Pac-Man Conquers Academia: Two Decades of Research Using a Classic Arcade Game

Philipp Rohlfshagen, Jialin Liu, *Member, IEEE*, Diego Perez-Liebana, *Member, IEEE*, and Simon M. Lucas, *Senior Member, IEEE*

***Abstract*—*Pac-Man* and its equally popular successor *Ms. Pac-Man* are often attributed to being the frontrunners of the golden age of arcade video games. Their impact goes well beyond the commercial world of video games and both games have been featured in numerous academic research projects over the last two decades. In fact, scientific interest is on the rise and many avenues of research have been pursued, including studies in robotics, biology, sociology, and psychology. The most active field of research is computational intelligence, not least because of popular academic gaming competitions that feature *Ms. Pac-Man*. This paper summarizes the peer-reviewed research that focuses on either game (or close variants thereof) with particular emphasis on the field of computational intelligence. The potential usefulness of games like *Pac-Man* for higher education is also discussed and the paper concludes with a discussion of prospects for future work.**

***Index Terms*—Artificial intelligence, competition, computational intelligence, evolutionary algorithms, games, Pac-Man, neural networks, reinforcement learning, survey, tree search.**

I. Introduction

PAC-MAN is the most successful arcade game of all time and has been a social phenomenon that elevated the game to cult status: Despite having been released more than three decades ago, interest in the game remains high and numerous research papers have been published in recent years that focus on the game in one way or another. Games in general have always been a popular testbed in scientific research, particularly in the field of computational intelligence, and also in other fields such as psychology, where games may be used as a tool to study specific effects using human test subjects. In the past, most games of interest were classical two-player board games such as *Chess* (and more recently *Go*), but today video games attract equal attention. Numerous competitions now take place every year where participants are asked to write software controllers for games such as *Super Mario, StarCraft, Unreal Tournament, Ms. Pac-Man*, and general video game playing (GVGP). There are also many reasons why research on games is on the rise, both from an academic and a commercial point of view.

Academically, games provide an ideal test bed for the development and testing of new techniques and technologies: Games are defined by an explicit set of rules and the gamer's performance in playing a game is usually defined unambiguously by the game's score or outcome. Games are also immensely flexible and vary greatly in complexity, from single player puzzles, to two-player board games, to extensively multiplayer video games. Techniques developed specifically for game playing may often be transferred easily to other domains, greatly enhancing the scope with which such techniques may be used.

There is significant commercial interest in developing sophisticated game AI. The video/computer games industry generates annual revenues in excess of $23 billion in the United States alone, with a world-wide market exceeding $94 billion [1]. Traditionally, most efforts (and finances) have been devoted to the graphics of a game. More recently, however, more emphasis has been placed on improving the behavior of nonplayer characters (NPCs) as a way to enrich game play.

This paper presents research that makes use of *Pac-Man, Ms. Pac-Man*, or close variants thereof. Despite the age of the game, the number of such studies, particularly those in the field of computational intelligence, has been increasing steadily over recent years. One of the reasons for this trend is the existence of academic competitions that feature these games (see Section III) and indeed, many papers that have been published are descriptions of competition entries. However, research is not restricted to the area of computational intelligence and this overview highlights how *Pac-Man* has also been used in the fields of robotics, brain–computer interfaces (BCIs), biology, psychology, and sociology. The goal of this study is to highlight all those research efforts and thereby paint a (hopefully) complete picture of all academic *Pac-Man* related research.

This paper is structured as follows. Section II introduces *Pac-Man* and *Ms. Pac-Man* and discusses why they constitute a useful tool for academic research. This is followed in Section III with an overview of ongoing academic gaming competitions that focus on *Ms. Pac-Man* and have led to a renewed interest in these games within the academic community. The overview of the peer-reviewed research is split into three parts: first, all research related to developing software controllers is reviewed

Manuscript received December 31, 2016; revised April 7, 2017 and July 6, 2017; accepted July 25, 2017. Date of publication December 14, 2017; date of current version September 13, 2018. This work was supported by EPSRC under Grant EP/H048588/1 titled "UCT for Games and Beyond." *(Corresponding author: Diego Perez-Liebana.)*

P. Rohlfshagen is with Daitum, Adelaide, S.A. 5061, Australia (e-mail: philipp@daitum.com).

J. Liu and S. M. Lucas are with the Department of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, U.K. (e-mail: jialin.liu@qmul.ac.uk; simon.lucas@qmul.ac.uk).

D. Perez-Liebana is with the Department of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K. (e-mail: dperez@essex.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TG.2017.2737145

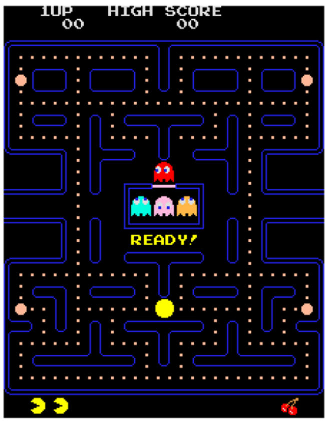

Fig. 1. Screenshot of the starting position in *Pac-Man*: *Pac-Man* (yellow disc) needs to eat the pills (small dots) while being chased by the four ghosts (red, cyan, pink, and orange). The large dots are power pills (energizers) that allow *Pac-Man* to eat the ghosts for a short period of time.

(see Sections IV and V). Then, all research related to game psychology (e.g., player profiles and entertainment values of games) is discussed in Section VI. Finally, Section VII outlines all studies in other areas of research. The paper is concluded in Section VIII where future prospects for *Pac-Man-related* research are discussed.

II. Games

A. *Pac-Man*

Pac-Man was developed by Toru Iwatani and released by Namco in 1980. The game was originally called Puckman but was renamed for the American market. It was released as a coin-operated arcade game and was later adopted by many other gaming platforms. It quickly became the most popular arcade game of all time, leading to a coin shortage in Japan [2]! The social phenomenon that followed the game's release is best illustrated by the song "*Pac-Man* Fever" by Buckner and Garcia, which reached position 9 in the single charts in the USA in 1981. The popularity of *Pac-Man* led to the emergence of numerous guides that taught gamers specific patterns of game-play that maximize the game's score (e.g., [3]). The full rules of the game as detailed below are paraphrased from the strategy guide "Break a Million! at Pac Man" [4].

The game is played on a 2-D maze, as shown in Fig. 1. The gamer has control over *Pac-Man* via a four-way joystick (which can be left centered in neutral position) to navigate through the maze. The maze is filled with 240 (nonflashing) pills, each worth ten points, and four (flashing) power pills (energizers) worth 50 points. The four ghosts start in the center of the maze (the lair), from which they are released one-by-one. Each ghost chases *Pac-Man*, eating him on contact. *Pac-Man* has two spare lives to begin with and the game terminates when all lives are lost. An additional life is awarded at 10 000 points. When *Pac-Man* eats a power pill, the ghosts turn blue for a limited amount of time, allowing *Pac-Man* to eat them. The score awarded to *Pac-Man* doubles with each ghost eaten in succession: 200–400–800–1600 (for a maximum total of 3050 points per power pill). The last source of points comes in the form of bonus fruits that

TABLE I
Characteristics of the Ghosts in *Pac-Man* [4]

Color	Orange	Blue	Pink	Red
Name	Clyde	Inky	Pinky	Blinky
Aggressiveness	20%	50%	70%	90%
Territory	Southwest	Southeast	Northwest	Northeast

appear at certain intervals just below the lair. There are eight different fruits with values of 100–5000 (higher-valued fruits appear in later levels only).

When all pills have been cleared, the game moves on to the next level. Technically the game is unlimited but a software bug in the original ROM code prevents the game going past level 255, which cannot be completed. The Twin Galaxies International Scoreboard[1] states that the highest possible score of the game (3 333 360 points by playing 255 perfect levels) was set by Billy Mitchell in 1999. A perfect score is possible as the behavior of the four ghosts is deterministic. Table I summarizes the main characteristics of the ghosts, which have variable degrees of aggressiveness: Clyde, for instance, is the least dangerous ghost that pursues *Pac-Man* only 20% of the time. Furthermore, each ghost has its primary territory where it spends most of its time. The eyes of the ghost indicate the direction they are traveling and ghosts cannot generally reverse direction unless *Pac-Man* consumes a power pill or an all-ghost-reversal event is triggered: Ghosts operate in one of the three modes (scatter, chase, or frightened) and a reversal event occurs after a number of transitions between these modes, for example, when going from scatter mode to chase mode, or vice versa. A detailed account of the game with all its wonderful intricacies may be found on Gamasutra.[2]

The speed of the ghosts is normally constant, except when travelling through the tunnel (where they slow down significantly) and except when edible (when they travel at half speed). Also, as the player is near the end of a level, ghosts may speed up (and the speed may depend on the type of ghost). The speed of *Pac-Man* is variable throughout the maze: Speed increases relatively to the ghosts in tunnels, around corners (cutting corners) and after eating a power pill, and decreases while eating pills. These variations in the relative speed of *Pac-Man* and the ghosts add significantly to the richness of game play, and can often make the difference between life and death. For example, an experienced player with the ghosts in hot pursuit may seek pill-free corridors and aim to execute multiple turns to exploit the cornering advantage.

B. *Ms. Pac-Man*

The strategy guides published for *Pac-Man* contain specific patterns that the player can exploit to maximize the score of the game. As pointed out by Mott [5], these patterns are not only important in mastering *Pac-Man*, but their mere existence is one of the game's weaknesses: "Lacking any particularly inspiring AI, *Pac-Man*'s pursuers race around the maze, following

[1] http://www.twingalaxies.com
[2] http://www.gamasutra.com/view/feature/3938/the_pacman_dossier.php?print=1

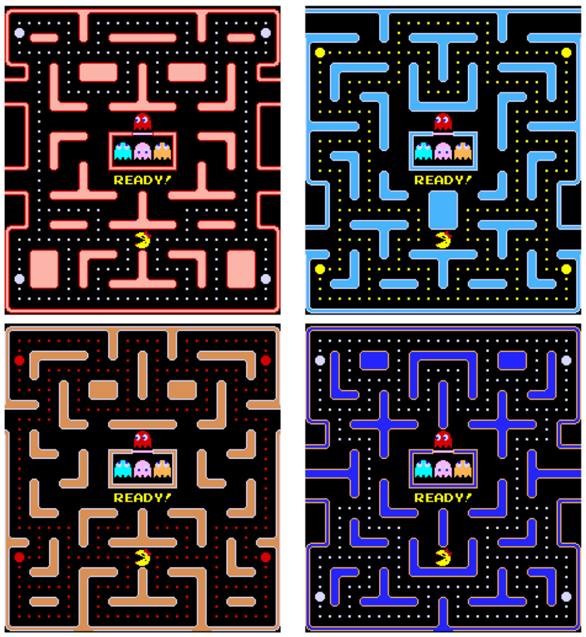

Fig. 2. Four mazes encountered during gameplay in *Ms. Pac-Man*: mazes are encountered left to right, top to bottom.

predictable paths, meaning players can effectively beat the game through memory and timing rather than inventive reactions." In other words, the determinism of the ghosts implies that there is a pure strategy that is optimal for playing the game. *Ms. Pac-Man*, released in 1982 and now sporting a female main character, changed this by introducing ghosts with nondeterministic behavior, that require gamers to improvize at times rather than follow strict and predictable patterns: "It is important to note that the monsters are much more intelligent in *Ms. Pac-Man* than in the original *Pac-Man* game. [...] This means that you cannot always predict what the monsters will do." [3, p. 4].

The overall gameplay of *Ms. Pac-Man* is very similar to that of *Pac-Man* and the objective of the game remains the same. However, apart from the new behavior of the ghosts, *Ms. Pac-Man* also introduced four new mazes, which are played in rotation. These new mazes are shown in Fig. 2. Additional changes include the following.

1) Clyde has been renamed to Sue.
2) Bonus fruits now move along the maze randomly.
3) Edible time of ghosts reduces as the game progresses but periodically increases again.

Unlike *Pac-Man* (thanks to the bug in the original ROM), *Ms. Pac-Man* never ends and new high scores are still being posted. As recently as 2006, a new high score was verified by Twin Galaxies: 921 360 points by Abdner Ashman.

C. Why These Games?

The remainder of this paper will show that a large number of research projects have made use of *Pac-Man*, probably more than any other video game. This begs the question: Why is *Pac-Man* such a useful tool in research? Undoubtedly, one of the strongest arguments for *Pac-Man* was its immense popularity upon release and the fact that the game remains popular even 30 years later: The game's characters feature on the covers of recently released books about arcade games, including *Arcade Fever* [6] and *The Art of Video Games* [7], and even feature in movies such as "Pixels" (Chris Columbus, 2015). Finally, websites such as the *World's Biggest Pac-Man* highlight how active the *Pac-Man* playing community is to this day: The website allows registered users to design mazes by hand using a graphical editor and whenever *Pac-Man* enters a tunnel, the game moves on to another maze. As of 2017, almost 250 000 mazes have been designed and 114 million games have been played using them. This popularity not only implies an immediate familiarity with this game across all age groups (making it easier for readers to relate to the research), but also validates the idea that the game is fun, entertaining and at the right level of difficulty.

Furthermore, numerous implementations of the game are available, many of which are open source. One of these that has been used in numerous academic studies is *NJam*.[3] *NJam* is a fully-featured interpretation of *Pac-Man* written in C++ (open source). The game features single and multiplayer modes, dual mode (players compete against each other to get more points), and a cooperative mode (where players attempt to finish as many levels as possible). There are three ghosts (although each type of ghost may feature in the game more than once): Shaddy, Hunter and Assassin, each of which has its own behavior. The game itself is implemented as a 2-D grid, where walls, empty spaces and the game's characters all have the same size.

Pac-Man has several additional attributes that make it interesting from an academic point of view (also see Section VII-E). In particular, the game poses a variety of challenges. The mazes may be represented using an undirected connected graph and hence one may make use of the many tools of graph theory, including path-finding. Furthermore, the game is real time, making it more challenging for computational methods to perform well (allowing the measurement of things like reaction times in human test subjects). Finally, the rules of the game are relatively simple and *Pac-Man* may be controlled using a simple four-way joystick. This makes the game suitable for studies that include primates, for instance. The game is visually appealing (important if human testing is involved) and the 2-D layout allows one to easily visualize the computational logic that drives the game's characters; this is particularly useful in higher education settings (see Section VII-E). Unlike almost all board games, the game's characters are heterogeneous and amenable to predator-prey scenarios: Techniques that work well for *Pac-Man* may not work well for the ghosts and hence the game provides a wider scope for research. Finally, the game is very expandable: It is easy to envision extensions to the game such as the design of new mazes, additional game characters or a modified set of rules.

III. Game Competitions

The University of Essex, Colchester, U.K., has been organizing game competitions centered around *Ms. Pac-Man* for

[3] http://njam.sourceforge.net/

the past ten years. The *Ms. Pac-Man* Screen-Capture Competition asks participants to develop software controllers to control the actions of *Ms. Pac-Man* for the original game using screen capture. The *Ms. Pac-Man* versus Ghosts Competition provides its own implementation of the game and competitors may develop software controllers for *Ms. Pac-Man*, or the four ghosts, that interface directly with the game engine. Based on the *Ms. Pac-Man* versus Ghosts Competition, the more recent *Ms. Pac-Man* versus Ghost Team Competition includes partial observability and allows for the ghosts to communicate with one another. Such competitions are essential to provide a common platform that academics may use to test, evaluate and compare their techniques against their peers. As Section III-D shows, prior to the competition, most studies made use of their own implementation of the game, making direct comparisons impossible. The reasons for this are manifold: Although the game has simple rules, not all the details are well documented and are sometimes difficult and time-consuming to implement. Furthermore, some studies require certain functionality that may not be available and/or difficult to implement on top of an already existing architecture. The code made available by the *Ms. Pac-Man* versus Ghosts Competition attempts to address these issues, as it provides a wide range of built-in functionality. The code is also easy to modify.

A. *Ms. Pac-Man Screen-Capture Competition*

The *Ms. Pac-Man* Screen-Capture Competition [8] ran from 2007 to 2011, with the last proper run in 2011 at the IEEE Conference on Computational Intelligence and Games (CIG).

The final of each competition event coincided with a major international conference in the field of computational intelligence. The competition makes use of the original game, either available as a Microsoft Windows application (Microsoft Revenge of Arcade version) or as a Java applet,[4] and requires screen capture to allow the algorithm to compute the next move in real time (a rudimentary screen-capture kit is provided). The screen-capture aspect is a significant component of the controller's architecture and past winners of the competition almost always dedicated significant effort to efficient feature detection that allowed the controller to make good decisions.

