Thoughts on the Beautiful Game, Popular Culture and Philosophy, vol. 51, pp. 75–88. Open Court Publishing Company, Chicago (2010)

37. Stone, P., Riley, P., Veloso, M.: The CMUnited-99 champion simulator team. In: Veloso, M., Pagello, E., Kitano, H. (eds.) RoboCup 1999. LNCS (LNAI), vol. 1856, pp. 35–48. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45327-X_2

38. Stone, P., Riley, P., Veloso, M.: Defining and using ideal teammate and opponent models. In: Proceedings of the 12th Annual Conference on Innovative Applications of Artificial Intelligence (2000)

39. Tanev, I., Yuta, K.: Epigenetic programming: genetic programming incorporating epigenetic learning through modification of histones. Inf. Sci. **178**(23), 4469–4481 (2008)

40. Tavafi, A., Nozari, N., Vatani, R., Yousefi, M.R., Rahmatinia, S., Pirdir, P.: MarliK 2012 soccer 2D simulation team description paper. In: RoboCup 2012 Symposium and Competitions: Team Description Papers, Mexico City, Mexico, June 2012 (2012)

41. Zuparic, M., Jauregui, V., Prokopenko, M., Yue, Y.: Quantifying the impact of communication on performance in multi-agent teams. Artif. Life Robot. **22**(3), 357–373 (2017)