A Survey of Learning Classifier Systems in Games

Kamran Shafi and Hussein A. Abbass

School of Engineering & Information Technology, University of New South Wales, Canberra, AUSTRALIA.

Abstract

Games are becoming increasingly indispensable, not only for fun but also to support tasks that are more serious, such as education, strategic planning, and understanding of complex phenomena. Computational intelligence-based methods are contributing significantly to this development. Learning Classifier Systems (LCS) is a pioneering computational intelligence approach that combines machine learning methods with evolutionary computation, to learn problem solutions in the form of interpretable rules. These systems offer several advantages for game applications, including a powerful and flexible agent architecture built on a knowledge-based symbolic modeling engine; modeling flexibility that allows integrating domain knowledge and different machine learning mechanisms under a single computational framework; an ability to adapt to diverse game requirements; and an ability to learn and generate creative agent behaviors in real-time dynamic environments. We present a comprehensive and dedicated survey of LCS in computer games. The survey highlights the versatility and advantages of these systems by reviewing their application in a variety of games. The survey is organized according to a general game classification and provides an opportunity to bring this important research direction into the public eye. We discuss the strengths and weaknesses of the existing approaches and provide insights into important future research directions.

I. Introduction

The use of computer games is widespread in human life. Computer games play diverse roles in shaping human life and knowledge, from being used for fun and entertainment [1] to being applied to strategic planning and decision support [2], [3], health [4], defense [5], education and

Corresponding Author: Kamran Shafi (k.shafi@unsw.edu.au; kamran.shafi@gmail.com)

learning [6], and analysis of complex phenomena [7]. In terms of revenue, the computer games industry is now one of the most lucrative businesses in the world with annual revenues in tens of billions of dollars [8]. This huge financial incentive is further fueling research and development in this domain.

Computational Intelligence (CI) based [9] approaches provide a powerful set of methods and tools for modeling artificial game-playing agents and making games more dynamic and interesting [10], [11]. The use of CI techniques in games has grown steadily over the years because of advancements in related research areas, such as evolutionary computation [12] and deep neural networks [13]. A useful review of CI techniques in games can be found in [14], [15].

Learning Classifier Systems (LCS) [16] are one of the earliest CI approaches. LCS bring important opportunities to games. First, LCS are rule-based, enabling easier integration with the classic knowledge bases that are used in games. Second, they are equipped with learning mechanisms that enable them to adapt the rules "on the fly." Third, recent advances in LCS introduce architectures and algorithms that make them a technique that is fast and suitable for real-time environments.

LCS were introduced as biologically inspired computational models for human-like cognition. Typical LCS implementations consist of a knowledge base or memory component that is encoded using some form of a symbolic representation, and a learning component that uses a combination of evolutionary and machine learning algorithms to adapt the knowledge base in given environmental conditions. Since their inception several decades ago, LCS have been applied to diverse problems in numerous application domains, including computer games. Several LCS surveys have been written over the years, with the more recent ones [17]–[20] focusing mainly on either algorithmic variations and advances in LCS or their applications. In addition, a small number of non-dedicated reviews (e.g., [21], [22]) have indirectly covered LCS and their applications under broader topics.

II. Motivation and Survey Strategy

The lack of a comprehensive survey covering LCS-based approaches in games is an information gap that this paper attempts to address. By synthesizing the existing LCS literature in computer games published over the last two decades, we offer insights into future research opportunities for both the computer games industry and LCS communities.

The search for relevant publications in this review was conducted through well-known databases such as Scopus, Google Scholar, IEEE Xplore and ACM Digital Library, using a combination of keywords. Some keywords included "game(s)," variants of "classifier systems," and specific system names such as XCS and UCS. This gave rise to several hundred matching results. The two filters applied in the search excluded (a) papers that focused on the use of other AI/CI approaches in computer games and merely listed LCS as a reference approach; and (b) LCS papers that did not have a focus on games but cited LCS work on games as a reference. In addition, this search excluded the general class of LCS-oriented agent-based modeling approaches. Examples of such work include LCS applications to classical benchmark multi-step problems (e.g., maze, woods, etc.) [23], in which LCS-based agents engage in typical reinforcement learning (RL) tasks, as well as agent applications in practical domains (e.g., economics [24], marketing [25], air traffic control [26], robotics [27], aircraft simulation [28], and multi-agent simulations [29]). While this line of research is important, such applications are not classified as "games" in the context of this paper. Additionally, these approaches have been covered in other reviews cited above, and are well known in the LCS literature. In contrast, LCS research in computer games is relatively unexplored, and LCS are relatively unknown in the community, and therefore relatively unused.

The rest of the paper is organized as follows: Section III introduces LCS and the important variants. Section IV provides a discussion of games and their different types, specifically those related to the reviewed approaches in this paper. Section V provides a comprehensive survey of LCS applications in different games. This survey is organized according to four categories: video games, combinatorial games, simulation games, and game theory. Section VI provides a summary of the review and highlights important points. Section VII lists the future research directions and insights based on this review and the authors' knowledge about emerging areas of research. The paper ends with a brief conclusion in Section VIII.

III. Learning Classifier Systems

John Holland introduced the concept of classifier systems as a way to design adaptive learning capabilities in artificial systems [30]–[32]. Holland was inspired by learning in natural cognitive systems. He attributed learning in these systems with the ability to build a representation or model of their environment, generalize from experience through exploiting environmental regularities, and continuously expand and improve their internal models and knowledge. With such philosophical underpinnings, he combined his ground-breaking work in genetic algorithms

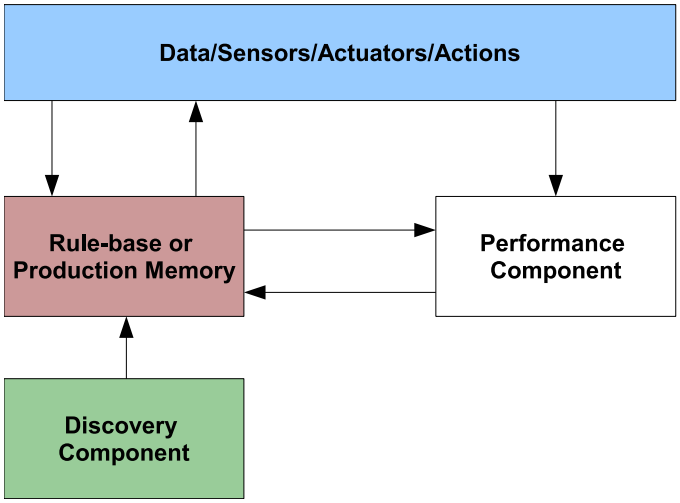

Fig. 1. Basic components of a learning classifier system. The production component interfaces with the environment, receives and processes input to the system and generates output using the current set of rules (knowledge-base or model) evolved by the system; the performance component deals with measuring the consistency of the evolved model based on the quality of the proposed actions in a task and the corresponding reward received from the environment; and the discovery component is tasked with generalization of the model (finding an accurate but minimal representation) through the application of evolutionary algorithms.

(GAs) [30] with a rule-based knowledge representation and RL mechanisms to set the foundations of LCS. Some useful introductory resources on LCS are available (e.g., [16], [33]). A brief introduction is provided in this section of this paper, specifically focusing on those components and the learning paradigms that are pertinent to games.

LCS are seen as a class of rule-based systems [34]. In Holland's words [32]: "Classifier systems are a kind of rule-based system with general mechanisms for processing rules in parallel, for adaptive generation of new rules, and for testing the effectiveness of existing rules." This definition highlights three basic functions that form the basis of most LCS implementations. Figure 1 depicts an abstract representation of these basic components and their interaction in a typical LCS implementation.

LCS share several features with RL systems. First, as with many RL systems, LCS were also originally applied to sequential learning problems, in which an agent interacts with its environment and learns through trial and error and through receiving feedback from the environment. Second, the rule evaluation in LCS borrows heavily from the RL literature. For instance, Holland used *bucket-brigade* – an epochal RL algorithm – to measure the goodness of classifiers in the early implementations of classifier systems [35]. Later incarnations of LCS leveraged the

contemporary developments in RL research and borrowed techniques that were more advanced, such as temporal-difference learning [36] and Q-learning [37], to evaluate rule quality. The goal of RL algorithms, in general, is to find an optimal policy and its valuation to guide an agent's behavior in an environment, based on the feedback or a reward signal received from the environment. Often, neural networks are used to approximate value functions when an exhaustive policy search is not practical, as in the case of high-dimensional or continuous problems. From that perspective, LCS provide an alternative RL mechanism that represents both the policy in an interpretable rule format as well as a rule-based approximation to value functions. Finally, the LCS framework allows for learning the policy in an online fashion, making it especially suitable for real-time applications and many computer games. See [38], [39] for a detailed comparison of the two approaches.

Traditionally, LCS models have been classified into two broad categories: the systems that were developed following Holland's work, known as Michigan-style LCS; and the approach popularized by De Jong and Smith [40], [41] known as Pittsburgh-style LCS. A major difference between the two approaches emerged based on their solution representation. In Michigan-style LCS, a *classifier* consists of a single condition-action-rule and a set of parameters that keep track of the rule's quality and other statistics. Subsequently, a classifier in these systems contributes partially to the entire solution or the knowledge base. A complete solution is represented by the set of all classifiers or the entire rule population. Michigan-style LCS are incremental learners; that is, they build their models online during interaction with the environment. Subsequently, the rule discovery algorithm, generally a GA, works in a non-generational fashion and often in the niches defined by the matching rule conditions to an environmental state or input.