The winner of the competition is the controller that achieves the highest score across multiple runs: Usually ten runs are executed prior to the conference and an additional three runs are demonstrated live at the conference. An overview of past competitions is shown in Table II. It is important to bear in mind when comparing the scores that although the game is identical in all cases, the underlying hardware used to execute the controllers differs. Furthermore, the performance of a controller should not be judged without considering the screen-capture mechanisms used.

Although the competition did not run formally for CIG 2012, Foderaro *et al.* [9] did demonstrate their entry and it performed in line with the results described in their paper. Their system used an internal model of the ghost decision-making processes in order to achieve high performance, with a high score of 44 630 (improving on the CIG 2011 winning score of 36 280). A refined version was developed and described in [10], which achieved a maximum score of 65 200, with an average score of 38 172 (more details about this controller in Section IV-A). Although this provided a clear improvement compared to previous approaches, its average score still falls a long way short of the human high score, indicating that there may still be interesting research to be done on making a super-human player of *Ms. Pac-Man* in screen-capture mode.

B. *Ms. Pac-Man Versus Ghosts Competition*

The *Ms. Pac-Man* versus Ghosts Competition [11] ran for four iterations, having built on the success of the *Ms. Pac-Man* Screen-Capture Competition. It took place twice a year and results were presented at major conferences in the field. The competition differs from the screen-capture competition in two important aspects: First, competitors interface directly with the game (i.e., no screen capture) and secondly, competitors may create controllers for either (or both) *Ms. Pac-Man* and the ghosts.

The game provided to the competitions is written entirely in Java, specifically for the competition. The original software was written by Lucas [12] and was used in numerous papers. This version came with three default ghosts teams showing different behaviors (*Random*, *Legacy*, and *Pincer*). It was later extended by Samothrakis *et al.* [13] and modified further by Rohlfshagen [11] for use in the competition. The current version of the software bears little resemblance to the original code and is continually improved in response to comments by the competition's participants. Care has been taken to implement the game faithfully in most respects, but it differs from the original game in several ways (and hence is not directly comparable to the version used in the screen-capture competition). For instance, there are no bonus fruits, the speed of all characters is constant unless the ghosts are edible and the tunnels are shorter than in the original game. The single most significant difference is the ghost control algorithms, since these are now provided by the ghost-team developers rather than being an intrinsic feature of the game.

All entries submitted compete with one another in a round-robin tournament to establish the best controllers: *Ms. Pac-Man* controllers attempt to maximize the score of the game, whereas the ghosts strive to minimize the score. There are no restrictions regarding the techniques or algorithms used to create the logic for either side, but controllers have only 40 ms per game step to compute a move. Each game lasts a maximum of 16 levels and each level is limited to 3000 time steps to avoid infinite games that do not progress. Whenever the time limit of a level has been reached, the game moves on to the next level, awarding half the points associated with the remaining pills to *Ms. Pac-Man*; this is to encourage more aggressive behavior of the ghosts and avoid the ghosts spoiling a game by grouping together and circling a few remaining pills. Ghosts are not normally allowed to reverse, but there is a small chance that a random reversal event takes place that reverses all the ghosts' movements.

[4]www.webpacman.com

TABLE II
SUMMARY OF RESULTS FROM THE *Ms. Pac-Man* SCREEN-CAPTURE COMPETITION

	CEC'07	WCCI'08	CEC'09	CIG'09	CIG'10	CIG'11
Entries	5	12	5	4	8	5
Functional	3	11	4	3	7	5
Winner	Default	Fitzgerald *et al.*	Thawonmas *et al.*	Thawonmas *et al.*	Martin *et al.*	Ikehata and Ito
Highest score	3810	15 970	13 059	30 010	21 250	36 280

TABLE III
SUMMARY OF RESULTS FROM THE *Ms. Pac-Man* VERSUS GHOSTS COMPETITION

	CEC'11	CIG'11	CIG'12	WCCI'12
Competitors	13	22	44	84
Countries	9	10	13	25
Controllers	19	33	56	118
Default controllers	3	4	2	4
Pac-Man controller (only)	7	7	24	24
Ghost controller (only)	3	8	8	16
Pac-Man and ghost controller	3	7	12	37

TABLE IV
SUMMARY OF RESULTS FROM THE *Ms. Pac-Man* VERSUS GHOST TEAM COMPETITION IN CIG'16

Entry	Average score	Minimum score	Maximum score
GiangCao	6348.85	1000	15 940
Dalhousie	5878.50	1140	13 620
Starter rule	2447.75	580	6730
Random	1629.85	250	5830

TABLE V
IMPLEMENTATION OF GAMES USED IN RESEARCH

Implementation	References
Original (Screen capture)	[9], [10], [15]–[35]
Public variant	[36]–[43]
Ms. Pac-Man versus Ghosts engine	[12], [13], [20], [44]–[67]
Ms. Pac-Man versus Ghost team engine	[14]
Own implementation	[68]–[92]

A summary of results from the past four iterations of the competition is shown in Table III. It is evident that the interest in the competition has increased each time, with a significant increase in the third iteration. Furthermore, an increasing number of participants have started to develop controllers for both *Ms. Pac-Man* and the ghosts. This is an encouraging trend, as the number of studies related to controllers for the ghosts is far outnumbered by those related to *Ms. Pac-Man*.

C. Ms. Pac-Man Versus Ghost Team Competition

Ms. Pac-Man versus Ghost Team Competition [14] was run for the first time at the 2016 IEEE CIG. Despite the reuse of the *Ms. Pac-Man* versus Ghosts Competition software, the framework is very different because of the partial observability implemented by Williams [14]. Both *Ms. Pac-Man* and the ghosts can only observe a first person view up to a limited distance or a wall. A messenger system is added to the game, which allows the ghosts to send default or personalized messages to either individual ghosts or all ghosts at once.

Participants are invited to submit one controller for *Ms. Pac-Man* or four controllers for the ghosts (a controller for each ghost). At each game step, a 40-ms time budget is allocated for deciding either one move for *Ms. Pac-Man* or four moves for the ghosts (the four ghost controllers thus share the time budget in a flexible way). All entries submitted for *Ms. Pac-Man* compete with all sample agents and entries submitted for ghosts in a round-robin tournament. The final ranking in CIG'16 is shown in Table IV. While only two entries were submitted, this may be due to the more difficult nature of the challenge (dealing with partial observability and messaging), and also the fact the final version of the software toolkit was only available two months before the deadline. The competition series is ongoing at the time of writing and aims to attract more entrants as the ways for dealing with partial observability and cooperation become better understood, and the software becomes more stable.

D. Research in Computational Intelligence

Computational intelligence is unsurprisingly the most active area of research centered around *Pac-Man* and has resulted in the most publications. Prior to the competitions described above, papers were largely fragmented, with each using their own, often much simplified version of the game. Following the first *Ms. Pac-Man* Screen-Capture Competition, numerous papers emerged that made use of the original ROM via screen capture; authors often proposed their own screen-capture mechanisms to improve the reliability of the process. More recently, controllers have been suggested that interface directly with the game using the current *Ms. Pac-Man* versus Ghosts game engine (or its predecessor). This also allowed researchers to start developing controllers for the ghosts in addition to *Ms. Pac-Man*. A summary of the variants of the game used in the development of the controllers is shown in Table V.

The next two sections discuss peer-reviewed work in computational intelligence related to (Ms) *Pac-Man* with Sections IV and V covering *Pac-Man* controllers and Ghost controllers, respectively. We purposely do not mention any scores reported in these papers as we believe this to be misleading: scores often depend significantly on the screen-capture method (if the original version of the game is used), the implementation of the game and the underlying hardware used for the experiments. The competitions provide a reliable platform to compare the performance of different controllers and these results may be accessed online. Table VI gives an overview of techniques that have been used to develop controllers for *Pac-Man*, *Ms.*

TABLE VI
SUMMARY OF PEER-REVIEWED RESEARCH RELATING TO *Pac-Man*, CATEGORIZED BY DISCIPLINE

Competitions		[8], [11], [14]	3
AI/CI	Rule-based & FSMs	[9], [10], [15], [16], [18], [23], [24], [52], [65], [71], [72]	67
	Tree Search & MC	[13], [20], [25], [26], [29], [30], [49], [51], [56], [59], [61], [74]	
	Evolutionary algorithms	[45]–[48], [50], [53], [57]–[60], [63], [68], [69]	
	Neural networks	[38], [70], [75]	
	Neuro-evolutionary	[12], [28], [31]–[33], [36], [37], [43], [44], [62], [64], [67], [77]	
	RL	[19], [21], [22], [34], [35], [41], [42], [73], [78], [82], [92]	
	Other	[17], [27], [54], [79], [90], [91]	
Game psychology		[93]–[99]	7
Psychology		[81], [100], [101]	3
Robotics		[102], [103]	2
Sociology		[104], [105]	2
BCIs		[83]–[85]	3
Biology and animals		[106]	1
Education		[80], [102], [103], [107]	4
Other		[39], [40], [108], [109]	4

Right-most column indicates the number of publications in each category.

Pac-Man or the ghosts. Many studies propose hybrid techniques or use multiple techniques for different aspects of the controller, and references are listed next to the technique that best describes it. The literature review follows the same classification.

IV. *Pac-Man* CONTROLLERS

Not surprisingly, most of the work performed in computational intelligence has been devoted to finding better controllers for (*Ms*) *Pac-Man*. This includes multiple approaches, from rule-based methods to tree search or learning, as well as several nature-inspired algorithms. This section summarizes these works and categorizes them according to the nature of the controllers implemented.

A. *Rule Based*

This section describes works in *Pac-Man* controllers, which have as a main component a set of *if-then* clauses, either in a hard-coded way or in a more structured manner, like finite state machines (FSMs) or behaviour trees, including fuzzy systems. In some cases, either the parameters of these rules or the structures themselves can be evolved. In general, these controllers, albeit effective, require an important amount of domain knowledge to code the rules, transitions, conditions, and actions.

In their paper, Fitzgerald and Congdon [15] detail their controller, RAMP (a Rule-based Agent for *Ms. Pac-Man*), which won the 2008 WCCI *Ms. Pac-Man* Screen-Capture Competition. It is a rule-based approach and, like many other competitors of the screen-capture competition, the authors optimized and refined the default screen-capture mechanism provided. The maze is discretized into a grid of 8×8 pixel squares and the locations of all pills/power pills are precomputed and stored as a connected graph. The nodes of the connected graph correspond to all turning points in the original maze (i.e., junctions and corners). Additional (fake) intersections are added on either side of the power pills to allow quick reversals at these points. Decisions are made only at junctions and a rule-independent mechanism is used to reverse if a ghost is present. The decision as to which path to follow is based on higher-level conditions and actions. The set of conditions include, among many others, the number of ghosts that are "close" or "very close" to *Ms. Pac-Man*, the number of remaining power pills, and whether ghosts are edible. The conditions also distinguish between the mazes and level and the progression throughout the level to apply the most suitable set of rules. The set of actions is composed of *Graze*, *Eat*, *Evade*, and *Run* and numerous parameters are used to construct the full set of rules. Finally, conflict resolution is applied in case multiple rules apply simultaneously. The authors compare two sets of rules and also discuss the use of evolutionary computation to choose subsets of rules and to fine-tune the numerous parameters of the rule-based controller, similar to [71].

Handa and Isozaki [16] employ an evolutionary fuzzy system to play *Ms. Pac-Man* (via screen capture): A set of fuzzy rules is designed by hand and the parameters of the rules are subsequently evolved using a $(1+1)$-ES. The screen is captured and the following information is extracted: distance to all ghosts (edible and nonedible), position of the nearest pill, and distance to the nearest junction from *Ms. Pac-Man* and the ghosts. Dijkstra's algorithm is used to precompute these distances using a coarse graph of the maze. The fuzzy rules are defined as avoidance, chase, and go-through. If none of the rules are activated, the default Eat is used instead (Eat is not a fuzzy rule but simply goes for the nearest pill). The rule for avoidance, for instance, is "IF a ghost IS close THEN *Ms. Pac-Man* goes in opposite direction." Each rule is applied to all four ghosts to determine an action for *Ms. Pac-Man*. Each rule has a set of parameters (such as minimum and maximum distance to determine membership) that are evolved using a $(1+1)$-ES, using as a fitness measure the length of time *Ms. Pac-Man* survives. The authors found that it is possible to improve performance of the fuzzy set using artificial evolution, but results are very noisy as the number of games played per controller had to be kept low (ten games) due to reasons of efficiency, causing a high variance of the results.

Thompson *et al.* [72] analyze the impact of looking ahead in *Pac-Man* and compare their approach to simpler controllers based on greedy and random decisions. The game used by the authors is based on *Pac-Man* but has some important differences, most notably nondeterministic ghosts (other differences relate to the lack of bonus fruits and speed of the ghosts, among others). The controller proposed is constructed from a knowledge base and a graph model of the maze. The knowledge base is a series of rules and the overall decision-making is facilitated by the use of an FSM. The FSM has three states: *Normal*, *Ghost is Close*, and *Energized*. Three different strategies are considered: *Random*, *Greedy-Random*, and *Greedy-Lookahead*. The former two simply make a move based on the current state of the maze. The latter performs a search in certain situations and subsequently employs A* to find the paths to the targets identified during the search. Ghost avoidance is dealt with explicitly in the FSM. The experiments compared all three controllers as well as a range of human players and found that the lookahead player, although not quite as good as the best human player, significantly outperforms the other two controllers.

Thawonmas and Matsumoto [18] describe their *Ms. Pac-Man* controller, ICE Pambush 2, which won the 2009 iteration of the screen-capture competition (the authors also wrote reports on subsequent versions of the algorithm for the years 2009–2011). The authors attribute the success of their controller to two aspects: advanced image processing and an effective decision-making system (rule based). The former improves on the standard toolkit supplied with the screen-capture competition in terms of efficiency (nonmoving objects are extracted only once at the beginning) and accuracy (a novel representation of the maze is proposed). The controller's decision-making is determined by a set of seven hand-crafted rules that make use of two variants of A* to calculate distances. The rules are applied in order of decreasing priority. The two variants of A* used differ in the way they calculate the costs of the paths and several cost functions are defined to maximize the likelihood of *Ms. Pac-Man*'s survival.

Thawonmas and Ashida [24] continue their work with another entry to the *Ms. Pac-Man* Screen-Capture Competition, ICE Pambush 3, which is based on their earlier effort, ICE Pambush 2 [18]. The controller architecture appears identical to [18] but the authors compare different strategies to evolve the controller's many parameters, including parameters used for distance calculations and for calculating costs. The authors found the most efficient approach is to evolve the distance parameters first and then the cost parameter, to achieve an improvement in score by 17%. The authors also address some interesting issues, including whether the same set of parameters is to be used for all mazes/levels (speed of ghosts changes as the game progresses) and whether all parameters should be evolved simultaneously. The EA used is a simple evolutionary strategy with mutation only. The authors found that not only did the optimized parameters improve playing strength but they also had an impact on the style of play.

The work by Bell *et al.* [23] describes their entry to the 2010 CIG *Ms. Pac-Man* Screen-Capture Competition. The controller is a rule-based system using Dijkstra's algorithm for shortest path calculations, a benefit-influenced tree search to find safe paths and a novel ghost direction detection mechanism. The algorithm first records a screenshot of the game and updates its internal model of the game state. The maze is represented as a graph with nodes at junctions and corners of the original maze. Once the internal state is updated, the controller determines the best rule to apply and then determines the best path (and hence direction) to choose. The authors employ "ghost iris detection" to determine the direction of each ghost. This approach is found to be more efficient than the more intuitive approach of comparing successive frames, and may provide slightly more up-to-date information when a ghost is turning at a junction (the idea being that the eyes point in the new direction before any movement has occurred). Furthermore, the authors address the issue of when a ghost enters a tunnel as it is momentarily not visible on the screen. This can pose problems for screen-capture kits. The authors overcome this by memorizing the positions of all ghosts for short periods of time. The core of the controller itself is based on six hand-crafted rules. One of the rules makes use of the benefit-influenced tree search algorithm that computes a safe path for *Ms. Pac-Man* to take: Starting from the current position of *Ms. Pac-Man*, the tree searches all possible paths, assigning a cost to each path depending on the distance and direction of the ghosts.

