Report from Dagstuhl Seminar 19511

# Artificial and Computational Intelligence in Games: Revolutions in Computational Game AI

**Edited by**
**Jialin Liu[1], Tom Schaul[2], Pieter Spronck[3], and Julian Togelius[4]**

1 **Southern Univ. of Science and Technology – Shenzen, CN,** `liujl@sustech.edu.cn`
2 **Google DeepMind – London, GB,** `schaul@google.com`
3 **Tilburg University, NL,** `p.spronck@gmail.com`
4 **New York University, US,** `julian@togelius.com`

## Abstract

The 2016 success of Google DeepMind's AlphaGo, which defeated the Go world champion, and its follow-up program AlphaZero, has sparked a renewed interest of the general public in computational game playing. Moreover, game AI researchers build upon these results to construct stronger game AI implementations. While there is high enthusiasm for the rapid advances to the state-of-the-art in game AI, most researchers realize that they do not suffice to solve many of the challenges in game AI which have been recognized for decades. The Dagstuhl Seminar 19511 "Artificial and Computational Intelligence in Games: Revolutions in Computational Game AI" seminar was aimed at getting a clear view on the unsolved problems in game AI, determining which problems remain outside the reach of the state-of-the-art, and coming up with novel approaches to game AI construction to deal with these unsolved problems. This report documents the program and its outcomes.

**Seminar** December 15–20, 2019 – http://www.dagstuhl.de/19511
**2012 ACM Subject Classification** Computing methodologies → Artificial intelligence, Computing methodologies → Machine learning, Software and its engineering → Software organization and properties, Software and its engineering → Contextual software domains, Software and its engineering → Virtual worlds software, Software and its engineering → Interactive games
**Keywords and phrases** artificial intelligence, computational intelligence, game theory, games, optimization
**Digital Object Identifier** 10.4230/DagRep.9.12.67

## 1 Executive Summary

*Pieter Spronck (Tilburg University, NL)*
*Jialin Liu (Southern Univ. of Science and Technology – Shenzen, CN)*
*Tom Schaul (Google DeepMind – London, GB)*
*Julian Togelius (New York University, US)*

The past decade has seen a rapid advent of new technologies in computational game playing. For a long time, artificial intelligence (AI) for 2-player deterministic board games was mainly implemented using tree search, employing the minimax algorithm with alpha-beta pruning, enhanced with some improvements which were often aimed at particular games. This approach worked well for most traditional games, but some games proved to be notoriously hard to tackle in this way. The textbook example of games for which regular tree search is inadequate, is Go.

Ten years ago, the novel technique of Monte Carlo Tree Search (MCTS) became popular, as it was shown that using MCTS, the quality of AI for Go improved significantly, albeit not yet to the level of top-level human players. Many experts predicted that around 2030 Go AI would surpass human-level play. Much to the surprise of many, however, already in 2016 Google's AlphaGo defeated the human world champion in Go, using a combination of MCTS and deep convolutional networks to evaluate Go board positions and perform move selection. The networks were trained using millions of examples of human play, combined with self-play. A short while later, it was demonstrated with AlphaZero that self-play by itself suffices to train the networks to reach the necessary quality.

There is a long history of research into computational game AI for 2-player deterministic board games. However, since the turn of the century computational techniques have also been applied to games of a different nature, such as games for 3 or more players, games with imperfect information, and video games. Such games bring their own challenges, and often need very different approaches for creating game AI. Nevertheless, computational techniques may be applicable. Recent successes have been achieved in the playing of Poker (multiple players, imperfect information) and DotA (team-based video game). Deep learning techniques have been used to teach a game AI to play old Atari video games, and the highly complex game Doom, by only observing the screen.

These computational approaches to AI game playing have been highly successful, and have caused great enthusiasm in researchers and laymen alike. However, while they have opened up new possibilities for implementing strong game AI, they are definitely not the one-size-fits-all solution for all problems in computational game playing. The aim of the seminar was to build upon the foundations laid by the state-of-the-art in computational game playing, and (1) identify for which game AI problems the current state-of-the-art is inadequate or unsuitable, including the reasons why; (2) propose and investigate which improvements to the state-of-the-art may open up ways to apply it to a wider range of game AI problems; and (3) form ideas on which novel techniques may be employed to solve problems for which the current state-of-the-art is simply not suitable.

For the purpose of the seminar, a "game" is considered any simulated environment in which decisions can be taken in order to achieve a particular goal. This includes board games, card games, video games, simulations, and VR/AR applications. Decisions in a game are taken by "players." In multi-player games, the goal is usually to "win" from other players, by reaching a pre-defined victory condition before any other player manages to do so. "Game AI" is a computer-controlled player. Good game AI takes decisions which are highly effective in achieving the goal. Cooperative games are also of interest, where the aim is for the players to work together to share in a victory. We wish to point out that games are often a reflection of some aspects of the real world, and allow investigating those aspects in a risk-free environment – good solutions for problems found in games may therefore have immediate applications in the real world.

The following is a (non-exhaustive) list of challenges to game AI which are hard to deal with using the current state-of-the-art. These challenges formed the basis for the discussions and investigations of the seminar.

- **Determining the limitations of MCTS and deep learning for computational game playing**: The state-of-the-art in computational game playing encompasses Monte-Carlo Tree Search (MCTS) and deep convolutional networks to store game information.

The recent successes of this approach in Go have made MCTS and deep learning the "go-to" techniques for implementing game AI. However, these techniques have many inherent limitations. MCTS needs extraordinary amounts of computer power and is therefore very expensive to use. While it can be parallelized easily, just adding more computer power has diminishing pay-offs. Moreover, there are many games for which MCTS clearly is not a suitable approach, for instance, games with a large branching factor where it is hard to come up with heuristics which pinpoint the branches which are most likely to contain the strong moves. As for deep learning, now that the early enthusiasm has waned a little, the first criticisms of it, which explain its many limitations, are already being published. Gaining insight into the limitations of MCTS and deep learning for game AI implementation will allow us to distinguish those games for which these techniques may be employed for strong game playing from those games for which different approaches are needed.

- **Defining more appropriate game complexity measures**: Game complexity is a measure which is supposed to indicate how difficult it is to implement game AI. It is usually expressed as the number of possible game states in base $log_{10}$. Beyond a complexity of 100 ($10^{100}$ game states), it is highly unlikely that a game will ever be "solved," i.e., will never be played perfectly. Researchers therefore aim for superhuman rather than perfect play. For a long time Go was considered the pinnacle of complexity in game playing, boasting a game complexity of 360. However, in the game AI domain, games have been researched with a much higher game complexity than Go. Typical examples of such games are:
  - *Arimaa*, a 2-player deterministic, perfect-information board game with a game complexity of 402.
  - *Stratego*, a 2-player deterministic, imperfect-information board game with a game complexity of 535.
  - *StarCraft*, a typical Real-Time-Strategy video game, with a varying game-tree complexity (depending on the parameters of the scenario) which is measured in the tens of thousands.

  The increased complexity of these games stems from multiple factors, such as an increased move complexity (e.g., in Arimaa players always make four moves in sequence), the introduction of imperfect information (e.g., in Stratego at the start of the game the players only know the location of their own pieces), or simply an explosion of pieces, moves, and non-deterministic influences (e.g., most video games). A common belief is that an increase in game complexity also entails an increase in difficulty of creating a game AI; however, previous investigations have shown that high game complexity does not necessarily equate high difficulty for achieving superhuman play. This indicates that "game complexity" might not be the most appropriate complexity measure for games. A theoretical investigation of game features may result in alternative ways to express game complexity, which may better relate to the difficulty of playing the game for an AI. Moreover, a better understanding of what makes a game difficult for an AI, might lead to new insights into how strong game AI can be built.
- **Learning game playing under adverse conditions**: In recent years, most research into game AI has moved towards "learning to play" rather than "implementing an algorithm." Game AI can learn from observing examples of human play (provided a large enough dataset is available) or from self-play. Such learning has lead to strong results in some cases, but in many cases fails under adverse conditions. Examples of such conditions are:

  - Imperfect information, i.e., the results of decisions of the AI depending partly on unknown data.
  - Continuous action spaces, i.e., the AI in principle being allowed to take an unlimited number of decisions in a small time period; thus, an AI not only has to decide what actions it wants to take, but also how many and with which intervals.
  - Deceptive rewards, i.e., situations in which positive results achieved by the AI in the short term, in practice drive it away from the ultimate goal of the game in the long term.

  To implement learning AI for wider classes of games, approaches must be devised to deal with such adverse conditions in systematic ways.
- **Implementing computational game AI for games with 3 or more players**: Most research into game AI is concerned with zero-sum 2-player games. The reason is obvious: in 2-player games, an objective "best move" always exists. With games that involve more than two players, which oppose each other, there often is no obvious "best move." For instance, if there are three players in the game, if two of those players band together, in general the third one will have a very hard time winning the game, even when taking into account that the other two are collaborating. The main problem is that when one player is obviously doing better than the other two, it is to the advantage of the other two to collaborate against the envisioned winner. Therefore, in games with three or more players, it may be advantageous not to play better than the other players, in order not to become the target of a collaborative assault of the opponents. A pessimistic perspective, where the AI assumes that the opponents will actually form a block, will in general lead to much worse play than a perspective wherein the AI tries to form collaborations itself. This means that the AI must incorporate in its reasoning the attitudes of the other players, for instance in the form of player models.

  The topic of AI for games of three or more players has been studied very little – a notable exception being the study of Poker, which is a relatively easy game in this respect considering the simplicity of the required player models and the very small state-space and game-tree complexities. Hundreds of thousands of games for three or more players exist, which by itself means that this challenge needs investigation. Moreover, when translating research findings to real-world challenges, the existence of more than two players is a given in most realistic situations.
- **Implementing AI for games with open-ended action spaces**: Certain classes of games have so-called "open-ended action spaces," i.e., the number of possible actions is basically unlimited. One example of such a type of game is found in interactive fiction: these are puzzle games which the player controls by typing sentences in plain English. While each game only understands a limited set of verbs and nouns, the player is generally unaware of this list. Designers of such games aim to allow the player to give any English command which is reasonable in the circumstances described by the game. Another example of such a type of game is a tabletop role-playing game, in which players are allowed to perform any action at all, and a game master determines (within boundaries of a complex ruleset) what the result of the action is. Creating an AI for either a game master or a player of such a game requires the AI to have at least a basic understanding of the game world to be successful. In practice, studies into game AI focus almost exclusively on games where the action spaces are closed, which makes regular learning algorithms applicable. For games with open-ended action spaces, as of yet no successful approaches have been found.

- **General Game Playing**: In the domain of General Game Playing (GGP), games are defined by a set of rules, specified in a General Game Description Language (GGDL). The goal of researchers in this domain is to create an artificial intelligence which is able to play such a game, based on only the rules. Yearly competitions are held where researchers pose their AIs against each other in games which are unknown to them at the start of the competition. The state-of-the-art in such competitions is using MCTS, enhanced according to some general assumptions on the types of games that need to be played. This approach is unsurprising, as MCTS does not require knowledge of the game in question in order to do reasonably well. In fact, this approach is so strong and so easy to implement that all competitors use it. The danger of the success of MCTS for GGP is that the research in this area gets stuck at a dead end – the same happened with the research into traditional game AI when for decades researchers only worked on small improvements to minimax and alpha-beta pruning, until MCTS came around to shake things up. It is highly unlikely that a blind and ostensibly "stupid" approach such as MCTS is the end-all of GGP AI implementations. It is therefore of particular interest to investigate novel approaches to GGP, which are not MCTS-based.
- **General Video Game Playing**: General Video Game Playing (GVGP) aims at designing an AI agent which is capable of successfully playing previously-unknown video games without human intervention. In the General Video Game AI (GVGAI) framework, video games are defined by a set of rules, sprites and levels, specified in a Video Game Description Language (VGDL). The VGDL was initially proposed and designed at the 2012 Dagstuhl Seminar on Artificial and Computational Intelligence in Games. The GVGAI framework has been expanded to five different competition tracks: (1) single-player planning, (2) two-player planning, (3) learning (in which no forward model is given), (4) level generation and (5) rule generation. In the planning tracks a forward model of every game is available; MCTS has been the state-of-the-art algorithm in these tracks. However, MCTS is not applicable to the learning track as no forward model is given and thus no simulation of game playing is possible. Deep reinforcement learning is a potential approach for the GVGAI learning track, but has not been investigated yet. Other methods might have potential too. Determining the applicability of different methods to the creation of GVGAI is a novel and topical challenge. Of particular interest in this respect is the creation of an AI for the domain of general Real-Time-Strategy (RTS) games.
- **Computation for human-like play**: Virtually all research into computational game AI focuses on building a game-playing AI which is as strong as possible. Strength can objectively be measured by pitting different AIs against each other. In video-game AI research, it has been recognized that playing strength is, in general, not a major goal – instead, much research in video game AI is aimed at making the AI play in an entertaining, interesting, or human-like manner. MCTS is notoriously unsuitable for selecting moves that are human-like, as it is simply based on finding the best outcome for the game as a whole. However, in situations where humans play against an AI, whether it is for entertainment or training, it is desirable that not only the best moves can be played by the AI, but also those moves which are interesting to explore or are on par with how a human might play. Almost no research has yet been done into computational human-like AI, which makes it a worthy challenge to take on.

The Dagstuhl seminar brought together computer scientists and industry experts with the common goals of gaining a deeper understanding of computational game AI, in particular to determine the limitations to the state of the art, to find new uses for the state-of-the-art, to explore new problems in the domain of computational game AI, and to investigate novel approaches to implementing computational game AI. Industry experts came not only from companies which specifically work in game AI research, but also from companies which use game AI in their products.

During the seminar we not only had discussions which investigate the topics theoretically, but also spent part of the seminar on trying to achieve practical results. We did the same in the 2015 and 2017 seminars, which was met with great enthusiasm and led to some strong follow-ups. As in the previous seminars, these practical sessions were partly to test out new ideas, and partly competition-based, where different approaches were used to implement AI for new problems, which were then compared to each other by running a competition.