A major distinction between different types of Michigan-style LCS is made based on how the fitness of classifiers is calculated in the system. The *strength*-based LCS relate a classifier's fitness directly to the accumulated payoff or reward it receives from the environment for its advocated action. In contrast, the *accuracy*-based LCS compute a classifier's fitness based on the accuracy of reward prediction, thereby allowing the system to evolve a complete action map of the problem. This subtle difference in fitness calculation has shown to have a significant effect on system performance and the evolutionary dynamics [42]. Holland's original implementation [35], known as CS-1, was based on this latter approach. However, in his revised implementation [43], which became known as the payoff-based LCS, he replaced the accuracy-based fitness with the strength-based fitness among other changes. Several LCS variants were later introduced,

building on Holland's work (see [20] for a recent historical review). However, after the initial appreciation of Holland's work, LCS largely remained disregarded until Stewart Wilson's work in 1990s revived the field.

Wilson, leveraging the advancements in the RL literature, introduced two different variants of LCS: the infamous ZCS [23] and the XCS [44]. XCS, in particular, incorporated a purely accuracy-based fitness computation and a Q-learning-based fitness update mechanism, among other features. These changes helped overcome some key performance bottlenecks in the payoff-based LCS, such as the proliferation of over-general classifiers in the population. XCS rejuvenated the field of LCS research and became the most popular and successful LCS. This current survey corroborated this observation and identified XCS as the most widely adopted LCS in the games literature, followed by Holland's LCS, ZCS, and a small number of uses of Pittsburgh-style LCS. Below, we provide a brief explanation of XCS, focusing on its main algorithmic components and its working, using a simple game example.

Figure 2 depicts a working cycle of an XCS. In a *Tic-tac-toe* game, for instance, an input to the system might consist of the current board configuration. A ternary vector of length nine could be used to model this environment. Here, the vector length corresponds to the number of boxes or positions on the board, which could be either empty or containing one of the two relevant symbols (i.e., a naught or a cross). A ternary set (e.g., $0, 1, 2$) could be used to map the three possible values for each position on the board. Subsequently, a quaternary vector of length nine could be used to represent the condition part in a classifier, where the extra literal allows generalization using a don't-care symbol (e.g., #). The action part of the classifier could be represented using a single integer, specifying the next move ($[1 - 9]$). That is, for placing a naught or a cross (depending upon which symbol is allocated to either competitor) on one of the available board slots, the population of classifiers ($[P]$) could be initialized randomly to a fixed size or incrementally, using the covering operator. Upon receiving an input, a matchset ($[M]$) is formed by all classifiers in the population that match the current input. A matching function can be defined based on an exact match or using some distance criterion [45]. The covering operator is activated if no matching classifiers are found in the existing population.

The covering creates one or more classifiers matching the current input, with a given probability of introducing don't-cares in the condition. Next, an action to be executed is selected from all advocated actions by the classifiers in $[M]$. A commonly used scheme is to alternate between a random action selection, encouraging exploration, and action selection based on the

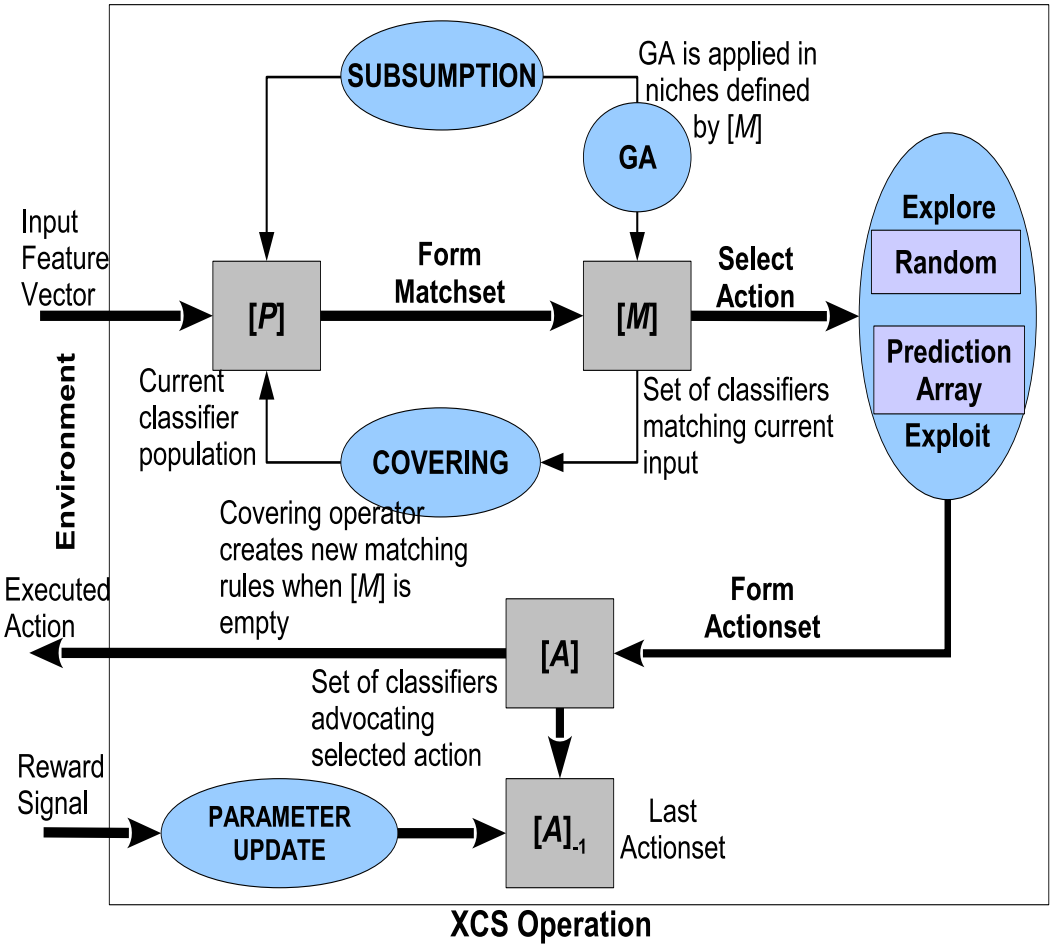

Fig. 2. Working of XCS in reinforcement learning tasks. The gray-shaded boxes represent different classifier sets used in the system and the blue-shaded ellipsoids represent different processes that take place in the system.

prediction strength of the matching classifiers, to exploit current knowledge. Finally, the fitness of all rules in the $[M]$ is updated based on the reward (e.g., 1000 for a win, -1000 for losing a game, and 0 otherwise) received from the environment for the selected action. A GA cycle is applied on the classifier population in either $[M]$ or $[A]$ based on the average experience of classifiers in either set.

In Pittsburgh-style LCS, a complete rule set constitutes a classifier. Hence, the system works on a population of variable-length rule sets using a generational GA, in which each rule set is considered a potential solution to the problem. The rule evaluation is performed in a batch-mode over the entire data set (or a sufficiently large number of examples), and the genetic operators are applied at the rule set level at the completion of a rule evaluation cycle. In other words, the GA in

these systems works more traditionally as an optimization heuristic to evolve the best solution in the form of a complete rule set. Since in this set-up the solutions require longer evaluation trials to measure their quality, these systems are more suitable for offline tasks and often are applied to concept learning problems using a supervised approach. Incremental variants of these systems which are potentially better candidates for use in computer games, have appeared recently [46]. However, this area is largely unexplored.

IV. Types of Games

Broadly, the LCS approaches included in this survey can be categorized as those related strictly to computer games and to game theory. Computer games such as video and board games are modeled and played in-silico. Game-theoretic studies are generally not categorized as computer games because of their mostly non-recreational applications. However, the relationship between game theory and computer games [47], [48] have been acknowledged in the literature. Moreover, as the use of computer games expands beyond recreation (e.g., in education [49] and management [2], [3] sectors), game-theoretic models could play an even greater role in game design. For these reasons, we included LCS-based game-theoretic studies in this review. Overall, this review has been organized into four types or categories of games, which are discussed below.

A. Video Games

A variety of video game genres exist [50]. This section provides an overview of the famous genres, with a focus on the types relevant to the LCS work in games. Among these, action games are one of the earliest and well-known types of video games, with a human player generally controlling and maneuvering a character in a given environment. Some of the famous action game sub-genres include arcade games, shooter games and platform games. In fact, many of the video game genres can be considered the extensions on or variations of action games.

Role-playing games (RPGs) are another popular type of video game. These games are generally characterized by a strong storyline and a set of formal game playing rules. One or more players take on a role or a character in a given story and perform actions that match the story line. Conversely, the story might unfold according to the players' actions. Players are usually characterized by personality traits that define their capabilities and action preferences. The massively multi-player online role-playing games (MMORPGs) are played among a large number of players on social media or on the internet. Non-player-characters (NPCs) are introduced in

these games to make the finctional settings more realistic. The CI-based techniques, including LCS, can be applied to model and control both the player and non-player characters in these games. *Fallout* [51], with its different versions, is an example of a famous RPG.

Real-time strategy (RTS) games are also a popular class of video games. In RTS games, players' actions are usually related to resource planning in a continuously changing environment. In these games, one or more players control a set of semi-autonomous entities to perform a real-time task, such as winning a military battle or a race. *Star Craft*, *War Craft* and *Supreme Commander* are examples of famous RTS games. The modeling of artificial agents for playing RTS games is quite challenging because of the diversity of multi-tasking that is inherent in these games.

B. Combinatorial Games

Combinatorial games (aka board games) usually involve two players who compete in sequential moves with perfect information about each game state. *Chess* and *Go* are famous examples of such games, which have been the subject of rigorous research in AI for a long time. AI approaches usually focus on modeling agents or algorithms that can compete with human experts in these games. A brute-force strategy used is to explore the space of all state-action possibilities and choose the best course of action to achieve a goal, such as be the first to complete a task. While for some simpler games (e.g., *Nim* and *Tic-tac-toe*), such a strategy works well, for more complex games (e.g., *Chess* and *Go*) an exhaustive search is impractical. Approximate techniques or other AI/CI techniques are commonly employed in these situations to model intelligent agents that could perform at the level of their human competitors.