Foderaro *et al.* [9] propose a decomposition of the navigable parts of the level in cells, to be then represented in a tree based on the cells' connectivities (and referred to by the authors as a connectivity tree). Additionally, the screen capture is analyzed by extracting the colors of each pixel, in order to identify certain elements of the game (fruits, pills, ghosts, etc.). As with Bell *et al.* [23], the eyes of the ghosts are analyzed to determine their direction of travel. Once all these features have been assigned to nodes in the tree, a function determines the tradeoff between the benefit predicted and the risk of being captured, and the tree is searched to decide the next action to play. For specific scenarios where the tracking of the ghosts' eyes is not reliable (i.e., when a power pill has just been eaten), a probability model is built in order to determine the most probable next location of the ghost. The authors continue their work [10] and construct a more complete mathematical model to better predict future game states and the ghosts' decisions. This also takes into account their different personalities. For instance, while Inky, Pinky, and Sue have the same speed, Blinky's speed depends on the number of pills in the maze. The authors show that their predictive model obtains an accuracy of 94.6% when predicting future ghost paths. We note that the tree search aspect of this paper shares some similarity with the prior tree search approach of Robles and Lucas (see Section IV-B), but the Foderaro approach used a better screen-capture module and accurate models of the ghosts' decision-making behavior.

B. Tree Search and Monte Carlo (MC)

This section summarizes the works performed on *Pac-Man* based on tree search techniques [from one-step look-ahead to Monte Carlo tree search (MCTS)] and/or have MC simulations used in the controller. These methods, especially MCTS, have shown exceptional performance, although they require the simulator to have a forward model to be applicable. Additionally, when MCTS is applied to *Pac-Man*, a limit needs to be applied to the length of rollout (unlike games such as Go where the rollouts proceed to the end of the game, at which point the true value is known). Since the rollouts usually stop before the end of the game, a heuristic is needed to score the rollout, and also to scale it into a range compatible with other search parameters, such as the exploration factor used in MCTS. The default heuristic is to use the current game score, but there are ways to improve on this.

Robles and Lucas [20] employ traditional tree search, limited in depth to 40 moves, with hand-coded heuristics, to develop a controller for *Ms. Pac-Man*. The controller is evaluated both on the original game, via screen capture, and the authors' own implementation of the game (the game implemented is based on [12] but has been extended significantly along the lines of [13]). The authors find their controller to perform approximately three times better on their own implementation of the game (where the controller can interface directly with the game but the game also differs in other aspects) than the original game played via screen capture. The controller creates a new tree of all possible paths at every time step of the game. The depth of the tree is limited to 40 moves and ignores both the direction and the state (i.e., edible or inedible) of the ghosts. Paths are subsequently evaluated by their utility: A safe path contains no ghosts, whereas unsafe paths contain a ghost (direction of a ghost is ignored). The authors consider two situations: the existence of multiple safe paths and the lack of safe paths. Safe paths are selected according to hand-coded rules that take into account the number of pills and power pills on the path. If no safe path exists, the controller chooses the path where the ghost is furthest from *Ms. Pac-Man*. Robles and Lucas found that the most elaborate path selection worked best, taking into account pills, power pills, and position within the maze at the end of the path.

Samothrakis *et al.* [13] were among the first to develop an MCTS controller for *Ms. Pac-Man*; they used a Max-n approach to model the game tree. The authors make use of their own implementation of the game, extended from [12]. The authors discuss several issues that become apparent when applying tree search to *Ms. Pac-Man*. They treat the game as turn taking and depth limit the tree. Furthermore, the movement of *Ms. Pac-Man* is restricted (no reversals) to reduce the size of the tree and to explore the search space more efficiently (i.e., the movement of *Ms. Pac-Man* is similar to the ghosts; it is still possible for *Ms. Pac-Man* to reverse as all directions are available from the root, just not the remainder of the tree). Leaf nodes in the tree may either correspond to cases where *Ms. Pac-Man* lost a life or those where the depth limit had been reached. To identify favorable leaf nodes (from the perspective of *Ms. Pac-Man*), a binary predicate is used to label the *game preferred node* (the value of one node is set to 1, all others set to 0). The target node is identified by the distance of *Ms. Pac-Man* to the nearest edible ghost, pill, or power pill. *Ms. Pac-Man* subsequently receives a reward depending on the outcome (e.g., completing the level or dying). The ghosts receive a reward inversely proportional to their distance to *Ms. Pac-Man* or 0 if *Ms. Pac-Man* clears the maze. The authors test the performance of their algorithm in several ways, comparing tree-variants *UCB1* and *UCB-tuned* [110], different tree depths as well as different time limits. Finally, the authors also test their controller assuming the opponent model is known, which subsequently led to the highest overall scores.

Around the same time as Samothrakis *et al.* [13], Ikehata and Ito [25], [29] also used an MCTS approach for their *Ms. Pac-Man* controller, with heuristics added to detect and avoid pincer moves. Pincer moves are moves that trap *Ms. Pac-Man* by approaching her from multiple junctions. To do this, the authors define the concept of a C-path, which is a path between two junctions (i.e., a path without any branching points). Similar to other studies, Ikehata and Ito model the maze as a graph with nodes at points that require a change in direction. *Ms. Pac-Man* subsequently moves from node to node, whereas the movement of the ghosts is chosen in a nondeterministic manner. The depth of the tree is limited given the real-time nature of the game. A set of rules is used to define a simplified model of the game used for the MC simulations. The rules are chosen to approximate the real game dynamics in a computationally efficient manner while preserving the essence of the game. *Ms. Pac-Man* and the ghosts move in a nondeterministic fashion according to a set of rules (behavioral policies). Simulations are terminated when the level is cleared and *Ms. Pac-Man* dies when a certain depth has been reached. Each nonterminal state is assessed by a reward function that takes the score of the game, as well as the loss of life, into account; the back propagation of values is determined by the survival probability of the child node only. The authors compare their approach against the then state of the art, ICE-Pambush 3 [24]. The proposed system outperforms ICE-Pambush 3 (both in terms of high score and maximum number of levels), but the authors highlight that their controller is somewhat less reliable. One shortcoming is the controller's inability to clear mazes, focusing instead solely on survival. The authors subsequently improve their controller to almost double the high score achieved: pills in dangerous places, as identified by a danger level map (similar to influence maps [17]), are consumed first, leaving pills in less dangerous areas for the end.

In a similar fashion to [29], Tong and Sung [26] propose a ghost avoidance module based on MC simulations (but not MCTS) to allow their controller to evade ghosts more efficiently. The authors' algorithm is based on [16]. The internal game state is represented as a graph, with nodes at junctions. Arbitrary points in the maze are mapped onto their nearest corresponding junction node, from which a path may be obtained, using the predefined distances. The screen capture is mapped onto a 28×30 square grid (each cell being 8×8 pixels); cells can be passable or impassable. A bit-board representation is used for reasons of efficiency. Movable objects are found using a pixel color counting method (based on [18]). The controller's logic itself is based on a multimodular framework where each module corresponds to a behavior. These behaviors include capture mode (chasing edible ghosts), ambush mode (wait near power pill, then eat pill, and enter capture mode), and the default behavior pill mode (tries to clear a level as quickly as possible). These behaviors are executed if *Ms. Pac-Man* is in a safe situation (this is determined using the precomputed distances). If the situation is dangerous, MC simulations are used. The look-ahead is used to estimate the survival probabilities of *Ms. Pac-Man*. To make the MC simulations more efficient, the authors impose several simplifications and restrictions on the gameplay that are possible. Most notably, the ghosts do not perform randomly but a basic probabilistic model is used to approximate the ghosts' real behavior. Given the real-time element of the game, the authors do not simulate until the end of the game (or level) but only until *Ms. Pac-Man* dies or has survived after visiting a predefined number of vertices. The state of *Ms. Pac-Man* (dead or alive) is subsequently back propagated. The authors found that the controller with the MC module almost doubled the score of the controller with a ghost avoidance module that is entirely greedy.

This concept is extended to employ MC for an evaluation of the entire endgame: Tong *et al.* [30] consider a hybrid approach where an existing controller is enhanced with an endgame module, based on MC simulations; this paper extends the authors' previous efforts (see [26]). The authors specifically consider the scenario of an endgame that entails eating as many of the remaining pills as possible (and thus ignoring many of the other aspects of the game such as eating ghosts). An endgame state has been reached when the number of remaining pills falls below a predefined threshold. The endgame module consists of two major components: path generation and path testing. The first module finds all the paths that connect two locations in the maze (typically the location of *Ms. Pac-Man* and the location of a remaining pill), whereas the second module evaluates the safety of each path. The latter is done using MC simulations. Using a similar approach to [26], a path is deemed safe if *Ms. Pac-Man* reaches the destination without being eaten. These data are accumulated into a survival rate for each path and if none of the paths is deemed safe, *Ms. Pac-Man* attempts to maximize the distance to the nearest ghost.

Pepels *et al.* [51], [61] present another MCTS controller, which managed second place at the 2012 IEEE World Congress on Computational Intelligence (WCCI) *Pac-Man* versus Ghost Team Competition, and won the 2012 IEEE CIG edition. The controller uses variable depth in the search tree: The controller builds a tree search from the current location where nodes correspond to junctions where *Ms. Pac-Man* is able to change direction. As these paths are of different lengths, not all leaves of the tree will have the same depth. In particular, the leaves will only be expanded when the total distance from the initial position is less than a certain amount. Another attribute of the controller is the use of three different strategies for the MC simulations: *pills*, *ghosts*, and *survival*. Switching from one strategy to the other is determined by certain thresholds that are checked during the MCTS simulations. The authors also implemented a tree reuse technique with value decay: The search tree is kept from one step to the next, but the values in the nodes are multiplied by a decay factor to prevent them from becoming outdated. Additionally, if the game state changes too much (*Pac-Man* dies, a new maze starts, a power pill is eaten, or a global reverse event occurs), the tree is discarded entirely. Finally, the authors also include long-term rewards in the score function, providing a rapid increase in the reward when eating entire blocks of pills or an edible ghost is possible.

Finally, Silver [74] uses a partially observable version of *Pac-Man*, called Poc-Man, to evaluate MC planning in large, partially observable Markov decision processes (POMDPs). In particular, Silver extends MCTS to POMDPs to yield partially observable Monte Carlo planning (POMCP) and the game itself functions as a testbed. The game was developed specifically for this purpose, and features pills, power pills, and four ghosts as usual (albeit with different layouts). Poc-Man can, however, only observe parts of the maze at any moment in time, depending on its senses of sight, hearing, touch, and smell (10 b of information): 4 b are provided for whether a particular ghost is visible or not, 1 b for whether a ghost can be heard, 1 b for whether food can be smelled, and 4 b for feeling walls in any of the four possible directions. Silver uses this game, in addition to some others, to test the applicability of MC simulations for online planning in the form of a new algorithm: MC simulations are used to break the curse of dimensionality (as experienced with approaches such as value iteration) and only a black-box simulator is required to obtain information about the states. Silver compares his POMCP algorithms on Poc-Man, both with and without preferred actions. Preferred actions are those better suited to the current situation, as determined by domain-specific knowledge. The algorithm performs very well for this domain given the severe limits on what the agent can observe, achieving good scores after only a few seconds of online computation. We will return to the concept of partially observable *Pac-Man* in Section III.

C. Evolutionary Algorithms

This section explores the work on evolutionary algorithms for creating *Pac-Man* agents. This includes mainly genetic programming (GP) (which evolves tree structures), grammatical evolution (GE) (where individuals in the population are encoded as an array of integers and interpreted as production rules in a context-free generative grammar), and also some hybridizations of these (note that neuro-evolution (NE) approaches are described in Section IV-E). The results of these methods are promising. Although to a lesser extent than rule-based systems, they still require an important amount of domain knowledge, but without the dependence of a forward model. This is an important factor that differentiates them from tree-based search methods: learning is typically done offline, by repetitions, with a more limited online learning while in play.

The work by Koza [68] is the earliest research on applying evolutionary methods to *Pac-Man*: Koza uses his own implementation of the game, which replicates the first level (maze) of the original game but uses different scores for the game objects and all ghosts act the same (strictly pursuing *Pac-Man* 80% of the time, otherwise behaving randomly). The author uses a set of predefined rules and models task prioritization using GP. In particular, Koza makes use of 15 functions, including 13 primitives and two conditionals. This leads to outputs such as "move toward nearest pill." This work was later extended by Rosca [69] to evaluate further artifacts of GP.

Alhejali and Lucas [47], using the simulator from [12] with all four mazes, evolve a variety of reactive *Ms. Pac-Man* controllers using GP. In their study, the authors consider three versions of the game: single level, four levels, and unlimited levels. In all cases, *Ms. Pac-Man* has a single life only. The function set used is based on [68] but is significantly modified and extended. It was divided into three groups: functions, data terminals, and action terminals. Functions include *IsEdible*, *IsInDanger*, and

IsToEnegerizerSafe. Most of the data terminals return the current distance of an artifact from *Ms. Pac-Man* (using shortest path). Action terminals, on the other hand, choose a move that will place *Ms. Pac-Man* closer to the desired target. Examples include *ToEnergizer*, *ToEdibleGhost*, and *ToSafety*, a hand-coded terminal to allow *Ms. Pac-Man* to escape dangerous situations. The controllers were evolved using a Java framework for GP called *Simple GP* (written by Lucas) and were compared against a hand-coded controller. The fitness was the average score over five games. Individuals in the final population are evaluated over 100 games to ensure the best controller is chosen. The authors found that it is possible to evolve a robust controller if the levels are unlimited; in the other cases, GP failed to find controllers that were able to clear the mazes.

In contrast to these GP methods that use high-level action terminals, Brandstetter and Ahmadi [50] use GP to assign a utility to each legal action from a given state. They use 13 information retrieval terminals, including the distance to the next pill, the amount of nonedible ghosts, etc. Then, the best-rated action is performed. After varying both the population size and the number of generations, the authors find that a moderate number of individuals is enough to lead to the convergence to a fairly good fitness value, which is the average score over the ten game tournaments during the selection.

Alhejali and Lucas later extend their work in [48] where they analyze the impact of a method called *Training Camps* to improve the performance of their evolved GP controllers. The authors weigh the change in performance against the additional computational cost of using Training Camps and conclude that they improve the controller's performance, albeit at the cost of manually creating these camps in the first place. A Training Camp is essentially a (hand-crafted or automatically generated) scenario that corresponds to a specific situation in the game (i.e., the game is divided into several subtasks). Using these to evaluate a controller's fitness addresses the issue of having to average over multiple complete games to account for the stochasticity of the game itself. The function set used is a revised set used by the same authors in a previous study [47]. The Training Camps are designed based on the observation that an effective *Ms. Pac-Man* controller needs to be able to clear pills effectively, evade nonedible ghosts, and eat edible ghosts. Numerous Training Camps were subsequently designed for each of these scenarios. Agents were trained on the scenarios and upon achieving a satisfactory performance, they were used as action terminals to create an overall agent for the game. The authors found that the new approach produced higher scores on average and also higher maximal scores.

More recently, Alhejali and Lucas [59] enhance an MCTS driven controller by replacing the random rollout policy used in the simulations by the policy evolved by GP. The evolved GP policy controls the rollouts by taking in to account the current game state and achieves an 18% improvement on the average score over 100 games simulated using the *Ms. Pac-Man* Screen-Capture Competition engine.

Galván-López *et al.* [45] use GE to evolve a controller for *Ms. Pac-Man* (using the implementation by Lucas [12]) that consists of multiple rules in the form "if *Condition* then perform *Action*"; conditions, variables, and actions are defined *a priori*. The authors compared the evolved controller to three other controllers against a total of four ghost teams. In GE, each genome is an integer array that is mapped onto a phenotype via a user-defined grammar in Backus–Naur form. This allows the authors to hand-code domain-specific high-level functions and combine them in arbitrary ways using GE. The actions considered include *NearestPill()* and *AvoidNearestGhost()*. These actions also make use of numerous variables that are evolved. The authors found that the evolved controller differed significantly from the hand-coded one. Also, the controller performed better than the other three controllers considered that came with the software kit (*random*, *random nonreverse*, and *simple pill eater*). The ghosts used for the experiment are those distributed with the code. The authors conclude that the improved performance of the evolved controller over the hand-coded one (although differences are small) is because the evolved controller takes more risks by heading for the power pill and subsequently eating the ghosts. This work was later extended by Galván-López *et al.* [46] to use position-independent grammar mapping, which was demonstrated to produce a higher proportion of valid individuals than the standard (previous) method.

Inspired by natural processes, Cardona *et al.* [57] use competitive coevolution to coevolve *Pac-Man* and Ghosts controllers. The coevolved *Pac-Man* controller implements a *MiniMax* policy based on a weighted sum of distance or game values as its utility function when the ghost is not edible and nearby, otherwise, the movement of the ghost is assumed to be constant (toward or away from the *Pac-Man*).

The design of the ghost team controller is similar. The weights in the utility function were evolved using an evolutionary strategy. A set of static controllers is used for *Pac-Man* and the ghost teams both in single-evolved and coevolved controllers during testing and validation. Four different coevolution variants are compared to the single-evolved controllers, distinguished by the evaluation and selection at each generation, and fitness function. The best performance was obtained when evaluating against the top 3 controllers from each competing population rather than just the single best.

