How to run a successful game-based AI competition

Julian Togelius

Abstract— **Game-based competitions are commonly used within the Computational Intelligence (CI) and Artificial Intelligence (AI) in games community to benchmark algorithms and to attract new researchers. While many competitions have been organised based on different games, the success of these competitions is highly varied. This short paper is a self-help paper for competition organisers and aspiring competition organisers. After analysing the fate of a number of recent competitions, some factors likely to contribute to the success or failure of a competition are laid out, and a set of concrete recommendations is offered. There is also a discussion of how to write up game-based AI competitions and what we can ultimately learn from them.**

Keywords: Competitions, benchmarking

I. Introduction

In the research field of artificial and computational intelligence in games, game-based competitions have come to assume a central place. Competitions are held each year in conjunction with the two main conferences IEEE Computational Intelligence and Games (CIG) and AAAI Artificial Intelligence in Interactive Digital Entertainment (AIIDE), as well as at major conferences dedicated to evolutionary computation, game design, machine learning etc. These competitions are based on games (computer games as well as digital versions of board games), and submissions to these competitions are in the form of either some CI/AI based software or the output of this software. The winner is the submission that best solves some problem posed by the game; often, the problem is to play the game as well as possible, but it could also be to generate fun levels for the game, accurately analyse player data or imitate human players.

In August 2013, I held a tutorial at CIG where I discussed what game-based AI competitions are, what we can learn from them and how we can make them better. The tutorial was based on my experience of running several such competitions and of being competitions chair for conferences. In the ensuing vigorous discussion it was suggested that the tutorial be written up and published. Thus this short paper.

II. Why do it?

First of all, let us motivate why we organise and participate in game-based AI competitions at all. One of the most important reasons, and probably the most academically respectable reason, is that competitions provide fair, transparent and reusable means of benchmarking algorithms. For example, variants of Pac-Man had been used several times in the past to test or demonstrate AI algorithms, such as genetic programming [1]. However, since different implementations of the game and different experimental conditions were used, there was no meaningful way to compare the results. After the introduction of the Ms. Pac-Man competition, however, researchers seeking to use some version of Pac-Man for their research have had access to (and generally used) software and experimental conditions as specified by the competition [2]. Anyone can check the league tables on the competition website, and compare their own algorithm with the results published there. Even after the competition has ceased to run officially, researchers have continued to use the competition software and rules to benchmark their solutions.

A strongly related benefit for researchers is that there is a benchmark available at all. Many researchers wishing to do research on problems relevant for games have in the past faced the problem of having to develop the game (or an API for an existing game) themselves. The existence of games already equipped with interfaces and in other ways prepared for AI research (such as being able to run in "headless" mode without graphics and with faster execution) allows researchers to focus on research. This has during the last few years allowed many researchers from other fields to enter the AI/CI in games field. It has also allowed many researchers already in the field to get more research done faster.

Game-based AI competitions are useful not only for research, but also for teaching. Universities now organise their own local competitions based on the software and rules used in the global competitions, and base course assignments on the competition software. The prospect of being able to work with a "real" game and benchmark the performance of their implementations against each other and against leading researchers is often appreciated by students [3].

The final reason for constructing these competitions and their associated benchmarks might seem trivial but really is not. It is that it makes it easier for people outside of our research field to understand and engage with what we are doing. It is not easy to explain to journalists, administrators or even academics from other departments what research in CI and AI is. Common testbeds such as pole balancing, TSP and machine scheduling often come across as incomprehensible and/or pointless. Everyone understands Pac-Man.

III. Which competitions are there?

In the rest of the paper, I will use concrete examples of game-based AI competitions in my argumentation. It will therefore be useful to start with talking about some of them. This paper is not a survey of game-based AI competitions; such a survey ought to be written, but would result in a considerably longer paper. Instead, this section is a selective enumeration of competitions (most of them associated with the CIG and AIIDE conferences) ordered by domain.