C. Simulation Games

Simulation games attempt to model some aspects of real-world situations, such as stock market trading or traffic control. These games have numerous uses in areas such as education, training, policy, and system analysis and design. While modeling and simulation is an established field for dealing with issues in these fields, games provide an engaging environment for the users to achieve similar objectives. For example, simulation games are used to train surgeons for complex surgeries [52]. Artificial agents can play various roles in these games. For instance, they may be used to co-learn their strategies from the human players [53] or to evolve challenging scenarios for the trainees [54].

D. Game Theory

Game theory is a well-established mathematical approach to study the competition between rational decision-making agents with wide-ranging applications in economics and, political, biological and computer sciences. CI-based approaches extend the game-theoretic models in simulated environments through developing agent models that aim to mimic purely rational and human-like behaviors [55]. In a typical two-player game (e.g., the Prisoner's Dilemma (PD) game), two strategic players make discrete choices independently (e.g., cooperate or defect). These games can be extended easily to become multi-player or iterative games, in which players consider the outcomes of their previous moves in deciding their actions at each iteration. Iterated Prisoner's Dilemma (IPD) is a well-known example of this type of game model.

Evolutionary game theory (EGT) [56] is a more recent branch of game theory that extends game-theoretic models to study the population-based evolutionary dynamics. The GA-based approaches have been well studied in EGT, as a result of Axelrod's seminal work [57]. Standard GA-based agents are conventionally used in the EGT models to study the evolution of strategies under varying game conditions. LCS-based strategic agents provide distinct advantages over traditional GA-based agent encoding, such as their ability to generalize across the strategy space and their ability to encode larger memory sizes. It is rather surprising that, despite this potential, only a handful of researchers have explored the use of LCS (reviewed in the next section) for modeling the game-theoretic agents.

V. LCS IN GAMES

This section presents a comprehensive survey of LCS applications under the four categories discussed in the previous section: video games, combinatorial games, simulation games, and game theory. The papers in each category (and sub categories) are reviewed in a chronological order. As discussed in a later section, the first major piece of work reported in the literature on the explicit use of LCS for games was Seredyski et al. [58]. Several game-like environments, such as the infamous "*maze*" or "*woods*" problems [23], were introduced as evaluation benchmarks in the early LCS research. Smith also tested his Pittsburgh-style LCS (LS-1) in a poker-betting domain [59]. However, this survey does not cover those studies, which focused on testing LCS capabilities in typical multi-step RL tasks.

A. Video Games

A small number of researchers have focused on modeling NPCs using LCS. Robert et al. [60]–[62] introduced *MHiCS*, a Modular and Hierarchical Classifier System, for modeling NPCs in MMORPGs. MHiCS is a multi-layer architecture in which each layer consists of multiple LCS. The first-level LCS aim at learning multiple high-level behaviors (e.g., attack, track and eat), based on different motivation values that are computed as a function of current environmental conditions and the given NPC personalities' traits. The second-level LCS are applied to produce a specific action based on the given motivation values and fitness of the first-level LCS. The last two levels of LCS are used to produce final actions, by diffusing outputs from the activated LCS in the previous levels, and to manipulate effectors or resources related to these actions. The overall architecture for MHiCS is quite involved and this precludes conducting an in-depth analysis of its utility [60]. The authors contend that LCS are better methods for modeling NPCs, as they fulfill four key agent design requirements: a) reactivity through event-action rules; b) proactivity through encoding flexibility; c) autonomy through evolutionary learning; and d) reconfigurability because of their interpretable knowledge base. Sanchez et al. [63] used an XCS within a behavioral animation framework, called Virtual Behaviors (ViBes), which was used to model and simulate NPCs. ViBes consists of four main components, including a decision-making module that is responsible for selecting given behaviors (e.g., eat, sleep or walk), based on a perceived input. An XCS was used in this module to learn rules adaptively for selecting different behaviors. The resulting system was tested in a game called *V-Man*, to model the agent behavior in a virtual kitchen environment. The authors observed that LCS-based agent models are able to provide reactive behaviors at the same time as planning a long sequence of actions needed for the virtual characters to operate in situated environments. Recently, Kop et al. [64] proposed the *Evolutionary Dynamic Scripting* (EDS) method to model NPCs in serious games. EDS borrows ideas from LCS, using a rule-based framework that is similar to that used in LCS but differs in two algorithmic components. First, instead of using a GA, EDS uses a tree representation for rules and genetic programming to evolve the rules. Second, EDS replaces the Q-learning-based RL component with a technique called *Dynamic Scripting* (DS). Unlike other RL methods that aim at learning the state-action mapping or a policy during interaction with the environment, DS works on the pre-defined rules (or a policy) and only adjusts the rule weights based on their applicability to different states and corresponding performance. An air combat simulation was

used to evaluate EDS and showed that EDS can produce slightly improved NPC behavior and can discover useful novel rules. The authors note that the evolutionary rule-based RL systems have an edge over traditional techniques because of their ability to produce continuous novel behavior in such scenarios and to allow the integration of external domain knowledge.

Several studies have explored the use of LCS for designing soft-soccer playing agents. Among these, Bonarini and Trianni's [65] work was the first application of fuzzy-LCS to simulated soccer games, specifically to *RoboCup*. The fuzzy-LCS are similar in effect to XCS, but differ by using fuzzy logic to encode rules. The experimental results showed a significantly better passing performance when communication between agents was enabled. These results provided evidence for a successful co-evolution of cooperation using LCS-based agents. Castillo et al. [66] used an XCS to model the players in RoboCup. The XCS-based players were built on top of an existing static rule-based agent architecture (11Monkeys [67]). Experiments with different settings were conducted. However, the best results were obtained when the agents used a mixed strategy in which the XCS population was always initialized with scripted rules taken from 11Monkeys. These rules were kept unchanged during the evolution, but additional rules were learned to complement this fixed strategy. Sato, along with different co-authors, later published a series of papers on an event-driven hybrid LCS approach for modeling online soccer playing agents [68]–[72]. The main idea behind this system is to employ LCS as a meta-learner or a hyper-heuristic algorithm that learn rules to select appropriate action-selection algorithms or strategies in an online soccer game. That is, different soccer-playing algorithms and strategies may perform differently in different game scenarios; the goal of meta-learning LCS is to map the most effective algorithms to the corresponding game scenarios. RoboCup, being a real-time and dynamic learning environment and requiring coordination between a team of agents, introduces interesting challenges for the agents' design. The above-mentioned studies have demonstrated the modeling power and flexibility that LCS provide in dealing with such an environment. Bonarini's work showed the integration of a fuzzy knowledge base with RL using the LCS framework. Castillo's work showed how LCS agents could be seeded with pre-scripted rules and evolved thereafter. Sato's work showed how LCS could be used as an agent as well as a meta-agent. These studies also demonstrated the ability of LCS to allow incorporating communication between agents and to evolve their knowledge cooperatively.

Among applications of LCS to other action and RTS games, Falke and Ross [73] used ZCS to model an adaptive agent in a war game in which models engaged in a battle between two armies.

ZCS was used to learn an adaptive strategy to control an army squad against human-controlled army squads. To avoid slow convergence, the system was provided with an immediate reward signal and was made to evolve classifiers at an abstract level. Learning in this more generalized or simpler search space helped ZCS to evolve increasingly complex controllers when competing against humans. This work showed that lightweight LCS models encoded with simpler conditions and abstracted actions could be more effective for RTS games than complex LCS models. Such models also provided better transparency and portability than previous attempts in that literature. Lujan et al. [74], [75] applied the standard XCS to model agents in *Wargus*. The system was tested with both immediate and delayed reward functions and it was found that the immediate reward function worked much better than the delayed version. The system was tested with a random player on four types of scenarios relating to different skills, such as training archers. The results showed that XCS performed well in terms of reward prediction and learning sensible rules. Lujan's work further emphasized the importance of appropriate design of reward functions for improving LCS performance in RTS games. Small and Congdon [76] applied a Pittsburgh-style LCS approach to model agents in an RTS game called *Unreal Tournament 2004*. The real-time nature of the learning environment did not allow the system to train on any winning scenario, and hence receive any positive reinforcement, leading to learning difficulties. To bypass this problem, the initial population was seeded by injecting several high-level handcrafted rules. Significant improvement in learning performance was reported with this strategy. This work has been perhaps the only study that tested Pittsburgh-style LCS in a real-time game environment. Computational time challenges were expected, owing to the batch-processing nature of this type of LCS. It would be interesting to compare XCS performance on the same game with the LCS used in this study. Nidorf et al. [77] used XCS to model agents for *RoboCode*, in which one or more (military) tanks engaged in a mutual battle. Agents needed to learn three kinds of strategies: scanning for enemies; targeting and firing at them; and maneuvering to create a strategic advantage. Apparently, a standard XCS implementation was used to encode all strategies singularly; that is, in the same population. The performance of XCS was compared with a neuro-evolutionary method (NEAT), and a better XCS performance was observed in some scenarios without a clear overall winner. The authors noted that a continuous, skewed payoff, multi-step environment proved challenging for XCS, especially with increasing difficulty in test scenarios. This has been perhaps the only study that compared the performance of LCS with another CI method in the same game environment. The authors noted that both algorithms were good at

learning competent strategies. Further, while XCS struggled with an increased number of actions, NEAT was found to be prone to overfitting. In our view, the former problem is much easier to handle and is more related to the environment representational issues than the overfitting problem, which is a more serious problem that may require algorithmic changes to address it. Tsapanos et al. [78] used ZCS to model agents in a military-flavored RTS game called *ORTS* [79], in which agents engaged in different tasks to build, defend, and attack military bases. The performance of ZCS was compared with a randomly acting agent as well as with an agent that used a standard RL algorithm (SARSA). The experimental results showed a better ZCS performance over other methods. This was one of the few studies that compared an LCS with a standard RL algorithm in a game environment and showed promising results. Clementis [80] applied XCS to a simplified "battleship" game. The authors tested two variations of the action selection mechanism. The first method used a probabilistic action selection mechanism, in which the action probabilities were determined from the statistics computed during the game play (e.g., by computing the number of hits over all hits for a specific action). The second method fed the action statistics to a neural network and used its output to predict actions indirectly. The results showed a significantly faster and better performance when using neural network-supported actions selection, highlighting the merits of hybridizing LCS-based game agents with other machine learning techniques.