D. Artificial Neural Networks (ANNs)

Several *Pac-Man* controllers have been implemented with ANNs. ANNs are computational abstractions of brains and work as universal function approximators. These can be configured as policy networks that choose actions directly given the current game state, or as value networks that rate the value of each state after taking each possible action. Most of this paper has used hand-coded features, but the ground-breaking work of Mnih *et al.* [34] showed it was possible to learn directly from the screen capture (pixel map), and recently this approach has also been applied to *Pac-Man*,[5] more of which is discussed later.

Bonet and Stauffer [70] use player-centered perceptrons to learn optimal playing strategies for both *Pac-Man* and the

[5]*Ms. Pac-Man* was included by Mnih *et al.* in the set of Atari 2600 games, but their method did not perform well compared with the best methods covered in this survey.

ghosts. The perceptrons are trained over time using reinforcements based on the outcomes of the games played. To simplify the task of learning, the authors use their own (flexible) implementation of the game and consider mazes of variable difficulty and complexity, starting with very simple cases, increasing the complexity once basic behaviors have been learned. The world is represented by a 2-D array of features. To enable more efficient learning, each type of feature has its own matrix, leading to a 3-D representation where each layer corresponds to the presence or absence of a particular feature. The characters have access to a 10×10 area that is centered on them to detect the features of the game. A separate perceptron is used for each of the four possible directions to determine the move to make. Each perceptron is updated based on a reinforcement learning (RL) scheme: the reward associated with each move is the immediate score of the game that results from the move (plus a percentage of the value of the previous move). The authors consider ten mazes of increasing complexity to learn essential behaviors first, followed by more sophisticated ones.

Yuan *et al.* [75] are interested in recreating the ability of novice human players to quickly learn basic skills in playing *Pac-Man*. They analyze the performance of ten human test-subjects unfamiliar with the game over 20 games, showing how performance increases monotonically. Computational techniques on the other hand usually require many more games before performance improves, making it a time laborious process. The authors argue that one of the reasons for this is that usually (at that time, when applying neural networks to *Pac-Man*) only the score at the end of the game is used as feedback, which provides little guidance to the in-game decision-making process.

In order to counter this, the authors show how the concept of "rationality" can be used to boost learning speed, and they show how the neural networks can learn over a small number of games when in-game training signals are provided based on eating fruit and not attempting to move into walls. However, their version of the game was simplified to the extent of making it nonchallenging, since it involved only one ghost and no power pills.

E. Neuro-Evolutionary Approaches

This particular approach, which hybridizes evolutionary algorithms and neural networks, has become a popular way to develop controllers for game agents. Evolution can be used to evolve the weights, topologies, and reward functions of the ANNs, and the literature shows that it can be further hybridized with other techniques.

Lucas [12] was the first to consider evolving ANNs to play *Ms. Pac-Man*. Lucas used a multilayer perceptron (MLP) with one hidden layer applied to his own implementation of the game. This implementation underwent further development to form the basis of many future experiments and competitions such as [11], [20], and [48]. Internally, the game is represented as a graph and the shortest path distances are precomputed. The ghosts were modeled to be aggressive and were not allowed to reverse apart from global reversal events that would trigger with small probability (and were beyond the influence of either controller). Several features were identified and used as input to the MLP. These include, among others, distance to edible/nonedible ghosts, current position, and nearest pill/power pill. The evolutionary algorithm used is an $(N + N)$ evolutionary strategy where $N \in \{1, 10\}$; this was used to evolve the weights of the MLP. The initial weights were drawn from a Gaussian distribution and different approaches to mutating these weights were considered. The move to be played was the one with the corresponding maximum value. The experiments compared the evolved MLP with a single-layer perceptron and a hand-crafted controller, showing best results with the evolved MLP.

The effect of noise was also studied, and it was shown that making the game deterministic by fixing the random seed (i.e., like the original *Pac-Man* as opposed to the original *Ms. Pac-Man*) made it much easier to evolve high performance controllers, though they would only perform well on the deterministic game. This also suggests there may be interesting research in testing the robustness of approaches such as deep Q networks that have shown such impressive results on learning to play Atari 2600 games [34], since these games are largely deterministic.

In [44], Burrow and Lucas compare the differences in performance between temporal difference learning (TDL) and evolved neural networks to play *Ms. Pac-Man*, using the same simulator as Lucas [12]. The authors find that TDL works best with a tabular function approximator and that evolved MLPs significantly outperform the TDL. The authors' aim is not to create a competitive controller but to analyze different learning techniques. Using their own implementation of the game, the controller makes use of a state value function (by looking ahead) to choose a move (look-aheads are simplified as the positions of the ghosts and the states of the pills are not updated). Two features are extracted from the game state: the relative distance to the nearest "escape node" (a node where ghosts are not approaching from all possible directions) and the distance to the nearest pill. A function approximator is used to represent the state value function: a multilayer interpolated table [111] and an MLP are considered. The MLP has $(2, 6, 1)$ nodes. The learning is done via one of two methods: TD(0) (one of the simplest forms of TDL) and evolution strategies. For the latter, a simple mutation-only $(15 + 15)$-ES is used.

Gallagher and Ledwich [36] make use of NE to determine whether a controller may be found that can play *Pac-Man* to a reasonable standard using as input the "raw" data of the screen. In other words, the input to the neural network is not a set of (hand crafted) features but instead the screen shot of the game itself. This work may be seen as a forerunner of this paper on using deep convolutional neural networks [34] to play directly from screen input. The neural network used is a MLP and it is evolved in response to the performance achieved in the game. The size of the hidden layer was determined *a priori* and a $(\mu + \lambda)$-EA was used to evolve the weights. To simplify the game, the authors often only considered a single ghost, also removing power pills and bonus fruits. The neural network has four outputs (one for each direction) with a (logistic) sigmoidal activation function. The game is divided into a grid of 31×28 squares. The network takes as input a window centered on the current position of *Pac-Man* with sizes considered 5×5, 7×7, and 9×9. The information (i.e., presence of walls, pills, and ghosts) is translated into a numerical representation. To overcome the limitations of the small input considered, global information pertaining to the number of remaining pills is also considered, leading to a total of $3w^2 + 4$ inputs, where w is the window width/height. The experiments addressed different ghost behaviors as well as different topologies, and the authors found that it is possible to use raw data as input, although none of the controllers managed to clear the maze.

Oh and Cho [28] propose a hybrid controller that combines hand-crafted rules based on Dijkstra's algorithm with evolved ANNs (based on NEAT; see [112]): The agent attempts to follow the rules if possible and if a safe route cannot be found, an evolved neural network is used instead. The authors use two sets of rules. First, if a ghost is near, *Ms. Pac-Man* immediately tries to move in the opposite direction. The second set of rules is based on paths obtained using Dijkstra's algorithm which, in turn, makes use of an abstract state representation: the maze is modeled as a grid of 28×31 nodes. The cost of each path equals its perceived danger, as measured by the proximity of nonedible ghosts as well as the presence of items such as pills and edible ghosts. The neural network is based on the NEAT method and has 20 input and four output nodes. The inputs are based on the locations of *Ms. Pac-Man*, the ghosts, the pills, and the power pills, and also the direction of *Ms. Pac-Man*. The fitness of an evolved network is equivalent to the game's score. The authors compare all components in isolation (rules only, neural network only, and hybrid) and conclude that the hybrid approach offers the best performance as it combines expert domain knowledge with a more costly yet also more flexible autonomous technique. An evolved neural network by itself is considered insufficient to cover the wide variety of scenarios *Ms. Pac-Man* may encounter throughout the game.

Tan *et al.* [31], [32] also investigate the hybridization of nature-inspired computational techniques by combining evolutionary algorithms with ANNs: The authors make use of evolution strategies to evolve the weights and biases of a feedforward neural network to automatically generate controllers to play *Ms. Pac-Man*. The authors evolve a two-layer feedforward neural network using a $1 + 1$-EA and compare its performance against a neural network found by random search as well as a random controller (in [32] only). The fitness of each ANN is determined by the average score over a number of games. The authors find that the former outperforms the latter.

More recently, Schrum and Miikkulainen [62] approach the problem from a multimodal perspective: the ghosts can be edible, a threat, or a mix in-between (when some are edible but others have been respawned already). In their technique, modular multiobjective NEAT, the authors use nondominated sorting genetic algorithm II (NSGA-II) as a multiobjective optimization technique to evolve artificial modular neural networks. They also employ operators that could add new modules. Two objectives are taken into account for NSGA-II: a *pill score* (number of pills eaten) and a *ghost score* (ghosts eaten). Experiments are run with one, two, and three modules, and results outperform previous work with nature-inspired algorithms in this domain.

In follow up work [64], the authors deepen their analysis by employing a variable number of modules and comparing different mutation operators. Additionally, the authors distinguish between split sensors (those that assess both the distance and the status–edible or threat–of the ghosts) and conflicting sensors (where the edible versus threat status is ignored) in order to get a more general and evolvable approach. Results show that networks with conflicting sensors are actually able to perform well in both situations, and evolution was able to find luring behaviors to maximize score when capturing ghosts after eating a power pill. Their work suggests that modular approaches are able to achieve a high performance of play in this game.

Inspired by [113], Miranda *et al.* [43] imitate a humanlike *Pac-Man* agent using NE. To reduce the search space, the game state is shrunk by considering only a window of 7×7 tiles. The authors focus mostly on the design of the fitness function. They compare an ANN (using back propagation) trained on a particular human player's movements to NE with a fitness based on the same human player's movements, and a similar system augmented with high-level game features such as the final score. The two NE approaches use a genetic algorithm (GA) to evolve the ANNs. The authors suggest that there is still room for improvement, for instance considering more information from the game by using a larger window, and emphasize the priority of improving the ANN.

F. Reinforement Learning

This section describes another set of offline learning methods that require episodic learning. In fact, the stochastic nature and unpredictable behaviors found in *Ms. Pac-Man* made this game a popular testbed for RL [41], [42], [78]. The approaches described here mainly used a form of Q-learning or TDL. Most of the work performed with RL methods have used different frameworks to the competitions, making general comparisons with other methods less clear.

One of the main challenges of these methods is the large state space and number of features. Bom *et al.* [78] extract the seven most important features of a game state and train their *Ms. Pac-Man* agent using single action single hidden layer neural networks with Q-learning; thus only seven input neurons (inputs) are required. At each time step, the action is chosen as the argmax of the four outputs of the neural networks. Using few high-order inputs leads to more effective and faster training. In addition, the good performance of neural networks trained with Q-learning is verified by transferring, i.e., testing the learned policy on a maze that has not been trained on. Tziortziotis *et al.* [41], [42] describe a game state using ten abstract features for speeding up the learning. Their *Ms. Pac-Man* agent is initialized by a policy trained without the presence of ghosts, then trained with the ghosts. The agent trained by two-stage learning is able to complete 40% of the games, simulated using the multiagent MASON toolkit [114].

Szita and Lorincz [73] use *Ms. Pac-Man* as an example in their demonstration of how one may deal with a range of combinatorial RL tasks. The authors make use of their own implementation of *Ms. Pac-Man* where ghosts chase *Ms. Pac-Man*

80% of the time and take random deceptions 20% of the time; ghosts may never reverse (as with [68]). The authors develop a set of high-level actions and observations and use RL to combine these into a suitable policy. The RL algorithm used is the cross-entropy method (CEM) that shares much in common with evolutionary approaches and performs policy optimization. The authors find that this hybrid approach performs better than either learning (from scratch) or a rule-based system in isolation. Szita and Lorincz use domain knowledge to preprocess the state information and to define action modules, and RL will subsequently combine these into a rule-based policy that may be used to play the game. The action modules are designed to be basic to minimize the amount of human knowledge required. Actions are temporally extended and may be executed in parallel. Conflict resolution between rules is achieved using priorities (which are also learned) and the agent can switch ON and OFF certain action modules so that the action to take is obtained only from modules active at the time. The learning takes place as follows: A generation of random policies is drawn according to the current parameter set. Policies are evaluated by playing the game. The parameter set is then updated using the CEM. The authors compared two approaches: one with randomly generated rules and one with hand-crafted rules, and CEM was compared to a simple stochastic hill climber. The best result was obtained using CEM and hand-coded rules.

Handa [21], [22] extends his previous work ([16]) to improve his controller by learning to identify critical situations in the game that allows the controller to survive considerably longer. This is done by adding a critical situation learning module where the learning is achieved by means of Q-learning using a cerebellar model articulation controller (CMAC) function approximator. Note that CMAC provides a crude form of interpolation when performing table lookup—A direct alternative with more precise retrieval can be found in Lucas [111] and also Abdullahi and Lucas [115]. The need for this module arises as the evolved fuzzy set that sits at the heart of the controller makes potentially poor decisions if rules are tied. The critical situation module thus acts as a tie breaker to ensure the right move is chosen. The input to CMAC is prepared for each intersection in the maze in the form of overpaying tilings. The controller thus only makes use of the critical situation module at junctions. The tiles are of size 6×6 and the tile set of size 5×5 or 6×6. The initial Q values are 0 and only a negative reward (-10) is given in the case of death. The controller used in the experiments is identical to [16] with only the addition of the new module. The author finds that the module can capture critical situations well but the method is too costly to be used in real time.

DeLooze and Viner [19] make use of fuzzy Q-learning to develop a controller for the screen-capture version of *Ms. Pac-Man*. Fuzzy Q-learning is a technique that combines fuzzy state aggregation with Q-learning, which may be applied naturally to the state aggregation obtained by the fuzzy sets: Fuzzy state aggregation builds states given multiple fuzzy sets, reducing the number of total states that need to be considered and making Q-learning an applicable technique. The authors consider the action of going to the nearest pill, the nearest power pill, or running away from the closest ghost. The controller decides on an action based on the current state of the game and on what has been learned about this situation in the past. The controller was trained by playing many games, taking one of the three actions available at random. Following a death, the actions that contributed to the death had their coefficients decremented (negative reward; using a window of 15 state-action pairs). When playing, the controller chooses the action corresponding to the highest coefficient. The authors tested numerous different setups in their experiments (e.g., the size of the fuzzy sets) and found that learning was ineffective given a lack of persistency (a new action would be chosen for each new screen capture) and hence the authors forced the controller to stick with the chosen action for a limited amount of time.

Recently, Vezhnevets *et al.* [82] proposed a model named STRategic Attentive Writer (STRAW) for learning macroactions and its variant STRAW-exploiter (STRAWe). Periodically, STRAW takes the high-level features extracted by a feature extractor from a game state (represented by a frame) and outputs a stochastic action plan during a certain horizon. This is referred to as an action-plan module. A module called commitment plan is used to determine at each time step whether the action plan needs to be replanned or not. Additionally, STRAWe contains a noisy communication channel between the feature extractor and the two modules of STRAW. STRAWe using a convolutional network as feature extractor and asynchronous advantage actor-critic as policy optimizer achieves significant improvement in scores on some Atari games, including *Ms. Pac-Man*. The score achieved by the STRAWe on *Ms. Pac-Man* is more than 50% higher than the one obtained by a recurrent long short term memory (LSTM) network. As we mention in the conclusions, it would be very interesting to see how well convolutional neural networks can learn to play the original *Ms. Pac-Man* game.

In a different work, Subramanian *et al.* [92] compare the way humans extract *options* (i.e., abstractions of the action space, as defined in the RL literature [116]) with that of automated methods, by collecting data from humans playing the Taxi and *Pac-Man* domains. The authors showed that human-created options provide a better performance for a Q-learning agent than the ones from automatic methods, showing (particularly in the *Pac-Man* domain) that human-crafted options bring not only a faster computation time, but also a higher average reward. This work also highlights that, however, optimal performance was not achievable with human options in *Pac-Man*, because of the way humans evaluated *Pac-Man* states (more interested on higher scores than survival time) and the fact that most human options were never executed until termination, which poses a problem due to the lack of an optimal method for option interruption.

Van Seijen *et al.* [35] proposed a new deep learning method, called hybrid reward architecture, to build an agent to play PacMan in a screen-capture setting, using the version of the game in the Atari 2600 collection. The authors decompose the reward function of the environment into *n* different (weighted) reward functions related to key aspects of the game, each one tackled by a different subagent. The authors show that the trained controller is able to achieve a performance above that of human players and other state-of-the-art RL approaches. The task decomposition approach is similar to the Training Camp method used by Alhejali and Lucas [48].

Finally, the work by Burrow and Lucas [44] fits within the category of Learning as it looks at the differences in performance between TDL and evolved neural networks to play *Ms. Pac-Man*. This work was reviewed in Section IV-E.