What was new for this particular seminar, is that we held expert talks during some of the evenings. These started with one or two experts giving a longer talk (between half an hour and an hour-and-a-half) on one of their specialisms, followed by a longer Q&A session and a discussion. One evening was spent this way on using modern communication media to inform people about research (Tommy Thompson), one evening was spent on the details of DeepMind's AlphaStar (Tom Schaul), and one evening was spent on advanced search techniques in board games (Olivier Teytaud and Tristan Cazanave). These evenings were greatly appreciated by participants, and should remain part of this series of seminars.

Reports on the working groups are presented on the following pages. Many of these working groups have lead to collaborations, which will lead to papers to be published at conferences and in journals in the coming year. All in all, the general impression was that the participants and the organizers found the seminar a great success, and an inspiration for future research.

## 2 Table of Contents

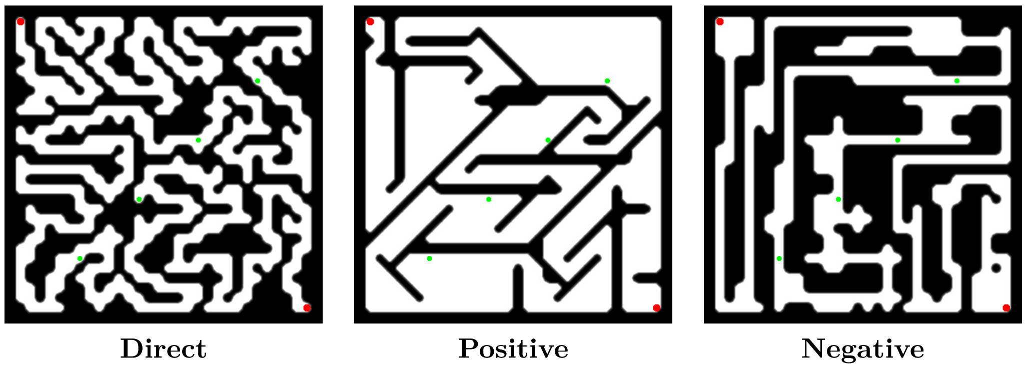

**Figure 1** Three level maps evolved with the same fitness function but different representation. The fitness function maximizes the path length between the two red dots. The first image designates cells of the map as "full" or "empty" with a binary gene, the second specifies the placement of walls, while the third specifies where to dig out rooms and corridors.

## 3 Working groups

### 3.1 Representation in Evolutionary Computation for Games

*Dan Ashlock (University of Guelph, CA), Tom Schaul (Google DeepMind – London, GB), Chiara F. Sironi (Maastricht University, NL), and Mark Winands (Maastricht University, NL)*

Representation is a key feature of the design of evolutionary algorithms. This abstract looks at applications of the principle of the importance of representation to automatic content generation and games. Three topics were identified for further study: parameterized tree policy to permit a space of MCTS like algorithms to be studied, co-evolving games and agents to partition and understand game content generation spaces, and graph motifs as an enabling representation for the evolution of narratives to underly games.

### Introduction

The topic of representation in evolutionary computation is very broad. The key point is that representation can have a large impact on the time required to solve evolutionary optimization problems and one the type of solutions located by an evolutionary algorithm system. Consider the problem of procedural generation of level maps. Figure 1 demonstrates the impact of representation on a simple procedural content generation problem [12].

The three example maps were all evolved to maximize the path length between a starting and an ending point, marked with red dots. The green dots were required to be in open space. The use of distinct representations caused a large impact on the appearance of the maps, demonstrating one of the impacts of choice of representation. Readers interested in additional reading on generating these sort of level maps might want to peruse [8, 1, 3, 4].

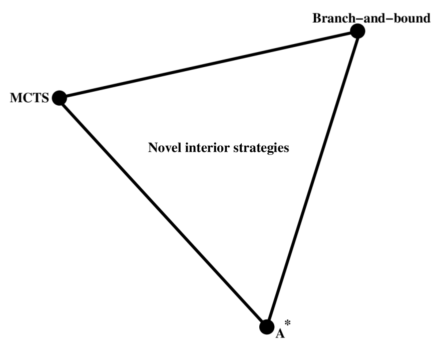

**Figure 2** A diagrammatic representation of the parameterized tree space. The vertices of the triangle represent extreme settings of the parameters that recover particular tree-search algorithms that are the basis of the space.

## Parameterized Tree Search

Monte-Carlo tree search [6] performs a directed partial search of a game space, functioning as an anytime algorithm that can refine its opinion about the correct next move until that move is required. The tree's policy is used to condition the balance between exploration and exploitation, favouring branches of the game tree that are promising or favouring branches of the game tree that are unexplored. Various *tree policies* are used for selecting which branch of the tree to follow and which new branch to try.

Many other tree search algorithms for games also exist [5]. The group identified the potential to write a parameterized tree policy that had MCTS, A∗, branch-and-bound, and potentially other tree search algorithms, as the extreme cases of the space defined by the policy function. This notion is shown in diagrammatic form in Figure 2. Using such a parameterized tree policy opens the door to a novel type of research on tree policies. Problems that could be treated include:

- The relationship between category of game and appropriate tree search algorithm.
- The utility of the algorithms that arise in the interior of the space of possible tree policies.
- The identification of new tree search algorithms by considering how to induce new behaviours with the parameterized tree search policy.

The group also identified a potential modification of MCTS that might grant it improved performance. At present most tree policies are based on the quality of the branched of the tree and the degree to which they have been explored. Two added pieces of information might be useful: the depth within the game tree of the node of the tree currently under consideration and the number of roll-outs performed since the search was started after the last move made. The incorporation of these features creates the potential for tree polices that are sensitive to the stage of the game and the overall quality of the tree that they are extending.

## Co-evolution of games and agents

A perennial, difficult topic is using computational intelligence to tune or create new games [11, 9]. The group identified a potential approach that would be interesting to try. When generating a new game or new pieces of procedural content there is a problem in that the fitness function used to drive evolution usually specifies much simpler tests that those implicit

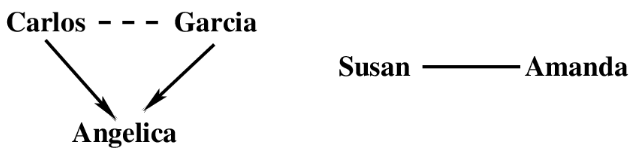

**Figure 3** Two graph motifs. The first designates a love triangle in which Carlos and Garcia are competing for the affections of Angelina. The second indicates that Susan and Amanda are friends.

in the play of the new game or of the use of the automatically generated content. One approach to this is to attempt to enumerate the several factors affecting performance and utility and using multi-objective optimization [7]. This is a difficult approach if the numbers of factors being assessed are large.

An alternative is to use co-evolution of games or game elements with agents that use the evolved content. As long as the level of agent success with a game element can be used to generate numbers it is possible to create a system that imitates some of features of multi-criteria optimization. During evolution, an agent's fitness is their best score on any game or piece of game content. The fitness of the game or game content is the number of agents on which that game content has the best score. This system creates a situation in which agents compete to become more skilled and game content evolves to attract agents. This situation is similar to *partitioning regression* [2] in which models of data are co-evolved to create models that specialize in different parts of the data space. The partitioning of game elements and agents provides a collection of possibilities analogous to the Pareto-frontier provided my multi-objective optimization.

It is important to note that the evaluation function of the agents on the game or game elements must be chosen with some care to evoke the partitioning effect. An incorrect choice will lead to very similar game elements or one game element collecting all the agents.

## Narrative generation via graph motifs

Another area of current research in games is the automatic generation of narratives underlying games [10]. The group identified the potential for search in spaces of social network graphs as a means of automatically establishing what actions a game engine should encourage to move a plot from an initial condition to a final condition specified by the game designer.

In this case, the social network graphs have vertices representing the players and non-player characters in the game as well as labelled edges that designate their relationship. Graph motifs are snippets of the graph that designate important or potentially important social situations. Figure 3 shows two graph motifs. In the first the arrows show interest, possibly romantic, or two young men in a young woman. The dotted line between the men shows that they know one another but are in competition. The second shows that two characters are friends. Characters and edges may participate in multiple graph motifs.

The game designer specifies and initial situation and a final situation. Search software is then use to connect the two social network graphs via a collection of *resolution operators* that transform motifs. With a rich enough collection of resolution operators, and the possibility of adding auxiliary characters to the scenario, almost any two graphs *can* be connected. The job of the search software is to find relatively short paths between the beginning and ending configurations as well as providing the game designer with a network consisting of multiple paths between the two configurations. This represents a rich space for manoeuvring the underlying narrative of the game toward the desired outcome.

Consider the Carlos, Garcia, Angelica triangle. Possible resolutions of this include the following.

- Angelica makes a choice, favouring one of the young men. The young men become {friends, strangers}.
- One of the young men kills the other to resolve the rivalry. Angelica {falls in love, is repulsed}
- An auxiliary character, Veronica is added to the social network graph and captures the affections of one of the young men.

Other possibilities are two characters who are friends of a third become friends, or possibly enemies. A friendship might sour. A romantic attraction may form. The key feature is that the space of motifs and the resolutions available for each motif must be rich enough to create a moderately rich network of social interaction graphs. This notion may be a bit difficult, but there is a network in which each node in the network is a social interaction graph, and the (directed) links in that network are resolution operators.

The possible relationships (edges), graph motifs, and resolution operators must be selected in a game-specific fashion, but the search engine can be generic once these quantities are specified. In a rich network space, evolutionary search is probably a good tool for traversing the network of graphs, which has a complex and not-too-structured fitness landscape. The group feels that development of this system could be quite valuable for narrative design in games.

## References

1 D. Ashlock and L. Bickley. Rescalable, replayable maps generated with evolved cellular automata. In *Acta Physica Polonica (B), Proceedings Supplement*, volume 9(1), pages 13–22, 2016.

2 D. Ashlock and J. Brown. Evolutionary partitioning regression with function stacks. In *Proceedings of the IEEE 2016 Congress on Evolutionary Computation*, pages 1469–1476, Piscataway, NJ, 2016. IEEE Press.

3 D. Ashlock, C. Lee, and C. McGuinness. Search based procedural generation of maze like levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):260–273, 2011.

4 D. Ashlock, C. Lee, and C. McGuinness. Simultaneous dual level creation for games. *Computational Intelligence Magazine*, 2(6):26–37, 2011.

5 D. Ashlock and C. McGuinness. Graph-based search for game design. *Game and Puzzle Design*, 2(2):68–75, 2016.

6 C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

7 C.A. Coello Coello, González Brambila S., J. Figueroa Gamboa, and et al. Evolutionary multiobjective optimization: open research areas and some challenges lying ahead. *Complex Intelligent Systems*, 2019.

8 Lawrence Johnson, Georgios N. Yannakakis, and Julian Togelius. Cellular automata for real-time generation of infinite cave levels. In *PCGames '10: Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, New York, NY, USA, 2010. ACM.

9 Jialin Liu, Julian Togelius, Diego Perez-Liebana, and Simon M. Lucas. Evolving game skill-depth using general video game ai agents, 2017.

10 Mark O Riedl and Robert Michael Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268, 2010.

**11** Julain Togelius and Noor Shaker. *Procedural content generation in games.* Springer, New York, NY, 2016.

**12** Julian Togelius, Georgios Yannakakis, Kenneth Stanley, and Cameron Browne. Search-based procedural content generation. In *Applications of Evolutionary Computation*, volume 6024 of *Lecture Notes in Computer Science*, pages 141–150. Springer Berlin / Heidelberg, 2010.

## 3.2 Game AI for a Better Society

*Alan Blair (UNSW – Sydney, AU), Christoph Salge (New York University, US), and Olivier Teytaud (Facebook – Paris, FR)*

The discussion in this group centered around two main themes: Game AI for Education, and Game AI for improved Mental Health.

There have been many efforts over recent years to develop specific games for vocational training, or to “gamify” components of the traditional academic syllabus.

Potential advantages of educational games include support for multi-view learning, scalability, equity of access and adaptability, promoting distinct capacities and complementary skills.

There is a diversity of opinion among educators as to whether students should be encouraged to play to their own strengths, leading to specialization; or to play to their own weaknesses, leading to good all-rounders. Studies have shown that when users are able to choose their own activity in educational games, those who already excel in abstract reasoning but not in social skills tend to choose strategic games like chess or checkers; and conversely, those with good social skills will often choose role-playing games. Counteracting this inherent self-selection bias may require careful incentives built into the system, or the intervention of a human instructor. On the other hand, the more important aim might be to empower the student and help them to better understand their own profile, and the effect of their choices on the coverage of their educational objectives.

One area in which Game AI could potentially have an impact is helping people to understand complex systems. Increasingly, a comprehension of phenomena arising from multiple interacting factors is needed in order to make even routine personal choices such as selecting a suitable mobile phone plan. Browsing effectively for accommodation, restaurants or leisure activities requires an awareness of the role of fake reviews and related phenomena. There are an increasing number of issues of public concern for which an informed citizenry with a general understanding of complex systems is necessary, in order to maintain a robust and healthy public discourse. These include electricity markets, university governance, fishing or water rights in a river system, economic and tax policies, anti-trust laws, geopolitical issues and trade agreements, as well as environmental issues and climate change.

Several existing games focus on issues such as Prisoner’s Dilemma or Tragedy of the Commons. With these kinds of games there is a risk of becoming overly simplistic or didactic, or forcing a limited range of decisions. Ideally, these games should be constructivist and open-ended in nature, as well as being fun to play. One good example is the ECO game (play-eco.com).

Another role for Game AI might be in helping people to understand the possible impact of AI on society, and the economic, philosophical, practical and societal issues that may arise. One good example of this is the Fallout universe, which began in the 1980's dealing with issues such as robot terminators, and was expanded into the 2000's to deal with a broader range of issues including AI embodiment, slavery and fundamental rights. For example: To what extent should traditional human rights be extended to robots? Would it be acceptable to "torture" an AI?