Other game applications include Irvan et al.'s work [81], which studied LCS-based agents in the game of *Pac-Man*. A multi-agent LCS architecture was proposed, in which multiple XCS-based agents successfully coordinated with each other using a shared memory concept in the Pac-Man game.

The above studies have shown the application of LCS in different roles in several categories of video games, including the modeling of NPCs; modeling of agents that compete with human players and other intelligent or scripted agents; modeling of multi-agent teams; and in learning adaptive game-playing strategies in real-time environments. Despite the diversity of the above-mentioned successful LCS applications, the utility of LCS in modern and more complicated games is not evident from the current literature. Given the LCS potential, as demonstrated by these studies, this seems to be a missed research opportunity.

B. Combinatorial Games

A small number of studies exist in this category. Browne et al. [82], [83] explored the performance of XCS in the *Connect4* game. Connect4 is a turn-based board game that has a

goal of placing the four counters of the same color or shape consecutively in any direction. The goal for an agent is to learn a winning strategy. An abstraction algorithm was introduced by the authors to overcome the scalability issues posed by the large, multi-step search space imposed by this game. The algorithm aimed at combining high fitness rules learned by an XCS and constructing rules with higher generalization. While this algorithm improved XCS performance (particularly, the number of wins), it also slowed the system. This work provided a solution to the scaling problem without losing the model transparency. The abstraction technique also aligned well with Holland's original vision of default hierarchies in LCS [84]. Other mechanisms that could can provide faster abstraction solutions in LCS have been proposed in the literature [85].

Sood et al. [86] attempted to test the ability of LCS to learn strategies in a complex supply-chain environment in the aviation industry. As their test bed for this purpose, they used three game environments, including Nim, IPD and matrix choice games [87]. An XCS was used to model agents playing these games and competing with other scripted agents. The experiments showed that XCS was able to learn better strategies than the other agents could. The authors listed model interpretability, effective generalization, and the flexibility to allow application be both the Markovian and non-Markovian environments as key reasons to choose LCS over other RL methods for this industrial problem.

Knittel and Bossomaier [88], [89] introduced LCS called the *Activation Reinforcement Classifier System* (ARCS) and evaluated its capabilities in the game of *Dots and Boxes*. ARCS was based on the concept of reusable features and a modular design to improve LCS scalability in combinatorial games. The main idea was to learn small fragments or features from the environment as rule constructs and then use networks of these features to define matching rules for given states of the game. The performance of ARCS and XCS was compared with increasing board sizes in the game. Both systems showed similar learning trends as the game size was varied. However, ARCS performed slightly better than XCS in catching-up speed when the problem size was changed. Once again, this work exemplified the flexibility of LCS to be abstracted to other evolutionary RL algorithms for meeting specific problem requirements.

LCS applications in combinatorial games are surprisingly limited, despite the fact that such games can replace the traditional synthetic problems used in basic LCS research in RL problems easily. In addition to being good benchmark problems, such games can become a source of innovation and improvement in LCS research, as evidenced by the works above.

C. Simulation Games

Simulation games are not common, owing to their generally serious applications. It is encouraging to see a small number of LCS applications being used in this category of games. Kobayashi and Terano [90] explored the use of XCS in designing a simulator for business education, as well as an artificial agent that engaged with the simulator along with the human users. The simulation results suggested that XCS was able to learn better business decisions, which could lead to better profits, than the other types of agents (including humans). The authors also conducted a sensitivity analysis of XCS parameters and suggested that a higher GA frequency (lower θ_{GA}) and a smaller exploration probability provided better and more robust outcomes. The authors highlighted the interpretability of the learnt model as a key advantage in this set-up, which allowed an easier integration with human users and other hand-coded agents. The rules learned by XCS were also deemed useful for developing new game scenarios.

Fernando [91] showed the ability of LCS to learn lexical and syntactic conventions for effective communication between two agents in a "language game". A language game models communication between at least two agents, with the agents trying to develop a language convention in order to understand each other, to achieve a common goal. The specific language game in this work used arbitrary concepts and sounds to represent language conventions. These were encoded as alphabetic strings and integers, respectively. Two implementations of XCS were employed in a co-evolutionary framework, one acting as a transmitter (speaker) and the other as a receiver (listener). Each XCS, in turn, consisted of two separate classifier populations, one covering the syntactic rules and the other covering the lexical rules. The results showed that the XCS-based agents were able to learn correctly the syntactic and lexical rules that represented the communicated concepts. This work demonstrated the application of LCS in implementing symbolic communication between agents. The authors found the incremental addition of new populations of classifiers an especially useful feature, as it allowed modification of syntactical conventions without interfering with lexical conventions. They also noted that the generalization pressure in XCS helped in learning systematic syntactic conventions [91].

Li and Liu [92] used LCS to model two types of agents (i.e., institutional investors and regulators) in a game that modeled regulatory dynamics in an artificial stock market. The two types of agents adapted their strategies during the simulation; that is, while the agents representing small and medium-sized enterprises engaged in making investment decisions. Unfortunately,

the authors did not describe the details of the LCS models used in the simulation, but listed adaptability as the key strength that encouraged them to adopt LCS for this problem.

As with the combinatorial games, the LCS applications under this category were limited and warrant further investigation. LCS can be used both as an agent interacting with other agents in the simulated environment and as a simulation miner with a goal to continuously improve the simulation and make it more interesting for its users.

D. Game Theory

Several studies have explored the use of LCS in game-theoretic setting. One of the first studies in this respect was presented in [93]. Instead of modeling game-playing agents, the authors used the concept of evolutionary stable strategy to provide theoretical foundations for convergence in LCS. Their empirical results, using a *Hawke and Dove* game, showed that the bucket-brigade algorithm leads to an evolutionary stable population, as long as the population is kept static; that is, the GA operation is suspended.

Seredyski et al. [58] were perhaps the first researchers to propose the use of LCS as a game-playing agent. They used CS-1 to encode agents that engaged in iterative two-by-two games with their immediate neighbors. The agents were placed on a ring to enforce the neighborhood structure. Later, this work was extended to develop a task scheduler for parallel computing and was reported in a series of papers, [94]–[96]. This work may also be considered the first application of LCS to spatial games. The work demonstrated the strength of game-theoretic modeling using LCS in designing effective and cooperative multi-agent systems for the environments requiring both parallel and distributed control.

Bagnall et al. [97] used XCS to model game-playing agents to study the bidding policies in an electricity market. Different XCS-based agents were introduced based on the type of electricity production (e.g., nuclear and gas). The agents interacted in a simulation setting with the goal of learning an optimal bidding strategy, across different electricity production scenarios, to optimize two different objectives: to minimize capital losses and to maximize daily profit. The agent architecture used two separate populations, each aiming to approximate the two objective functions independently. An agent controller was used to determine the final action (i.e., the per unit bidding price), based on the prediction obtained from the underlying populations. The experiment's results showed that the agents were able to adapt to different marketing mechanisms. Some evidence of evolving a cooperative behavior was also found under certain scenarios.

Interest in the role of agent-based systems in an electricity market has been renewed since the popularization of the smart-grid concept [98]. Bagnall's work demonstrated the potential of game-theoretic LCS agent models for this important domain.

Meng and Pakath [99] and later, Gaines and Pakath [100], [101] used CS-1 and XCS, respectively, to learn strategies in the traditional IPD game. The experiment's results showed better performance of LCS-based agents against pre-scripted strategies, including the infamous *Tit-for-Tat*. A comparison of the two LCS approaches (i.e., CS-1 and XCS) was performed in a study by [101]. The results showed that XCS performed better than CS-1 when dealing with deterministic opponents, while CS-1 performed better when dealing with stochastic opponents. IPD is one of the most studied games in EGT [102], with evolutionary algorithms commonly used to evolve evolutionarily stable strategies. However, most work in this area has focused on the monolithic agent models, with a single population used to evolve the best strategy. LCS, as shown in the above studies, provide an intuitive and clear way to extend this work in a multi-agent setting, allowing the investigation of heterogeneous agents' interactions in the iterative game environment.

Hercog and Fogarty [29] used a ZCS in a multi-agent simulation setting to study emergent dynamics in an "El Farol Bar" (EFB) problem. EFB, later generalized as "minority games" [103], models social dilemma situations in which unique game solutions may not exist and agents may need to take a more practical approach to reach a decision, such as forming hypotheses or learning from experience. This leads to a continuously evolving equilibrium. More recently, Hercog [104] extended the above study and used an XCS to model agents for an EFB variant that considered multiple bars. The authors contended that such modeling and simulation studies could be beneficial for benchmarking multi-agent learning and analyzing coordination problems. This work highlighted the ability of game-theoretic LCS agents to provide insights into both micro- and macro-level behaviors in complex dynamics.

Takadama et al. [105] used a Pittsburgh-style LCS for modeling an evolutionary agent in a "bargaining game". Although the focus of this study was the validation of multi-agent simulations, it was one of the few pieces of work reported in this review to use Pittsburgh-style LCS in a game setting. Three different agent learning architectures were tested, including LCS, evolutionary strategy and an RL-based agent. The bargaining game was used as the simulation model. This work highlighted that because of high variation between multiple simulation runs, more research is needed to design LCS methods that can be validated in these simulation environments.