G. Other Approaches

This section describes controllers implemented using other techniques that do not fall under the main categories described in this section. Examples are ant colony optimization (ACO), influence maps, graphs, and constraint satisfaction problems.

Martin *et al.* [27] propose a controller based on ACO to play *Ms. Pac-Man*. The authors identify some of the game's objectives and specify different types of ants that correspond to these objectives: collector ants and explorer ants. The former tries to maximize the score of the game by collecting pills, power pills, and eating ghosts, whereas the latter attempts to find safe paths that evade the ghosts. The distance an ant may explore is limited due to the real-time nature of the game. Ants are launched from positions adjacent to *Ms. Pac-Man*'s current position and the controller chooses the path found by the best explorer ant if a ghost is near, and the path found by the best collector ant otherwise. The nodes visited by the ant are chosen according to the pheromone distribution across the graph (proportional rule). The concept of a dead ant is used to either label a collector ant that has reached the maximum distance without scoring any points, or an explorer ant has been eaten by a ghost. The parameters of the ACO, including the number of ants, distances travelled, and learning rates, are fine-tuned using a GA.

Wirth and Gallagher [17] propose a simple controller for *Ms. Pac-Man* based on influence maps: An influence indicates the desirability (or lack thereof) of certain regions on the maze. In particular, all game objects exert positive or negative (from the perspective of *Ms. Pac-Man*) influences onto to the surroundings and the influence map is simply the sum of all these local influences. The authors constructed the influence map based on pills, power pills, and ghosts (edible and nonedible). The influences were designed to encode the basic intuition as to what constitutes a good *Ms. Pac-Man* player: eat pills and edible ghosts, avoid nonedible ghosts. For the controller to make a decision during the game, all surrounding nodes are evaluated using the influence map and the maximum value is subsequently selected: influences are local and their impact decays geometrically (Euclidean distances have been used) inversely to distance. In their experiments, the authors also considered the ability of a simple hill climber to optimize the parameters of the influence map and found that good parameters may be found quickly. The authors conclude that their controller behaves sensibly but would at times oscillate if the influence values of neighboring nodes were very similar (this would be resolved dynamically during gameplay).

Anthony *et al.* [79] explore the idea of maximizing *empowerment* as an action decision mechanism for intelligent agents, and employ a simplified version of *Pac-Man* (without pills, turning it into a survival predator–prey game). Empowerment is a measure that determines how much an agent can influence its environment by performing actions on it. The authors introduce several versions of empowerment techniques to propose its use for general game playing, as it can be used without the need of game or domain knowledge. Furthermore, this study proposes a mechanism to group actions into strategies, showing that the combination of both ideas provides the agent with more control over the environment and policies that are preferred by humans.

Svensson and Johansson [54] design influence-map-based controllers for both *Ms. Pac-Man* and the ghosts. The *Ms. Pac-Man* controller takes into account seven influence maps (the influence of lookahead positions of the *Pac-Man*, the distance to the nearest pill, power pill, ghost, and edible ghost, and the influence of freedom of choice), which are measured by five parameters. At the first stage, the authors picked up the two most influential parameters by running lots of experiments and studying the landscape of the game's score over the parameter search space. Then, these two parameters are fixed at their optimal values to optimize the other parameters. A ghost controller is designed in the same way but only three influence maps (measured by three parameters) are considered: the distance between the *Pac-Man* and the nearest power pill, the lookahead positions of the ghosts, and the distances between ghosts.

More recently, Costa *et al.* [90] model a simplified version of the Pac-Mac game using typed graph grammar with negative application conditions. The game objects (*Pac-Man*, ghosts, berry, and block) are represented by nodes in the graphs. An arrow between two nodes represents that the game object at the arrowhead can be taken by the game object at the tail, for instance, a *Pac-Man* can move to a block or a berry. Rules are represented by such graphs.

Finally, Koriche *et al.* [91] transfer the *Pac-Man* game to a stochastic constraint satisfaction problem with 93 variables and 22 constraints, and design three *Pac-Man* agents using upper confidence bounds for trees (UCT), maintaining arc consistency together with upper confidence bound (MAC-UCB) method and classical forward checking (FC) together with UCB (FC-UCB), respectively. The authors show that MAC-UCB statistically outperforms both UCT and FC-UCB.

H. General Video Game Playing

GVGP [117] is a discipline that lies at the intersection of game AI and artificial general intelligence. The challenge of GVGP is to design controllers that can play any video game from a diverse set of possible games without knowing the details of the games in advance. This makes for a much greater challenge, since game-specific heuristics and control modules are of limited (or zero) value.

One of the most notable works on GVGP was done by Mnih *et al.* [34], who applied deep Q-learning in order to achieve human level of play in 49 of the games from the classic Atari 2600 collection. Although each network was trained separately for each specific game, the generality of their approach resides in the fact that they all share the same architecture, hyperparameters, and training procedures. The network employed received only the screen capture and the current score of the game, and gave the action to take as output. This method has been used successfully to outperform human players in many other games but performed inferior to humans in the case of *Pac-Man*.

The game of *Pac-Man* has also been featured in other popular GVGP frameworks such as VGDL [118], developed in to pyVGDL by Schaul [86], [87] and the general video game AI (GVGAI) framework (www.gvgai.net; by Perez *et al.* [88], [89]). In these works, *Pac-Man* (along with other games) is implemented in the video game description language, which allows 2-D arcade games to be specified in simple, plain text. These frameworks also enable the possibility of writing controllers, which can interact with the engine by receiving game state and supplying an action every 40 ms.

The GVGAI framework was a reimplementation of the PyVGDL engine in Java for the GVGAI competition. During the first edition of this contest, when *Pac-Man* featured as an unknown game: the winning approach of the competition (Open-Loop Expectimax Tree Search [88]) achieved a 100% victory rate in this game (meaning it cleared all levels). This implementation of *Pac-Man* differs greatly from the original *Pac-Man* and *Ms. Pac-Man*, with ghosts that chase the player in a simpler way. However, it is still worth pointing out that the agent was able to clear all mazes without prior knowledge of the game that was being played.

V. Ghost Controllers

Traditionally *Pac-Man* is viewed from the perspective of the gamer, with the ghosts as opponents that are part of the game. Far fewer papers have been published so far that aim to develop better strategies for the ghosts (see Table V). The data in Table III show that a slight preference for *Ms. Pac-Man* controllers remains but this difference is diminishing and it is reasonable to expect more papers to be published in the near future centered around the ghost team. This section reviews all studies concerned with the ghosts, as well as a small selection of research on predator–prey models (of which *Pac-Man* is a special case). The methods involved in designing controllers for ghosts are summarized in Table VI.

A. Rule-Based Approaches

Gagne and Congdon [52] design a rule-based controller, namely FRIGHT, for the ghosts. In total, 15 high-level conditions (parameters), including hard and soft conditions, are extracted from current game state for rule selection. The determination of a rule is similar to a SATisfiability problem. Only the rule that has all the conditions satisfied is selected. Each rule refers to one single action in $\{$*Retreat*, *Evade*, *Surround*, *Attack*, *Protect*$\}$. If more than one rule is satisfied, the one that meets most soft conditions is selected. If no rule is satisfied, the *Attack* action is applied. Besides hand-coded rules, some rules are generated using evolution strategies.

B. Nature-Inspired Heuristics

Recio *et al.* [53] develop an ant-colony-based controller for the ghost team. The objective of the ghost team is to cut across *Ms. Pac-Man*'s path, using two different types of ants. For the *Ms. Pac-Man* agent, explorer ants are launched from all the adjacent nodes to her location, indicating a measure of optimality for each one of the possible paths she can take. Additionally, hunter ants are launched from the nodes contiguous to each one of the ghosts' locations, keeping their current directions. In this study, the proposed approaches are compared against the benchmark NPCs and the other entries of the 2011 IEEE Congress on Evolutionary Computation (CEC) *Ms. Pac-Man* Versus Ghost Team Competition.

Tamura and Torii [58] generate a controller for the ghosts using GE and Backus–Naur form grammars. A population of 100 individuals is evolved during 50 generations. The controller aims to minimize the average score for *Pac-Man* over ten simulations of each individual. The designed controller is compared to three hand-coded ghost controllers using the *Ms. Pac-Man* versus Ghost Competition engine, except that, only one out of the four levels is played and the *Pac-Man* is not awarded one more life at 10 000 points. The authors defined grammars that design more aggressive ghosts and avoid two ghosts taking the same routes.

Liberatore *et al.* [60] design flocking strategies (FS), a swarm intelligence technique, for the ghost team. The authors classify the ghosts as three types of actors according to their states: normal, hunted, and blinking. Each FS is a mapping of a ghost state and the type of interacted actor, among the five actor types (including *Pac-Man* and power pill), to a flocking rule that calculates the next move for the ghost. In total, 50 FSs are randomly initialized and evolved offline as individuals in a GA aiming to minimize *Ms. Pac-Man*'s score. Neither online learning nor centralized control is required. Then, more FS for the team of ghosts are compared by Liberatore *et al.* [63]. Concretely, the authors present a GA with lexicographic ranking to optimize FS-based ghost controllers. A comparison is made between flocks of homogeneous and heterogeneous individuals, and the results are matched with those from the agents present in the *Ms. Pac-Man* versus Ghosts Competition framework, and some other approaches in the literature. The authors found that their approach obtained better results than those agents present in the framework, and then some of the other controllers employed, with a better performance in the case of homogeneous teams.

C. Reinforcement Learning

Beume *et al.* [37] are interested in creating effective controllers for the NPCs of the game (i.e., the ghosts), motivated by the need for entertaining NPCs (citing the work by Yannakakis and Hallam [94]). The focus is on how NPCs may develop different behaviors if different learning strategies are used. An extended version of the NJam *Pac-Man* clone is used, reproducing the original first level and creating numerous test scenarios (maps and states of power pills); simple rules are used to control *Ms. Pac-Man*. Learning is done offline and two approaches are compared: model based and direct learning. The model-based approach uses patterns/scenarios, whereas the direct approach utilizes the success/failure of completed games. As in POMDPs, the NPCs only receive local knowledge (i.e., unlike in many modern video games where NPC-intelligence derives from global, normally inaccessible, knowledge). Feedforward networks using back propagation are used for the model-based case and evolutionary algorithms for the direct case. The inputs to the networks and the EA are identical and related to the state and position of all game objects (i.e., pills, *Pac-Man*, etc.) that are within the range of sight of the ghost. The behaviors of the ghosts are broken down into three states: *roam*, *evade*, and *chase*; a separate network is used for each and gets activated whenever the corresponding scenario occurs in the game. The networks are then trained offline with patterns that correspond to these scenarios. The EA is used to learn improvements to a set of rules that are subsequently used for game play. Two EAs are used to evolve both pure and mixed strategies. One $(1+1)$-EA is used for each of the four ghosts in the game, with the fitness of a ghost being determined by the number of correct decisions made while playing the game. The authors conclude that both forms of learning were effective in shortening the time it took to eat *Pac-Man*.

D. Neuro-Evolutionary Approaches

Similar to Beume *et al.* [37], Wittkamp *et al.* [38] explore the use of computational intelligence (CI) techniques for real-time learning to evolve strategies for the ghost team. Using a neural network to control the ghosts, the focus is on team-work development that makes use of continuous short-term learning to regularly update the (overall) strategy of the ghosts. The authors make use of NEAT (see [112]) to continuously evolve the ANN's topology and weights. The goal is for the ghosts to learn as a team to exploit the weaknesses of *Pac-Man*, which is controlled by a hand-coded controller (*pacbot*). Each ghost has its own neural network, which acts as a move evaluator, and is evolved by playing a series of games (offline). Real-time learning is subsequently used to learn team-specific behaviors in the short term. The fitness of each neural network is determined by the performance of the team, not the individual. Four separate instances of NEAT (four populations) are executed in parallel and the best one is chosen for the actual game play (in the paper, the authors actually disregard real time and do the learning sequentially). Each differs according to the distance of the ghost to *Pac-Man*. Each neural network has 19 inputs regarding the current state of the game, including distances to *Pac-Man* and the objects in the game. The authors comment that their approach of short-term learning allows the ghosts to avoid having to learn complex general game-playing strategies. In the experiments, numerous different behaviors are learned and performance is compared against the game's original ghost team. One of the most important results is the emergence of structured team play where the ghosts successfully limit the number of possible escape routes for *Pac-Man*.

Hasan and Khondker [77] evolve neural networks for ghosts using a $(10+10)$-ES, with each network having 20 hidden neurons (this a similar setup to the one used by Lucas [12]). The noteworthy aspect of their implementation of the game is its integration in to social media using Heroku.

Finally, in contrast to the work focused on controlling the entire team of ghosts, Dai *et al.* [33] build an agent using evolutionary neural networks particularly for the red ghost, Blinky, which is the most aggressive one. In this work, only the weights of the networks are evolved. The authors show that the evolved controller alone is able to capture PacMan more often than the default implementation included in the software, and the presence of the evolved ghost in the team makes a positive difference with regards to how quickly PacMan is captured.

E. Tree Search and MC

Nguyen and Thawonmas [49], [56] introduce the use of MCTS to control the ghost team, presenting the bot that won the first *Ms. Pac-Man* versus Ghost Team Competition at 2011 IEEE CEC. In this approach, one of the ghosts (Blinky) moves with a set of predefined rules, whereas the other three employ MCTS. This was implemented like this in order to balance the complexity of many hand-coded rule-based ghosts and the reliability of MCTS controllers. In [49], *Ms. Pac-Man*'s movements are predicted by the k-nearest-neighbor algorithm.

In [56], *Ms. Pac-Man*'s movements and position are predicted using MC simulations from her current location, where it is assumed that the *Pac-Man* agent is trying to minimize the distance to certain objectives, according to her moves in previous time steps. Instead of simulating moves on a tick per tick basis, the authors simulate actions as moving from one crosspoint to another, providing a deeper lookahead for the simulations. Nodes are evaluated according to different criteria, such as inverted *Pac-Man* score, spread in the location of ghosts and relative distance to *Pac-Man*. The authors show that the combination of MCTS and rule-based ghosts outperforms that of solely MCTS controllers, as this provides a natural way of introducing domain knowledge into the problem while keeping the search capabilities of MCTS.

F. Predator–Prey Scenarios

Finally, it is also worth considering some related work that focuses on the more general case of predator–prey scenarios (as reviewed in [11]). For instance, Haynes *et al.* [119], [120] strive to generate programs for the coordination of cooperative autonomous agents in pursuit of a common goal. The authors consider a simple predator–prey pursuit game, noting that the problem is easy to describe yet extremely difficult to solve. An extension of GP was used to evolve teams of agents with different strategies for their movements. Similarly, Luke and Spector [121] consider different breeding strategies and coordination mechanisms for multiagent systems evolved using GP. In particular, the authors are interested in the performance of homogeneous and heterogeneous teams: In a heterogeneous team of agents, each agent is controlled by a different algorithm, whereas homogeneous agents are all controlled by the same mechanism. The problem considered is called the Serengeti world: a toroidal, continuous, 2-D landscape, inhabited by gazelles and lions.

Examples of more recent work regarding predator–prey scenarios include Rawal *et al.* [122], Rajagopalan *et al.* [123], and Cardona *et al.* [57]. In the first two cases, the authors consider the coevolution of simultaneous cooperative and competitive behaviors in a complex predator–prey domain. The authors propose an extended neural-network architecture to allow for incremental coevolutionary improvements in the agents' performance. This mechanism demonstrates the hierarchical cooperation and competition in teams of prey and predators. The authors further note that due to sustained coevolution in this complex domain, high-level pursuit-evasion behaviors emerge. The third case has been discussed previously in Section IV-C, in which competitive coevolution was applied to evolve both *Pac-Man* and ghost controllers.

VI. Player Profiles and Measures of Fun

The research reviewed in Sections IV and V is primarily concerned with the development of controllers that play the game as well as possible. *Pac-Man* has also been used extensively as a tool to gain a better understanding of the behavior of gamers and what constitutes fun in a game. These studies are outlined next.

Beume *et al.* [97] are interested in measuring the perceived level of fun a player experiences when playing *Pac-Man* (based on NJam). Instead of questioning gamers directly regarding their enjoyment of the game, the authors propose to use theory of flow,[6] as defined in the field of psychology, and query whether this provides a more reliable indicator of this subjective subject matter. The authors conduct a sizeable study with human players (85 samples were effectively used in the study), using various techniques to control the ghosts, including neural networks and evolutionary algorithms. The study combines measures of fun with questionnaires to validate the feasibility of the measures.