The author is with the Center for Computer Games Research, IT University of Copenhagen, Rued Langgaards Vej 7, 2300 Copenhagen, Denmark. e-mail: julian@togelius.com

The earliest game-based AI competitions were based on classical board games; Chess has long been studied in AI and cognitive science research, and a Computer Chess Olympiad has been running since 1974 [4]. Checkers has also been the subject of much research, at least until it was solved in 2007 [5]. Recently, much effort has gone into playing Go, motivating the development of the Monte Carlo Tree Search algorithm [6], [7]. In the CIG community, there has been a competition in Othello-playing at some conferences; competitors submitted board evaluation functions which would work with a shallow MiniMax search [8].

Strategy games have in common with classic board games the control over multiple units and the focus on long-term strategy. Surprisingly few competitions have been based on turn-based strategy games. One of them is Planet Wars, a mechanically simple game that was the basis of a Google competition. There was also briefly a competition based on the nuclear strategy game DefCon in 2009. Instead, popular competitions in the CI/AI community have been based on real-time strategy games (RTS). ORTS (Open RTS) was developed by Buro and used for a competition that ran 2003–2009 [9]. In recent years, focus has shifted to the StarCraft competitions. These are based on the now-classic Blizzard RTS game together with a third-party interface called BroodWar API; there is an annual competition at CIG and another at AIIDE, and both are currently very active [10].

Car racing games could be positioned at the other end from classic board games on several dimensions of the games continuum; while they feature elements of tactics (such as overtaking) and occasionally strategy (such as damage and fuel management), the greatest emphasis is on the continuous control problem of driving the car so that lap times are minimised. The first simulated car racing competition, in 2007, was based on an idiosyncratic two-dimensional racing game with simple physics and attracted a large number of submissions [11]. This competition morphed into the Simulated Car Racing Championship, based on the more complex game TORCS, which has been running since 2008 [12].

Several competitions have been based on classic arcade games or console games from the 80s. The Ms. Pac-Man Screen Capture competition, where agents interfaced to an emulated version of the original Ms. Pac-Man game, ran from 2007 to 2011 [2]. A Java-based Ms. Pac-Man competition ran in 2011 and 2012, where the core mechanics and levels of the game have been replicated in a framework to which the controllers can interface directly [13]. The Mario AI Championship ran in various versions from 2009 to 2012 [14], [15]. This competition had four tracks. In one track submitted controllers competed on playing unseen levels as well as possible, in another on learning to play specific levels, in a third on playing in a human-like manner and in the fourth on generating levels. Competitions have also briefly existed based on the 2D space shooter Xpilot [16], and on the cooperative platformer Geometry Friends [17].

Another category of games is first-person shooter (FPS) games. The Unreal Tournament 2004 is perhaps the only commercial FPS for which a third-party interface for agent control exists, and as a result it has been used extensively for competitions. The 2k BotPrize was a Turing-test-like competition for developing the bot that played in the most human-like manner in a multiplayer game; it ran from 2008 until 2013, when a bot finally managed to convince more than half of the judges that it was a human player [18]. The same game has also been used for a "deathmatch" competition in 2009, where the aim was simply to be the last bot standing.

Finally, there are a few competitions that are not based on existing games or game genres. One is the physical travelling salesman problem (PTSP) competition, which can be seen as the hybrid of the traditional TSP problem and a racing game. Controllers compete for reaching a number of checkpoints in minimum time, while taking turning radius, momentum etc into account [19]. Another is Cellz, which is a game where the task of the controller is to control multiple "cellz", which move about in a 2D environment, eat smaller cellz and multiply [20]. A more famous example is the general game playing competition (GGP), which tasks submitted controllers with learning to play an arbitrary unseen board-game-like game, after being provided with the rules [21].

Competitions are also held in other fields, including reinforcement learning, evolutionary computation and planning, and much could be learnt from how those are organised. An interesting competition to learn from is the HUMIEs, focusing on human-competitive results from evolutionary computation in any domain, which has several times been won by game-playing agents [22]. Much could probably also be learnt from how human-human game-playing competitions (eSports) are organised.

IV. Success and failure

Not all game-based AI competitions are equal: some have more impact than others. A competition may get no entrants, run only once, stagnate, or evolve and keep being relevant.