Mailliard et al. [106] used a simplified version of CS-1 as a learning mechanism for social behavioral rules in an organized theory of actions framework. Their simplified LCS implemented neither a GA for rule discovery nor the bucket-brigade algorithm for the credit apportionment. Instead, they focused on modeling the action selection in LCS, based on different social satisfaction levels, computed as a function of stakes or weights associated with each social relationship and the corresponding expected payoffs. An exploratory study was conducted using the standard PD game to investigate the types of action strategies learned by LCS and the corresponding effects on the game dynamics. This work showed the utility of LCS in performing strategic analysis using game-theoretic models.

Xianyu and Yang [107] used LCS-based agents in a co-evolutionary spatial game-theoretic setting to study the fairness behavior in an "ultimatum game". In contrast with classical EGT, which allows homogeneous mixing of agents, spatial games [108] provide a useful platform to model and study the effect of neighborhood structure on the evolution of cooperation. The conventional approaches use replicator dynamics to model evolutionary dynamics in spatial games. This work was one of the few studies to explore the use of machine learning agents, and perhaps the only one after Seredyski et al. [58] to explore the use of LCS in spatial games. CS-1 was used to model agents and the game was studied on both small-world and scale-free networks under perfect and imperfect information conditions. This work further supported the observations we made above on Pakath's work [99], that LCS provide far richer modeling opportunities for EGT than conventional GA-based models. In summary, the selected work reviewed in this section has highlighted the strengths of LCS as a versatile modeling framework, flexible knowledge representation, learning mechanism and communication interface for studying complex dynamics using strategy games.

VI. Discussion

Table I summarizes the literature survey presented in Section V across three features: type and sub-type of games to which LCS is applied; names of games; and types of LCS used to model agents in these games. The earliest application of LCS to games occurred in the mid 1990s. The initial studies mostly used Holland's CS-1 in their models. Later, the focus shifted to the use of XCS, which became the most frequently and successfully used classifier system in this literature. However, the original LCS models continue to be used to this day.

TABLE I

SUMMARY OF LCS LITERATURE ON GAMES.

LEGEND: YOP=YEAR OF PUBLICATION; TOG=TYPE OF GAME; SUB-CAT=SUB-CATEGORY OF GAME; NOG=NAME OF GAME;TOLCS = TYPE OF LCS

Ref	YOP	TOG	Sub-Cat	NOG	TOLCS
[58], [94]	1995	Strategy	Restricted games	Ring Game	CS-1
[97]		Strategy	Non-cooperative games	Electricity Market	XCS
[99]	2001	Strategy	Non-sequential, non-cooperative	Iterated Prisoner's Dilemma	CS-1
[29]	2002	Strategy	Minority	El Farol Bar	ZCS
[60]–[62]	2002	Role playing	MMORPG	Ryzom	MHiCS
[66]	2003	Role playing	Multi-agent	RoboCup	XCS
[73]	2003	Role playing	Multi-agent	Tank Battle	ZCS
[90]	2003	Simulation	Education, policy	Business Simulator	XCS
[82], [83]	2005	Combinatorial	Board	Connect4	XCS
[86]	2005	Strategy	Various	Nim, IPD, Matrix Choice	XCS
[68]–[72]	2005	Role Playing	Multi-agent	Robosoccer	XCS
[63]	2006	Role Playing	Single-agent	V-Man	XCS
[106]	2007	Strategy	Non-sequential, non-cooperative	Prisoner's Dilemma	CS-1
[74], [75]	2008	Role Playing	Single-agent	Wargus	XCS
[76]	2009	Role Playing	Single-agent	Unreal Tournament	Pittsburgh-style
[107]	2010	Strategy	Sequential, Non-cooperative	Ultimatum Game	CS-1
[77]	2010	Role Playing	Multi-agent	Robocode	XCS
[81]	2010	Role Playing	Single-agent	Pac-Man	OCS
[88]	2011	Combinatorial	Board	Dots and Boxes	ARCS
[78]	2011	Role Playing	Single-agent	Battle Game	ZCS
[91]	2011	Language game		Language Game	XCS
[104]	2013	Strategy	Minority	Multi EFB	XCS
[80]	2013	Role Playing	Single-agent	Battleship	XCS
[92]	2014	Simulation	Policy	Stock Market	XCS
[64]	2015	Role Playing	Single-agent	Air-Combat	EDS

A key feature that has been highlighted by this survey is the flexibility of LCS, which allows modeling, both simple as well as meta-learning agent architectures, under the same framework. This powerful feature can be leveraged in scaling LCS to higher-fidelity and more complex game environments. The methods used to represent the interfacing environments, or the type of input to the system, and encoding classifiers seemed to play a significant role in the performance of LCS-based agents. While binary encoding was the dominant method used in the reviewed approaches, this could be limiting in providing adequate control resolution and the range of behaviors that the agents could exhibit. The use of real-valued [109], [110] or other types of encoding [111], [112] could be explored, with consideration given to their own costs [113].

Most of the reviewed works were published in conferences or workshops. A major shortcoming of most publications was that they did not provide important implementation details for non-LCS users, making it harder to adopt LCS for games outside the LCS research community. The availability of online source codes for specific LCS-based game agent implementations could help improve the visibility of these approaches, as well as their acceptance by the wider research community. Many of the reviewed works did not provide a deep analysis of the system and its performance, especially for more complex system implementations, such as MHiCS. Therefore, the development of analysis tools for game environments is another important research direction in this field.

The application to games of classical AI techniques, such as finite-state-machines, scripting, case-based models, and decision trees, has a long history and the techniques continue to be used to this day. However, the need to depart from such techniques and begin to use more advanced AI developed at the beginning of the new millennium, as the demand for more dynamic, challenging, and interactive game design increased [114]–[116]. This trend continues to grow, as the maturation of new technologies, such as virtual reality and immersive games, demands more creative and real-time behavior from artificial agents. Traditional AI techniques offer simplicity, a faster response time, and lower demand for resources and therefore, are often preferred over the advanced AI/CI techniques. However, such techniques are mostly inflexible as they are constrained with their own pre-programmed knowledge. While LCS share many of these difficulties with their counterparts, they have major advantages for game developers, such as their design flexibility; their ability to be applied to real-time environments; their ability to learn continuously and adapt in dynamic environments; and the model transparency.

VII. Future Directions

A wide range of games exists in all of the categories discussed in Section IV in which LCS-based agent approaches have not been applied. Comparative studies between LCS and other CI-based agent approaches in popular games (e.g., Dove and Hawkes, Snow Drift, Chess, and Go), are warranted. Simple game environments, such as RoboCode, can be used as a useful platform for the evaluation and further development of LCS, especially in multi-step environments.

It will also be interesting to explore other types of LCS for games. Specifically, there are limited applications of Pittsburgh-style LCS in games. Recent advances in this discipline [117], [118] provide important research opportunities to be explored.

In this section, we offer ideas for future directions related to LCS for games. Each of these ideas represents an under-explored research area that has the potential to advance both LCS and games research.

A. Architectures for Games

One primary advantage of LCS that has not been fully explored is that these systems come with an architecture. Such an architecture can offer a structured way to decompose and manage complex problems. For example, a large body of literature exists on evolutionary games and evolutionary spatial games, which are dominated by the application of conventional GA-based agent approaches. LCS-based agent technologies can contribute significantly to this field by providing more flexible and function-rich agent architectures that allow agents to learn from previous interactions, in addition to being evolutionary.

B. Symbolic Non-Symbolic Dilemma

For most recreational games, possibly the most important performance-based objectives are to deliver believable actors and to win the game. Believability of character is a challenging issue. It can be handled in a data-driven manner, through fast exploration of a large search space to make wise and smart decisions. This has created a demand for methods that are fast, can explore a large space of possibilities, and make decisive winning decisions. Non-symbolic methods that are fast, such as neural networks [119], deep networks [120], and Monte-Carlo tree search [121], [122], offer alternatives to symbolic methods and have been shown in the literature to have different levels of success. However, these approaches lack the expressive power to explain decisions in classic reasoning. Another way to achieve believability is to explore symbolic methods. In the

absence of optimized implementations and appropriate architectures [85], these methods can be slow in contexts requiring believability. However, they can offer rational decisions that can be explained by traversing the rules in the knowledge base.

Implementations of LCS (e.g., XCS), integrate non-symbolic RL and symbolic rules mechanisms. In a similar fashion, it is plausible to investigate a hybrid approach that uses LCS to learn meta-strategies while a neural network or a Monte-Carlo method is employed to implement the specifics of these strategies. The combination of symbolic and non-symbolic methods in LCS has had successful outcomes in other areas, such as data mining [112], [123] and robotics [27]. Other recent work has explored the use of genetic network programming [124] and Boolean networks [125] representations within the hybrid LCS models.

C. Reasoning in LCS for Games

The symbolic nature of LCS offers an opportunity to reason in games. This level of reasoning can be used to explain actions for users, which could help them to diagnose their performance, train faster, or conduct post-action reviews. However, the learning ability of LCS comes with a disadvantage; that is, many rules can be generated, leading to large rule sets and hence low interpretability. Moreover, some implementations, such as XCS, generate rules locally, and this causes the rule set to increase rapidly, with many overlapping rules.

The above problems have been addressed in the LCS literature, but these solutions have not been tried in the application of LCS to games. Some of these methods clean the population in an offline model [126], [127], while others compress the population in real time as learning occurs [128].

D. Role of LCS Role in Interactive Simulation

Interactive simulations have many uses for education, training, and planning [129]. The value of these simulations increases when the system is able to diagnose users' actions and offer corrective actions. In this scenario, the objective is not to learn the game or the simulation itself, but to build a model of the user to explore ways of improving user performance. Therefore, interaction can be adaptive in response to the users' performance.

Non-symbolic methods are not useful for handling this problem type. They cannot offer diagnostic capabilities for users; nor can they offer a reasoning chain to explain to the users the

implications of their actions. LCS offer an opportunity for these simulations to diagnose and reason about users' actions.