The work is based on the assumption that flow is experienced when the player's skill is close to what is required to achieve the task (based on [124]). In contrast to the work by Yannakakis *et al.* (see below), Beume *et al.* take the player's point of view to measure what is perceived (rather than the analysis of game statistics from automatically generated games). Beume *et al.* measure the time fraction of the game in which the player is confronted with interesting situations. These are defined as interactions with the ghosts (based on proximity), which should also be correlated to the perceived difficulty of the game, and hence can be matched to skill to establish flow. The experiments attempt to establish whether flow is measurable and whether it is a good measure of fun. The authors also compare their interaction measure against that of Yannakakis and Hallam [93]. They found divergence in the results, concluding that neither measure is able to accurately capture the degree of fun experienced by the players.

Cowley *et al.* [98], [99] present a series of studies aimed at analyzing player behavior in the game *Pac-Man*. The goal of this work is to gain a better understanding of player profiles (in particular, skill and preference for certain situations) and to use this information for in-game adaptation to create games better suited to individual gamers (dynamic player modeling). The authors use their own, somewhat simplified, implementation of the game where the ghosts move entirely randomly. Their approach is based on low-level in-game data capture that measures deviations from optimal choices throughout the game, potentially revealing the gamer's skill and play preference. These data constitute key information in the analysis of optimal player experience. The authors also use theory of flow and consider games as information theory systems, using decision theory to model the player's choices. Acknowledging that players seldom play perfectly, due to skill and preference, deviations from optimal decisions may reveal useful information regarding the player's profile. In particular, the authors relate the difficulty of decision-making on a move-by-move basis to the overall challenge of the game.

In the first study, Cowley *et al.* define utility functions for five different states that correspond to basic behaviors in the game (such as hunt and flee). The authors find that although prediction rates are not overly reliable (47%–60%), players are categorized reasonably well, although not sufficiently so far in-game content adaptation. Then, the authors increase the reliability and granularity of predictions, albeit at the cost of speed: The improved approach takes into account all relevant features of the state (weighted feature vector), looking ahead in time using tree search. The authors find that the overall accuracy is lower than before (44%) but that some features can lead to robust prediction of player movements. Cowley *et al.* [99] finally improve the previous two approaches to increase the prediction accuracy significantly (by 26%–70.5%) as tested on a variety of gamers in an extended experimental study.

The concept of "fun" is difficult to measure and quantify. Yannakakis and Hallam [93], [94], [96] and Yannakakis and Maragoudakis [95] present a series of studies aimed at better understanding what constitutes fun and enjoyment in video games. The original emphasis is on predator–prey multicharacter video games and *Pac-Man* was chosen as the case study throughout all these works. The authors use their own implementation of the game, which does not feature power pills. In [93], Yannakakis and Hallam view *Pac-Man* from the ghosts' perspective and attempt to evolve neural-controlled ghost teams that play effectively against a fixed strategy *Pac-Man* player. The authors find that near-optimal play makes the game less interesting, as the player is constantly eliminated (see [125] cited in [93]). The authors subsequently propose a general metric of interest for predator–prey video games and this measure is then used to adapt opponent (i.e., ghosts) strategies online, to maintain high levels of interest throughout the game: starting with near-optimal offline trained behaviors, the ghost controller is adapted online according to the level of interest (based on the player's behavior). The authors investigate different learning procedures to achieve this reliably in real time. A fully connected multilayered feedforward neural network is used to control the ghosts and three fixed (nonevolving) strategies are used as a control (including a near-optimal strategy).

To define a measure of interesting behavior, Yannakakis and Hallam first define criteria that make a game interesting and then

[6]The concept of flow basically refers to the involvement of a person in a task: heavier involvement minimizes the perception of external events. Flow is subsequently defined as an optimal experience whereby a person is so engaged in what they are doing, everything else (including their self-consciousness) is perceptually eliminated.

quantify these attributes into a mathematical formula. These behaviors are subsequently measured based on game data and combined into an overall measure of interest. Each neural controller is evolved offline by playing a series of games. The authors observe both the performance of the controller as well as its impact on the interest of the game. The authors show that the best controllers, in terms of performance, do not have the highest rating for interest, illustrating the tradeoff one has to make. The best performing ghosts are then used for online adaptation using the interest of the game as the goal. The latter experiment includes different strategies for *Pac-Man* that may change midgame, and results show a successful adaptation to the changing game dynamics.

In [95], Yannakakis and Maragoudakis continue this line of work to take into account the player's contribution to the emergence of interest using a player modeling mechanism. In particular, the authors use Bayesian networks, trained on player data, to infer appropriate parameter values for the online learning procedure. The model of the player is based on numerous features of the game, including score and time played. The authors find that the player model improves the online adaptation process. In [94], Yannakakis and Hallam continue this work to test the online learning mechanism over more complex stages (mazes) of the game. The authors also consider the relationship between the interest measure and the topology of each stage (a complexity measure of each topology is also proposed). They find that the interest measure is robust and independent of the stage. All these studies are summarized and extended in [96], including motivation and background.

Sombat *et al.* [55], [66] focus on understanding user preference (as correlated to a perception of what constitutes fun) and collect data from human players using the resources of the *Ms. Pac-Man* versus Ghosts Competition. Every human player is required to play two games against each of the ghost teams and select their preferred team. Interestingly, the ghost controllers, which ranked highest in the competition, are usually not voted as the most enjoyable. Similarly, Baumgarten [76] collects game data of more than 200 players online, then studies game feature and playing style using linear discriminant analysis on the discretized survey data. The physical interaction with the device is determined to be a key feature during the game playing. More analysis and discussion on feature influence may be found in [76].

VII. Research in Other Fields

Not all work in *Pac-Man* has aimed to create a better AI controller for the game, or even to estimate and control aspects of player experience: There has also been work on using it as a testbed to study some other phenomena. An example of this is a cross-disciplinary analysis written by Wade [126], in which he analyzes the game from several different points of view. His work touches on game design (using sprites and animations to suggest a nonviolence intent), as well as including references to classic culture (such as the *Theseus' Minotaur Labyrinth* and the breadcrumbs of *Hansel and Gretel*) and, mainly, a sociological approach focused on the need to strike a balance between survival and the consumerism of modern times. Wade makes a direct connection between the success of the game and the societal and political systems of the western world.

A. Sociology

Cheok *et al.* [104], [105] use the concept of *Pac-Man* to propose human *Pac-Man*, an interactive implementation of the game where teams of Pac-Men and ghosts move in the real world using a variety of sensors and head-mounted displays. The goal of this work is to pioneer a new form of gaming where virtual worlds and reality merge seamlessly to allow gamers a socially interactive experience. Players move freely in the real world, equipped with a variety of sensors tracking their position and a head-mounted display that superimposes virtual entities on the real world. These entities resemble the pills of the original game. Other entities include actual physical objects equipped with bluetooth, allowing gamers to physically interact.

The game is played by a team of Pac-Men and a team of ghosts and the rules of the game are quite similar to the original. Catching an opponent is done by tapping a sensor on the opponent's shoulder. Each gamer has a helper who is not part of the game but instead sits at a desktop showing an overview of the game, and may help the gamers to achieve their objectives. The sensors include GPS, dead-reckoning modules, and inertial sensors to keep track of the players' positions, both inside buildings and outside. The players' heads are also tracked to display virtual entities via head-up displays. Bluetooth is used to allow the gamers to interact with physical objects scattered around the world. Communication to a central server that maintains the state of the game is done wirelessly and the game state is translated into a virtual world where the positions of all players correspond to those in the real world. In [105], Cheok *et al.* carry out a user study consisting of 23 test subjects to test the viability of their approach. Given the feedback from the gamers, collected via questionnaires, the authors find that most aspects of the game are well perceived while some aspects, such as the amount of hardware required, are somewhat cumbersome.

B. Psychology

Ohno and Ogasawara [100] use *Pac-Man* as an example domain to develop a cognitive model of information acquisition processes in highly interactive tasks. The goal of this work is to provide a cognitive model with the potential to estimate the performance of a human in performing a bidirectional interactive task on a computer that dynamically responds to user inputs. Such models may help interface designers to optimize the user experience. The authors propose the information acquisition model that focuses on the symbol encoding process on the computer display. Data were obtained by tracking eye movement of the user while playing *Pac-Man*. The authors use their own implementation of the game, which differs in many ways from the original but is able to record sequences of moves and keystroke information. The experiment challenged each participant to maximize the game's score and included an exercise where some objects would disappear from the screen midgame with the user asked to note where these objects had been. The primary goal of the study was a comparison between the authors' proposed model and GOMS (goals, operators, methods, and selection rules), a simpler model on which their model was based. One of the results shows that a key factor in improving the gamer's skill is the ability to focus on the most relevant parts of the screen, given the very limited time to execute a move, and how quickly the relevant game objects are identified.

Mobbs *et al.* [101] investigate the reaction of humans (14 subjects were tested) in the face of threats, using a predator–prey scenario where volunteers are pursued through a 2-D virtual maze and experience pain if captured by the predator. This work relates to studies that examine the ability of any organism to switch flexibly between defensive states in response to a threat. The authors monitored the subjects' brain activity using magnetic resonance imaging and found that certain shifts occurred in brain activity as the predator moved closer. The scenario only corresponds loosely to *Pac-Man* as only a single predator (ghost) was used and the mazes were created specifically for the task at hand (they were much simpler than the original mazes consisting of 9×13 cells in a rectangular grid).

Based on the psychological belief-desire theory of emotion, seven emotions are determined by a rule-based partition using the belief and desire of the given state. Moerland *et al.* [81] study the models of emotion generation for two emotions in particular: fear and hope. The fear and hope of a state can be estimated though forward planning using RL agents. Both ϵ-greedy and *softmax* policies have been applied to estimate the fear and hope of *Pac-Man* with the presence of a ghost nearby or not.

C. Brain–Computer Interfaces

A BCI is a form of human–machine interaction that analyzes signals from the brain cortex, in response to certain stimuli. The variations of electrical activity can be measured with appropriate technology and used for machine control. Due to the limitations of these techniques as forms of user input and data collection, researchers typically implemented their own versions of *Pac-Man* to address these difficulties.

In their work, Krepki *et al.* [83] read and classify electroencephalogram (EEG) and electromyogram signals to determine when *Pac-Man* will turn, and in which direction. The authors employed motor imagery (in which the algorithm learns to associate imaginary movements with certain outputs) with two classes: turn right and left. Information about the intended command is fed back to the player (i.e., the direction PacMan's *nose* is pointing). Users were able to clear the level (eating pills and finding an *exit* added to the maze) by means of this approach, although the time spent to do so was much longer than when using a conventional keyboard.

Girouard *et al.* [84] employ this technique as a way to provide an additional input, rather than using BCI to control and play the game. This form of *Passive* BCI works in conjunction with traditional input means (arrow keys on a keyboard). The authors employed functional near-infrared spectroscopy, which measures changes in hemoglobin concentrations, in order to differentiate between the state of play in two dimensions: play versus rest, and different difficulty levels (easy versus hard). Results showed an accuracy of 94.4% in the first dimension, and 61.6% in the second.

Finally, Reuderink *et al.* [85] used a modified version of *Pac-Man* to analyze the effect of frustration in EEG signals. Frustration is induced in two different ways: at the input level (15% of the key presses are missed at random) and at the visual output stage (freezing the screen with a probability of 5% for two to five frames). Analysis of the collected data shows that frustration can deteriorate BCI performance and that it should be possible to detect player state changes (boredom, fatigue, etc.) during game play.

D. Biology and Animals

In [106], Van Eck and Lamers investigate as to whether it is possible to play video games against animals: Motivated by the entertainment value of playing with pets, the authors pose the question whether a similar experience could be obtained in a video-game-like scenario. In particular, Van Eck and Lamers replace the computer AI with field crickets in a simplified version of *Pac-Man*, recreated as a real maze. The authors concentrate on the differences that emerge when gamers interact with traditional NPCs or real animals. Four crickets are used to replace the ghosts and camera tracking is employed to update the game state of the video game. The instinctive behavior of the animals leads to some interesting observations: when agitated, the animals move erratically through the maze to later group together and remain stationary. Also, the speed varies whenever a cricket is moving through the maze. To create greater interaction between the animals and the gamer, the authors use vibrations, which the crickets perceive as danger, to either lead the crickets toward or away from *Pac-Man*.

E. Education

Pac-Man (or *Ms. Pac-Man*) has featured in research centered around education as well as being used as a teaching aid. In many cases, *Pac-Man* has been used merely as a metaphor, whereas in other cases the game took on a central role. Squire [127] presents a general survey of video games in education, arguing that educators have ignored the educational potential of gaming. The survey highlights how *Pac-Man* was among the games that sparked the "*Pac-Man* Theory of Motivation" [128], which posed the question as to whether the magic of *Pac-Man* can be exploited for student involvement, enjoyment, and commitment ([128] cited in [127]).[7]

DeNero and Klein [107] use their own implementation of the game to teach fundamental concepts in their introductory course on artificial intelligence. The authors comment that the breadth of techniques in AI may appear incoherent to students new to the subject matter and subsequently propose a series of programming projects centered around the game of *Pac-Man* to tightly integrate the different techniques covered in the course. These topics include state-space search, Markov decision processes,

[7]Bowman [128] analyzes *Pac-Man* players using the concept of flow (see [124]; this concept is also exploited by [97] to measure fun in games).

RL, and probabilistic tracking [107]. DeNero and Klein motivate their choice of *Pac-Man* as the game is interesting and fun to play, and it supports a range of domains that may be tackled using a variety of techniques. Furthermore, the framework lends itself nicely to the visualization of techniques in 2-D space. The framework is publicly available and has been adopted by several other universities.

Their software is implemented to approximate the original game sufficiently well while offering the flexibility to create different scenarios. Furthermore, the software lends itself to automatic assessment using scripts. Interestingly, DeNero and Klein often include (mini) competitions for each exercise to determine the best technique, hinting at the potential of game competitions in education, particularly in conjunction with online courses (which have recently grown in popularity). Similarly, the game competitions outlined in Section III have contributed to the wider use of *Pac-Man* in higher education. In particular, the *Ms. Pac-Man* versus Ghosts Competition framework has been used by several universities in their AI courses (e.g., University of California Santa Cruz, Georgia Institute of Technology, New York University) and numerous students have made use of the software for their B.Sc. and M.Sc. projects.

Following [107], Silla [80] presents an assignment that allows their students to learn GAs using another Java *Pac-Man* framework, developed by Ted Grenager, which is less known than the *Ms. Pac-Man* versus Ghosts Competitions.

Another example of how *Pac-Man* is used in higher education is given by Dickinson *et al.* [102] from Brown University: The authors recreate *Pac-Man* as a physical game played with robots. The fifth floor of the computer science building functions as the maze, complete with pills, power pills, and ghosts. The implementations of the students subsequently compete against one another to achieve the highest score. The hardware setup is purposely kept inexpensive, using off-the-shelf Roomba robots (Roomba robots are vacuum cleaners and hence can collect pills by suction). The Roomba *Pac-Man* task is used throughout the course as a common theme to guide students through a series of labs and projects that include obstacle avoidance, object seeking, AI behavior by means of subsumption, MC localization, and path planning. The authors conclude that this approach has been successful and that students were motivated by the challenge the game represented.

Rao [103] is also interested in using games to provide a stimulating environment grounded in real-world problems. Their intent is to teach robotics. The setup here is similar to [102] in terms of hardware used but the game is simplified to two robots, one being *Pac-Man* and the other a ghost. The maze used consists of a small square arena with four blocks near the corners. The behavioral model of *Pac-Man* tries to escape the ghost, which in turn is trying to catch the *Pac-Man*.

Finally, *Pac-Man* has a role to play in continuing education. For example, Tose[8] was not a computer science researcher but taught himself Java especially to enter the *Ms. Pac-Man* versus Ghosts Competition, and went on to win the *Pac-Man* category, and finish third in the ghost-team category. We do not have details of the entries beyond these slides[9] but they contain a number of good ideas such as using a recursive flood fill to work out how ghosts can propagate through a maze that takes in to account their current directions of travel, and hence goes beyond a simple shortest path analysis. This was used in order to analyze danger points in the maze. We have not included this in the previous sections on *Pac-Man* and ghost controllers as it has not been written up in the peer-reviewed literature, but it is a good example of the wide and deep appeal of the game.

F. Other

Yacoub *et al.* [129] review the PRotocol MEta LAnguage (PROMELA) and its three extensions for modeling and verification of software, and present their own extension called Discrete-Event PROMELA (DEv-PROMELA). As implied by its name, DEv-PROMELA is based on the concept of discrete-event simulation. To demonstrate how DEv-PROMELA can be applied, the authors use *Pac-Man*: The game is modeled as an asynchronous protocol between *Pac-Man* and the ghosts. The game state, player events (i.e., an action), and event(s) from the ghosts are checked every 0.1 unit of time. The DEv-PROMELA can help the game designers check the specifications at an early stage, before implementation.