Game AI raises many issues in relation to mental health. Modern tablet and smartphone games can be highly addictive and thus detrimental to the mental, physical and economic well-being of susceptible users. Perhaps, tougher legislative control is needed – particularly in relation to freemium games, loot boxes and games targeted to children.

On the other hand, Game AI could present an opportunity to improve mental health and wellbeing, by providing a companion agent, to replace or supplement a therapist, or to help people deal with depression, loneliness or alienation. One example of this is Ally from Spirit AI (spiritai.com) which is a conversational bot to help combat issues such as online bullying – including effective moderation, automatic intervention and behavior prediction.

There is a growing interest in the idea of using AI technology to make a positive impact on society, and we in the Game AI community should be looking for further opportunities to make Game AI part of this movement.

## 3.3 AI for Card Games with Hidden Information

*Bruno Bouzy (Nukkai – Paris, FR), Yngvi Björnsson (Reykjavik University, IS), Michael Buro (University of Alberta – Edmonton, CA), Tristan Cazanave (University Paris-Dauphine, FR), Chiara F. Sironi (Maastricht University, NL), Olivier Teytaud (Facebook – Paris, FR), Hui Wang (Leiden University, NL), and Mark Winands (Maastricht University, NL)*

This document sums up the discussion we had in Dagstuhl on Thursday 19 December about AI for card games with hidden information. We mostly discussed on the base of our ongoing experience on Skat and Bridge. Card games such as Bridge and Skat are Imperfect-Information Games (IIG).

In Bridge and Skat, a perfect solver with open cards is very useful. In Bridge, the perfect solver with open cards is named the Double Dummy Solver (DDS). A similar tool exists in Skat. Flat Monte Carlo (MC) using a Perfect Solver with open cards is the current state-of-the-art algorithm used by playing programs in Bridge and Skat. Another name for this algorithm is Perfect Information MC (PIMC). Flat MC may use simulators different from a perfect solver. Examples are the random simulator or any player which performs a game to the end, or evaluates a position.

In IIG, a node in the game tree can be an information set (IS). An IS is defined by the previous history visible so far by a given player. An IS depends on the viewpoint of a given player. An IS contains a set of states, each state containing the full information of the game at a given time. Because the player has partial information only, a given IS of this player contains all the states that this player cannot discriminate.

A problem identified with PIMC is strategy fusion. With an open card solver, the best strategy output by the solver assumes that each state has its best action, this action depending on the state it belongs to. Therefore, there is no reason that two states belonging to the same IS shares the same best action. This is the strategy fusion problem. However, an open card solver used in Flat MC remains the current best approach to Bridge and Skat.

Another issue in IIG is non-locality. You may consider two game trees: the IIG tree and the Complete Information Game tree (CIG tree). In Bridge or Skat, at the beginning of a game, the cards are dealt, then the play occurs on this specific card distribution. In the CIG tree, the root node is a chance node, and a search with open cards occurs in the corresponding sub-tree below this chance node. However, when searching with hidden cards in the IIG tree by performing actions over IS, the search meets IS with viewpoints from diffrrent players, and consequently, the search meets sub-trees of the CIG tree, different from the sub-trees in which the search started. This is the non-locality problem.

Recursive Flat MC (RFMC) was tried with success in Skat in 2013. RFMC has been tried recently on Bridge with success as well despite the computing time used.

In our discussion, we mentioned IS-MCTS, either with the Single Observer (SO) version, or with the Multiple Observer (MO) version. Various results were obtained with Flat MC and IS-MCTS on IIG. For games with more uncertainty, Flat MC was better than IS-MCTS. But on specific and small games, IS-MCTS could surpass Flat MC.

Alpha-mu is an algorithm which plays the role of the declarer in Bridge card play assuming the defense has perfect information. Alpha-mu solves the strategy fusion problem by playing the same action in all the states belonging to the same IS. Alpha-mu considers a set of worlds (card distributions) for which the set of outcomes are gathered in a vector. A set of vectors of outcomes (0 or 1) can be associated to a node in the tree. Specific operators on these vectors are used to back up information from nodes to nodes.

We also discussed about zero learning. Zero learning does not mean no learning, but learning without external knowledge, by self-play and starting with the rules of the games only.

We mentioned the importance of performing inferences in an IIG. We briefly discussed perspectives on Bridge and Skat, such as performing inferences with neural networks or solving the sub-game of winning the next trick in Bridge.

## 3.4 Human-AI Coordination

*Katja Hofmann (Microsoft Research – Cambridge, GB), Duygu Cakmak (Creative Assembly – Horsham, GB), Peter I. Cowling (University of York, GB), Jakob Foerster (University of Toronto, CA), Setareh Maghsudi (TU Berlin, DE), Tómas Philip Rúnarsson (University of Iceland – Reykjavik, IS), Pieter Spronck (Tilburg University, NL), and Nathan Sturtevant (University of Alberta – Edmonton, CA)*

Human-AI coordination is emerging as a new frontier for machine learning research. While historically a large part of the machine learning community has been focused on supervised learning tasks, new application areas require AI systems to collaborate and coordinate with humans in complex environments. Real-world applications include self-driving cars, virtual assistant technologies, game AIs, and household robotics.

Interestingly, much of the progress on multi-agent learning has focused on competitive settings, in particular two player zero-sum games. In these settings all Nash equilibria are interchangeable, so going from a self-play setting to playing optimally against a human opponent is trivial.

Coordination games are at the opposite end of the spectrum. In these games the optimal policy is crucially dependent on the strategy played by the other agents in the team. Yet, humans have remarkable skills at coming up with strategies that allow them to coordinate with an unknown team-mate on a novel task in a zero-shot setting. In contrast, there has been very little progress on this kind of problem in the machine learning and reinforcement learning community.

In this Dagstuhl seminar working group, we identified key directions for driving research towards more effective learning of human-AI coordination. First, since machine learning communities are benchmark and data driven, we believe that one reason for the lack of progress in this important domain is due to missing appropriate benchmarks. Second, we identified promising directions towards solving this challenge. Below, we briefly summarize these.

The team identified the following considerations for a benchmark for driving progress in human-AI coordination: (1) it should promote general solutions to avoid overfitting to individual specific game instances, (2) it should provide scope for collecting or providing data on human-human coordination, as a means of training and evaluation, and (3) it should provide a reliable measure of success that can reliably quantify progress towards human-like coordination.

Addressing these considerations, we propose the following benchmark setup. The key idea is to utilize a large number of novel *pure coordination* games, in which the optimal action crucially depends on the assumptions made about the gameplay of the (unknown) team-mate. One surprisingly entertaining example game is the *Guess 100* game, in which two players repeatedly choose numbers between 1 and 100 with the goal of converging to the same number as fast as possible. In this game humans typically play one of a number of heuristics that solve the game after a small number of guesses with an unknown team mate, displaying *zero-shot* coordination. Each of the proposed coordination games is accompanied by a large number of example game plays from humans playing online in an ad-hoc setting (ie. with a randomly assigned team mate) which can be used to develop new methods and to evaluate the performance of existing and novel learning algorithms. The goal of the benchmark would be to push for methods which produce human-like policies for the hold-out (test) games, without having access to the human game play data

Directions towards solving the proposed benchmark challenge include a wealth of research in game theory, machine learning, multi-agent systems, cognitive science, economics and many other areas. For example, one direction considers approaches that mimic the human ability to interpret the actions and intents of others, commonly called *theory of mind* (ToM). It involves interpreting the communicative intent when observing the actions of others, but also allows for finding strategies that can be interpreted successfully when observed by others.

Initial software to let AI play the *Guess 100* game, developed during the seminar, can be found here: https://github.com/katja-hofmann/dagstuhl-hac.

## 3.5 Evolutionary Computation and Games

*Jialin Liu (Southern Univ. of Science and Technology – Shenzen, CN), Dan Ashlock (University of Guelph, CA), Günter Rudolph (TU Dortmund, DE), Chiara F. Sironi (Maastricht University, NL), Olivier Teytaud (Facebook – Paris, FR), and Mark Winands (Maastricht University, NL)*

Evolutionary Computation (EC) techniques have been widely used in games. In this working group, we summarised the use of EC techniques in games mainly into the following categories:

- Planning in games: evolving action sequences for playing games [2].
- Agent training/adaptation: evolving/adapting (general) AI agents [5].
- Game design: level generation / puzzle design [3], game balancing [4], enhancing PCGML [6], design of NPCs.
- Evolutionary game theory [1].

In this working group, we brainstormed the pros and cons of EC for games, as well as the opportunities and problems. This abstract summarises the brainstorming discussions during the session.

An incomplete list of advantages and weaknesses of EC for games are listed as follows.

- Advantages:
  - **Easy to struct** Evolutionary Algorithms (EAs)[1] are easy to be structured and applied to different situations in games, e.g., for game design or game playing; single- or multi-player games; collaborative or adversarial games; with full or partial observation, etc.
  - **Derivative-free** EAs are derivative-free algorithms and meet well the needs of optimisation problems in game.
  - **Exploratory** EAs are powerful for their ability of exploring the search space.
  - **Robust to useless variables** EAs are robust to useless variables.
  - **Parallelism** EAs are population-based and can be easily paralleled.
  - **Diversity-convergence** EC takes into account both the diversity and convergence when evolving action sequences or game contents.
  - **Multi-objective** Multi-objective EAs can be used in the multi-objective optimisation problems in games.
  - **Partial observation** EAs can be applied to playing partially observable games by sampling.
- Weaknesses:
  - **Computationally costly** EC techniques can be computationally costly to find a reasonably good result (e.g., when using a single parent or having many restricted variables).
  - **Dense results** The results can be very dense and hard to understand.
  - **Representation matters** Designing good representation is hard. A bad one will significantly reduce the efficiency and effectiveness of EAs.
  - **Fitness** When there is no optimisation fitness, EC techniques are no more applicable.
  - **Hard to reuse data**

[1] In this abstract, we use the term "Evolutionary Algorithm (EA)" to refer to a whole family of algorithms, instead of a particular algorithm.

Among the above weaknesses, designing good representation and defining good fitness are particularly hard and important. For instance, when applying EC techniques to evolving game levels, how to design a suitable agent-based game fitness evaluation?

We agreed with the potential of using ECs for agent configuration, in particular for evolving selection strategies in MCTS.

**References**

1 Dan Ashlock, Mark D Smucker, E Ann Stanley, and Leigh Tesfatsion. Preferential partner selection in an evolutionary study of prisoner's dilemma. *BioSystems*, 37(1-2):99–125, 1996.

2 Raluca D Gaina, Adrien Couëtoux, Dennis JNJ Soemers, Mark HM Winands, Tom Vodopivec, Florian Kirchgeßner, Jialin Liu, Simon M Lucas, and Diego Perez-Liebana. The 2016 two-player gvgai competition. *IEEE Transactions on Games*, 10(2):209–220, 2017.

3 Ahmed Khalifa, Diego Perez-Liebana, Simon M Lucas, and Julian Togelius. General video game level generation. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 253–259, 2016.

4 Jialin Liu, Julian Togelius, Diego Pérez-Liébana, and Simon M Lucas. Evolving game skill-depth using general video game ai agents. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2299–2307. IEEE, 2017.

5 Chiara F Sironi, Jialin Liu, and Mark HM Winands. Self-adaptive monte-carlo tree search in general game playing. *IEEE Transactions on Games*, 2018.

6 Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M Lucas, Adam Smith, and Sebastian Risi. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 221–228, 2018.

## 3.6 Abstract Forward Models

*Diego Perez Liebana (Queen Mary University of London, GB), Raluca D. Gaina (Queen Mary University of London, GB), and Tommy Thompson (AI and Games – London, GB)*

Statistical Forward Planning (SFP) algorithms are those methods that use a Forward Model (FM) to plan an action or a sequence of actions from a given game state. Examples of these methods are Monte Carlo Tree Search (MCTS) and Rolling Horizon Evolution (RHEA). One of the main benefits of SFP approaches is that they are able to make new plans every time a new decision needs to be made, giving them flexibility and adaptability to face unexpected and previously unseen circumstances. MCTS and RHEA have been widely used in games research, from general (video) game playing [1, 2] to state-of-the-art approaches in Computer Go [4], Shogi, Chess and Atari [5]. Despite their popularity, however, SFP methods have rarely reached the games industry, with not many commercial titles using these methods. Notable exceptions are Creative Assembly's Total War series and AI Factory's card and board games.

One of the main reasons for the small uptake of SFP methods is their need for an FM. In particular, an FM needs to provide two pieces of functionality: a *copy* function, which creates exact copies of a game state, and a *next* function that provides a possible next game state given an original one and an action to apply to it. MCTS (or RHEA) normally select better moves if the number of iterations (or, respectively, generations) is higher, which is heavily influenced by the computational cost of the *copy* and *next* functions. Many commercial games (especially the ones that would benefit from the use of SFP methods) have complex mechanics and large game states, making the use of FMs unaffordable.

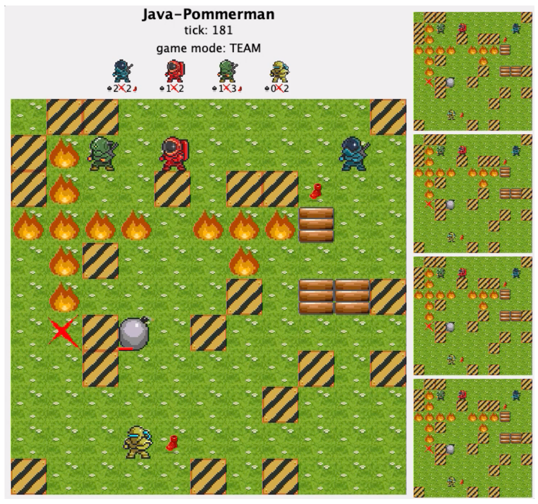

**Figure 4** Java Pommerman framework.

This workgroup focused on approaching this issue by studying how *abstract* forward models can be created. Rather than learning forward models (as done in [5] for games of reduced complexity and size), the objective was to analyze how game states can be simplified into models that SFP methods can use without incurring in a significant loss of performance. In order to test this, we used a Java implementation of Pommerman, a 4-player simultaneous move game in which agents try to be the last one standing to win. Figure 4 shows a screenshot of this game. For more information about this game, the reader is referred to [3].