Get no entrants. Both the Cellz competition and the Xpilot competition ran once, and received no submissions. The first Geometry Friends competition had only one entrant. This is of course the least desired fate for a competition, so it is important to analyse the reasons for the lack of competitors. In the case of Cellz, a probable cause is that the game was not similar enough to any existing computer game, and was not really playable by humans. (In the later stages of the game, a human would have to control dozens of cellz simultaneously in real-time.) Therefore, it is hard to compare the performance of a controller to how a human would have played, and also hard to intuit strategies. Additionally, perhaps few people perceived Cellz as an interesting or important problem to work on. Xpilot on the other hand is a fairly well-known game with strong similarities to classic arcade games such as *Asteroids* and *Defender*. However, it was rather complicated to get started with developing an entry for the competition. There was no interface for any high-level language in common use, such as Java; instead there was a Scheme interface. The Geometry Friends competition

software did not include a sample controller, which probably dissuaded many potential competitors.

Run only once. The first version of the Simulated Car Racing competition ran only one year, before being replaced with a competition with the same name and partly the same organisers but based on a very different racing game. The Unreal 2k4 Deathmatch competition ran only once, attracting submissions from three competitors, and was not followed by another such competition. A key reason for competitions being discontinued is almost certainly lack of time and energy on part of the organisers. Running competitions is hard work, and sometimes thankless. If the number of submissions is low, it is tempting to not re-run the competition. Another possible reason is that the problem appears solved or close to solved in its current form. In the first version of the simulated car racing competition, the top competitors reached very similar scores, and seemed to be playing the game near-optimally. Some sort of change was needed to make the problem more challenging; thus the radical redesign.

Stagnate. The new version of the Simulated Car Racing competition kept attracting better and better submissions for two or three years, but then it stagnated: in 2012 and 2013, none of the new submissions beat the best submission from 2011. The controllers neither performed better nor were they more sophisticated. There were also fewer submissions in these latter years. As a result, the competition is not running in 2014 and it is unclear whether it will run again. Something similar happened for the Gameplay track of the Mario AI competition but in a shorter time span: 2009 saw two competition events and 2010 saw three; in between each event, the framework evolved. The sophistication of the submissions increased throughout 2010, even though there were fewer of them. In 2011 the Gameplay track saw very few entrants, and in 2012 there were too few entrants to run.

One important possible reason for stagnation is that no further progress seems attainable, either because the problem is perceived as "solved", or because the submitted controllers seem so sophisticated that it would be impossible to improve on them in time for the deadline. This is a problem particularly for submissions by teams of students. Other important possible reasons are that the organisers of the competition fail to sufficiently advertise the competition, make results of previous years available, and make source code of previous years' competitors available. It is worth noting that even if a competition stagnates, the competition software can go on to be widely used for research papers that report experiments done using that software but not submitted to the competition. This is the case for both the Simulated Car Racing competition and the Mario AI competition.

Keep evolving and being relevant. The 2k BotPrize managed to get numerous submissions during the six years it was running. More importantly, these submission generally became better at fooling the judges, until an entry (as mentioned above) finally managed to win the prize in 2013. In its lifetime, it saw at least one major overhaul of its rules, and there was continual improvement of the Pogamut framework it builds on. This is an example of a competition that managed to keep evolving and stayed relevant until the very end. Another good example here is the StarCraft competitions. These competitions have run since 2010, and the submissions to the competitions have been getting better and more sophisticated each year. This is somewhat surprising, as the complexity of the best controllers is such that it would deter many students and other more casual participants from participating. But apparently, many of the best competitors in that competition are either researchers on long-term or permanent contracts, or developers outside of academia working in their spare time.

Competitions that keep being relevant have at least two things in common: (1) the organisers keep investing time in e.g. improving the usability of the software, bug-fixing both rules and software, keeping results tables up to date and promoting the competition, and (2) the underlying problem is not perceived as solved, but it is seen as realistic to make short-term advancement towards solving the problem.

V. How to succeed

The following advice is based on the fates of existing competitions discussed above, and on my own experience of running competitions. It can be thought of as "the ten habits of highly successful competition organisers". The list is ordered by how well-corroborated these habits are by existing competitions. I regard the first five as purely beneficial habits that you should follow, whereas the five latter are positive in most cases though there might be situations where they can or should not be followed.

Choose a fun game. A game that humans play out of their own free will is more interesting to write AI for. It also helps if the game is famous so that many have already played it.

Be transparent and reliable. Lay down the rules for your competition early on. Be clear about how scoring will happen and whether you have an open-source requirement for your submissions. Stick to your rules. Keep your website updated.