VIII. Conclusions

LCS are one of the earliest evolutionary computing techniques and are a continuously growing field of research, providing a natural development platform for computer games. In particular, due to their roots in computational cognition and symbolic representation, LCS are ideal to facilitate interaction between a game and a human, in addition to their natural use as an adaptive learner.

This survey is the first to provide a review of the extant LCS research in games. By reviewing diverse LCS applications in a variety of games, this survey brings to attention the potential of LCS for this field. The survey has offered insights into several future directions, ranging from examining learning architectures and symbolic and non-symbolic dilemmas, to reasoning and the use of LCS to mediate adaptively in interactive games. These directions offer significant opportunities to researchers in both LCS and games.

This survey is timely, in that both computer games and LCS are growing research areas with many advancements. These advancements have obvious implications for either field. It is hoped that by presenting a consolidated review of the extant work, this survey will prove to be a stepping-stone in triggering the required research interest in leveraging on these advancements and exploring this important area of research.

References

[1] E. A. Boyle, T. M. Connolly, T. Hainey, and J. M. Boyle, "Engagement in digital entertainment games: A systematic review," *Computers in Human Behavior*, vol. 28, no. 3, pp. 771–780, May 2012.

[2] H. T. J. Smit and L. Trigeorgis, *Strategic Investment: Real Options and Games*. Princeton University Press, Princeton, New Jersey, 2012.

[3] T. Ben-Zvi, "The efficacy of business simulation games in creating decision support systems: An experimental investigation," *Decision Support Systems*, vol. 49, no. 1, pp. 61–69, April 2010.

[4] S. A. Adams, "Use of "serious health games" in health care: A review," *Studies in Health Technology and Informatics*, vol. 157, pp. 160–166, June 2010.

[5] G. Fong, "Adapting COTS games for military experimentation," *Simulation & Gaming*, vol. 37, no. 4, pp. 452–465, December 2006.

[6] D. R. Michael and S. L. Chen, *Serious Games: Games That Educate, Train, and Inform*. Course Technology, Cengage Learning PTR, Mason, OH, USA, 2005.

[7] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems," *Proc. of National Academy of Sciences*, vol. 99, no. suppl 3, pp. 7280–7287, May 2002.

[8] Entertainment Software Association (ESA), "The 2015 Essential Facts About the Computer and Video Game Industry," http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf, 2015, accessed 20 November 2015.

[9] A. P. Engelbrecht, *Computational Intelligence: An Introduction*. John Wiley & Sons, West Sussex, England, 2007.

[10] S. M. Lucas, "Computational Intelligence and AI in Games: A new IEEE Transactions," *IEEE Trans. Comput. Intell. AI in Games*, vol. 1, no. 1, pp. 1–3, March 2009.

[11] D. Loiacono, P. L. Lanzi, J. Togelius, E. Onieva, D. A. Pelta, M. V. Butz, T. D. Lönneker, L. Cardamone, D. Perez, Y. Sáez, M. Preuss, and J. Quadflieg, "The 2009 simulated car racing championship," *IEEE Trans. Comput. Intell. AI in Games*, vol. 2, no. 2, pp. 131–147, June 2010.

[12] A. Ghosh and S. Tsutsui, *Advances in Evolutionary Computing: Theory and Applications*, ser. Natural Computing. Springer, Berlin, 2012.

[13] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, January 2015.

[14] S. M. Lucas, "Computational intelligence and games: Challenges and opportunities," *International Journal of Automation and Computing*, vol. 5, no. 1, pp. 45–57, January 2008.

[15] G. Yannakakis and J. Togelius, "A panorama of artificial and computational intelligence in games," *IEEE Trans. Comput. Intell. AI in Games*, vol. 7, no. 4, pp. 317–335, December 2015.

[16] L. Bull, "Learning classifier systems: A brief introduction," in *Applications of Learning Classifier Systems*, L. Bull, Ed. Springer, Berlin, 2004, pp. 1–12.

[17] O. Sigaud and S. W. Wilson, "Learning classifier systems: A survey," *Soft Computing*, vol. 11, no. 11, pp. 1065–1078, September 2007.

[18] R. J. Urbanowicz and J. H. Moore, "Learning classifier systems: A complete introduction, review, and roadmap," *Journal of Artificial Evolution and Applications*, vol. 2009, pp. 1–25, January 2009.

[19] A. Fernández, S. García, J. Luengo, E. Bernadó-Mansilla, and F. Herrera, "Genetics-based machine learning for rule induction: Taxonomy, experimental study and state of the art," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 913–941, December 2010.

[20] L. Bull, "A brief history of learning classifier systems: from CS-1 to XCS and its variants," *Evolutionary Intelligence*, vol. 8, no. 2-3, pp. 55–70, September 2015.

[21] K. Shafi and H. A. Abbass, "Biologically-inspired complex adaptive systems approaches to network intrusion detection," *Information Security Technical Report*, vol. 12, no. 4, pp. 209–217, December 2007.

[22] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial Intelligence Review*, vol. 18, no. 2, pp. 77–95, June 2002.

[23] S. W. Wilson, "ZCS: A zeroth level classifier system," *Evolutionary Computation*, vol. 2, no. 1, pp. 1–18, March 1994.

[24] Z. Guessoum, L. Rejeb, and R. Durand, "Using adaptive multi-agent systems to simulate economic models," in *Proc. of Third International Joint Conference on Autonomous Agents and Multiagent Systems*, Washington DC, July 19-23 2004, pp. 68–75.

[25] L. A. Wehinger, M. D. Galus, and G. Andersson, "Agent-based simulator for the German electricity wholesale market including wind power generation and widescale PHEV adoption," in *Proc. of 7th international conference on the European Energy Market*, Madrid, June 23-25 2010, pp. 1–6.

[26] V. V. Pham, L. T. Bui, S. Alam, C. Lokan, and H. A. Abbass, "A Pittsburgh multi-objective classifier for user preferred trajectories and flight navigation," in *Proc. of 2010 IEEE Congress on Evolutionary Computation*, Barcelona, July 18-23 2010, pp. 608–615.

[27] J. Hurst and L. Bull, "A neural learning classifier system with self-adaptive constructivism for mobile robot control," *Artificial Life*, vol. 12, no. 3, pp. 353–380, July 2006.

[28] R. E. Smith, A. El-Fallah, B. Ravichandran, R. K. Mehra, and B. A. Dike, "The fighter aircraft LCS: A real-World, machine innovation application," in *Applications of Learning Classifier Systems*, L. Bull, Ed. Springer, Berlin, 2004, pp. 113–142.

[29] L. M. Hercog and T. C. Fogarty, "Co-evolutionary classifier systems for multi-agent simulation," in *Proc. of 2002 IEEE Congress on Evolutionary Computation*, vol. 2, Honolulu, May 12-17 2002, pp. 1798–1803.

[30] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975, republished by the MIT press, 1992.

[31] ——, "Adaptation," in *Progress in theoretical biology*, R. Rosen and F. M. Snell, Eds. Plenum, New York, 1976.

[32] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard, *Induction: Processes of Inference, Learning, and Discovery*. MIT Press, Cambridge, Massachusetts, 1986.

[33] P. L. Lanzi, W. Stolzmann, and S. W. Wilson, *Learning Classifier Systems: From Foundations to Applications*. Springer, Berlin, 2000.

[34] J. H. Holland, L. B. Booker, M. Colombetti, M. Dorigo, D. E. Goldberg, S. Forrest, R. L. Riolo, R. E. Smith, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, "What is a learning classifier system?" in *LNCS 1813: Learning Classifier Systems: From Foundations to Applications*, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, Eds. Springer, Berlin, 2000, pp. 3–32.

[35] J. H. Holland and J. S. Reitman, "Cognitive systems based on adaptive algorithms," in *Pattern-directed Inference Systems*, D. A. Waterman and F. Hayes-Roth, Eds. Academic Press, New York, 1978, reprinted in: Evolutionary Computation. The Fossil Record. David B. Fogel (Ed.) IEEE Press, 1998.

[36] G. Tesauro, "Temporal difference learning and TD-Gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, March 1995.

[37] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[38] M. Dorigo and H. Bersini, "A comparison of Q-learning and classifier systems," in *From Animals to Animats 3: Proc. of Third International Conference on Simulation of Adaptive Behavior*. Brighton, United Kingdom: MIT Press Cambridge, MA, 1994, pp. 248–255.

[39] P. L. Lanzi, "Learning classifier systems from a reinforcement learning perspective," *Soft Computing*, vol. 6, no. 3-4, pp. 162–170, June 2002.

[40] S. F. Smith, "A learning system based on genetic adaptive algorithms," Ph.D. dissertation, University of Pittsburgh, Pittsburgh, 1980.

[41] K. De Jong, "Learning with genetic algorithms: An overview," *Machine Learning*, vol. 3, no. 2-3, pp. 121–138, October 1988.

[42] T. Kovacs, "Strength or accuracy? Fitness calculation in learning classifier systems," in *Learning Classifier Systems: From Foundations to Applications*, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, Eds. Springer, Berlin, 2000, pp. 143–160.

[43] J. H. Holland, "Adaptive algorithms for discovering and using general patterns in growing knowledge-bases," *International Journal of Policy Analysis and Information Systems*, vol. 4, no. 3, pp. 245–268, September 1980.

[44] S. W. Wilson, "Classifier fitness based on accuracy," *Evolutionary Computation*, vol. 3, no. 2, pp. 149–175, June 1995.

[45] K. Shafi, H. A. Abbass, and W. Zhu, "The role of early stopping and population size in XCS for intrusion detection," in *LNCS 4247: Proc. of 6th International Conference on Simulated Evolution and Learning: SEAL 2006, Hefei, China, October 15-18, 2006.*, T.-D. Wang, X. Li, S.-H. Chen, X. Wang, H. A. Abbass, H. Iba, G.-L. Chen, and X. Yao, Eds. Springer, Berlin, 2006, pp. 50–57.

[46] M. A. Franco, N. Krasnogor, and J. Bacardit, "GAssist vs. BioHEL: Critical assessment of two paradigms of genetics-based machine learning," *Soft Computing*, vol. 17, no. 6, pp. 953–981, June 2013.

[47] J. H. Smith, "The games economists play-implications of economic game theory for the study of computer games," *Game Studies*, vol. 6, no. 1, pp. 1–15, December 2006.

[48] M. Müller, "Computer Go as a sum of local games: An application of combinatorial game theory," Ph.D. dissertation, ETH, Zürich, 1995.

[49] A. Amory, K. Naicker, J. Vincent, and C. Adams, "The use of computer games as an educational tool: Identification of appropriate game types and game elements," *British Journal of Educational Technology*, vol. 30, no. 4, pp. 311–321, October 1999.

[50] T. H. Apperley, "Genre and game studies: Toward a critical approach to video game genres," *Simulation & Gaming*, vol. 37, no. 1, pp. 6–23, March 2006.

[51] S. M. Iversen, "In the double grip of the game: Challenge and Fallout 3," *Game Studies*, vol. 12, no. 2, December 2012.

[52] M. K. Schlickum, L. Hedman, L. Enochsson, A. Kjellin, and L. Felländer-Tsai, "Systematic video game training in surgical novices improves performance in virtual reality endoscopic surgical simulators: A prospective randomized study," *World Journal of Surgery*, vol. 33, no. 11, pp. 2360–2367, November 2009.

[53] S. L. Wang, K. Shafi, C. Lokan, and H. A. Abbass, "An agent-based model to simulate and analyse behaviour under noisy and deceptive information," *Adaptive Behavior*, vol. 21, no. 2, pp. 96–117, April 2013.

[54] S. Alam, K. Shafi, H. A. Abbass, and M. Barlow, "Evolving air traffic scenarios for the evaluation of conflict detection models," in *Proc. of 6th Eurocontrol Innovative Research Workshop*, Eurocontrol Experiment Research Center, Paris, Dec. 4-6 2007, pp. 237–245.

[55] C. Camerer, *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, Princeton, New Jersey, 2003.

[56] J. W. Weibull, *Evolutionary Game Theory*. MIT press, Cambridge MA, 1997.

[57] R. Axelrod, "The evolution of strategies in the Iterated Prisoners Dilemma," in *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed. Pitman, London, and Morgan Kaufman, Los Altos, CA, 1987, pp. 32–41.

[58] F. Seredyński, P. Cichosz, and G. P. Klebus, "Learning classifier systems in multi-agent environments," in *Proc. of First IEE/IEEE Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield, Sept. 12-14 1995, pp. 287–292.

[59] S. F. Smith, "Flexible learning of problem solving heuristics through adaptive search," in *Proc. of Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, 1983, pp. 422–425.

[60] G. Robert, P. Portier, and A. Guillot, "Classifier systems as 'animat' architectures for action selection in MMORPG," in *Proc. of 3rd International Conference on Intelligent Games and Simulation*, London, Nov. 29-30 2002.

[61] G. Robert and A. Guillot, "MHiCS, A modular and hierarchical classifier systems architecture for bots," in *Proc. of 4th International Conference on Intelligent Games and Simulation*, London, Nov. 19-21 2003, pp. 140–144.

[62] ——, "MHiCS, A motivational architecture of action selection for Non-player characters in dynamic environments," *International Journal of Intelligent Games & Simulation*, vol. 4, no. 1, pp. 5–16, June 2006.

[63] S. Sanchez, H. Luga, and Y. Duthen, "Learning classifier systems and behavioural animation of virtual characters," in *LNCS 4133: Proc. of 6th International Conference on Intelligent Virtual Agents*, J. Gratch, M. Young, R. Aylett, D. Ballin, and P. Olivier, Eds., CA, Aug. 21-23 2006, p. 467.

[64] R. Kop, A. Toubman, M. Hoogendoorn, and J. J. Roessingh, "Evolutionary dynamic scripting: Adaptation of expert rule bases for serious games," in *LNCS 9101: Current Approaches in Applied Artificial Intelligence: Proc. of 28th International*

Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2015, Seoul, South Korea, June 10-12, 2015, M. Ali, S. Y. Kwon, C.-H. Lee, J. Kim, and Y. Kim, Eds. Springer, Berlin, 2015, pp. 53–62.

[65] A. Bonarini and V. Trianni, "Learning fuzzy classifier systems for multi-agent coordination," *Information Sciences*, vol. 136, no. 1, pp. 215–239, August 2001.

[66] C. Castillo, M. Lurgi, and I. Martínez, "Chimps: An evolutionary reinforcement learning approach for soccer agents," in *Proc. of 2003 IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, Washington, D.C., Oct. 2003, pp. 60–65.

[67] S. Kinoshita and Y. Yamamoto, "11Monkeys description," in *LNCS 1856: RoboCup-99: Robot Soccer World Cup III*, M. Veloso, E. Pagello, and H. Kitano, Eds. Springer, Berlin, 2000, pp. 550–553.

[68] Y. Sato and T. Kanno, "Event-driven hybrid learning classifier systems for online soccer games," in *Proc. of 2005 IEEE Congress on Evolutionary Computation*, vol. 3, Edinburgh, Sept. 2-5 2005, pp. 2091–2098.

[69] Y. Sato, Y. Akatsuka, and T. Nishizono, "Reward allotment in an event-driven hybrid learning classifier system for online soccer games," in *Proc. of 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, July 8-12 2006, pp. 1753–1760.

[70] Y. Akatsuka and Y. Sato, "Reward allotment considered roles for learning classifier system for soccer video games," in *Proc. of 2007 IEEE Symposium on Computational Intelligence and Games*, Honolulu, HI, Apr. 1-5 2007, pp. 288–295.

[71] Y. Sato, Y. Inoue, and Y. Akatsuka, "Applying GA to self-allotment of rewards in event-driven hybrid learning classifier systems," in *Proc. of 2007 IEEE Congress on Evolutionary Computation*, Singapore, Sept. 25-28 2007, pp. 1800–1807.

[72] Y. Inoue and Y. Sato, "Applying GA for reward allotment in an event-driven hybrid learning classifier system for soccer video games," in *Proc. of 2008 IEEE Symposium on Computational Intelligence and Games*, Florida, Dec. 15-18 2008, pp. 296–303.

[73] W. J. Falke II and P. Ross, "Dynamic strategies in a real-time strategy game," in *Proc. of 5th Annual Conference on Genetic and Evolutionary Computation*. Chicago: Springer, Berlin, July 9-11 2003, pp. 1920–1921.

[74] A. Lujan, R. Werner, and A. Boukerche, "Generation of rule-based adaptive strategies for a collaborative virtual simulation environment," in *Proc. of 2008 IEEE International Workshop on Haptic Audio Visual Environments and Games*, Ottawa, Oct. 18-19 2008, pp. 59–64.

[75] A. Lujan, "Generation of rule-based adaptive strategies for games," Ph.D. dissertation, University of Ottawa, Ottawa, Canada, 2009.

[76] R. Small and C. B. Congdon, "Agent Smith: Towards an evolutionary rule-based agent for interactive dynamic games," in *Proc. of 2009 IEEE Congress on Evolutionary Computation*, Trondheim, Norway, May 18-21 2009, pp. 660–666.

[77] D. G. Nidorf, L. Barone, and T. French, "A comparative study of NEAT and XCS in Robocode," in *Proc. of 2010 IEEE Congress on Evolutionary Computation*, Barcelona, July 18-23 2010, pp. 86–93.

[78] M. T. Tsapanos, K. C. Chatzidimitriou, and P. A. Mitkas, "A zeroth-level classifier system for real time strategy games," in *Proc. of 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 2, Lyon, France, Aug. 22-27 2011, pp. 244–247.

[79] M. Buro, "ORTS: A hack-free RTS game environment," in *LNCS 2883: Third International Conference on Computers and Games, Revised Papers*, J. Schaeffer, M. Müller, and Y. Björnsson, Eds. Edmonton: Springer, Berlin, July 25-27 2003, pp. 280–291.

[80] L. Clementis, "Learning classifier system improvement based on probability driven and neural network driven approaches," in *Proc. of 3rd Eastern European Regional Conference on the Engineering of Computer Based Systems*, Budapest, Aug. 29-30 2013, pp. 143–148.

[81] M. Irvan, T. Yamada, and T. Terano, "Is XCS approach good for organizational-learning oriented classifier systems?" in *Proc. of 24th Annual Conference of The Japanese Society for Artificial Intelligence*, Florida, June 9-11 2010.

[82] W. Browne and D. Scott, "An abstraction algorithm for genetics-based reinforcement learning," in *Proc. of 7th Annual Conference on Genetic and Evolutionary Computation*, Washington DC, USA, June 25-29 2005, pp. 1875–1882.

[83] W. Browne, D. Scott, and C. Ioannides, "Abstraction for genetics-based reinforcement learning," in *Reinforcement Learning: Theory and Applications*, M. E. Cornelius Weber and N. M. Mayer, Eds. I-Tech Education and Publishing, Vienna, Austria, 2008, pp. 187–198.

[84] J. H. Holland, "Concerning the emergence of tag-mediated lookahead in classifier systems," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1, pp. 188–201, June 1990.

[85] K. Shafi, H. A. Abbass, and W. Zhu, "Real time signature extraction from a supervised classifier system," in *Proc. of 2007 IEEE Congress on Evolutionary Computation*, Singapore, Sept. 25-28 2007, pp. 2509–2516.

[86] N. P. Sood, A. G. Williams, and K. A. De Jong, "Evaluating the XCS learning classifier system in competitive simultaneous learning environments," in *Proc. of 7th Annual Workshop on Genetic and Evolutionary Computation*, Washington, D.C., June 25-29 2005, pp. 112–118.

[87] S. R. Beckman, "Cournot and Bertrand games," *The Journal of Economic Education*, vol. 34, no. 1, pp. 27–35, March 2003.

[88] A. Knittel and T. Bossomaier, "Re-usable features in a hierarchical concept network for autonomous learning in complex games," in *Proc. of 2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems*, Paris, Apr. 11-15 2011, pp. 155–162.

[89] A. Knittel, T. Bossomaier, and A. Snyder, "Concept accessibility as basis for evolutionary reinforcement learning of dots and boxes," in *Proc. of 2007 IEEE Symposium on Computational Intelligence and Games*, Honolulu, HI, Apr. 1-5 2007, pp. 140–145.

[90] M. Kobayashi and T. Terano, "Learning agents in a business simulator," in *Proc. of 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2003*, vol. 3. IEEE, 2003, pp. 1323–1327.

[91] C. Fernando, "Co-evolution of lexical and syntactic classifiers during a language game," *Evolutionary Intelligence*, vol. 4, no. 3, pp. 165–182, September 2011.

[92] Y. Li and S. Liu, "Agent-based modeling simulation analysis on the regulation of institutional investor's encroachment behavior in stock market," *Journal of Industrial Engineering and Management*, vol. 7, no. 2, pp. 434–447, March 2014.

[93] D. F. Yates and A. Fairley, "Evolutionary stability in simple classifier systems," in *LNCS 865: Evolutionary Computing: AISB Workshop Leeds, U.K., April 11–13, 1994 Selected Papers*, T. C. Fogarty, Ed. Springer, Berlin, 1994, pp. 28–37.

[94] F. Seredyński, "Task scheduling with use of classifier systems," in *LNCS 1305: Evolutionary Computing: AISB International Workshop Manchester, UK, April 7–8, 1997 Selected Papers*, D. Corne and J. L. Shapiro, Eds. Springer, Berlin, 1997, pp. 287–306.

[95] ——, "Distributed scheduling using simple learning machines," *European Journal of Operational Research*, vol. 107, no. 2, pp. 401–413, June 1998.

[96] F. Seredynski and C. Z. Janikow, "Learning Nash equilibria by coevolving distributed classifier systems," in *Proc. of 1999 IEEE Congress on Evolutionary Computation*, vol. 3, July 6-9 1999, pp. 1619–1626.

[97] A. J. Bagnall and G. D. Smith, "Game playing with autonomous adaptive agents in a simplified economic model of the UK market in electricity generation," in *Proc. of IEEE-PES/CSEE International Conference on Power System Technology*, vol. 2, Perth, Australia, Dec. 4-7 2000, pp. 891–896.

[98] L. Hernandez, C. Baladron, J. M. Aguiar, B. Carro, A. Sanchez-Esguevillas, J. Lloret, D. Chinarro, J. J. Gomez-Sanz,

and D. Cook, "A multi-agent system architecture for smart grid management and forecasting of energy demand in virtual power plants," *IEEE Communications Magazine*, vol. 51, no. 1, pp. 106–113, January 2013.

[99] C.-L. Meng and R. Pakath, "The Iterated Prisoner's Dilemma: Early experiences with learning classifier system-based simple agents," *Decision Support Systems*, vol. 31, no. 4, pp. 379–403, October 2001.

[100] D. A. Gaines and R. Pakath, "Alternate adaptive agent architectures and behavioral consequences," in *Proc. of Tenth Americas Conference on Information Systems*, New York, Aug. 6-8 2004, pp. 25–28.

[101] ——, "An examination of evolved behavior in two reinforcement learning systems," *Decision Support Systems*, vol. 55, no. 1, pp. 194–205, April 2013.

[102] G. Kendall, X. Yao, and S. Y. Chong, *The Iterated Prisoners' Dilemma: 20 Years on*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2007.

[103] D. Challet, M. Marsili, and Y.-C. Zhang, *Minority Games: Interacting Agents in Financial Markets*. Oxford University Press, Oxford, UK, 2013.

[104] L. M. Hercog, "Better manufacturing process organization using multi-agent self-organization and co-evolutionary classifier systems: The multibar problem," *Applied Soft Computing*, vol. 13, no. 3, pp. 1407–1418, March 2013.

[105] K. Takadama, Y. L. Suematsu, N. Sugimoto, N. E. Nawa, and K. Shimohara, "Cross-element validation in multiagent-based simulation: Switching learning mechanisms in agents," *Journal of Artificial Societies and Social Simulation*, vol. 6, no. 4, October 2003.

[106] M. Mailliard, F. Amblard, C. Sibertin-Blanc, and P. Roggero, "Cooperation is not always so simple to learn," in *Agent-Based Approaches in Economic and Social Complex Systems IV*, ser. Springer Series on Agent Based Social Systems, T. Terano, H. Kita, H. Deguchi, and K. Kijima, Eds. Springer, Japan, 2007, vol. 3, pp. 147–154.

[107] X. Bo and J. Yang, "Evolutionary ultimatum game on complex networks under incomplete information," *Physica A: Statistical Mechanics and its Applications*, vol. 389, no. 5, pp. 1115–1123, March 2010.

[108] M. A. Nowak and R. M. May, "Evolutionary games and spatial chaos," *Nature*, vol. 359, no. 6398, pp. 826–829, October 1992.

[109] S. Wilson, "Get Real! XCS with Continuous-Valued Inputs," in *LNCS 1813: Learning Classifier Systems: From Foundations to Applications*, P. Lanzi, W. Stolzmann, and S. Wilson, Eds. Springer, Berlin, 2000, pp. 209–219.

[110] H. H. Dam, H. A. Abbass, and C. Lokan, "Be real! XCS with continuous-valued inputs," in *Proc. of 7th Annual Conference on Genetic and Evolutionary Computation*, Washington, DC, USA, June 25-29 2005, pp. 85–87.

[111] M. V. Butz, "Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system," in *Proc. of 7th Annual Conference on Genetic and Evolutionary Computation*, Washington DC, USA, June 25-29 2005, pp. 1835–1842.

[112] H. H. Dam, H. A. Abbass, C. Lokan, and X. Yao, "Neural-Based Learning Classifier Systems," *IEEE Trans. on Knowl. and Data Eng.*, vol. 20, no. 1, pp. 26–39, January 2008.

[113] J. Drugowitsch, *Design and Analysis of Learning Classifier Systems: A Probabilistic Approach*, ser. Studies in Computational Intelligence. Springer, Berlin, 2008.

[114] J. Laird and M. VanLent, "Human-level AI's killer application: Interactive computer games," *AI Magazine*, vol. 22, no. 2, p. 15, June 2001.

[115] P. Sweetser and J. Wiles, "Current AI in games: A review," *Australian Journal of Intelligent Information Processing Systems*, vol. 8, no. 1, pp. 24–42, April 2002.

[116] A. Nareyek, "AI in computer games," *Queue*, vol. 1, no. 10, p. 58, February 2004.

[117] J. Bacardit and N. Krasnogor, "Performance and efficiency of memetic Pittsburgh learning classifier systems," *Evolutionary Computation*, vol. 17, no. 3, pp. 307–342, November 2009.

[118] D. A. Calian and J. Bacardit, "Integrating memetic search into the BioHEL evolutionary learning system for large-scale datasets," *Memetic Computing*, vol. 5, no. 2, pp. 95–130, June 2013.

[119] M. Hausknecht, J. Lehman, R. Miikkulainen, and P. Stone, "A neuroevolution approach to general Atari game playing," *IEEE Trans. Comput. Intell. AI in Games*, vol. 6, no. 4, pp. 355–366, December 2014.

[120] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, January 2016.

[121] C. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Trans. Comput. Intell. AI in Games*, vol. 4, no. 1, pp. 1–43, March 2012.

[122] D. Perez, E. J. Powley, D. Whitehouse, P. Rohlfshagen, S. Samothrakis, P. I. Cowling, and S. M. Lucas, "Solving the physical traveling salesman problem: Tree search and macro actions," *IEEE Trans. Comput. Intell. AI in Games*, vol. 6, no. 1, pp. 31–45, March 2014.

[123] P. Rojanavasu, H. H. Dam, H. A. Abbass, C. Lokan, and O. Pinngern, "A self-organized, distributed, and adaptive rule-Based induction system," *IEEE Trans. on Neural Netw.*, vol. 20, no. 3, pp. 446–459, March 2009.

[124] X. Li and K. Hirasawa, "A learning classifier system based on genetic network programming," in *Proc. of 2013 IEEE International Conference on Systems, Man, and Cybernetics*, Manchester, Oct. 13-16 2013, pp. 1323–1328.

[125] R. J. Preen and L. Bull, "Discrete and fuzzy dynamical genetic programming in the XCSF learning classifier system," *Soft Computing*, vol. 18, no. 1, pp. 153–167, January 2014.

[126] S. W. Wilson, "Compact rulesets from XCSI," in *LNCS 2321: Advances in Learning Classifier Systems: 4th International Workshop, IWLCS 2001 San Francisco, CA, USA, July 7–8, 2001 Revised Papers*, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, Eds. Springer, Berlin, 2001, pp. 197–210.

[127] P. Dixon, D. Corne, and M. Oates, "A Ruleset Reduction Algorithm for the XCS Learning Classifier System," in *Learning Classifier Systems: 5th International Workshop, IWLCS 2002, Granada, Spain, September 7-8, 2002. Revised Papers*, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, Eds. Springer, Berlin, 2003, pp. 20–29.

[128] K. Shafi and H. A. Abbass, "An adaptive genetic-based signature learning system for intrusion detection," *Expert Systems With Applications*, vol. 36, no. 10, pp. 12 036–12 043, December 2009.

[129] L. Chittaro and R. Sioni, "Serious games for emergency preparedness: Evaluation of an interactive vs. a non-interactive simulation of a terror attack," *Computers in Human Behavior*, vol. 50, pp. 508–519, September 2015.