In order to test in the conditions of slow FM models, we artificially included delays in the different update rules of the game, such as computations for bomb explosions and movement, flame propagation and avatar collision checks. This created a version of the game that, because of the time required to carry out FM calls to the function *next*, was unusable by SFP methods. Then, we included the possibility for MCTS to decide i) which rules should actually be updated by the FM; and ii) a distance *threshold*, from the player's avatar, beyond which game state information is ignored by the FM's *next* calls. In effect, this created the possibility for MCTS to use an abstract (incomplete, inaccurate) FM that was faster to execute than the perfect one.

Our initial tests showed some interesting results. We ran MCTS players that use an abstract FM (AFM-MCTS) against the one that uses the complete and expensive FM (simply MCTS). They played in Pommerman's TEAM mode, 200 repetitions per pair. Two versions of AFM-MCTS where used: `Threshold-3`, which uses an FM that updates all the rules of the game except those occurring farther than 3 tiles away; and *Incomplete*, which uses an FM with infinite threshold where most game rules are not triggered at all.

For the `Threshold-3` case, the AFM-MCTS method was able to achieve a higher winning rate than the same MCTS algorithm which had access to a perfect FM. Simply by limiting the rules that can be updated to those taking place in close proximity to the agent, MCTS was able to run more iterations and therefore increase the quality of action selection, leading to a higher victory rate (62%, with 0.5% draws). When the FM abstraction is more severe (as in *Incomplete*, where some rules do not trigger in any case), the AFM-MCTS method victory rate falls, but only to 44% (1% draw rate). Considering this abstract FM is highly dysfunctional, it is remarkable that MCTS still achieves a decent winning rate using it.

In a way, this work shows how it is possible to *inject* Partial Observability to the model used for planning without heavily penalizing the performance of the agents. Which by itself opens interesting lines of future work. For instance, it would be interesting to investigate if it is possible to automatically detect which rules should be updated, and when. It could be possible to define priorities for these rules, which may depend on distances or ad-hoc heuristics. It could also be possible to statistically model (or learn) those expensive feature updates that are omitted from the FM, creating a hybrid between abstract and learnt FM.

In general, this work proposes two lines of research for the future. First, understanding how models can be simplified to a minimum expression without reducing the performance of the AI agents. Can these models be learnt automatically? What can we learn from a game design point of view, if some features can be ignored for AI decision making? Is it possible for these models to maintain a symbolic representation, hence improving the explainability of their decisions? Secondly, it is also worth investigating how SFP methods can themselves be improved to work with an abstract and inaccurate FM. This would not only aid in the use of abstract FMs for commercial games, but also in applications of automatically learning FMs with model-based reinforcement learning approaches.

**References**

1 Michael Genesereth, Nathaniel Love and Barney Pell. *General game playing: Overview of the AAAI competition*. AI magazine, 26:2, 62-73, 2005.

2 Diego Perez-Liebana, Jialin Liu, Ahmed Khalifa, Raluca D Gaina, Julian Togelius and Simon M. Lucas. *General Video Game AI: A Multitrack Framework for Evaluating Agents, Games, and Content Generation Algorithms*, IEEE Transactions on Games 11:3, 195-214, 2019.

3 Diego Perez-Liebana, Raluca D Gaina, Olve Drageset, Ercüment Ilhan, Martin Balla, and Simon M Lucas. *Analysis of Statistical Forward Planning Methods in Pommerman*, Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 15:1, 66-72, 2019.

4 David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche and others. *Mastering the Game of Go with Deep Neural Networks and Tree Search*. Nature, 529:7587, 484-504, 2016.

5 Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt and others. *Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model*. arXiv preprint arXiv:1911.08265, 2019

## 3.7 The Leonardo Project: Open-ended skill discovery

*Sebastian Risi (IT University of Copenhagen, DK), Alan Blair (UNSW – Sydney, AU), Bruno Bouzy (Nukkai – Paris, FR), Nantas Nardelli (University of Oxford, GB), Tom Schaul (Google DeepMind – London, GB), Jacob Schrum (Southwestern University – Georgetown, US), and Vanessa Volz (modl.ai – Copenhagen, DK)*

One of the open problems in artificial intelligence is open-endedness. The question is how can we create a system that, similiary to evolution in nature, can create an unlimited variety of different innovations.

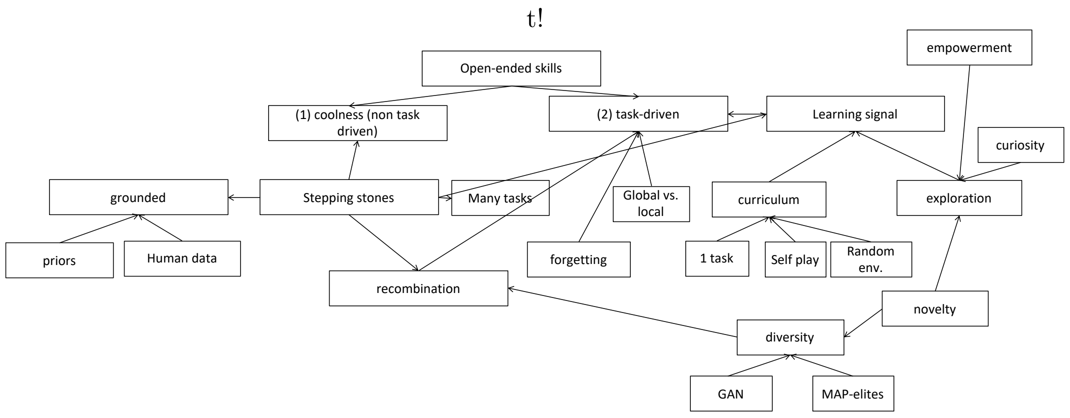

**Figure 5** Overview of the different aspects of open-endedness.

While approaches modeled after evolution in nature (i.e. evolutionary algorithms (EAs)) have shown promise in optimization problems, no current EA does produce innovations in a truly open-ended fashion.

## Overview of Open-ended Approaches

We sketched the landscape of open-ended approaches in Figure 5. First, approaches to OE can be divided into approaches that ultimately want to solve a certain task and approaches where the final task does not matter.

Additionally, the current approaches to discover diversity can be divided into the following approaches:

1. Co-evolving or complexifying environments
2. Self-play (curriculum but no diversity of skills)
3. Imitation Learning: Learning from human demonstrations (might not work all the time; need enough data)

## Promising current approaches

One of the first examples of the idea of open-ended innovation is Minimal Criterion Coevolution (MCC) [1]. In MCC both the agent and the environment co-evolve to solve increasingly more difficult mazes. Recent work building on these ideas is POET [3], which deals with the more challenging OpenAI gym bipedal walker domain. POET is a good example of an approach in which solutions to a particular obstacle-course can function as stepping stones for solving another one. In fact, for the most difficult environments it was not possible to directly train a solution; the stepping stones found in other environments were necessary to solve the most ambitious course.

However, neither MCC nor POET are fully open-ended. Both approaches are restricted in the type of environments they can generate, and the goals the agents pursue. Some creative problem solving occurs, but the space of actions/behaviors in these example domains is fairly restricted. Therefore, it is worth considering what types of domains can actually support open-ended skill discovery, and what algorithmic ingredients would make it possible.

## Necessary ingredients for Open-Endedness?

Several existing algorithmic tools may contribute to the discovery of open-ended behavior. Ones worth considering are listed below.

1. Reinforcement Learning
2. Quality Diversity
3. Co-evolution
4. Domain randomization
5. Populations
6. Curiosity-driven Exploration

The success of RL, and Deep RL in particular, in various domains makes it a technique that is impossible to ignore in the search for interesting behaviors. However, narrow pursuit of maximizing rewards in a particular domain seems to often converge to a narrow range of behaviors. Population-based approaches provide a way to store a variety of interesting behaviors across different individuals rather than inside of a particular agent's policy, but can also be susceptible to convergence. However, techniques such as Quality Diversity and Co-evolution encourage new behaviors to constantly emerge in evolving populations. Curiosity-driven Exploration can also explicitly seek new and different behaviors. Finally, domain randomization could also force an agent to support multiple behaviors, because a variety of unexpected domains means a variety of behaviors may be required to perform well. There may be interesting ways to combine these techniques in yet unseen ways to encourage open-ended behavior.

## A new domain to study open-endedness

Here we propose a new domain to test the ability for open-ended innovation of different algorithms. In this domain, called the *Leonardo Project*, the agent has to try to learn to paint any painting with arbitrary given constrains. Given the success of other recent computational art approaches [2], it is important to emphasize that the new goal here is to have a single learning/training process that produces, in a single run, the ability to generate a wide variety of art in many different styles.

**References**

**1** Brant, J. C., and Stanley, K. O. Benchmarking open-endedness in minimal criterion coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2019), pp. 72–80.

**2** Ganin, Y., Kulkarni, T., Babuschkin, I., Eslami, S., and Vinyals, O. Synthesizing programs for images using reinforced adversarial learning. *arXiv preprint arXiv:1804.01118* (2018).

**3** Wang, R., Lehman, J., Clune, J., and Stanley, K. O. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.0175* (2019).

## 3.8 Applications of Artificial Intelligence in Live Action Role-Playing Games (LARP)

*Christoph Salge (New York University, US), Mike Preuß (Leiden University, NL), Spyridon Samothrakis (University of Essex – Colchester, GB), Emily Short (Spirit AI – Oxford, GB), and Pieter Spronck (Tilburg University, NL)*

This abstract covers the results of the working group on how artificial intelligence techniques, particularly those commonly used in AI for Games, could be applied to Live Action Role-Playing (LARP) games and similar experiences. The results have been expanded on in a paper which has been submitted to a high-level conference.

### Introduction

How could an artificial intelligence help a pretend paladin hunt an orc through a forest? The research field of Artificial Intelligence in Games, which should technically be able to answer this question, separates into two sub-fields. One dealing with how to make AIs that can play games to win, the other sub-field asking how AI can enhance the game experience. For both fields it is an interesting exercise to expand the scope of what a game is. Live-Action Roleplaying (LARP) is usually considered a game, or game-like experience, and it is gaining in popularity. In 2001 there were an estimated 100,000 active LARP players worldwide. In 2019 the German LARP calendar[2] lists 517 public events, with sometimes hundreds to (in rare cases) thousands of participants.

Using an AI to play such a game well seems to be a somewhat far-fetched endeavor – but as past AI predictions have shown, one should be careful in prophesying. It is imaginable that some day we will have a fully embodied AI that convincingly plays make-believe in a shared imagined world, navigating both the physical and the interpersonal challenges, while figuring out how to actually "win" in an open-ended, game-like interaction.

For now, we are more interested in AI applications to enhance the game experience in LARP. AI in games research in the past has also focused on game design, believable characters, world building, story telling, automatic game balancing and player modeling [10]. Naively, AI might seem like a poor fit for a game genre that is often associated with a deliberate lack of modern technology. But there are already early attempts to integrate modern technology into LARP [6, 7, 2].

We argue that there is a role for Artificial Intelligence in LARP – especially when focusing on those AI technologies that have already been successfully applied to other game genres. Furthermore, we think that the unique properties of LARP open up some interesting research directions for game AI. Moveover, AI can open some new possibilities for LARP gaming, e.g., super-LARPs that are hardly imaginable without AI. We should note that, going forward, we use the word AI rather loosely, and inclusively, to refer to a range of technologies from machine learning and neural networks to rule based systems and statistics. We also include those technologies resulting from such approaches, such as natural language processing.

[2] http://www.larpkalender.de/

**Figure 6** Illustrative photo depicting a scene from a typical fantasy LARP. Players in costumes act out a scene in front of a backdrop of forest and period tents. Photo by anonymous, with permission.

### Overview

The most common form of LARP sees participants come together, dressed as characters from a fictional world, and interact with each other in a live setting (Fig. 6). A LARP can be explained as an improvisational theater play in which you are playing a character, know about the other characters and the world, but do not have a script to follow [9]. There are many other traditions and corresponding descriptions of what LARP is. [3] provide a good overview. Central are usually a physical embodiment of at least some players as their character.

One interesting, and for our purposes relevant, distinction between table-top (role-playing) games and LARPs is the decomposition and realization of the different functions of the game master. In the first the game master (GM) usually interacts verbally with a group of players, sitting at the same table with them. The GM acts as a storyteller, quest provider, information provider, arbiter of rules, world simulator, and often also host. A lot of those functions operate on the fictional world that resides in the GMs mind, and players can interface with this world verbally. By stating their actions or speaking they affect this world, and the resulting effects are usually relayed to them with speech by the GM, all through one person. In LARP, particularly in larger ones, these different functions can be performed by different crew members, or in some cases other players, or the actual physical world. It is also typical that the different members in the organization team have different roles and responsibilities. While some crew members might be in charge of the games plot or overall event details, others might just be there to monster [8], i.e. play opponents and non-player characters, without having a understanding of the overall game or plot. This decomposition

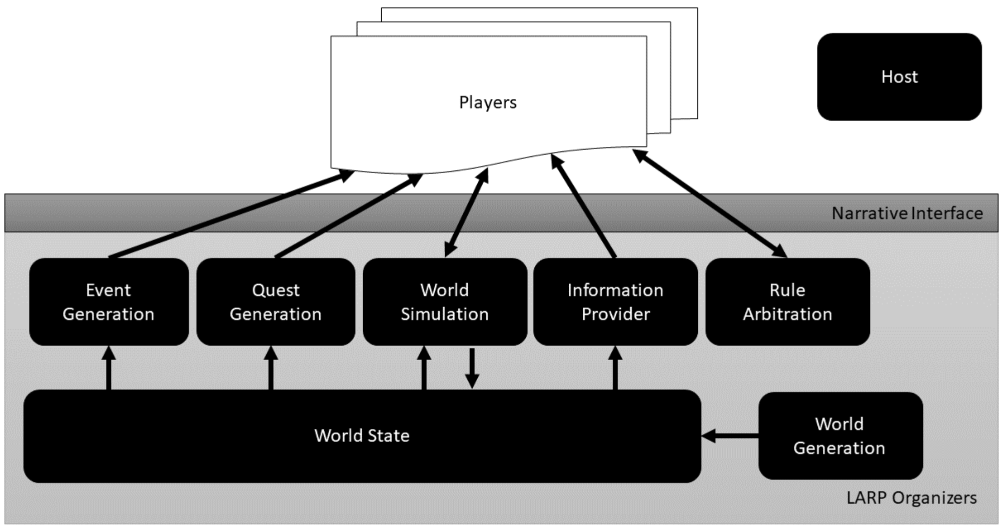

**Figure 7** Decomposition of different functions / roles for LARP organizers. Arrows denote flow of information. In tabletop role play all functions in the grey box are usually performed by the game master, a single person, and all interactions are mediated by a single, narrative interface. The interface in LARP is between each function and the players, and can take different forms.

provides a challenge, but also an opportunity to solve different aspects of the AI game master problem separately. Fig. 7 provides a diagram of the functions we now describe in detail. We should note that this is largely oriented at big, entertainment focused, UK-style LARPS. Other LARP traditions may have similar roles, but their functions or limitations might be slightly different.

## World State

The game world is at least partly fictitious. If a player were to kill an non-player character, then the character's death would be part of this fictitious world state. Facts like, for example, the identity of the chancellor of a fictitious in-world nation is also part of this world state. All the relevant information about the game world has to be (a) generated in the first place, (b) kept consistent with actions taken by the player and events in the world, and (c) used to inform events that happen. This data must generated at the start of the game, and be stored and synchronized, sometimes over multiple sessions.

**World Simulations**: When players act in the world, the effect of their actions has to be determined. Some actions of the player have to be perceived and acted upon by the organizers. This kind of world simulation, or effect on the world, is a typical bottleneck. The more players are participating in a given LARP, the more difficult it becomes to feed back all this information on the larger world state, and react to it appropriately.

**Event Generation / Storytelling**: Many LARPs feature a plot presented to the players in the form of events. The scale of LARP makes it hard to produce events tailored to specific players. As a result, many LARPs often have one person who keeps track of the current state of the world, decides on most major events, and integrates new information into the world model. That person is often overworked, struggling to keep track, or required to make design decisions. The limits of a single person's memory and speed of decision-making puts an upper bound on how much the players can influence the world and the planned order of events.

**Quest Generation**: In LARPs players often get quests. It is difficult to provide different quests for all people. A common quest for a group, or even all people is common. To compensate, players often come up with their own personal quests, similar to how real people might decide on life goals. Self-given, intrinsic goals tend to be of a different nature than extrinsic goals given by the game master. Quest generation ideally should be connected to the overall state of the world: quests should make sense, or be responsive to what is happening in the world, or their success should update the world state.

**Information Provider**: A common role for non-player characters (NPCs) is to provide information about the world to the player. Players are usually only aware of what they observed, so it is important to inform them about events or relevant facts. Information can be provided by documents or objects, but the most common approach is to have a NPC that brings up the relevant information "naturally" in conversation, or is specifically sent somewhere to inform players. This means that these NPCs need to be briefed.

**Rules Arbitration**: Most LARP events rely on an honor system in which player self-police their compliance with the game rules. For example, it is common that players have life points, and count themselves how many they lose in any given fight, determining themselves when they die. Nevertheless, there are usually a given number of referees that are there to arbitrate in case of conflict. Rules arbitration is particularly needed when hidden information is involved.

**Hosting**: In addition to managing the game's fiction, a LARP organizer is usually also responsible for hosting the event. This might include ensuring that there are adequate sleeping, food and hygiene arrangements, and that everyone is feeling safe and comfortable. This creates obligations to ensure a space that is free of dangers, harassment, etc. As with other games, it should be possible to withdraw from the game at any point without any negative real world consequences.

**Summary: Decomposition**: The described roles are not exhaustive, and they do overlap and interact to a certain degree. In smaller LARP events, several roles might be carried out by the same person. Nevertheless, the decomposition of roles is a distinguishing feature of LARP, and gives rise to some of the following challenges and opportunities:

### Physicality

Most of a LARP is carried out in the physical world. This entails that real world phenomena interact with the fictional world and the world state.

### Coherence

In an ideal world every meaningful player action updates the world state, and this update is communicated to all relevant parties. Such level of responsiveness is often not feasible for LARP. Because there is no objective external reality against which this information can be verified, wrong information propagates more within the LARP than it would in real life. Communication technology can address many of these issues. However, organizers often design their games to combat this decoherence – sometimes by providing fictional reasons why players cannot influence world state too much, and NPCs do not have more than a limited amount of information about the world.

### Scalability

Scaling up a LARP often means adding more people to organizing roles, or splitting the roles among more people, and hence increases the challenge of keeping everyone in sync. It also means that more (coherent) content must be created to keep the game interesting for all players.

### Immersion

There is an aim to keep a LARP immersive by removing everything that is counter to the world that is being presented, i.e., modern technology might be banned, even though such technology can be used to help the organization. Devices can be camouflaged as something else, and the often present concept of "magic" can be used to explain modern equipment or capabilities. It is also possible to have technology work in the background, and then design an appropriate interface that connects that technology to the game and the players.

### Robustness

In many LARPs there is an understanding among players that they are not just consumers of an experience, but are actively helping to create the same experience for others. Player are usually willing to overlook minor problems and help to make LARPs work. This comes both in the form of being willing to adapt and interpret inconsistent clues, and in a willingness to improvise to fill the gaps. This gives LARPs a certain degree of robustness.

## Possible AI Applications

In this section we want to present some sketched out examples of how existing, or conceivable AI approaches, particularly those from the AI in Games domain [10, 11], can be applied to LARP. The goal is to both make organizing LARPs easier, by overcoming the previously outlined challenges, and to enhance gameplay in a way not possible without AI.

### Conversational Agents

There is currently tremendous interest from the AI community towards building conversational agents – commonly known as chatbots. The technology is not mature yet to the point where general conversations can be had. How could these chatbots be used in LARP?

One common interaction between players and referee is asking for rules clarification, such as "Can I dodge epic damage?" This is a role that could be filled by a straightforward question-answering system. As this is an "out of game" interaction, such a system could be deployed on a smart phone, or similar device, and there would be no need to simulate a character for that AI. This would already free up a large chunk of the time spend by the organization team.

Taking this a step further would be a conversational system that could provide "in game" information, such as "Who is the current ruler of this place?" In a most simple case, this could also just be a digital device with an question-answering system trained on a fixed text corpus, detailing the background of the world. But there are opportunities for improvement here. To further the immersion the conversational AI could be imbued with character traits that manifest in the way it speaks. Taking this even further would be to employ techniques that give an appearance of agency. Interactions with the AI could lead to changes in its mood – and the forms of interaction available might depend on past interaction.

There is also increasing interest in the concept of "virtual humans" – persistent non-player characters who might interact with a player via a combination of games, VR experiences, and social media. The technology developed to support such characters for marketing, theme park, and entertainment applications might also be suitable for deployment in LARPs.

### Embodied AI Agents

There are also opportunities to having the previously mentioned AI agents physically embodied in the world. For example, a smart phone could be stored in a talking magical book. It could use its GPS sensor to determine its current location, and then trigger certain interactions when it is carried into a certain area. A conversational AI disguised as a familiar might turn into a game companion that rides around on the player's shoulder.

### Drama Management: AI director

Narrative designers of conventional video games often use a system of storylets or quality-based narrative, in which story events are triggered whenever some pre-conditioned state is reached in the game world. They write individual moments that they want the player to experience at some point, and then allow the system to select the point when those moments are best presented during a particular player's playthrough. LARP creators have written about building LARPs with similar gameplay beats in mind. In the case of existing LARPs, it typically falls to human GMs to determine when the moment has arrived to deliver a story beat, and there is little room for last-minute customization. An AI system able to track key elements of world state, however, would be able to select when to activate particular storylets, and potentially use grounding techniques similar to those used in video games to fill in elements of the delivery, customizing the story moment to the exact parameters that allowed it to be fired off.

### AI Content Generation

Before a LARP is first run, there is usually a phase where the organizers create a world and setting. There are two ways in which procedural content generation could help human designers with this. In a mixed initiative co-creative approach an AI system could produce fictional history or relationships, and a human designer could then select and refine. The system could also be used to generate inspiration of ideas.

On a more practical level an AI system could provide more complexity after the rough brush strokes have been filled in by a human designer. This might be useful to engage players who want to engage in a more scholarly play style. Once a system like this is set up, it would be easily scaleable – in theory one could provide a whole library worth of research that is both related to the world, and allows for relevant research to be played out.

Logic puzzles are a common element to LARPSs to serve as proxies for scholarly investigation. So a tablet representing a digital lock might require you to solve a sudoko or play a match three game to hack it. Existing AI techniques can already create a wide range of puzzles of a given type, allowing to add variability, or even design personalized puzzles in real time, related to the stats of the character impersonated by the player, or related to the world.

### AI Story Hooks

Another opportunity for AI content generation is to produce story hooks or quests for players – providing for more micro-questing in larger LARPs. There are already existing approaches to automatic quest generation, usually looking at existing NPCs, their role in the world ontology, and their desires [4, 5, 1]. These tools could be easily modified to provide a range of quests to players, if a data based representation of the world and the players exists. These quests could then be offered in between events to players, as part of their event briefing package, or even prior to the event in digital form. This would allow the player to accept and reject certain type of quests – which could enable player modeling and more personalized quest generation. A player might, for example, not be interested in performing any illegal activity, as it clashes with their character concept.

### AI-aided World Simulation

A big element of LARP, an a defining characteristic, is how the player interacts with the real and the game world. Many interactions of the players have immediate consequences provided by the environment and the other players. But there are usually also interactions with the fictional world where the player affects parts of the world that are not physically present. Here an referee usually has to interpret the players actions and determine how this affects the fictional, invisible world, and what consequences in they physical game world result from this. This is usually a massive bottleneck, resulting in design choices that aims to reduce the amount of this kind of interaction. AI could help by providing part of this complex world simulation.

### Super LARP and Information Spread

Even more AI support than for single AI characters transferring information from Game Masters to players would be necessary if two or more LARPs would be connected in a way that they play within a common framework storyline. The more simple case would be succession in game time, so that one LARP takes up the open ends from a previous LARP. If, however, AI can support design and organization of a single LARP, there is no principal obstacle for several coordinated LARPs taking place at the same time.

## Conclusion

In summary, we believe that LARP is a domain well suited for the application of AI and AI and Games techniques. The listed, existing approaches demonstrate this, and the speculative examples show a range of relatively straight-forward extensions of existing AI and Games techniques, so they would be suitable for LARP. Doing so could overcome several existing challenges for LARP organizers, such as scalabiltiy and content generation issues. It could also provide for new forms of play that would not be possible without AI. LARP also provides an interesting test-bed for AI applications, particularly those that want to explore the interface between humans and AI, or how AI can interact with the physical world. Here the robustness of LARP, caused by the willingness of the participants to correct errors on the fly, could provide valuable for researchers. In general, AI in LARP research offers several unexplored opportunities, both to enhance the experience of players, and to explore the limitations and challenges of AI.

### References

1 Calvin Ashmore and Michael Nitsche. The Quest in a Generated World. *DiGRA Conference*, 2007.

2 Ella Dagan, Elena Márquez Segura, Ferran Altarriba Bertran, Miguel Flores, and Katherine Isbister. Designing "True Colors": A Social Wearable That Affords Vulnerability *CHI '19*, Association for Computing Machinery, New York, NY, US, 2019.

3 J Tuomas Harviainen, D Simkins, J Stenros, E MacCallum-Stewart, and D Hitchens. Live-action role-playing games. *Role-playing game studies: transmedia foundations.* Routledge, London, 2018.
4 Ben Kybartas and Rafael Bidarra. A survey on story generation techniques for authoring computational narratives. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(3):239–253, 2016.
5 Lee Young-Seol and Sung-Bae Cho. Dynamic quest plot generation using Petri net planning. *Proceedings of the Workshop at SIGGRAPH Asia*, 47–52, 2012.
6 Elena Márquez Segura, Katherine Isbister, Jon Back, and Annika Waern. Design, appropriation, and use of technology in LARPs. *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 1–4, 2017.
7 Elena Márquez Segura, James Fey, Ella Dagan, Samvid Niravbhai Jhaveri, Jared Pettitt, Miguel Flores, and Katherine Isbister. Designing Future Social Wearables with Live Action Role Play (LARP) Designers *CHI '18*, ACM, New York, NY, US, 2018.
8 Laura Mitchell. Volunteers as monstrous workers:'monsters' in UK live-action roleplay game organizations. *Culture and Organization*, 25(3):233–248, 2019.
9 Lizzie Stark. *Leaving Mundania: Inside the transformative world of live action role-playing games.* Chicago Review Press, 2012.
10 Georgios N. Yannakakis and Julian Togelius. A Panorama of Artificial and Computational Intelligence in Games. *IEEE* Trans. Comput. Intellig. and AI in Games, 7(4):317–335, 2015.
11 Georgios N Yannakakis and Julian Togelius. *Artificial intelligence and games* 2, Springer, 2018.

## 3.9 Procedural Content Generation for Minecraft

*Christoph Salge (New York University, US), Alan Blair (UNSW – Sydney, AU), Alex J. Champandard (nucl.ai – Wien, AT), Michael Cook (Queen Mary University of London, GB), Raluca D. Gaina (Queen Mary University of London, GB), Diego Perez Liebana (Queen Mary University of London, GB), Jacob Schrum (Southwestern University – Georgetown, US), Emily Short (Spirit AI – Oxford, GB), and Vanessa Volz (modl.ai – Copenhagen, DK)*

**Main reference** Christoph Salge, Michael Cerny Green, Rodrigo Canaan, Julian Togelius: Generative design in minecraft (GDMC): settlement generation competition. FDG 2018: 49:1-49:10

This abstract reports both on a talk and on a subsequent working group with the same title. The initial presentation was focused on the GDMC AI Settlement Generation Challenge in Minecraft [1]. This is a competition where participants write an algorithm that can generate an "interesting" settlement for a given, previously unknown Minecraft map. The general idea behind the competition is to put some focus of creativity and co-creativity as part of intelligence, and consequently as part of artificial intelligence. Currently, the main focus is on creating a settlement that fits into, and reacts to the environment given by an input map. The output should be similar to what a human could do. Further down the line there are plans to further extend the competition, to have AIs complete partially built settlements, and possibly do so from an embodied perspective. Currently, the AI interact with the map as a 3D representation. While the GDMC challenge hopes to stimulate research in this direction, it also aims to a.) solicit some ideas and approaches from the general public and b.) offer a joint framework where different technical approaches, such as wave form collapse or PCGML can be compared with each other.

The first part of the working group centered on discussing the details of the competition, its merits to science, and how to further extend and enhance the competition. Currently, all generated settlements are evaluated by humans, who are given a set of criteria – namely adaptivity, functionality, evocative narrative and aethetics. We discussed if those criteria could be automated – and how. Several of the working group attendees were actually expert judges for the competition, and shared their experiences. We also reviewed the results from the two previous years and looked at generated settlements from past competitions and discussed the used PCG techniques.[3]

We discussed different submission formats, and the newly added Chronicle challenge [2]. The goal there is to generate a text that is somehow about the generated settlement – for example, a historical narrative or a tourist guide. The main criterion for this is "fit", i.e. how well does a written text fit with, and only with, a certain settlement.

The second half of the working group was focused on coding up examples and trying out different approaches. Participants looked at various ideas, such as generating signs that reflect the features of the landscape, piping information from the settlement generation process into an existing travel diary generator, or developing a twitter bot that could walk around the settlement and take screenshots of interesting places and automatically tweet them out.

While there is no direct output planned for this working group, we should note that the GDMC competition is running again in 2020 (Deadline is the 30th of June), and several participants are considering to submit their own entries at this point.

**References**

1 Christoph Salge, Michael Cerny Green, Rodgrigo Canaan, and Julian Togelius. Generative design in minecraft (gdmc) settlement generation competition. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, pages 1–10, 2018.

2 Christoph Salge, Christian Guckelsberger, Michael Cerny Green, Rodrigo Canaan, and Julian Togelius. Generative design in minecraft: Chronicle challenge. page 5, 6 2019. 10th International Conference on Computational Creativity , ICCC 2019.

## 3.10 Black Swan AI

*Spyridon Samothrakis (University of Essex – Colchester, GB), Nantas Nardelli (University of Oxford, GB), Günter Rudolph (TU Dortmund, DE), Tómas Philip Rúnarsson (University of Iceland – Reykjavik, IS), and Tom Schaul (Google DeepMind – London, GB)*

Currently deployed AI systems has trouble acting on patterns what they were not trained with, though there are hints that humans can do that without much trouble. In this group we discussed how this might be possible and ways that machines could replicate human abilities.

[3] Material available online at http://gendesignmc.engineering.nyu.edu/

## Introduction

The great strides we have seen in (Game) AI in the last 10 years or so seem to suffer from a serious lack of robustness; agents fail to learn concepts that will help them generalise in cases were the training distribution (which often comprises of carefully curated sets of data or example games) is different then the testing distribution (which could include a large subset of real world scenarios). One can make the case that there is probably enough information in the data to act reasonably in an out-of-distribution fashion, but the current crop of AI systems seem to fail at the task. Given that distributions of real-world events might be somewhat "flat", this poses serious problems, as there is no way that an agent could be provided with enough data that contains all the possible configurations of the known universe.

## "Black swan" AI

We broadly and loosely define black swan AI[4] as systems that can act reasonably on a world configuration that they have not directly perceived during their training phase. For example a self-driving car that has never seen a pedestrian crossing the highway should be able to maintain internal coherence and deduce the right action (e.g. push the break pedal).

## Training robust AIs

A serious issue posed in designing "black-swan" robust systems is the difficulty of designing correct testing procedure. Once a test scenario has been recognised, it is somewhat easy for a system designer to include the relevant inductive biases in their algorithm and effectively solve the specific problem instance, without addressing the bigger issue. A potential solution to this methodological problem would involve two researchers, a test generator and a test solution provider. The generator would create a novel problem on a weekly/monthly/annual basis and provide a subset of the data, while keeping an out-of-distribution test set hidden. A solution provider would then attempt to solve the problem and the two would compare the results for the test set only once, as to avoid any information spillage.

## How do humans do it?

We postulate that humans can act on completely fictive scenarios, both in terms of state and action space. For example it is very easy for a human to imagine a battle between her and a TV-come-to-life, though the scenario is extremely unlikely (a television set cannot really fight). It also known that humans can base their decisions and societal organisation on completely fictive thoughts and abstract concepts (e.g. what is a corporation?). In that sense, no world permutation is unimaginable (and hence no black swans in the way we defined it above) – utterly outrageous counterfactuals can be acted upon.

[4] "Black Swan" as a term was popularised by Taleb [1]

### How could machines possibly do it?

Though the training/test set problem identified above is a major issue, ideas that are closely aligned to procedural content generation[2] could prove helpful; one can treat certain environments as one of many possible instances of a generic concept (for example, the game of Mario as a specific realisation within a family of Mario-like environments, where Mario can take radically different actions (e.g. fly), while the shapes of obstacles change from square to random). Another possible way of thinking, especially in the context of vision, is learning computer vision by first learning computer graphics[3]. One could learn a generative model of graphics and go back to learning models of acting on generated instances, which can now be made to comprise of the totality of vision.

**References**

1 Nassim Nicholas Taleb. *The black swan: The impact of the highly improbable*, volume 2. Random house, 2007.
2 Noor Shaker, Julian Togelius, and Mark J Nelson. *Procedural content generation in games*. Springer, 2016.
3 Geoffrey Hinton. Taking inverse graphics seriously, 2013.

## 3.11 Learning to Learn

*Spyridon Samothrakis (University of Essex – Colchester, GB), Bruno Bouzy (Nukkai – Paris, FR), Michael Buro (University of Alberta – Edmonton, CA), Setareh Maghsudi (TU Berlin, DE), Tómas Philip Rúnarsson (University of Iceland – Reykjavik, IS), and Tom Schaul (Google DeepMind – London, GB)*

In this workshop we discussed the design of machine learning systems that learn their learning mechanisms (instead of just learning from data).

One can easily imagine a taxonomy of AI systems scafflolded alongside a path of recursive learnability. Systems of hardcoded rules would form the beginning of a ladder, while systems that learn would from data would be an immediate step up. An obvious and natural extension to a further step would be to have systems that learn how to learn.

This form of meta-reasoning has extensively studied in the literature. We broke down the problem into a set of *interfaces* and *objectives*. Interfaces highlight the method that a learning to learn algorithm would be implemented in, while objectives define goals that we would like to achieve by employing a learning to learn mechanism.

Overall the goal would be to search the algorithmic space defined by a learning process and adapt the learning mechanism. As such, one can easily envision a recurrent neural network being used as the learning mechanism for neural networks, replacing generic variations of SGD with problem specific learning mechanisms. Along the same line of thought, one can easily envision novel hebbian rules [1] being learned, again as mechanisms to achieve a certain objective.

The difficulty of designing good algorithms in this niche seems to lie in the fact that searching through a wide algorithmic space is hard (and possibly prohibitively computationally expensive). If these algorithms are to more broadly accepted as part of the standard AI toolbox, careful design of what is allowed to learn would need to take place.

**Table 1** Examples of interfaces and objectives. Interfaces define the mechanism, while objectives define the goal of a learning to learn mechanism.

| Interfaces | Objectives |
|---|---|
| Learning network connectivity | Sample efficiency |
| Learning to explore | Avoiding catastrophic forgetting |
| Learn initial weights | Robustness |
| Learning hyperparameters (e.g. $\gamma, \lambda$) | Reward maximisation |
| Learning the learning rate | Exploitation Maximisation |
| Learning the update rule | |
| Learning the opponent adaptation | |

## References

1 Joseba Urzelai and Dario Floreano. Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments. *Evolutionary computation*, 9(4):495–524, 2001.

## 3.12 Self-Adaptive Agents for General Game Playing

*Chiara F. Sironi (Maastricht University, NL), Dan Ashlock (University of Guelph, CA), Cameron Browne (Maastricht University, NL), Tom Schaul (Google DeepMind – London, GB), Hui Wang (Leiden University, NL), and Mark Winands (Maastricht University, NL)*

When implementing General (Video) Game Playing agents that play a wide variety of (video) games, it is always hard to find a game-playing strategy or a setup for the agent that is optimal for all games in all circumstances. Therefore, the aim should be to design a game-playing approach that adapts various characteristics of the agent to each game being played. When designing self adaptive agents, multiple aspects should be taken into consideration. It is important to define the objectives to be achieved by adapting the agent. Moreover, we should decide which characteristics of the agent we want to adapt and whether the adaptation should happen on-line or off-line. Finally, we should consider that the structure of game trees might vary substantially from game to game. This increases the complexity of adapting agents to each game. These aspects are discussed next.

## Objectives

First of all, multiple objectives can be identified towards which agents could be adapted. For example, the agent could be adapted to each game to increase its playing *strength*, its *efficiency* or its *robustness*. *Fun* is another possible objective. For instance, when playing against humans, agents could be adapted to the level of the human player to keep the game entertaining. Agents could also be adapted in order to *avoid delusion*. A simple example is when an agent believes to be winning the game because it is actually not searching the parts of the game tree where losing states are present. In this case, search could be adapted to each game to explore interesting parts of the game tree. Finally, another possible objective is *transparency*. A desirable feature of an adaptive agent would be the ability to explain why it is making certain decisions and why it is searching certain parts of the tree for each game it plays.

## Approach

There are also many ways in which the agents themselves can be adapted, for example the *algorithm* they use to play each game. The agent could not only change the algorithm used to play, but it could also adapt the *parameters* that control the game-playing algorithm and also the *heuristics* that such algorithm is using. As a concrete example of adaptation of the search algorithm, Genetic Programming could be used to explore the space of action-selection functions, looking for the one that is most suitable for each game at hand. The performance of an agent on different games or at different stages of the game might also depend on how further ahead the agent is looking. Therefore, the search *horizon* could also be adapted. Another alternative consists in adapting the representation of the *actions* or of the *game model* or of the *rules*. For example, depending on certain characteristics of the game a representation might be more efficient than another one. Finally, for most of the games the search space is too large to be visited exhaustively. Therefore, agents could be designed to adapt the *sampling strategy* that chooses which parts of the tree to visit and how many times.

## Timing

Another aspect to take into account is when the adaptation takes place. A wide spectrum of possibilities is available, that goes form fully on-line adaptation (i.e. while a single run of the game is being played) to fully off-line adaptation. An example of the first would be adapting the agent after each *trajectory* that is explored during the search. A less fine grained alternative would be adapting the agent after each *move* that is performed in the game. Moving towards the off-line end of the spectrum we can find adaptation to each *episode* (or game run), to each game *level*, to each *game* or to each game *genre*.

## Complexity

One final aspect that should be considered is that in General (Video) Game Playing the domain structure is partly unknown. Moreover, game trees have a huge variation in branching factor and depth. Therefore, which technique works on which game might depend on the structure of the game tree. This means that there is the need to understand the game search space in order to be able to adapt the search and explore such space efficiently. An interesting direction for future research would be looking into how search spaces can be visualized. In addition, it would be interesting to visualize and analyze search trees. Investigating metrics that can be applied to search trees might enable us to discover desirable or undesirable properties in a game. These properties might help the process of adapting the agent.

## 3.13 NPC Understanding

*Pieter Spronck (Tilburg University, NL), Duygu Cakmak (Creative Assembly – Horsham, GB), Jialin Liu (Southern Univ. of Science and Technology – Shenzen, CN), and Mike Preuß (Leiden University, NL)*

Non-Player Characters (NPCs) in games exist in a computer-controlled game world, where they exhibit behaviors which they express through conversations and actions. Game developers use a high variety of techniques to create Artificial Intelligence (AI) which controls these NPCs. Common approaches include behavior trees, state machines, and straightforward scripts. The reason to employ such basic techniques is that computational power for running AI in a game is limited. Moreover, these techniques lead to AI which is easy to create, debug, and maintain.

A disadvantage of the classic approaches to implementing NPC behavior is that the NPCs only react in specific ways to events which happen. For instance, if a human player takes an item in the game which, according to the game, belongs to the NPC, the NPC might get angry and attack the player. This is because implicitly the game AI assigns a motivation to the human player for taking the item, labeling the human player as a "thief." However, the motivation of the human player might be very different. No games exist, however, in which the human player can express their motivation, or in which game AI derives motivations in a more intelligent way. For commercial video games, that might actually suffice, but for many serious games, it is necessary that plausible motivations for human player behavior are derived by the game AI.

To assign plausible motivations to players, NPCs need to have some rudimentary kind of "understanding" of the player's behavior in the game world. For example, consider the following situation: an NPC is sitting at a table and eating food from a plate, when the human player enters, grabs some food from the plate, and eats it. Depending on an understanding of the role of the player in the game world, different motivations can be assigned to the player. If the player is in a romantic relationship with the NPC, this might be considered reasonably normal. If the player is a good friend of the NPC, this might be considered a joke. If the player is a stranger to the NPC, this might be considered an insult. Proper responses might be a friendly smile, a confused look, or an angry scowl, respectively.

Ultimately, with a system of understanding, game AI may be able to make suitable predictions on what is supposed to happen after a human player interacts with the game world, and make suitable backwards predictions on how a certain situation has come to pass. For example, an NPC who finds the player, who is a stranger in the village, on their front porch, then discovers that their front door is unlocked and a valuable sword from their home is stolen, might not actually know that the player stole the sword, but backwards reasoning should provide him with at least enough of a suspicion to ask a local guard to frisk the player.

To supply an NPC with a rudimentary understanding of a game world, the following components are needed:

- An entity-relationship model which describes the objects in the NPCs world, and the relationships between the objects (e.g., buildings and their owners). This ER-model can be shared by all NPCs.
- A network of social structures (i.e., what social relationships exist between NPCs).
- A description of social norms, which assign a rating of 'normality' to each action in combination with the objects and social structures involved; if social norms are culture-dependent, then NPCs may have different sets of social norms.

- Optionally, an overview of the history of the game world.
- For each NPC, an internal state, including personal attributes and goals in the world.

Using these components, and a set of potential NPC actions, NPCs can generate a reasonable action list for each moment in time. From the reasonable action list they can choose an action to perform based on their state and attributes. Considering that without interaction of a human player, the 'life' of a group of NPCs will be repetitive, unchanging from day to day, using the set of components listed above, an expected world state can be generated and stored for each moment in a day. The expected world state is simply the state of the world which exists if only NPCs interact with the world.

The expected world state can be compared with the actual world state, and any discrepancies (which are probably caused by interactions of the human player) need to be interpreted. This interpretation is influenced by the place of the human player in the social structure, and an NPC's selection from the reasonable action list can be influenced by it. In a realistic implementation, if an NPC cannot derive a good motivation for a human player's behavior, it can get the option to simply ask for it.

The structure described above can also be used to generate new daily behaviors for NPCs when a player interferes dramatically with the game world. For example, suppose that in a virtual village many NPCs get their daily nutrition from eating a baker's bread, and the player kills the baker, then the NPCs can no longer achieve their goal of "having their hunger quenched" by buying and eating bread. Consequently, they will select different actions from their reasonable action list. If this system is implemented well, it could lead to a different stable state for the village, in which a new baker is appointed or villagers get their food from different sources.

Naturally, implementing a system as described above is a challenging undertaking, but in principle, since game worlds are so much simpler than the real world, a small-scale could be run. Such an experiment could be a lead-in to, in the far future, create AI which has an understanding of the real world, rather than a game world.

## 3.14 AI for Accessibility in Games

*Tommy Thompson (AI and Games – London, GB), Bruno Bouzy (Nukkai – Paris, FR), Alex J. Champandard (nucl.ai – Wien, AT), Michael Cook (Queen Mary University of London, GB), Diego Perez Liebana (Queen Mary University of London, GB), and Emily Short (Spirit AI – Oxford, GB)*

Despite there being approximately 164 million people in the United States playing video games, there has historically been a lack of support for players with accessibility issues. In this workgroup we discuss the open challenges yet to be explored and opportunities for artificial intelligence to be employed as means to enhance player experience.

### Introduction

Video games have become increasingly more pervasive in modern culture, with an estimated 164 million people in the United States playing video games, accounting for 65% of the country [1]. However, while this figure is considerable, there has historically been a lack of support for people who have issues accessing games due to physical or mental health issues or disabilities. Given the nature of the medium and the input devices often required to interact with them, video games can present a variety of challenges for players due to pre-existing health conditions.

It is estimated that of the aforementioned 164 million people in the United States, 33 million of them are disabled [2]. This results in a need for different solutions to address the likes of reading text prompts in user interfaces, to interacting with ‘dual stick’ gamepad controllers on popular games consoles. At the time of writing, the range of accessibility options in games is slowly increasing. With new tools and peripherals such as the Xbox Adaptive Controller providing alternative input methods for common interfaces, to the introduction of hard requirements for video games to provide basic accessibility options in areas of communication [3]. Ranging from text-to-speech systems enabling for users to have menus and other text-heavy regions read out to them as seen in games such as *Tom Clancy’s The Division 2* [5], to the ability to customise font size, colour and background for spoken dialogue and narration exhibited in *Gears 5* [6].

Despite this, many of the accessibility options available still fail to address many of wide-ranging and often bespoke issues that players face. New regulations on accessibility options introduced by the Federal Communications Commission in the United States have taken longer to appear in video games than they have done in other entertainment mediums such as film, and television. With the original regulation introduced as part of the ‘21st Century Communications and Video Accessibility Act’ (“CVAA”) in 2010, With a waiver applied to the video games industry until the end of 2018. While this waiver no longer applies to the industry as a whole, there are still dispensations afforded to smaller independent games companies due to the expected costs when compared to the budget of the project. Furthermore, the FCC regulation on deals exclusively with a subset of accessibility concerns in relation to communication. Hence many other concerns are still largely ignored.

Hence in this workgroup, we aimed to have a discussion about the state of artificial intelligence applications to accessibility in games. The areas within which AI can be applied and envisioned some potential applications based upon existing technologies and more novel applications that are within our capabilities as a research community.

## Accessibility: Wants and Needs

The notion of ‘accessibility’ is rather broad and could be considered to encapsulate a variety of different users, including:

- Players who would enjoy additional features to enhance their experience.
- Players for whom accessibility features help remove issues that have a small impact upon gameplay.
- Players for whom accessibility features enable them to enjoy playing a given game. Without these features, it is highly probable they would not be able to play the game.

What is clear is that there is a difference of *wants* versus *needs* when identifying and designing potential accessibility tools. In the majority of cases, a tool that satisfies a need will cater to all audiences. A good example is use of subtitles for spoken dialogue. Given players with hearing loss need these solutions to understand what it happening, while others may prefer them to enhance their experience. This behaviour has been exhibited in recent video game releases, with publisher Ubisoft acknowledging 60% of players turned on subtitles in games released in 2017. However, when they made subtitles enabled by default in later releases, only around 5% of players turn them off [4].

## Opportunities for AI

At present there are not many artificial intelligence applications, if any, that focus specifically on providing support to video game players with accessibility needs. However, when we consider specific problems areas, there are existing applications and research areas that could turn towards game-specific challenges. For example, text-to-speech (screen reader) technologies could be adapted to read the game world in a concise and useful manner for a player to make informed decisions. This brings its own challenges of understanding what information is pertinent to the player at that time and would make for an interesting body of research: given the needs and priorities of the reader would shift for each player and game that they play. Meanwhile input interpolation and prediction used in motion control could be adapted to better understand the intended movement from a users with limited motor functions.

Conversely, there are established problems emerging within the video games industry that could benefit from further research and development. Colour blindness filters for user interfaces are recognised as a desired feature within combat games, but there is little consensus on how to solve the problem. Filters are provided for known colour blindness types (deutronopia, tritonopia), but there is a lack of consistency from one game to another. Working with players through an interactive learning process to train the desired colour blind filter could be a useful feature – applied at the level of a PC or console operating system such that it can be applied and later modified or improved on a per-game basis.

## Future Steps

While the authors are confident of the capabilities of artificial intelligence and the community's capacity to address these concerns, what we lack is a strong understanding of the problem. It would be presumptuous to initiate a concerted research effort into these problems without understanding what are the larger concerns of disabled players and those who support them. As such, further research in this area should commence by initiating discussions with the one or more charities and other registered bodies who provide support, resources and advice to disabled players. Two notable examples being the 'AbleGamers'[5] charity within the United States, as well as 'Special Effect'[6] in the United Kingdom. With a more thorough understanding of some of the larger issues faced by disabled players a more concise and informed position can be made on how to proceed.

### References

1 Entertainment Software Association. *2019 Essential Facts About the Computer and Video Game Industry*, August 2019.

2 AbleGamers *The AbleGamers Foundation – Helping Disabled Gamers Everyday!*, YouTube (AbleGamers), 2012.

3 Frankfurt Kurnit Klein & Selz PC. *Video Games With Advanced Communications Services Must Now Be Accessible to Players With Disabilities*, January 2019.

4 GamesIndustry.biz. *Ubisoft Sees High Acceptance for Opt-Out Subtitles*. June 2019.

5 Massive Entertainment. *Tom Clancy's The Division 2*. March 2019

6 The Coalition. *Gears 5*. September 2019

[5] https://ablegamers.org
[6] https://www.specialeffect.org.uk

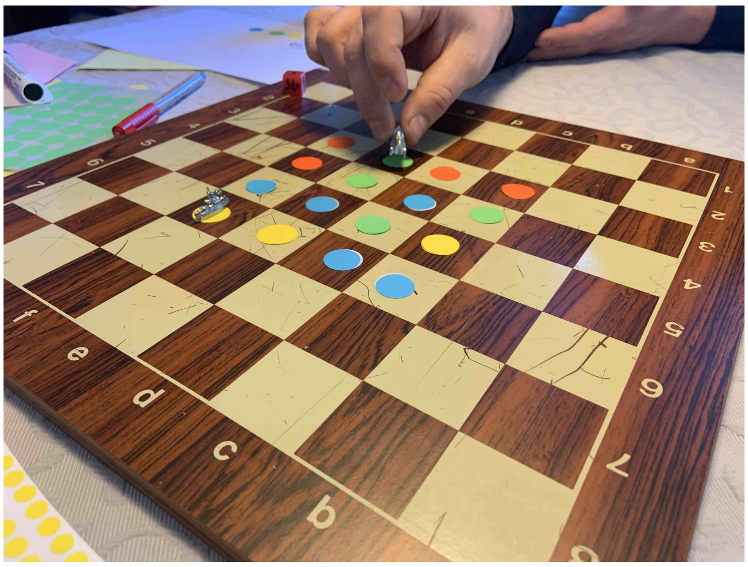

**Figure 8** A game being played by us during the workshop.

## 3.15 Consistor

*Julian Togelius (New York University, US), Jialin Liu (Southern Univ. of Science and Technology – Shenzen, CN), Mike Preuß (Leiden University, NL), Sebastian Risi (IT University of Copenhagen, DK), Jacob Schrum (Southwestern University – Georgetown, US), Tommy Thompson (AI and Games – London, GB), and Georgios N. Yannakakis (University of Malta – Msida, MT)*

Consistor is a game where you must always take the same decision when faced with the same situation. In other words, you are only free to choose what action to take the first time you are faced with a new situation. Our workshop designed single- and two-player versions of the game and AI-based methods for refining the game. We foresee the use of search algorithms and evolutionary computation to help playtest and analyze the game. More (or less) interestingly, we also made connections between the central ideas of the game and certain ideas from ethics, psychology, and media studies.

### Single-player Consistor

Firstly, we designed a single-player version, in which a player starts from a vertex of the game board and aims at reaching the vertex at the diagonal line. When the game starts, no rule has been defined, the player designs the game rules by making a decision at his/her first visit to each colour of tiles. And then, when a colour is visited again, the player should follow the defined rules. Figures 8 and 9 provides some illustrative examples. We have tested the game by generating boards randomly and moving the starting and target location around. In general, this game is easy to solve when the board dimension is low.

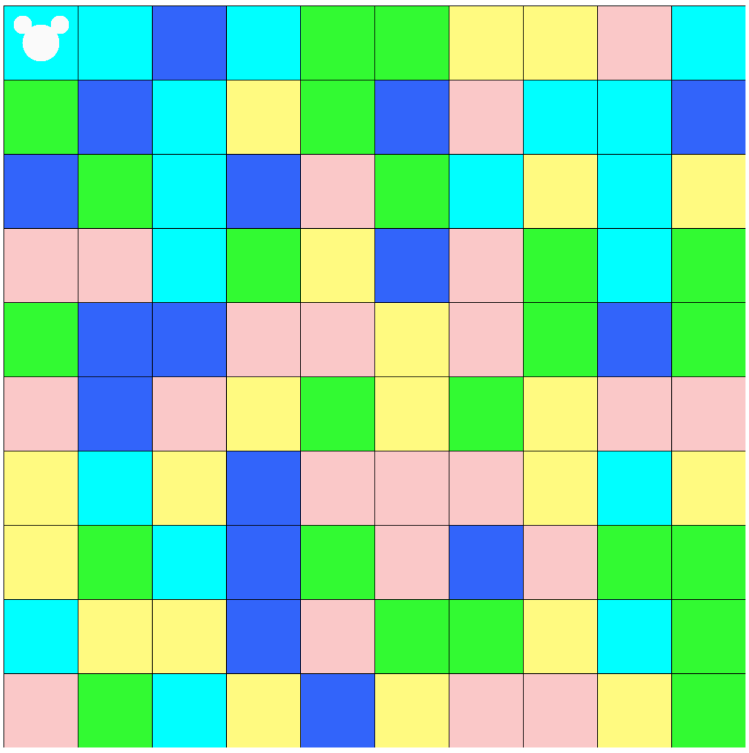

Figure 9 A randomly generated playable 10 × 10 board for single-player Consistor.

## Two-player Consistor

It is straightforward to extend the Consistor game to a two-player version. We considered a simple cooperative version and a more complex version with a payoff matrix in order to imitate the ethics choices and and psychology conflicts in real life.

### Cooperative Consistor

In the cooperative version, two plays start from different locations and take actions in turn, thus, they define the game rules together. The result of a game can be determined by the following rules: (i) when a player reaches the target location, then both win the game; (ii) if a player moves out of the boundary of the board, then both lose; (iii) if a player fails into an infinite loop, then both lose. The players need to be careful to do not kill the other by defining detrimental games rules which force the other to move out of the boundary or repeatedly go left-right or up-down in tiles of certain colours.

Collaboration in this game is not as easy as we thought. Out of 4 playtests by human players, only one ended with a win, two ended with an "infinite loop" and one ended with the "death" of a player. Often, one player puts his teammate in danger when trying hard to approach the target location. The attraction of winning the game by reaching the target alone certainly makes the players act in a selfish way and forget about the situation of their teammates. Two examples are given as follows.

**Example 1**: Considering Figure 10 and the fact that

- it's the player A to make a move, and
- the rule for green-coloured tiles is not defined yet.

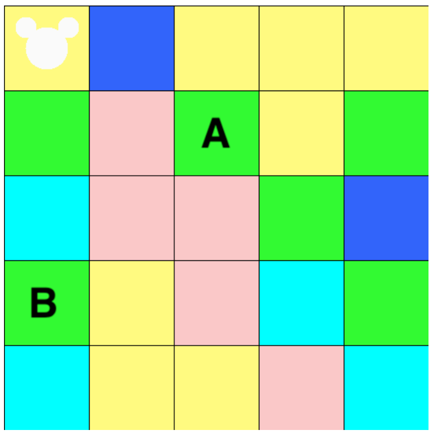

**Figure 10** Player A kills player B by moving left.

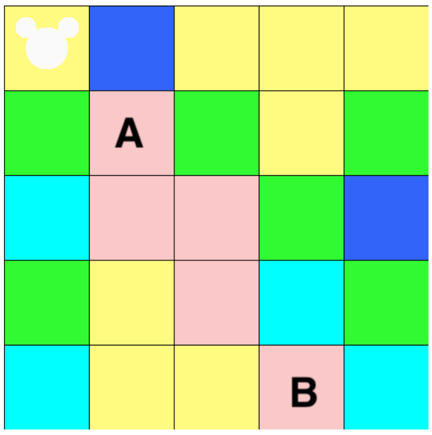

**Figure 11** Player A kills player B by moving left.

If the player A moves left, then the player B will have to follow the rule by moving left and loses the game according to the criterion (ii).

**Example 2**: Considering Figure 11 and the fact that

- it's the player A to make a move,
- the rule for pink-coloured tiles is not defined yet, and
- the rule for yellow-coloured tile is already defined as "move to right".

If player A moves left, then the player B will have to move left and arrive at the yellow-coloured tile. The player B will fail into infinite loop and lose the game according to the criterion (iii).

### Consistor with a payoff matrix

To further imitate the ethics choices and and psychology conflicts in real life, we design a more complex two-player Consistor with a payoff matrix, as given in Table 2.

**Table 2** "O" refers to being out of the boundary. "I" refers to being in infinite loop. "K" refers to killing the other player, i.e., the other player is in state O or I. "M" refers to meeting each other.

| $State_A$ | $State_B$ | $Score_A$ | $Score_B$ |
|---|---|---|---|
| I/O | I/O | 0 | 0 |
| K | I/O | 1 | 0 |
| M | M | 3 | 3 |

Surprisingly, the same human players in the previous playtests showed better collaboration in this game than in the actual cooperative one and often ended with a double-win. The observation makes us rethink our design of cooperative games. Are the games actually designed as how we wanted? Or in other words, do the players behave as how we expect/predcit when designing the games?

## Summary and Future Steps

Our initial motivation was to design a game based on consistency to imitate the human behaviour when making decisions in same situations, but we ended with some interesting observations when playing the designed two-player versions. This helps us make connections between the central ideas of the game and certain ideas from ethics, psychology, and media studies. A work is progress is using evolutionary computation to generate playable game boards of different difficulty levels and using search algorithms to solve and evaluate the designed games.

## 3.16 Game-Playing Agent Evaluation

*Vanessa Volz (modl.ai – Copenhagen, DK), Yngvi Björnsson (Reykjavik University, IS), Michael Buro (University of Alberta – Edmonton, CA), Raluca D. Gaina (Queen Mary University of London, GB), Günter Rudolph (TU Dortmund, DE), Nathan Sturtevant (University of Alberta – Edmonton, CA), and Georgios N. Yannakakis (University of Malta – Msida, MT)*

The goal of this working group was to analyse and improve how game-playing agents are evaluated. While various competitions (GVGAI, pathfinding) and milestone-benchmarks (e.g. Chess, StarCraft II) exist, it still remains difficult to draw general conclusions about the strengths and weaknesses of a given algorithm from existing results. This is mainly due to the lack of comparability between different ways of evaluating an algorithm.

The same issues make reporting on the performance of game-playing agents in publications difficult. Evaluation results are (1) likely not generalisable to other games / other domains and (2) it is often difficult to find comparable results from other publications.

As a first step towards alleviating these issues, we developed a general framework that describes several aspects of evaluating game-playing agents with the intention to define a standard of best-practices. If adopted widely, this could facilitate comparisons of different algorithms and at the same time ensure good scientific practise.

The main output of this working group was a list of aspects under which algorithms can be compared, grouped into different dimensions. In lieu of presenting the whole list, we give an overview of these dimensions below. The full list, along with 2 example applications can be found in the supplementary material.

1. Objective Measure: To compare AI approaches, the goal they are trying to achieve should be clearly stated. In-game score or winrate are the most common objectives in current research, but more subjective aspects, such as e.g. believability, play an important role as well.
2. Constraint Measures: This describes under which constraints a comparable AI should be operating. This includes e.g. computational, model and implementation complexity, as well as explainability and human knowledge engineering.
3. Problem Definition: This describes the environment in which the agent is operating. It includes the interface to the game as well as the type and number of different settings the agent is tested in.
4. Scientific Method: This relates to general best-practices in reporting empirical results, including reproducibility, hypothesis-driven analysis and self-reporting.

Our list of dimensions is hopefully a starting point to more discussions regarding meaningful and scientifically sound evaluations of game-playing agents. We plan to develop this further into more concrete guidelines in the future.

## 3.17 Search in Video Games

*Mark Winands (Maastricht University, NL), Dan Ashlock (University of Guelph, CA), Michael Buro (University of Alberta – Edmonton, CA), and Chiara F. Sironi (Maastricht University, NL)*

To make sound decisions in video games, players also have to plan ahead, just as they have to do for classic board games such as chess and checkers. This is not trivial, as it is not always obvious which actions can be taken to accomplish a goal in an effective way. For an example, for a real-time strategy game a scenario could be that agents first have to cut some trees in order to reach a mining location much quicker.

Video-games can have several properties that make search much harder than in abstract games. First, they have the real-time property, there are no turns, and players can make a decision at any time in a continuously changing world. Next, there can be an infinite number of actions, which also can be durative. These games have also a high degree of non-determinism, i.e., an action could have several outcomes. They are also partially observable, players cannot see the complete map. There is also incomplete information, the exact rules and underlying structure of the game is not known in advance. Rewards could therefore be inaccurate or even deceptive. This impact could be even stronger as rewards are sparse. Finally, there are even multiple agents (so-called NPCs) for which a player can make separate decisions at the same time.

The challenge is to abstract the search space of such a complex video game into an extensive form game, which makes game-tree search algorithms feasible. The amount of detail left out is dependent on the situation, and an abstraction mechanism should be therefore adaptive.

Abstraction can be twofold. 1) They either abstract action sequences into so-called macro-actions. Search techniques investigate these macro-actions in order to find a rational strategy. Scripts will then execute these macro-actions by making the low-level decisions, so called micro-actions. 2) They abstract the states as well. For example, by transforming the game levels to a tile-based graph. The associated actions in these abstracted worlds are high-level as well. For instance, movement can be a macro-action by going from one tile to another, where the micro-actions would handle the obstacles in the tile themselves.

In order to be able to search, the macro-actions should have access to the forward model. As the forward model is used to simulate the micro-actions, it should run faster than the game speed. The built-in forward models of many video games are rather slow, making effective search daunting. The solution is to learn a fast forward model. The approach would be as follows. A set of macro-actions is constructed, and their outcomes when applying it on the existing forward model are recorded. The next step is to learn to predict the outcome of a particular macro-action for a particular state. The resulting learned model will replace the existing forward model, which will speed up the search, as there are no micro-actions to be simulated anymore. An AlphaZero approach could be used to learn these mappings.

### 3.18 Forward Game Design Models

*Georgios N. Yannakakis (University of Malta – Msida, MT), Alex J. Champandard (nucl.ai – Wien, AT), Michael Cook (Queen Mary University of London, GB), Mark J. Nelson (American University – Washington, US), and Julian Togelius (New York University, US)*

Forward models underpin many approaches to general game playing, allowing agents to reason about the future state of the game and perform more efficient search. In this working group we reframed the way forward models are traditionally conceived and instead utilised the notion of a forward model for game design. A model that reveals the underlying processes of game design and predicts the game design state when a particular game design action is taken would allow computational game designers to predict the effect of changes to a game design. Such predictions can, in turn, assist in the automatic generation of entirely new games that adhere to particular game design goals and styles. Capturing the process of game design also offers an autonomous way for unfolding aspects of player behavior and experience as both are naturally embedded within that process. In this working group we explored how one might construct such a forward model, and suggested experimental setups that would allow for data on game design decisions to be gathered.

## 4 Panel discussions

### 4.1 Closing Session

*Pieter Spronck (Tilburg University, NL)*

During the closing session, the participants evaluated the seminar, reflecting on what did and did not work, and looking forward on how the seminar series should continue. The topics below were discussed.

The opening session of the week contained a "game" in which each participant would tell who they are, and then would give a true/false statement about themselves, for which they had to achieve the "biggest possible entropy" (i.e., the difference between number of people who think the statement is true and number of people who think the statement is false being as small as possible). This went fast and was still sufficiently memorable to be a good introduction, and also worked as an ice-breaker.

On the first day we asked participants to anonymously (so that junior participants would not feel intimidated by senior ones) provide a title for a working group, and then wrote the working groups on the blackboard. After that we reduced the number of working groups to an amount for which each group would have about five people. Working groups which were not held were kept on the blackboard, and would either be held later in the week, or canceled by the person who proposed them, and new working groups could be proposed during the week. This approach worked well.

Every morning of the seminar we started with brief talks, usually by participants who wanted to start a working group on a particular topic. This worked well. While the seminar is focused on working groups, and is considered successful in particular because of this, the longer talks in the evenings, which were considered optional and usually visited by about half the participants, were really appreciated. These longer talks (which could take the form of tutorials) should be incorporated explicitly for a follow-up seminar, so that more preparation is possible. This would provide a good mix of working groups and optional longer talks and plenary discussions.

There was some criticism of the intensity of the program. Some participants felt that more "moments of rest" should be included. For instance, it wouldn't be a bad idea to have a walk after lunch almost every day. The practice of taking breaks should be normalized.

There was some criticism on the "diversity" of the participants, though the organizers felt that we actually were not doing too badly at that. However, the suggestion that more (senior) people could be involved in suggesting names for invitees was taken to heart.

There was some criticism on the fact that there was a strong sense of community among a large number of participants, and that those who came from 'outside' this community sometimes felt intimidated. The organization should make more attempts to stimulate those 'outsiders' to talk. It was recognized that there definitely had been made such attempts, it still seems that more attention should be given to this issue. One possibility is to offer every participant the possibility to get a 'mentor' or 'buddy' before the seminar starts, who can discuss with them how they should approach the seminar and what is expected of them. This could also take the form of a list of 'volunteers' who can be contacted to provide such information. In this respect it might be a good idea to make someone reach out privately to potential participants before they receive the invitation, so that they know what the seminar is about and why it should not be missed.

It was suggested that because the participants of the seminar tend to be high-ranked senior researchers and promising junior researchers, due to the style of seminar (mostly working groups) it is the ideal event in which some of the future developments in the field can be determined. It would be a good idea to explicitly take this position early in the week, preferably during the opening session, by discussing "what is new right now and what will the future be like."

A list of potential topics for a follow-up seminar was discussed, among which: (1) bringing back lost topics; (2) 3+ player interaction; (3) human-game AI cooperation and interaction; (4) human-game interfaces; (5) using languages in games; and (6) game evaluation and metrics. Some of these topics (in particular the last two) may be more suitable for a small seminar.

Finally, potential relaxation activities for a future seminar were discussed, among which: (1) playing a MOBA; (2) exploring local food; (3) role-playing games; (4) mini-LARP; and (5) programming simple machines such as Arduino and Makey-makey.

The closing session took about 90 minutes, which is quite long, but the organizers found it stimulating. It showed how engaged the participants were with the seminar, and how interested they were in making a follow-up seminar even better.

## Participants

- Dan Ashlock
University of Guelph, CA
- Yngvi Björnsson
Reykjavik University, IS
- Alan Blair
UNSW – Sydney, AU
- Bruno Bouzy
Nukkai – Paris, FR
- Cameron Browne
Maastricht University, NL
- Michael Buro
University of Alberta – Edmonton, CA
- Duygu Cakmak
Creative Assembly – Horsham, GB
- Tristan Cazanave
University Paris-Dauphine, FR
- Alex J. Champandard
nucl.ai – Wien, AT
- Michael Cook
Queen Mary University of London, GB
- Peter I. Cowling
University of York, GB
- Jakob Foerster
University of Toronto, CA
- Raluca D. Gaina
Queen Mary University of London, GB
- Katja Hofmann
Microsoft Research – Cambridge, GB
- Jialin Liu
Southern Univ. of Science and Technology – Shenzen, CN
- Setareh Maghsudi
TU Berlin, DE
- Nantas Nardelli
University of Oxford, GB
- Mark J. Nelson
American University – Washington, US
- Diego Perez Liebana
Queen Mary University of London, GB
- Mike Preuß
Leiden University, NL
- Sebastian Risi
IT University of Copenhagen, DK
- Günter Rudolph
TU Dortmund, DE
- Tómas Philip Rúnarsson
University of Iceland – Reykjavik, IS
- Christoph Salge
New York University, US
- Spyridon Samothrakis
University of Essex – Colchester, GB
- Tom Schaul
Google DeepMind – London, GB
- Jacob Schrum
Southwestern University – Georgetown, US
- Emily Short
Spirit AI – Oxford, GB
- Chiara F. Sironi
Maastricht University, NL
- Pieter Spronck
Tilburg University, NL
- Nathan Sturtevant
University of Alberta – Edmonton, CA
- Olivier Teytaud
Facebook – Paris, FR
- Tommy Thompson
AI and Games – London, GB
- Julian Togelius
New York University, US
- Vanessa Volz
modl.ai – Copenhagen, DK
- Hui Wang
Leiden University, NL
- Mark Winands
Maastricht University, NL
- Georgios N. Yannakakis
University of Malta – Msida, MT