The difficulty of *Pac-Man* has been commented on in a few studies. DeNero and Klein [107] point out that *Pac-Man* is a challenging game as just eating all the pills in as few time steps as possible corresponds to a nonplanar travelling salesman problem. Viglietta [109] takes this notion further, showing that the game is indeed NP-hard: Viglietta aims to single out recurrent features/mechanics in video games that allow reduction of the game to a known hard problem. The focus is on "script-less" games from the period 1980–1998. Several *metatheorems* are defined and applied to a variety of games to establish their hardness. The metatheorems focus on various characteristics such as doors, switches, and pressure plates. Among the many games considered, the authors show that *Pac-Man* is NP-hard: The decision problem considered is whether a level may be cleared without losing a life. Viglietta assumes full configurability of the game, including the number of ghosts and their behaviors, the locations of pills, power pills, and walls: Given a very specific setup, the author shows the existence of *single-use paths*, a metatheorem proven to lead to NP-hardness given a reduction from a Hamiltonian cycle.

Øgland [108] makes use of *Pac-Man* in a rather abstract manner (similar to the *Pac-Man* theory of motivation) and uses the game to produce quality plans for organizational development frameworks. Øgland is motivated by research that shows playing video games may have an impact on people's decision-making processes, including conflict resolution. The purpose of his study is to identify a *Pac-Man* model for developing optimal strategies for Total Quality Management (TQM). In particular, Øgland shows how *Pac-Man* strategies may be used effectively as TQM strategies. To achieve this, *Pac-Man* is viewed (from a game-theoretic perspective) as a five-player game of

[8]http://www.diego-perez.net/DarylTose.pptx

[9]See footnote 8.

imperfect information. TQM policies are expressed in terms of the *Pac-Man* model, assigning *Pac-Man* scores to TQM activities. For instance, eating a dot could correspond to document verification. A long-term TQM strategy is subsequently described in terms of playing the game, and strategies effective in the game are translated to effective strategies for TQM management.

Becroft *et al.* [39] implement a game-independent behavior tree tool, called AIPaint, aimed at helping game designers build behavior decision trees for AI agents by drawing simple schema using different shapes provided: for instance, using an arrow to show the direction to move toward. No additional programming by a human designer is required. AIPaint has been used for designing Blinky and Clyde's behaviors, evaluated by some undergraduate students with little programming experience, and obtained positive feedback.

Finally, Maycock and Thompson [40] try to improve the human playing experience on touchscreen devices and implemented an android game as a testbed, based on the original *Ms. Pac-Man* Screen-Capture Competition engine. An A^* search is used to visit the checkpoints periodically and help better understand the screen taps and navigation. Although the improvement in terms of game scores and number of successful taps is small, this work is still particularly interesting as Maycock and Thompson combine the AI methods to commercial products to improve user experience.

VIII. Conclusion and Prospects for the Future

This paper presents a detailed overview of peer-reviewed studies that focus, in one way or another, on the video game *Pac-Man* (or any of its many variants) as part of scientific research. The overview highlights the wide variety of research that exists, including studies in computer science, neural engineering, psychology, sociology, robotics, and biology. The reasons for the interest in this game are discussed, including the potential of *Pac-Man* in higher education (as a tool for teaching and assessment). The renewed interest in the game, most likely spurred on by the recent academic game competitions that focus on *Ms. Pac-Man*, is easily justified: *Pac-Man* remains a promising platform for research due to its many characteristics that make it stand out from other games such as Chess or Go. It has a suitable degree of complexity, with challenges that include real-time components and heterogeneous player types. The noise of the game caused by the stochasticity of opponents and rules (e.g., random reversals) is a particular challenge that requires a controller to be robust to perform well.

On the other hand, the complexity of the game is contained, allowing for relatively efficient representations of game states (e.g., graphs, heatmaps). Furthermore, the action set is quite limited and performance can be judged precisely by the game's score. This balance of complexity, combined with the immense popularity of the game (past and present) and the recent availability of the game in various formats (screen capture, various game engines), makes it a promising choice for research in artificial intelligence and beyond. In the future, other computer games may become more popular for research, but until then there are still plenty of opportunities to advance research in *Pac-Man*.

In particular, rule-based approaches are still a dominant force in terms of performance when it comes to playing the game as well as possible (although other approaches such as MCTS have recently caught up with them). It is interesting to observe how specific aspects of gameplay may make a significant impact on the performance of the controller, such as the order of how the pills are consumed (see [29]) or the absolute position in the maze (see [20]). However, even if computational techniques should converge into a near-optimal playing strategy in the near future, the game as a concept remains interesting. For example, the work of Silver [74] has shown how giving the agent a restricted view of the game can transform the problem into a POMDP. This area of research has also been captured by the most recent *Ms. Pac-Man* competition, where characters can only observe the game state partially [14].

The screen-capture competition has highlighted how the game may pose auxiliary challenges (i.e., developing good screen readers). An important experiment to carry out is to test the performance of deep Q networks on the original *Ms. Pac-Man* game, as we only know of reports of their performance on the Atari 2600 version [34], [35], which is not only easier but also lacks the wealth of human play data for comparison purposes. Challenges may also be extended to include new aspects such as automatic content generation (mazes, rules).

We also envisage versions of the game that move closer toward the challenges of general game playing, where the maze layout may be varied and the rules may be modified or extended in a variety of ways (e.g., with additional power-ups such as missiles, or the ability to block corridors or unlock doors), the exact details of which would not be known to the agent prior to the commencement of a competition. Given the range of possibilities and the continued human interest in the game, we see a bright future for *Pac-Man* variants in AI research, and also the possibility of AI contributing to the development of new versions of the game that are even more fun for people to play.

Acknowledgment

The authors would like to thank all participants of the *Ms. Pac-Man* Screen-Capture Competition, the *Ms. Pac-Man* versus Ghosts Competition, and the *Ms. Pac-Man* versus Ghost Team Competition. Many competitors contributed actively toward improving the competitions and making them popular.

References

[1] Entertainment Software Association, Washington, DC, USA, "The 2015 report: A Year of innovation and achievement," 2016.
[2] K. Berens and G. Howard, *The Rough Guide to Videogames*. London, U.K.: Rough Guides Ltd., 2008.
[3] Video Game Books, Inc., *Playing Ms. Pac-Man to Win*. New York, NY, USA: Simon and Schuster, 1982.
[4] E. Zavisca and G. Beltowski, *Break a Million! at Pac Man*. New York, NY, USA: Delair Publ. Company, 1982.
[5] T. Mott, *1001 Video Games You Must Play Before You Die*. London, U.K.: Cassell, 2010.
[6] J. Sellers, *Arcade Fever: The Fan's Guide To The Golden Age of Video Games*. Philadelphia, PA, USA: Running Press, 2001.

[7] C. Melissinos and P. O'Rourke, *The Art of Video Games: From Pacman to Mass Effect*. New York, NY, USA: Welcome Books, 2012.

[8] S. M. Lucas, "Ms *Pac-Man* competition," *ACM SIGEVOlution Newslett.*, vol. 4, pp. 10–11, 2008.

[9] G. Foderaro, A. Swingler, and S. Ferrari, "A model-based cell decomposition approach to on-line pursuit-evasion path planning and the video game *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 281–287.

[10] G. Foderaro, A. Swingler, and S. Ferrari, "A model-based approach to optimizing *Ms. Pac-Man* game strategies in real time," *IEEE Trans. Comput. Intell. AI Games*, vol. 9, no. 2, pp. 153–165, Jun. 2017.

[11] P. Rohlfshagen and S. Lucas, "Ms *Pac-Man* versus Ghost Team CEC 2011 competition," in *Proc. IEEE Congr. Evol. Comput.*, 2011, pp. 70–77.

[12] S. M. Lucas, "Evolving a neural network location evaluator to play *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, 2005, pp. 203–210.

[13] S. Samothrakis, D. Robles, and S. Lucas, "Fast approximate Max-n Monte-Carlo tree search for Ms *Pac-Man*," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 2, pp. 142–154, Jun. 2011.

[14] P. R. Williams, D. Perez-Liebana, and S. M. Lucas, "*Ms. Pac-Man* versus Ghost Team CIG 2016 competition," in *Proc. IEEE Conf. Comput. Intell. Games*, 2016, pp. 1–8.

[15] A. Fitzgerald and C. Congdon, "RAMP: A rule-based agent for *Ms. Pac-Man*," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 2646–2653.

[16] H. Handa and M. Isozaki, "Evolutionary fuzzy systems for generating better *Ms. Pac-Man* players," in *Proc. IEEE Int. Conf. Fuzzy Syst. (IEEE World Congr. Comput. Intell.)*, 2008, pp. 2182–2185.

[17] N. Wirth and M. Gallagher, "An influence map model for playing *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, 2008, pp. 228–233.

[18] R. Thawonmas and H. Matsumoto, "Automatic controller of *Ms. Pac-Man* and its performance: Winner of the IEEE CEC 2009 software agent *Ms. Pac-Man* competition," in *Proc. CD-ROM Asia Simul. Conf.*, vol. 2009, 2009, pp. 1–4.

[19] L. DeLooze and W. Viner, "Fuzzy Q-learning in a nondeterministic environment: Developing an intelligent *Ms. Pac-Man* agent," in *Proc. IEEE Symp. Comput. Intell. Games*, 2009, pp. 162–169.

[20] D. Robles and S. Lucas, "A simple tree search method for playing *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, 2009, pp. 249–255.

[21] H. Handa, "Detection of critical situations by CMAC+ Q-learning for PacMan agents," in *Proc. Int. Conf. Netw., Sens. Control*, 2009, pp. 124–129.

[22] H. Handa, "Constitution of *Ms. Pac-Man* player with critical-situation learning mechanism," *Int. J. Knowl. Eng. Soft Data Paradigms*, vol. 2, no. 3, pp. 237–250, 2010.

[23] N. Bell, X. Fang, R. Hughes, G. Kendall, E. O'Reilly, and S. Qiu, "Ghost direction detection and other innovations for *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, 2010, pp. 465–472.

[24] R. Thawonmas and T. Ashida, "Evolution strategy for optimizing parameters in Ms *Pac-Man* controller ICE Pambush 3," in *Proc. IEEE Symp. Comput. Intell. Games*, 2010, pp. 235–240.

[25] N. Ikehata and T. Ito, "Monte Carlo tree search in *Ms. Pac-Man*," in *Proc. 15th Game Program. Workshop*, 2010, pp. 39–46.

[26] B. Tong and C. Sung, "A Monte-Carlo approach for ghost avoidance in the *Ms. Pac-Man* game," in *Proc. Int. IEEE Consum. Electron. Soc. Games Innov. Conf.*, 2010, pp. 1–8.

[27] E. Martin, M. Moises, R. Gustavo, and S. Yago, "Pac-mAnt: Optimization based on ant colonies applied to developing an agent for *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, 2010, pp. 458–464.

[28] K. Oh and S. Cho, "A hybrid method of Dijkstra algorithm and evolutionary neural network for optimal *Ms. Pac-Man* agent," in *Proc. 2nd World Congr. Nature Biol. Inspired Comput.*, 2010, pp. 239–243.

[29] N. Ikehata and T. Ito, "Monte-Carlo tree search in *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, 2011, pp. 39–46.

[30] B. K.-B. Tong, C. M. Ma, and C. W. Sung, "A Monte-Carlo approach for the endgame of *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, 2011, pp. 9–15.

[31] T. G. Tan, J. Teo, and P. Anthony, "Nature-inspired cognitive evolution to play *Ms. Pac-Man*," in *Proc. Int. Conf. Math. Comput. Biol.*, 2011, vol. 9, pp. 456–463.

[32] T. Tan, J. Teo, and P. Anthony, "Automating commercial video game development using computational intelligence," *Amer. J. Econ. Bus. Admin.*, vol. 3, no. 1, pp. 186–190, 2011.

[33] J.-Y. Dai, Y. Li, J.-F. Chen, and F. Zhang, "Evolutionary neural network for ghost in *Ms. Pac-Man*," in *Proc. Int. Conf. Mach. Learn. Cybern.*, 2011, vol. 2, pp. 732–736.

[34] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[35] H. van Seijen, M. Fatemi, R. Laroche, J. Romoff, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," *Advances Neural Inform. Process. Syst.*, pp. 5398–5408, 2017.

[36] M. Gallagher and M. Ledwich, "Evolving *Pac-Man* players: Can we learn from raw input?" in *Proc. IEEE Symp. Comput. Intell. Games*, 2007, pp. 282–287.

[37] N. Beume *et al.*, "To model or not to model: Controlling *Pac-Man* ghosts without incorporating global knowledge," in *Proc. IEEE Congr. Evol. Comput. (IEEE World Congr. Comput. Intell.)*, 2008, pp. 3464–3471.

[38] M. Wittkamp, L. Barone, and P. Hingston, "Using NEAT for continuous adaptation and teamwork formation in PacMan," in *Proc. IEEE Symp. Comput. Intell. Games*, 2008, pp. 234–242.

[39] D. Becroft, J. Bassett, A. Mejía, C. Rich, and C. L. Sidner, "AIPaint: A sketch-based behavior tree authoring tool," in *Proc. Artif. Intell. Interactive Digit. Entertainment*, 2011, pp. 2–7.

[40] S. Maycock and T. Thompson, "Enhancing touch-driven navigation using informed search in *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, 2013, pp. 1–2.

[41] N. Tziortziotis, K. Tziortziotis, and K. Blekas, "Play *Ms. Pac-Man* using an advanced reinforcement learning agent," in *Hellenic Conference on Artificial Intelligence*. New York, NY, USA: Springer-Verlag, 2014, pp. 71–83.

[42] N. Tziortziotis, "Machine learning for intelligent agents," Ph.D. dissertation, Univ. Ioannina, Greece, 2015.

[43] M. Miranda, A. A. Sánchez-Ruiz, and F. Peinado, "A neuroevolution approach to imitating human-like play in *Ms. Pac-Man* video game," in *Proc. 3rd Congreso de la Sociedad Española para las Ciencias del Videojuego*, 2016, pp. 113–124.

[44] P. Burrow and S. Lucas, "Evolution versus temporal difference learning for learning to play *Ms. Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, 2009, pp. 53–60.

[45] E. Galván-López, J. Swafford, M. O'Neill, and A. Brabazon, "Evolving a *Ms. Pac-Man* controller using grammatical evolution," in *Applications of Evolutionary Computation*. New York, NY, USA: Springer-Verlag, 2010, pp. 161–170.

[46] E. Galván-López *et al.*, "Comparing the performance of the evolvable π grammatical evolution genotype-phenotype map to grammatical evolution in the dynamic *Ms. Pac-Man* environment," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1–8.

[47] A. Alhejali and S. Lucas, "Evolving diverse *Ms. Pac-Man* playing agents using genetic programming," in *Proc. U.K. Workshop Comput. Intell.*, 2010, pp. 1–6.

[48] A. Alhejali and S. Lucas, "Using a training camp with genetic programming to evolve Ms *Pac-Man* agents," in *Proc. IEEE Conf. Comput. Intell. Games*, 2011, pp. 118–125.

[49] K. Q. Nguyen and R. Thawonmas, "Applying Monte-Carlo tree search to collaboratively controlling of a ghost team in Ms *Pac-Man*," in *Proc. IEEE Int. Games Innov. Conf.*, 2011, pp. 8–11.

[50] M. F. Brandstetter and S. Ahmadi, "Reactive control of *Ms. Pac-Man* using information retrieval based on genetic programming," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 250–256.

[51] T. Pepels and M. H. M. Winands, "Enhancements for Monte-Carlo tree search in Ms *Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 265–272.

[52] D. J. Gagne and C. B. Congdon, "Fright: A flexible rule-based intelligent ghost team for *Ms. Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 273–280.

[53] G. Recio, E. Martin, C. Estébanez, and Y. Saez, "AntBot: Ant colonies for video games," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 4, pp. 295–308, Dec. 2012.

[54] J. Svensson and S. J. Johansson, "Influence map-based controllers for *Ms. Pac-Man* and the ghosts," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 257–264.

[55] W. Sombat, P. Rohlfshagen, and S. M. Lucas, "Evaluating the enjoyability of the ghosts in Ms *Pac-Man*," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 379–387.

[56] K. Q. Nguyen and R. Thawonmas, "Monte Carlo tree search for collaboration control of ghosts in *Ms. Pac-Man*," *IEEE Trans. Comput. Intell. AI Games*, vol. 5, no. 1, pp. 57–68, Mar. 2013.

[57] A. B. Cardona, J. Togelius, and M. J. Nelson, "Competitive coevolution in *Ms. Pac-Man*," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 1403–1410.