Be platform-agnostic. It is often erroneously assumed that academic AI researchers are willing to spend time learning any particular programming language or build system that the competition might require, or go out and buy the hardware or software it requires. A good competition builds on software that runs on any platform, and can easily interface with several major programming languages in use in academia (such as Java, c and Python).

Be persistent. Running a competition once is of limited value. Running a competition ten times over the space of five years is of much higher value, and does not require more initial effort, but does require much more persistence. Repeating a competition allows competitors to improve and tune their submissions, and allows people who see earlier competition events to get inspired and participate in later events. Therefore it is worth running again even if you get no submissions the first time. Establish a clear governance structure, and make sure there is someone there to take over running the competition when you or your students graduate.

Run a discussion group. Even though you keep your website updated, if your competition is popular there will be more questions than you can handle, especially regarding technical details. So start a discussion group (e.g. using Google Groups) and invite all competitors and other users of the software to do technical support for each other.

Offer money. Not only are researchers like other humans motivated by money, but the fact that someone is prepared to pay money to the winner makes the competition seem more serious.

Avoid network communication. It should be possible to link directly to the competition software, as overhead for communication via IP can slow down the software substantially.

Make it possible to speed up the game. Most forms of reinforcement learning, in particular evolutionary computation, require many thousands of trials. If you want your competitors to be able to use learning algorithms effectively, make sure your game can be sped up by several orders of magnitude. This unfortunately causes problems for using existing closed-source commercial games as a basis for a competition, as they can rarely be sped up significantly.

Make it really easy to get started. This is surprisingly often overlooked. An average postgraduate student or faculty AI researcher, who might not be a stellar programmer, should be able to have a proof-of-concept submission up and running within hours. Instructions should be less than a page.

Keep everything open source. Competition entries should be open source so as to allow competitors to learn from each other, and so as to prevent cheating. It might be best to release the source code of entrants only after each competition event, so as to avoid copying techniques during the run-up to an event. The competition software itself should be open source if possible, as it is the only fully complete specification of the competition rules. This is a question of fairness: closed-source software can always be decompiled by anyone with sufficient time and resources. (The parameters, levels or datasets used for the final test can be kept secret until the competition event.) Note that it is not enough to stipulate the open access to source code – all the source code should be available on the website.

VI. Writing up competitions

Peer-reviewed publications are the universal currency of academia. Publication output and citation numbers factor heavily in hirings and promotions. Running competitions takes time and effort, and can usually not be discounted as "teaching". If we agree that competitions are valuable for the research community, we need to incentivise running them by making it possible to publish on the running and results of competitions. For the same reason, we need to incentivise taking part in competitions, especially for those who put serious efforts into their competition entries. It is also important for future researchers, and for future competitions organisers, that results of competitions are available in a permanent and citable form. (Competition web pages fail on both accounts.) Looking at the competitions used as examples in this paper, a number of different approaches to publishing their organisation and results have been taken. They can be grouped into four publication models: *solipsist*, *dictator*, *anarchist* and *big hippie family*. I will here comment on the pros and cons of each.

Solipsist. The solipsist model is that nothing is published, as if the outside world did not exist. This leaves no permanent record of the competition. Such a solution essentially negates the value of organising the competition, as nobody will be able to build on it. It has no advantages beyond simplicity.

Dictator. In the (benevolent) dictator model, the organiser(s) write a paper of their own, and do not include the competitors as authors, even though the paper includes results and might include descriptions of the competitors' entries. The big advantage of this model is authorial control: the organiser can write a stylistically, conceptually and factually coherent paper and get it submitted on time. Another potential advantage is that the paper has few authors – for some people (though certainly not all) this is important. The main disadvantage is that competitors, who are crucial for the success of the competition, might not get sufficient credit as they are not included as authors. (This might be perfectly fine, e.g. with competitors from outside academia that are not interested in citations, or when competitors are anyway planning to publish their contributions separately.) Another disadvantage is that the organiser might not know the submissions well enough to write good descriptions of them. This model was used by the Pac-Man vs Ghost Teams competition [13] and the 2k BotPrize [18] among others.

Anarchist. One way to make sure that everybody gets credit for their work is to make an arrangement where the organisers write a short paper about the organisation of the competition but not about the competition entries, and that each of the competitors submits a paper about their own entry. In theory this is great, as every part of the competition is described by someone who worked on it, and everybody gets due credit. The main disadvantage is that it likely won't work. Students graduate and faculty are perennially overcommitted, so the papers might not get submitted. Some competition entries might not be substantial or novel enough to publish a paper on in isolation, only as part of the comparison that the competition provides. Another big disadvantage is that no single publication will contain the definitive record of the competition, and be able to make a really informed comparison of the entries. This model was used by for example the Gameplay Track of the Mario AI Championship, where papers by the organisers [14], [15] were followed by papers by some (but far from all) competitors, e.g. Bojarski and Congdon [23].

Big hippie family. In the big hippie family model, the competition organisers organise the writing of a joint paper including both organisers and competitors as authors. This paper contains discussions of all aspects of the competition, and short descriptions of the competitors. Perhaps the greatest advantage of this model is that it results in a single paper which will become the definitive point of reference for the competition. Naturally, this paper will also be the

paper to cite when discussing the competition or its underlying benchmark game software. It also allows everybody to write about what they know most about, resulting in more initiated technical sections. If done right, this allows deep comparisons between entries.

One way to do write such a paper is for the competition organisers to give detailed instructions to competitors for how to describe their entries. This could be in the form of a list of questions that must be answered in the section describing the entry, and a strict page limit and deadline. In the final paper, the first third could be devoted to the organisation of the competition, about half to descriptions of the competition entries, and the rest to results. The author order could be that organisers go first, followed by competitors in order of descending results ranking. This model was used for the Simulated Car Racing Championship [11], [12] and for the Level generation track of the Mario AI Championship [24].

There are also disadvantages to the big hippie family model. The main disadvantage is probably the substantial editorial effort required to assemble the paper, including chasing competitors that do not contribute their descriptions in time, and editing the descriptions. Given the page limits of conferences and journals, there might not be space to describe the contributions in sufficient level of detail (though this could be solved with online appendixes and extended versions on repositories such as arXiv). Also, for those judged by committees that count fractional citations, spreading authorship so thinly might not be ideal.

The publication models described above are all geared towards publication in existing conferences and journals. However, it would be interesting to experiment with new forms of publication specially geared to reporting competitions. One such model could be a bundle of short or very short articles, with the introductory article written by the competition organisers and every competing team writing their own article, published simultaneously and back-to-back. Such a bundle would be reviewed and accepted/rejected as a single entity, but it would be composed of a number of smaller entities for authorship and citation purposes.

VII. Learning from competitions

What can we learn from a game-based AI competition? This question is harder to answer than it seems. Let us consider some alternatives:

That the problem can be solved algorithmically? Most competition software comes with example code that can solve the given problem to some extent, e.g. drive a lap around a track in TORCS, create a playable Mario level or kill a few enemies in Unreal Tournament. However, the winning competitor might be able to solve the problem better. After much work, some competition submissions drive faster than a human, create levels that humans find interesting or trick human judges that they are human. These are impressive "existence proofs" of AI solutions to game problems.

How best to solve the problem? We cannot answer this question with a high degree of certainty based on the results of a competition. Just because e.g. neuroevolution is the basis for the best-performing Pac-Man controller, this does not mean that neuroevolution is the best way to play the game – not even out of these methods that have been tried. Tomorrow, someone might submit a controller that performs better based on a new, exotic method, or on some method that has already been tried but where the previous attempt got some detail wrong. However, if a competition has many entrants it might be possible to say something generic about the relative performance of methods. For example, MCTS seems to be a better method than MiniMax for playing Go.

How not to solve the problem? No. The fact that all controllers based on neural networks have done comparatively badly in the Gameplay track of the Mario AI competition does not prove the hypothesis that neural networks are unsuited to this kind of problem. Maybe neural nets work excellently if you just change the input representation, or get rid of the bugs in the previous submissions. At best, the results corroborate the unsuitability hypothesis.

How to solve a more general version of the problem? Just because a given methods works very well for e.g. solving instances of the physical travelling salesman problem, this is no guarantee that the same method will work well for other similar problems. It does not even guarantee that they work well for the same problem after adding noise or multiple players, or for radically different instances of the problem.

Interestingly, this echoes the situation with competitions in non-game fields: they mostly provide existence proofs, that something can be done and how it concretely can be done in one case. The DARPA Grand Challenge showed us that it was possible for vehicles to drive a long distance autonomously, but the specific solution that won was not necessarily the best way to control an autonomous vehicle. In the NetFlix Challenge, it was shown that it was possible to produce 10% more accurate movie recommendations, but the winning entry was not adapted into a production environment and it could be argued that it teaches us little about which machine learning algorithms work best.

One thing we do learn from competitions is how to run competitions. In recent years we have seen an increase in sophistication of competitions, and I hope that this paper will help to codify the knowledge we are gaining in this process.

VIII. Evolving competitions

Below, I will outline a few steps we as a community could take to ensure the continued popularity and increased scientific relevance of game-based AI competitions.

We should work towards ensuring that competition benchmarks are used in more papers, so as to improve the comparability of results in our field. When we review conference and journal papers, we need to start demanding that new algorithms and methods are tested on the same benchmarks as are used for the competitions, unless there is a very good reason not to do so. For example, if in a paper the authors test an algorithm on a "Pac-Man-like game" or a "generic racing game", demand that in the revision, the algorithm is tested on the competition version of Ms. Pac-Man or TORCS, so that the results can be fairly compared with other methods.

In order to ensure that competitions become more permanent and reliable, we could consider distributing the responsibility for running them, so that they become community efforts. This would mean that "ownership" of the competitions is transferred from individuals to the organisations behind them, such as the IEEE CIS Games Technical Committee.

As game design evolves, we need to evolve our offering of competitions to make use of the possibilities afforded by new game genres. We have not yet seen any competitions based on new game genres such as physics-based puzzles (e.g. *Angry Birds*, *Tower of Goo*) or Tower Defence (e.g. *Plants vs. Zombies*, *Kingdom Rush*). This is a question both about keeping the competition menu fresh and interesting, and about exploiting the exploration of human cognitive capacities that good game designers engage in. *Carpe ludem!*

We should try to counteract the tendency of competition participants to overfit their solution to particular problems and problem parameters, so as to make the results of competitions more generally valid. This could be done through parameterising the competitions, and testing the submissions on multiple random parameter sets. Almost any game could easily be parameterised, either through numerical parameters such as gravity, tire grip, speed or amount of hit points, or through generating level geometry or other non-numerical structures. For example, the PTSP competition was made more generic through evolving problem instances [25].

In the extreme, when parameterising all aspects of the game including its rules, each parameterisation is a completely new game. The idea of general video game playing [26] is to apply this idea to games with graphical logic; there is ongoing work to develop a general video game playing competition based on a special-purpose language called the Video Game Description Language [27], [28]. This is the road less travelled, and the shape of things to come.

REFERENCES

[1] J. R. Koza, "Genetic programming: on the programming of computers by means of natural selection (complex adaptive systems)," Cambridge, MA, 1992.

[2] S. M. Lucas, "Ms pac-man competition," *ACM SIGEVOlution*, vol. 2, no. 4, pp. 37–38, 2007.

[3] J. Carpio Cañada, T. Mateo Sanguino, J. Merelo Guervós, and V. Rivas Santos, "Open classroom: enhancing student achievement on artificial intelligence through an international online competition," *Journal of Computer Assisted Learning*, 2014.

[4] M. Newborn, *Computer chess*. John Wiley and Sons Ltd., 2003.

[5] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen, "Checkers is solved," *science*, vol. 317, no. 5844, pp. 1518–1522, 2007.

[6] C.-S. Lee, M.-H. Wang, G. Chaslot, J.-B. Hoock, A. Rimmel, O. Teytaud, S.-R. Tsai, S.-C. Hsu, and T.-P. Hong, "The computational intelligence of mogo revealed in taiwan's computer go tournaments," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 1, no. 1, pp. 73–89, 2009.

[7] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 4, no. 1, pp. 1–43, 2012.

[8] S. M. Lucas and T. P. Runarsson, "Temporal difference learning versus co-evolution for acquiring othello position evaluation," in *Computational Intelligence and Games, 2006 IEEE Symposium on*. IEEE, 2006, pp. 52–59.

[9] M. Buro, "Orts: A hack-free rts game environment," in *Computers and Games*. Springer, 2003, pp. 280–291.

[10] S. Ontanón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, "A survey of real-time strategy game ai research and competition in starcraft," *IEEE Transactions on Computational Intelligence and AI In Games*, 2013.

[11] J. Togelius, S. M. Lucas, H. Duc Thang, J. M. Garibaldi, T. Nakashima, C. H. Tan, I. Elhanany, S. Berant, P. Hingston, R. M. MacCallum, T. Haferlach, A. Gowrisankar, and P. Burrow, "The 2007 IEEE CEC Simulated Car Racing Competition," *Genetic Programming and Evolvable Machines*, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10710-008-9063-0

[12] D. Loiacono, P. L. Lanzi, J. Togelius, E. Onieva, D. A. Pelta, M. V. Butz, T. D. Lonneker, L. Cardamone, D. Perez, Y. Sáez, *et al.*, "The 2009 simulated car racing championship," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 2, no. 2, pp. 131–147, 2010.

[13] P. Rohlfshagen and S. M. Lucas, "Ms pac-man versus ghost team cec 2011 competition," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 70–77.

[14] J. Togelius, S. Karakovskiy, and R. Baumgarten, "The 2009 mario ai competition," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010, pp. 1–8.

[15] J. Togelius, N. Shaker, S. Karakovskiy, and G. N. Yannakakis, "The mario ai championship 2009-2012," *AI Magazine*, vol. 34, no. 3, pp. 89–92, 2013.

[16] G. B. Parker and M. Parker, "Evolving parameters for xpilot combat agents," in *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*. IEEE, 2007, pp. 238–243.

[17] J. B. G. Rocha, "Geometry friends," Master's thesis, IST, University of Lisbon, 2009.

[18] P. Hingston, "A Turing test for computer game bots," *IEEE Transactions on Computational Intelligence and AI In Games*, vol. 1, no. 3, 2009. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5247069

[19] D. Perez, P. Rohlfshagen, and S. M. Lucas, "The physical travelling salesman problem: Wcci 2012 competition," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012, pp. 1–8.

[20] S. M. Lucas, "Cellz: a simple dynamic game for testing evolutionary algorithms," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1. IEEE, 2004, pp. 1007–1014.

[21] M. Genesereth, N. Love, and B. Pell, "General game playing: Overview of the aaai competition," *AI magazine*, vol. 26, no. 2, p. 62, 2005.

[22] K. Kannappan, L. Spector, M. Sipper, T. Helmuth, W. Lacava, J. Wisdom, and O. Bernstein, "Analyzing a decade of human-competitive ("humie") winners: What can we learn?" in *Genetic Programming Theory and Practice (GPTP)*, 2014.

[23] S. Bojarski and C. B. Congdon, "Realm: A rule-based evolutionary computation agent that learns to play mario," in *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*. IEEE, 2010, pp. 83–90.

[24] N. Shaker, J. Togelius, G. N. Yannakakis, B. Weber, T. Shimizu, T. Hashiyama, N. Sorenson, P. Pasquier, P. Mawhorter, G. Takahashi, *et al.*, "The 2010 mario ai championship: Level generation track," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 3, no. 4, pp. 332–347, 2011.

[25] D. Perez, J. Togelius, S. Samothrakis, P. Rohlfshagen, and S. Lucas, "Automated map generation for the physical travelling salesman problem," *IEEE Transactions on Evolutionary Computation*, 2013.

[26] J. Levine, C. B. Congdon, M. Ebner, G. Kendall, S. M. Lucas, R. Miikkulainen, T. Schaul, and T. Thompson, "General video game playing," in *Artificial and Computational Intelligence in Games*, S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, Eds. Saarbrücken/Wadern: Dagstuhl Publishing, 2013.

[27] M. Ebner, J. Levine, S. Lucas, T. Schaul, T. Thompson, and J. Togelius, "Towards a Video Game Description Language," in *Artificial and Computational Intelligence in Games*, S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, Eds. Saarbrücken/Wadern: Dagstuhl Publishing, 2013.

[28] T. Schaul, "A video game description language for model-based or interactive learning," in *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013, pp. 1–8.