[58] K. Tamura and T. Torii, "Development of ghost controller for Ms *Pac-Man* versus ghost team with grammatical evolution," *J. Adv. Comput. Intell. Intell. Informat.*, vol. 17, no. 6, pp. 904–912, 2013.

[59] A. M. Alhejali and S. M. Lucas, "Using genetic programming to evolve heuristics for a Monte Carlo tree search Ms *Pac-Man* agent," in *Proc. IEEE Conf. Comput. Intell. Games*, 2013, pp. 1–8.

[60] F. Liberatore, A. M. Mora, P. A. Castillo, and J. J. Merelo Guervós, "Evolving evil: Optimizing flocking strategies through genetic algorithms for the ghost team in the game of *Ms. Pac-Man*," in *European Conference on the Applications of Evolutionary Computation*. New York, NY, USA: Springer-Verlag, 2014, pp. 313–324.

[61] T. Pepels, M. H. M. Winands, and M. Lanctot, "Real-time Monte Carlo tree search in Ms *Pac-Man*," *IEEE Trans. Comput. Intell. AI Games*, vol. 6, no. 3, pp. 245–257, Sep. 2014.

[62] J. Schrum and R. Miikkulainen, "Evolving multimodal behavior with modular neural networks in *Ms. Pac-Man*," in *Proc. Annu. Conf. Genetic Evol. Comput.*, 2014, pp. 325–332.

[63] F. Liberatore, A. M. Mora, P. A. Castillo, and J. J. Merelo, "Comparing heterogeneous and homogeneous flocking strategies for the ghost team in the game of *Ms. Pac-Man*," *IEEE Trans. Comput. Intell. AI Games*, vol. 8, no. 3, pp. 278–287, Sep. 2016.

[64] J. Schrum and R. Miikkulainen, "Discovering multimodal behavior in *Ms. Pac-Man* through evolution of modular neural networks," *IEEE Trans. Comput. Intell. AI Games*, vol. 8, no. 1, pp. 67–81, Mar. 2016.

[65] P. McClarron, R. Ollington, and I. Lewis, "Effect of constraints on evolving behavior trees for game AI," in *Proc. Int. Conf. Comput. Games, Multimedia Allied Technol.*, 2016, pp. 1–6.

[66] W. Sombat, "Optimising agent behaviours and game parameters to meet designer's objectives," Ph.D. dissertation, Univ. Essex, Colchester, U.K. 2016.

[67] J. B. Schrum, "Evolving multimodal behavior through modular multiobjective neuroevolution," Ph.D. dissertation, Artif. Intell. Lab., Univ. Texas Austin, Austin, TX, USA, 2014.

[68] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[69] J. Rosca, "Generality versus size in genetic programming," in *Proceedings of the First Annual Conference on Genetic Programming*. Cambridge, MA, USA: MIT Press, 1996, pp. 381–387.

[70] J. Bonet and C. Stauffer, "Learning to play *Pac-Man* using incremental reinforcement learning," in *Proc. Congr. Evol. Comput.*, 1999, pp. 2462–2469.

[71] M. Gallagher and A. Ryan, "Learning to play *Pac-Man*: An evolutionary, rule-based approach," in *Proc. IEEE Symp. Comput. Intell. Games*, 2003, pp. 2462–2469.

[72] T. Thompson, L. McMillan, J. Levine, and A. Andrew, "An evaluation of the benefits of look-ahead in *Pac-Man*," in *Proc. IEEE Symp. Comput. Intell. Games*, 2008, pp. 310–315.

[73] I. Szita and A. Lorincz, "Learning to play using low-complexity rule-based policies: Illustrations through *Ms. Pac-Man*," *J. Artif. Intell. Res.*, vol. 30, no. 1, pp. 659–684, 2007.

[74] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2164–2172.

[75] B. Yuan, C. Li, and W. Chen, "Training a *Pac-Man* player with minimum domain knowledge and basic rationality," in *Advanced Intelligent Computing Theories and Applications: Proceedings of the 6th International Conference on Intelligent Computing*, vol. 93. New York, NY, USA: Springer-Verlag, 2010, p. 169.

[76] R. Baumgarten, "Towards automatic player behaviour characterisation using multiclass linear discriminant analysis," in *Proc. AISB Symp., AI Games*, 2010, pp. 1–4.

[77] M. M. Hasan and J. Z. Khondker, "Implementing artificially intelligent ghosts to play *Ms. Pac-Man* game by using neural network at social media platform," in *Proc. Int. Conf. Adv. Elect. Eng.*, 2013, pp. 353–358.

[78] L. Bom, R. Henken, and M. Wiering, "Reinforcement learning to train *Ms. Pac-Man* using higher-order action-relative inputs," in *Proc. IEEE Symp. Adapt. Dyn. Program. Reinforcement Learn.*, 2013, pp. 156–163.

[79] T. Anthony, D. Polani, and C. L. Nehaniv, "General self-motivation and strategy identification: Case studies based on Sokoban and *Pac-Man*," *IEEE Trans. Comput. Intell. AI Games*, vol. 6, no. 1, pp. 1–17, Mar. 2014.

[80] C. N. Silla, "Teaching genetic algorithm-based parameter optimization using Pacman," in *Proc. IEEE Frontiers Educ. Conf.*, 2016, pp. 1–6.

[81] T. Moerland, J. Broekens, and C. Jonker, "Fear and hope emerge from anticipation in model-based reinforcement learning," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 848–854.

[82] A. Vezhnevets *et al.*, "Strategic attentive writer for learning macro-actions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3486–3494.

[83] R. Krepki, B. Blankertz, G. Curio, and K.-R. Müller, "The Berlin brain-computer interface (BBCI)–Towards a new communication channel for online control in gaming applications," *Multimedia Tools Appl.*, vol. 33, no. 1, pp. 73–90, 2007.

[84] A. Girouard *et al.*, "Distinguishing difficulty levels with non-invasive brain activity measurements," in *IFIP Conference on Human–Computer Interaction*. New York, NY, USA: Springer-Verlag, 2009, pp. 440–452.

[85] B. Reuderink, A. Nijholt, and M. Poel, "Affective Pacman: A frustrating game for brain computer interface experiments," in *International Conference on Intelligent Technologies for Interactive Entertainment*. New York, NY, USA: Springer-Verlag, 2009, pp. 221–227.

[86] T. Schaul, "A video game description language for model-based or interactive learning," in *Proc. 2013 IEEE Conf. Comput. Intell. Games*, 2013, pp. 1–8.

[87] T. Schaul, "An extensible description language for video games," *IEEE Trans. Comput. Intell. AI Games*, vol. 6, no. 4, pp. 325–331, Dec. 2014.

[88] D. Perez-Liebana *et al.*, "The 2014 general video game playing competition," *IEEE Trans. Comput. Intell. AI Games*, vol. 8, no. 3, pp. 229–243, Sep. 2016.

[89] D. Perez-Liebana, S. Samothrakis, J. Togelius, S. M. Lucas, and T. Schaul, "General video game AI: Competition, challenges and opportunities," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 4335–4337.

[90] A. Costa, R. Machado, and L. Ribeiro, "Evolving negative application conditions," *CIP–CATALOGAÇÃO NA PUBLICAÇÃO*, vol. 1, pp. 73–82, 2016.

[91] F. Koriche, S. Lagrue, É. Piette, and S. Tabary, "General game playing with stochastic CSP," *Constraints*, vol. 21, no. 1, pp. 95–114, 2016.

[92] K. Subramanian, C. Isbell, and A. Thomaz, "Learning options through human interaction," in *Proc. IJCAI Workshop Agents Learn. Interactively Hum. Teachers*, 2011, pp. 1–7.

[93] G. Yannakakis and J. Hallam, "Evolving opponents for interesting interactive computer games," *Animals Animats*, vol. 8, pp. 499–508, 2004.

[94] G. Yannakakis and J. Hallam, "A generic approach for generating interesting interactive *Pac-Man* opponents," in *Proc. IEEE Symp. Comput. Intell. Games*, Colchester, U.K., 2005, pp. 94–101.

[95] G. N. Yannakakis and M. Maragoudakis, "Player modeling impact on player's entertainment in computer games," in *User Modeling 2005*. New York, NY, USA: Springer-Verlag, 2005, p. 151.

[96] G. Yannakakis and J. Hallam, "Towards optimizing entertainment in computer games," *Appl. Artif. Intell.*, vol. 21, no. 10, pp. 933–971, 2007.

[97] N. Beume *et al.*, "Measuring flow as concept for detecting game fun in the *Pac-Man* game," in *Proc. IEEE Congr. Evol. Comput. (IEEE World Congr. Comput. Intell.)*, 2008, pp. 3448–3455.

[98] B. U. Cowley, "Player profiling and modelling in computer and video games," Ph.D. dissertation, Univ. Ulster, Coleraine, U.K., 2009.

[99] B. Cowley, D. Charles, M. Black, and R. Hickey, "Analyzing player behavior in Pacman using feature-driven decision theoretic predictive modeling," in *Proc. IEEE Symp. Comput. Intell. Games*, 2009, pp. 170–177.

[100] T. Ohno and H. Ogasawara, "Information acquisition model of highly interactive tasks," in *Proc. ICCS/JCSS*, 1999, vol. 99, pp. 288–293.

[101] D. Mobbs *et al.*, "When fear is near: Threat imminence elicits prefrontal-periaqueductal gray shifts in humans," *Science*, vol. 317, no. 5841, pp. 1079–1083, 2007.

[102] B. Dickinson, O. Jenkins, M. Moseley, D. Bloom, and D. Hartmann, "Roomba *Pac-Man*: Teaching autonomous robotics through embodied gaming," in *Proc. AAAI Spring Symp. Robots Robot Venues, Resources AI Educ.*, 2007, pp. 35–39.

[103] M. Rao, "An implementation of Pacman game using robots," *Indian J. Comput. Sci. Eng.*, vol. 2, no. 6, pp. 802–812, 2011.

[104] A. Cheok, S. Fong, K. Goh, X. Yang, W. Liu, and F. Farzbiz, "Human Pacman: A sensing-based mobile entertainment system with ubiquitous computing and tangible interaction," in *Proc. 2nd Workshop Netw. Syst. Support Games*, 2003, pp. 106–117.

[105] A. Cheok *et al.*, "Human Pacman: A mobile, wide-area entertainment system based on physical, social, and ubiquitous computing," *Pers. Ubiquitous Comput.*, vol. 8, no. 2, pp. 71–81, 2004.

[106] W. Van Eck and M. Lamers, "Animal controlled computer games: Playing *Pac-Man* against real crickets," in *Entertainment Computing–ICEC 2006*. New York, NY, USA: Springer-Verlag, 2006, pp. 31–36.

[107] J. DeNero and D. Klein, “Teaching introductory artificial intelligence with *Pac-Man*,” in *Proc. Symp. Educ. Adv. Artif. Intell.*, 2010, pp. 1885–1889.

[108] P. Øgland, “Using game theory for producing quality plans: A *Pac-Man* simulation experiment,” in *Proc. Syst. Res., Lessons Past - Progress Future*, 2009, pp. 1–20.

[109] G. Viglietta, “Gaming is a hard job, but someone has to do it!” in *Proceedings of the Sixth International Conference on Fun With Algorithms*. New York, NY, USA: Springer-Verlag, 2012.

[110] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.

[111] S. M. Lucas, “Temporal difference learning with interpolated table value functions,” in *Proc. IEEE Symp. Comput. Intell. Games*, 2009, pp. 32–37.

[112] K. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.

[113] J. Ortega, N. Shaker, J. Togelius, and G. N. Yannakakis, “Imitating human playing styles in Super Mario Bros,” *Entertainment Comput.*, vol. 4, no. 2, pp. 93–104, 2013.

[114] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, “Mason: A multiagent simulation environment,” *Simulation*, vol. 81, no. 7, pp. 517–527, 2005.

[115] A. A. Abdullahi and S. M. Lucas, “Temporal difference learning with interpolated n-tuples: Initial results from a simulated car racing environment,” in *Proc. IEEE Conf. Comput. Intell. Games*, 2011, pp. 321–328.

[116] R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” *Artif. Intell.*, vol. 112, no. 1–2, pp. 181–211, 1999.

[117] J. Levine *et al.*, “General video game playing,” *Artif. Comput. Intell. Games*, Dagstuhl Follow-Ups, vol. 6, pp. 77–83, 2013.

[118] M. Ebner, J. Levine, S. M. Lucas, T. Schaul, T. Thompson, and J. Togelius, “Towards a video game description language,” *Artif. Comput. Intell. Games*, Dagstuhl Follow-Ups, vol. 6, pp. 85–100, 2013.

[119] T. Haynes, S. Sen, D. Schoenefeld, and R. Wainwright, “Evolving a team,” in *Proc. Working Notes AAAI Symp. Genetic Program.*, 1995, pp. 23–30.

[120] T. Haynes and S. Sen, “Evolving behavioral strategies in predators and prey,” in *Adaption and Learning in Multi-Agent Systems*. New York, NY, USA: Springer-Verlag, 1996, pp. 113–126.

[121] S. Luke and L. Spector, “Evolving teamwork and coordination with genetic programming,” in *Proceedings of the First Annual Conference on Genetic Programming*. Cambridge, MA, USA: MIT Press, 1996, pp. 150–156.

[122] A. Rawal, P. Rajagopalan, and R. Miikkulainen, “Constructing competitive and cooperative agent behavior using coevolution,” in *IEEE Symposium on Computational Intelligence and Games*. Piscataway, NJ, USA: IEEE Press, 2010, pp. 107–114.

[123] P. Rajagopalan, A. Rawal, and R. Miikkulainen, “Emergence of competitive and cooperative behavior using coevolution,” in *Proc. 12th Annu. Conf. Genetic Evol. Comput.*, 2010, pp. 1073–1074.

[124] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*. New York, NY, USA: Harper Perennial, 1990.

[125] S. Rabin, *AI Game Programming Wisdom*. Boston, MA, USA: Cengage Learn., 2002.

[126] A. Wade, “Dots, fruit, speed and pills: The happy consciousness of *Pac-Man*,” *J. Cultural Res.*, vol. 19, no. 3, pp. 248–261, 2015.

[127] K. Squire, “Video games in education,” *Int. J. Intell. Games Simul.*, vol. 2, no. 1, pp. 49–62, 2003.

[128] R. Bowman, “A *Pac-Man* theory of motivation: Tactical implications for classroom instruction,” *Educ. Technol.*, vol. 22, no. 9, pp. 14–17, 1982.

[129] A. Yacoub, M. Hamri, and C. Frydman, “Using DEv-PROMELA for modelling and verification of software,” in *Proc. 2016 Annu. ACM Conf. SIGSIM Principles Adv. Discr. Simul.*, 2016, pp. 245–253.

Philipp Rohlfshagen received the Ph.D. degree in computer science from the University of Birmingham, Birmingham, U.K., in 2007.

He is a cofounder and the Director of Daitum, Adelaide, S.A., Australia, an Australian-based AI company specializing in decision analytics. His research interests include optimization, machine learning, and games, with more than 30 peer-reviewed publications. His most recent activities focus on solving complex real-world optimization problems and making optimization technology widely accessible.

Jialin Liu (M’16) received the B.Sc. degree in optoelectronics from Huazhong University of Science and Technology, Wuhan, China, in 2010, the M.Sc. degree in bioinformatics and biostatistics from the Université Paris-Sud, Orsay, France, and the École Polytechnique, Palaiseau, France, in 2013, and the Ph.D. degree in computer science from the Université Paris-Saclay, Paris, France, in 2016.

She is currently a Postdoctoral Research Associate at Queen Mary University of London, London, U.K. Her research interests include reinforcement learning, black-box noisy optimization, portfolio algorithms, and artificial intelligence in games.

Diego Perez-Liebana (M’12) received the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 2015.

He is currently a Lecturer in Computer Games and Artificial Intelligence at the University of Essex. He has authored or coauthored in the domain of game AI, with interests in reinforcement learning and evolutionary computation. He organized several game AI competitions, such as the physical traveling salesman problem and the general video game AI competitions, held in IEEE conferences. He has programming experience in the video games industry with titles published for game consoles and PC.

Simon M. Lucas (SM’98) received the Ph.D. degree in electronics and computer science from the University of Southampton, Southampton, U.K., in 1991.

He is currently a Professor in Artificial Intelligence and Head of the School of Electronic Engineering and Computer Science, Queen Mary University of London, London, U.K. His research interests include games, evolutionary computation, and machine learning, and has authored or coauthored widely in these fields with more than 180 peer-reviewed papers. He is the inventor of the scanning n-tuple classifier.

Dr. Lucas is the Founding Editor-in-Chief of the IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES.