Using a Team of General AI Algorithms to Assist Game Design and Testing

Cristina Guerrero-Romero
Queen Mary University of London
London, UK
c.guerreroromero@qmul.ac.uk

Simon M. Lucas
Queen Mary University of London
London, UK
simon.lucas@qmul.ac.uk

Diego Perez-Liebana
Queen Mary University of London
London, UK
diego.perez@qmul.ac.uk

***Abstract*—General Video Game Playing (GVGP) has become a popular line of research in the past years, leading into the existence of a wide range of general algorithms created to tackle this challenge. This paper proposes taking advantage of this research to help in game design and testing processes, introducing a methodology consisting on using a *team* of Artificial General Intelligence agents with differentiated goals (winning, exploring, collecting items, killing NPCs, etc) and skill levels. Using several agents with distinct behaviours can provide substantial information to influence design and bug fixing. Two methods are proposed to aid game design: 1) the evaluation of a game based on the expected performance in the behaviour of each of the agents; and 2) the provision of visual information to analyse how the experience of the agents evolves during play-through. Having this methodology available to designers can help them to decide if the game or level created that is under analysis fits the initial expectations. Including a *Logging System* can also be used to detect anomalies while the development is still in an early stage. We believe this approach allows the flexibility and portability to be easily applied to games with different characteristics.**

I. Introduction

Games evolve during their development process, both in terms of implementation and design. New ideas are put into practice during the production phase, which need to be tested quickly and efficiently. While using human play-testing is a broad practice, there's no denial that this impacts the company in terms of resources (human and technological), time and money. Agent-based testing is a suitable alternative for automatic game testing and there is a prolific body of work (as described later in this paper) that uses game specific agents to evaluate content and agent behaviour. Although this can provide some advantages for fast and reliable testing, the design and implementation of these specific agents may not be adaptable enough to the changes designers and developers are regularly introducing. The use of general algorithms, on the other hand, provides a level of generalisation, portability and flexibility that can't be matched by game-specific ones.

This paper proposes a methodology consisting on a *team* of Artificial General Intelligence (AGI) agents with differentiated goals to aid the design and testing processes during the development of a game. Having a team of general agents focused in distinct objectives (winning, exploring, collecting items, killing NPCs, etc.) and skill levels provides a flexibility that would not be possible using just one. Providing the game under evaluation, choosing the agents and setting expected targets of performance to them, a Logging System and two types of performance reports are generated. The first one gives information about how accurate the estimated performance for each of the behaviours is, compared to the actual results. The second one shows a graph that provides visual feedback of how certain information of the game is retrieved by the agents and evolves during its play-through.

This team of general agents is meant to respond to changes and updates across multiple dimensions of game design:

1) ***Rules****:* These are the base of every game. Making any change on the rules can trigger unexpected outcomes and affect other rules in a way the designer did not plan to. Using general agents, independent from these rules, allow to check that everything is working as it should by running them without having to be modified in any way. This grants the possibility of carrying out an immediate testing to detect anomalies as soon as they appear. It provides flexibility to the methodology and it is one of the core ideas of the approach proposed in this paper. It would be able to work with any game, without having to adapt the algorithms to its specifications.

2) ***Levels****:* Where the action takes place. They shape how the game is presented to the player: increasing level of difficulty, reachable areas, distribution of the elements of the game (proportion of enemies by stages, collectable items dispersed uniformly, etc). Having available a report of each one of the general agents after they have played a certain level could provide the information needed to check that these points are covered as expected. An example would be analysing the evolution of the number of NPCs eliminated by the *Killer* (III-B8). The designer should be able to notice peaks and an abrupt increase of the numbers in those stages where a big confrontation is expected. If those peaks are not presented in the play-through graph, the level should be reviewed and fixed in order to work as desired.

3) ***Non-Playable Characters (NPCs)****:* NPCs' performance and interactions with the player have a big impact on the experience while playing the game. Any update on their implementation should be tested and their impact on player experience analyzed and measured. Analysing the general agents' reports and behaviour could provide an insight of this. For example, checking the number of deaths vs. kills of the *Killer* after a change is done to the NPCs, comparing the difference on percentage of life lost between two *Killers* with

known disparate level of mastery, or tracking the whereabouts of the NPCs (logging similar information to the one measured for the *team*) for behavioural checking.

4) ***Game Parameters****:* Even small updates in the parameters can have a big impact in the game. An example is updating the height of the jump of the avatar: if it is set to a very low value, they might not be able to reach some areas of the game, affecting the exploration. Analysing the information provided by the agents can give a clue to know if the parameters are set properly. In this example, the percentage of the exploration reached by the *Map Explorer* (III-B2) when the height of the jump is modified could increase or decrease abruptly.

The methodology works within the game. If using algorithms specific to it, every time any of its elements is changed the algorithms would need to be updated as well. Having general algorithms, with general goals independent from the rules, implies that they do not have to be modified every time a change is done. This is considered to be one of the biggest strengths of the method proposed in this paper. Another discussed benefit is being able to use the same algorithms, without modifications, in different levels of the same game as they are being created. Because of the general goals, the heuristics would not need to be updated to fit the specifications of a new level. It would allow checking if it fulfils the expectations almost immediately after being included in the game. Finally, as Section II-A states, there are many types of GVGP algorithms, which provide a wide range of options depending on the technology, characteristics and implementation of the game considered. An example would be the availability of a forward model or not.

It is worth mentioning that the use of General AI does not mean that some game-specific tweaks should not be added in order to improve the heuristics performance, as long as the main general goals are not changed. The strength of the proposed approach is based on the generality, flexibility and robustness concepts of general AI, which can adapt to significant changes in the game design.

This paper provides an initial overview of General AI frameworks and automatic testing approaches in Section II, followed by the description of the proposed methodology in Section III. Section IV describes the limitations of this approach, and conclusions and possible extensions are detailed in Section V.

II. Background

A. General Video Game Playing

General Video Game Playing (GVGP) aims to develop algorithms capable of playing video games without having prior knowledge about them, with mere access to the state of the game and the available actions [1]. The interest in the research in this area has grown in the past years.

The most common techniques to tackle the problem, with different implementations, are the use of *Reinforcement Learning* (RL), *Tree Search* and *Evolutionary* algorithms. In order to provide the resources to be able to develop and study these different approaches, a series of frameworks have been created. The main open-source frameworks available to the research community are the Arcade Learning Environment (ALE) [2], the General Video Game AI Framework (GVGAI) [3], OpenAI Gym [4] and Project Malmo [5], among others. These frameworks have in common the desire of encouraging the study of the Artificial General Intelligence (AGI) but have different characteristics, leading to the creation of numerous types of algorithms, which keep growing and being improved.

One of the first general frameworks is **ALE**, a testbed for comparing and evaluating planning and learning algorithms, providing an interface to Atari 2600 games, like *Space Invaders* and *Ms. Pac-Man* [2]. Most of the research carried out using this framework has focused on *Reinforcement Learning*, as Mnih et al. work [6]. They showed how using Deep Q-Networks (DQN) receiving only a screenshot and the game score as inputs through a set of 49 games, it was possible to achieve a level of performance comparable to a professional human player in many of the games tested. The environment also allows the use of planning algorithms, but the research in this area using this framework is very rare. A reason for this could be the complexity in finding heuristics general enough to have a good performance overall the games [7].

The **GVGAI Framework** has been used for the ongoing GVGAI Competition since it was first run in 2014 [3]. The games used to benchmark the algorithms are described in the Video Game Description Language (VGDL) [8], originally implemented in *Python* by Tom Schaul [9]. It allows the implementation of single and two-player 2D Arcade games.

Providing a forward model that allows agents to for see the possible states originated by taking any of the available actions, the initial competition encouraged the submission of single player planning algorithms. Over the years, the competition has been expanded to cover other novel areas of general AI, releasing two-player [10], level [11] and rule generation [12], and a learning track [13], which removes the availability of the forward model and provides a screen capture to promote research on other learning algorithms.

The algorithms developed to work in this framework and submitted to the competition are very assorted: several variations of Monte Carlo Tree Search (MCTS) [14], including its Open-Loop variations (OLMCTS) that works better in stochastic environments, and evolutionary algorithms, like Rolling Horizon Evolutionary Algorithm (RHEA) [15] and Random Search (RS). Two algorithms that, so far, have shown best overall performance, and therefore, have been claimed winners of some of the competitions, are Adrien Couëtoux's Open Loop Expectimax Tree Search (OLETS) [3] and Joppen et al.'s YOLOBOT [16]. An insight of the framework, its wide use, algorithms implemented and a complete list of the winner algorithms per year can be found in a recent survey [17].

Another popular toolkit is **OpenAI Gym**, oriented to test *Learning* approaches providing a common interface for a collection of environments based in pre-existent RL benchmarks [4]. It includes, among others, *Atari*, which uses ALE. This collection grows over time.

In contrast with other frameworks, it provides abstraction

for the environment instead of the agent and does not provide a hidden test set. *OpenAI* encourages peer review and collaboration by sharing the code and a description of the approach followed, instead of arranging a competition between the algorithms. The framework focuses on both the performance of an algorithm and the amount of time it takes to learn. It keeps a strict version number scheme every time a change is made in an environment.

Finally, it is also worth mentioning **Project Malmo**, a platform built on top of *Minecraft*, designed to support AGI research in reinforcement learning, planning, multi-agent systems, robotics and computer vision [5]. In this framework agents are exposed to a 3D environment with complex dynamics that provides the experimenters the tool to set complicated tasks. In 2017, the *Malmo Collaborative AI Challenge*[1] was run using this environment to encourage the research in collaborative Artificial Intelligence. The goal of the competition was creating agents capable of learning to achieve high scores when working with a range of both artificial and human partners.

The existence of these frameworks (and others) implies that there is a huge variety of algorithms with different characteristics, weakness and strengths, at everyone's disposal. There is a large and active community of researchers currently working on improving those general algorithms and creating new ones.

B. Automatic Testing and AI Assisted Game Design

When creating a game or adding new levels to it, they should go through a testing process to make sure their characteristics are aligned with the expectations and that there are no bugs affecting the gameplay. The Quality Assurance (QA) of the games are usually carried out by either members of the development team or game testers, being a manual task. Automated testing aims to facilitate the QA by using automatic processes. They are generally game-dependent, which is a big limitation as they should be implemented specifically for the game under development. This paper proposes a methodology general enough to be easily adaptable to any game, without having to invest a lot of time in game-specific setting ups.

Intrinsic motivation refers to a series of physical needs that motivates a certain behaviour without the direct existence of a external reward like score. In [18] S. Roohi et al. state how the emerging field of simulated-based game testing looks promising. Being able to use simulated agents instead of human players to provide feedback during the game design process could increase the speed and reduce the costs. These authors review the existing literature in intrinsic motivation in player modeling, focusing in simulation-based game testing. They come to the conclusion that its application to automatic testing is sparse and hope that their work would provide new ideas to the research community. This paper takes this inspiration and suggests using a series of agents (not necessarily just intrinsically motivated) for simulation-based game testing. These agents have a range of goals to have available distinct behaviours that can provide diverse information to the game designer.

In [19] Holmgård et al. present an approach for automated playtesting using *archetypal generative player models* called *Procedural Personas*. In this work they use a variation of Monte Carlo Tree Search (MCTS) where its Upper Confidence Bound (UCB) equation is adapted by evolution to be able to create players with differentiated goals and behaviours. They create four *Procedural Personas* (*Runner*, *Monster Killer*, *Treasure Collector* and *Completionist*) to play their test game *Minidungeons 2* focused in four different primary objectives. These are, in order, reaching the exit, killing enemies, collecting items and consuming any game object that is possible to be collected or killed. All of them were also given a secondary goal: reaching the exit as quick as possible for the *Runner* or just being able to reach it, for the rest of them.

The authors ran a series of experiments in order to compare the performance between the evolved personas and the baseline algorithms and to test how different they interact with the environment. In the results they show how all these evolved personas perform better than baseline UCB1 ones regarding computational time required to reach the exit and, therefore, finishing the levels. They notice how these evolved personas, even when all managed to reach the end of the level, had differentiated play-styles depending on their primary goal and realising that they were also affected by the patterns of the level played. They discuss how these personas with differentiated behaviours can be used for evaluation of levels, either providing feedback to a human game designer, or assisting the improvement of automatic generated levels, driven by their distinct play-traces. Because they are oriented to provide useful feedback to the game or level designer, they argue how the utility functions should be defined by them in order to fulfil the priorities of the design.

The methodology presented in this paper is inspired by this work but aims to have a more general and portable approach, capable of being applied to several different games without having to design specific types or utility functions to fit the game under consideration. We believe that extending the idea to use general agents, developed with general goals that can be applied to several different games can provide important advantages. Having a *team* of pre-defined types with general goals and approaches gives the designer the chance to choose which agents fit the characteristics of the game.

S. Nielsen et al. used the *Relative Algorithm Performance Profile* (RAPP) approach to estimate the quality of a certain game based on the performance of general agents [20]. They compared the performance between known algorithms in a range of hand-designed, mutated and random generated VGDL games. Their premise argued that a game that has a high skill differentiation is likely to be a good one. In spite of the results backing their hypothesis, complexity does not necessary infer quality and, by its own, this approach is not able to provide further information about the game under evaluation.

[1] https://www.microsoft.com/en-us/research/academic-program/collaborative-ai-challenge/

The evaluation that this paper proposes is based on the performance of the general agents, but using a different approach. Although RAPP can be used to make the methodology stronger, there are some important differences to highlight. Firstly, in [20] they used seven different general algorithms, including an *Explorer*, but they only based their performance in the winning rate and difference of score. In our case, the agents used, if with distinct goals, are not compared between them. However, they provide particular information about their own play-through and performance based on each of their objectives. Also, our methodology is expected to provide deeper feedback and richer information than a mere state of the good/bad quality of the game.

T. Machado et al. built the Computationally Intelligent Collaborative EnviROnment (*Cicero*) [21], which is a general-purpose AI-assisted tool for 2D tile-based game design. It was built on top of the GVGAI Framework, assisting the creation and development of VGDL games. It provides a game editing mechanism to add the sprites and rules that conform the game and includes a *mechanics recommender*. It suggests certain sprites and rules based on the ones added. It also grants an automatic testing feature that shows game rule statistics in real time and a level visualization. To test the game, it is possible to either play it manually or select one of the general agents available in the framework. Running the game with a general agent provides heat-maps of the player and the NPCs. Also, during the automatic gameplay, a list with the different rules and the stats for each of the interactions of the game are shown. Cicero was expanded to include *SheekWhence*, a *retrospective analysis tool for gameplay session* [22]. This extension includes recording of the gameplay to analyze the sequence of events, being able to go forward and backwards in the session. Their limitations are that it is very oriented to the VGDL language and the GVGAI framework, impeding its application to a wider range of games. Also, although when general algorithms are used for evaluation, it is not taking advantage of a lot of information that could be extracted from their play-through to provide richer information to the designers. Also, these agents' ultimate goal is winning, so their behaviour is not as assorted as including a *team* with different objectives.

III. General AI Team to Assist Game Design

A. Overview

This paper proposes a methodology capable of assisting the design and testing process during the development of a game. The evaluation uses a *team*; a series of General AI algorithms with differentiated goals (Section III-B). Each one of the agents play and behave differently within the same game. Extracting certain information from their play-through, and having the right tools to interpret it, can help the designer to check if they are in the right path, or if a change needs to be carried out to get aligned with the expected outcome.

The methodology needs a series of entities to work. First, the **Designer**, final user and responsible of the game. They want to make sure that the content (game or level) under development fits the expectations of the design without errors. They would provide the part of the **Game** and set up the processes required for its evaluation.

Three types of outputs are generated after the methodology is applied to evaluate the game, two of which are reports. First, the **Target Reports** provide the results of evaluating the game based on each of the agents' behaviours, compared to expected targets. These targets are set before the tests are run. Next, the **Visual Reports** provide visual information about the evolution of the information retrieved by each of the agents during their play-through. This information is presented in a series of graphs. Last, the **Logging System** records the logs resulting from the algorithms' play-through to provide support for testing and debugging (Section III-D).

The main steps of this methodology are as follows:

1) ***Setting up the team****:* There is a range of general agents of different types and with a range of skills. The designer is able to choose, and optimize, the ones they believe fitting the characteristics of the game and design expectations. They can also set an expected performance for each of the agents.

2) ***Integrating the game****:* The methodology is focused on being portable and flexible enough to be used with different games. However, it is needed to set it up to be able to run the algorithms, extract information from their play-through and record the metrics in the Logging System.

3) ***Evaluation process****:* The types of agents and skills chosen by the designer are run a certain number of times in the game provided. Each play-through of each of the agents logs a series of metrics and errors triggered to be able to have detailed information about what happened. Also, two types of reports are generated.

4) ***Generate reports****:* The information provided by each of the agents (Section III-B) is processed to generate the two different type of reports presented in Section III-C.

B. The Team

This paper proposes using a series of general algorithms with differentiated goals, capable of playing a game focusing on their specific objectives. Differentiating the heuristics in General Video Game Playing was introduced by Guerrero-Romero et al. in [23], and some of the members of the suggested *team* have been inspired by their work.

Also, it has been influenced by R. Bartle's *player types* [24]. This work presented four approaches to play MUD games, showing how the same game can be played in different ways depending on the motivation of the players, leading to distinct behaviours. Even when this work is specific for MUDs, it has been a reference to find types applicable to different games.

This is a non-exclusive list of members proposed and the targets for each of them:

1) ***Winner****:* Focused on winning the game; maximizing the score when a winning state is not immediately reachable. The information provided by an agent of this type can be the number of wins, game ticks to victory, or strategy followed when more than one option is available.

*2) **Map Explorer**:* Focused on covering the reachable areas as much as possible. The information provided by an agent of this type can be the amount of different positions of the map visited, total percentage of the map explored, or game ticks required to finish the exploration.

*3) **Novelty Explorer**:* An alternative for an exploratory agent is considering states instead of positions; going through as many different game states as possible and providing this number as result. It is related to the *Novelty* appraisal common in intrinsically motivated agents in AI [18]. Also, the inspiration for this kind of agent comes from the work done by Bellemare et al. in [25]. They proposed connecting the information gained through the learning process and count-based exploration, which guides agents' behaviour to reduce uncertainty. This approach is designed to explore the environments in a more practical and efficient way.

*4) **Curious**:* Focused on interacting as much as possible with the elements of the game, always prioritizing those that has not been interacted with before. The information provided can be the number of elements interacted with, actions triggered when they happened, or game ticks required to interact with the different elements of the game.

*5) **Competence seeker**:* Based on the model of *empowerment* of intrinsic agents, which denotes the degree of control the agent feels having over the environment [18]. It is related to the amount of information the agent is capable of collecting when a series of actions are performed. It can provide information about the level of expertise gained during its play-through.

*6) **Scorer**:* Focused on maximizing the score, without paying attention to the chances of winning the game. The information provided can be the number of points obtained, or game ticks required to maximize the score.

*7) **Collector**:* Focused on collecting the items available in the game. The information provided can be the number of items collected, counts per type of item, or game ticks required to collect the different items present in the game.

*8) **Killer**:* Focused on removing from the game as much Non-Playable Characters (NPCs) as possible. The information provided by an agent of this type can be the number of NPCs killed, number of times killed by an NPC, counts per type of NPC encountered, or game ticks required to kill all the enemies present in the game.

*9) **Scholar**:* Focused on learning the outcome of the actions available, taking as much knowledge about the game as possible. The information provided by an agent of this type is the percentage of accuracy of the knowledge gained during the duration of the gameplay. As it is needed to have concrete information about the rules and outcomes of the interactions with the game in order to be able to check the quality of the predictions, the generality of this type of agent is improbable. However, an agent with this kind of objective is an interesting addition to the team as it can be used to detect anomalies during gameplay. There is a high chance that an agent focused on this kind of task finds unexpected rules or bugs on the existent ones that should be fixed.

Having a *team* with differentiated agents provides a flexibility that would not be possible using just a specific one. The general objectives presented in this section cover different aspects, that could be present, or not, in a game. The designer can accommodate the methodology to adapt their intentions and needs by choosing the agents to include to fit to the characteristics of the game. Also, having several differentiated behaviours provides a lot of information that can help to decide if the version of the game, or level, under evaluation is as expected, or it should be adjusted.

C. Assisting Game Design

As indicated above, two different type of reports are provided in order to check the validity of the design of the game.

The first kind is **Performance-target based reports**, thought to evaluate the game based on the expected performances in the behaviour of each of the agents. In the experiments carried out in the GVGAI Framework for the work presented in [23], results for same heuristics algorithms showed a clear distinction depending on the type of game. A clear example is the results obtained using the Exploration Maximization Heuristic (EMH). Algorithms using this heuristic were focused on maximizing the exploration of the level. Their performance was calculated by obtaining the percentage of the level explored dividing the number of different positions visited, by the total. In completely accessible maps in games like *Butterflies*, the agents using the EMH ended up with an average percentage of performance higher than 80% in most of the cases. Whereas, in games with large maps, or where a series of steps were needed to unlock the access to the different areas, like *Roguelike*, any of those agents got an average higher than 45%[2]. The presence of these differences on performance can be used in designer's benefit, providing an estimation of performance that agents should achieve depending on the type of game designed.

Before running the *team*, the designer would be able to choose the agents considered appropriate for the game under evaluation, and to set an estimated desired percentage of performance for each of them. After a series of runs carried out by each of the agents, an error for the expected values would be obtained and returned, to inform if there is an agreement between the ideal values and the reported ones. For example, if a designer plans a game to be easily accessible but difficult to win, they would assign a high desired value to the *Map Explorer* and a low value to the *Winner*. After several runs of the agents, the errors would be reported by calculating the difference between the targets and the real values. With this information, they would be able to check if the design matches the expectations, or how distant the values are.

A second type of information retrieved is captured *in game*, which can be easy interpreted by the designer in the form of **Visual reports**. These are meant to provide graphs that analyse how the information retrieved by the agents evolves

[2]This percentage was not explicitely mentioned in the paper, but it has been taken from the same results obtained in those experiments

during play-through. Another way to extract information from the agents would be generating a series of graphs with the information provided by each of them (number of different positions or states visited, number of elements interacted with, etc) by point in time. Analysing the shape and evolution of the plotted values, the designer should be able to extract and conclude interesting information about their game. A continuous trend means that the agent is capable of getting information without much impediments, improving uniformly. If, otherwise, the growth is stuck for a period of time, either there is nothing more to be discovered, all targets for the agent were reached; or there was an obstacle (or a series of obstacles) blocking the agent to achieve its goals. Let's take a possible play-through graph obtained for the *Map Explorer* as an example. It could show an uniform growth until certain point, keep still for a while to end up increasing uniformly again. This shape could be interpreted as follows: the map of the game is divided in two areas and an action from the player is required to progress in the game.

This method could also be used to analyse the distribution of different elements of the game. In the example of the *Collector*, the growth of the graph would show peaks in those areas where there are several items to collect.

D. Logging System

The *Logging System* keeps track of the information resulting from running each of the agents: position by time, actions taken, elements of the game interacted with, responses triggered, etc. These logs can help to detect anomalies and broken states of the game.

M. Nelson proposed seven strategies to extract information from the game [26]. In [27], V. Volz et al. gather a list of measures envisioned to be included in the GVGAI framework in order to be able to extract information from the game play. They differentiate between direct and indirect loggable measures, agent-based measures and interpreted measures. Because of the generality of the framework this list was collected for, it could be taken as a reference to use in this methodology. Having a *team* of agents instead of a unique one, can cover much more game states, being able to trigger errors that would be difficult to catch otherwise.

Another application for these logs is verifying how the agents with distinct goals behave differently, checking which actions the agents would take in similar states. Because of their differentiated motivations, it should be possible to analyse the reasoning behind their decisions.

E. Variations

Same algorithms with different parameters have different strengths. The *team* can include several versions of the algorithms with same objectives, but different levels of mastery, based on those parameters. There are several methods to be able to arrange a series algorithms by measuring their performance, used for several competitions and online rankings. The most distinguished are the *Bayes Elo system* [28], *Glicko* [29] and *TrueSkill* [30], the skill rating system used in Xbox Live and recently extended.

The designer can be given the opportunity to choose between differentiated skilled agents and even perform Relative Algorithm Performance Profiles checks (II-B). This enlargement allows an even bigger range of choices and richer information available.

IV. Limitations

The approach proposed in this paper has a clear strength, but there are a series of limitations that should be taken into consideration, a list of which is presented in this section.

The time needed to perform the evaluations should be taken into consideration in order to arrange enough time to analyse the reports and to plan the actions to be taken as a result. More complex the game, more time the evaluation would take, as the agents would need more time to run and finish the play-through in order to provide feedback. A feasible solution would be presenting the game split in stages or levels; analysing small chunks each time. Also, the complexity of the game affects the performance of the algorithms as General AI has some limitations solving complicated environments, which should be kept in mind.

The methodology presented here would obviously be strengthened if these limitations would not exist or should they be minimized. Thus, it is also an aim of this paper to motivate and encourage research on these areas:

A. Reinforcement Learning

One of the limitations when working with *Reinforcement Learning* (RL) algorithms, is that they need off-line training and their performance depends on the size and intricacy of the system. RL algorithms must explore the environment, having to decide between exploitation and exploration as it learns which actions lead to rewards [31]. The more complex the game is, the more it struggles as more the rewards are delayed in time.

There has been clear advances in RL methods, showing good performance in well defined problems. An example is *AlphaGo* being able to master *Go* [32]. Although the rules of the game are simple, the game has certain characteristics that impeded mastering it with AI for a long time: deep games, large branching factor and, above all, lack of a good state evaluation function. Other examples showing the progress in RL, applied to videogames, come from the work done by Mnih. et al [6] (see Section II) and the research done in the *VizDoom* platform [33].

Despite this progress, RL has not yet provided world winning approaches for more complex games, such as *Starcraft* [34]. Not only games like this require multiple levels of abstraction and reasoning, but also include many real-world features that limit the application of these techniques. Examples are, in this and other games, the presence of a continuous state and action space, stochasticity, partial observability (fog of war is present in multiple strategy games) and multi-agent systems.

B. Planning Algorithms

These algorithms do not require off-line training (setting aside the parameters optimization that will be discussed in the next section), and therefore have a quick set-up. However, they require a forward model in order to be able to simulate future possible states to choose the best action available. Therefore, there exists the challenge of having to create a forward model from scratch to include it in the game if it didn't exist before; or to work with abstract and/or not precise forward models available.

Moreover, the number of roll-outs (that depends on processing time and resources) have a big impact on the behaviour and performance, as the more simulations they are allowed to see, more information they get about the future. In [35], the authors compare the differences in performance in the Physical Travelling Salesman Problem when a budget of 40ms or 80ms is provided to MCTS, RHEA and RS. It has an impact in the number of roll-outs available per turn for the MCTS and the number of individuals for the Genetic Algorithms (GA). In the General AI scope, M. Nelson [36] ran a series of experiments through 62 games that conform the GVGAI Framework. The goal was checking how the performance of the MCTS is affected when varying the time budget provided to return an action, which influences the number of roll-outs it gets to take.

C. Parameter Optimization

General AI algorithms use a series of parameters that have a big impact on their performance and behaviour and, in most of the cases, need to be optimized. As mentioned in the previous section, if the number of roll-outs available to the planning algorithm is modified, the number of predictions will be reduced or extended and, therefore, the information available to take a decision will be affected, influencing the results. In evolutionary algorithms, the size of the population has an impact on the performance. Gaina et al. proved the veracity of this for the RHEA. They compared how the winning rate of the general algorithm was influenced by the size of the population and individual length [15], so the optimization of the parameters is important indeed.

Optimizing the parameters to the game under evaluation could take time. If not enough time is allowed, their expected performance could drop, which would end up on providing misleading reports. To provide enough time to reach a certain level of performance, optimization has usually been done off-line. However, there has been some recent progress in this area and an online adaptive parameter tuning mechanism for MCTS has been implemented in GVGP, showing promising results [37].

The N-Tuple Bandit Evolutionary Algorithm (NTBEA) shows ways to mitigate some of the limitations presented by this parameter optimization. Lucas et al. describes the NTBEA as a simple, informative and efficient model capable to be applied to numerous of optimisation-related problems [38]. In the referenced work they show how to apply this approach to optimise the parameters of RHEA. In [39] Kunanusont et al. use this algorithm to evolve the game parameters of *Space Battle*, affecting its design. They argue how the results obtained in the experiments carried out show how NTBEA could be used for AI-Assisted Game Design.

The *team* should be well-tuned to allow the agents to recognise and carry out the actions expected to reach their goals, in order to obtain proper results that fit the expectations and interpret the feedback accordingly.

D. The Challenge of General AI

Developing algorithms capable of working through different games is a challenging task as it is not possible to use game-specific information to guide the algorithms. Because of the difficulties of the problem, several approaches have been created and are being investigated in order to tackle it.

General AI is an ongoing research. Even considering the latest improvements, it's still not good enough to generalise properly to every kind of game. This is clearly shown by checking the results of the *GVGAI Competition*[3]. Even when the agents perform well in some games, there are games with a very low percentage of success; and any algorithm manages to perform uniformly good through all of them.

Furthermore, general algorithms can be applied to several areas in games: from one player simple games to multi-player collaborative games, where they need to work together to achieve a common objective. The variety on the problems to tackle increases the complexity of the generalisation, leading to an existent work in progress for a very complex task.

V. Conclusion

This paper proposes a new methodology using General AI for assisting game design and testing and presents its features. A series of differentiated goals to be applied to the general agents are presented. Having agents focusing in targets that go beyond simply winning the game leads to distinct gameplays and to extract information driven by those. The two type of reports and logging system generated by the methodology can help the designer to check if the expectations they have about their game under evaluation are fulfilled. Because of the independence of the rules given by the generality of the AI, this approach allows an early integration in a game under development without requiring mayor modifications when the game is extended or modified.

This proposed methodology is rooted in previous work on the field of general game AI, automatic play testing and AI-assisted game design. It takes into account the needs of the games industry for efficient and accurate game testing and highlight interesting areas of future research. Several extensions in the methodology are possible. As it has been presented, it evaluates a single instance of a game with a team of agents to meet certain predefined criteria. However, this can be seen as a single evaluation in a learning process that automatically tunes and tweaks game content. For instance, a possibility would be to employ a Multi-Objective Evolutionary Algorithm, where each agent in the team represents a different objective to fulfil.

[3]http://www.gvgai.net/

We believe that using general algorithms is the next step for automatic game design and testing in order to provide a portability non-existent in the approaches followed to date. Furthermore, introducing the option to choose between several algorithms with differentiated behaviours and skills adds flexibility to adapt the methodology to the characteristics of the game under evaluation.

ACKNOWLEDGMENT

This work was funded by the EPSRC CDT in Intelligent Games and Game Intelligence (IGGI) EP/L015846/1.

REFERENCES

[1] J. Levine, C. Bates Congdon, M. Ebner, G. Kendall, S. M. Lucas, R. Miikkulainen, T. Schaul, and T. Thompson, "General Video Game Playing," 2013.

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The Arcade Learning Environment: An Evaluation Platform for General Agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[3] D. Perez-Liebana, S. Samothrakis, J. Togelius, T. Schaul, S. M. Lucas, A. Couëtoux, J. Lee, C.-U. Lim, and T. Thompson, "The 2014 General Video Game Playing Competition," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 3, pp. 229–243, 2016.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[5] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell, "The Malmo Platform for Artificial Intelligence Experimentation." in *IJCAI*, 2016.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-Level Control Through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[7] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, and M. Bowling, "Revisiting the Arcade Learning Environment: Evaluation protocols and open problems for general agents," *Journal of Artificial Intelligence Research*, vol. 61, pp. 523–562, 2018.

[8] M. Ebner, J. Levine, S. M. Lucas, T. Schaul, T. Thompson, and J. Togelius, "Towards a Video Game Description Language," 2013.

[9] T. Schaul, "A Video Game Description Language for Model-Based or Interactive Learning," in *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013, pp. 1–8.

[10] R. D. Gaina, D. Pérez-Liébana, and S. M. Lucas, "General Video Game for 2 Players: Framework and Competition," in *Computer Science and Electronic Engineering (CEEC), 2016 8th*. IEEE, 2016, pp. 186–191.

[11] A. Khalifa, D. Perez-Liebana, S. M. Lucas, and J. Togelius, "General Video Game Level Generation," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2016, pp. 253–259.

[12] A. Khalifa, M. C. Green, D. Perez-Liebana, and J. Togelius, "General Video Game Rule Generation," in *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*. IEEE, 2017, pp. 170–177.

[13] D. Perez-Liebana, S. Samothrakis, J. Togelius, S. M. Lucas, and T. Schaul, "General video game ai: Competition, challenges and opportunities," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 4335–4337.

[14] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[15] R. D. Gaina, J. Liu, S. M. Lucas, and D. Pérez-Liébana, "Analysis of Vanilla Rolling Horizon Evolution Parameters in General Video Game Playing," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2017, pp. 418–434.

[16] T. Joppen, M. U. Moneke, N. Schröder, C. Wirth, and J. Fürnkranz, "Informed Hybrid Game Tree Search for General Video Game Playing," *IEEE Transactions on Games*, vol. 10, no. 1, pp. 78–90, 2018.

[17] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, "General Video Game AI: a Multi-Track Framework for Evaluating Agents, Games and Content Generation Algorithms," *arXiv preprint arXiv:1802.10363*, 2018.

[18] S. Roohi, J. Takatalo, C. Guckelsberger, and P. Hämäläinen, "Review of Intrinsic Motivation in Simulation-based Game Testing," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 2018, p. 347.

[19] C. Holmgård, M. C. Green, A. Liapis, and J. Togelius, "Automated Playtesting with Procedural Personas through MCTS with Evolved Heuristics," *arXiv preprint arXiv:1802.06881*, 2018.

[20] T. S. Nielsen, G. A. Barros, J. Togelius, and M. J. Nelson, "General Video Game Evaluation Using Relative Algorithm Performance Profiles," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2015, pp. 369–380.

[21] T. Machado, A. Nealen, and J. Togelius, "CICERO: Computationally Intelligent Collaborative EnviROnment for Game and Level Design," in *3rd workshop on Computational Creativity and Games (CCGW) at the 8th International Conference on Computational Creativity (ICCC'17)*.

[22] ——, "Seekwhence a retrospective analysis tool for general game design," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*. ACM, 2017, p. 4.

[23] C. Guerrero-Romero, A. Louis, and D. Perez-Liebana, "Beyond Playing to Win: Diversifying Heuristics for GVGAI," in *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*, 2017, pp. 118–125.

[24] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit MUDs," *Journal of MUD research*, vol. 1, no. 1, p. 19, 1996.

[25] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Advances in Neural Information Processing Systems*, 2016, pp. 1471–1479.

[26] M. J. Nelson, "Game Metrics Without Players: Strategies for Understanding Game Artifacts." in *Artificial Intelligence in the Game Design Process*, 2011.

[27] V. Volz, D. Ashlock, and S. Colton, "Gameplay Evaluation Measures," *Artificial and Computational Intelligence in Games: AI-Driven Game Design (Dagstuhl Seminar 17471)*, p. 122, 2018.

[28] A. E. Elo, *The Rating of Chessplayers, Past and Present*. Arco Pub., 1978.

[29] M. E. Glickman, "Example of the Glicko-2 System," *Boston University*, 2012.

[30] T. Minka, R. Cleven, and Y. Zaykov, "TrueSkill 2: An Improved Bayesian Skill Rating System," Tech. Rep., March 2018.

[31] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[32] 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, 2016.

[33] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A Doom-Based AI Research Platform for Visual Reinforcement Learning," in *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 2016, pp. 1–8.

[34] S. Ontanón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, "A Survey of Real-Time Strategy Game AI Research and Competition in Starcraft," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 5, no. 4, pp. 293–311, 2013.

[35] D. Perez, S. Samothrakis, S. Lucas, and P. Rohlfshagen, "Rolling Horizon Evolution Versus Tree Search for Navigation in Single-Player Real-Time Games," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 351–358.

[36] M. J. Nelson, "Investigating Vanilla MCTS Scaling on the GVG-AI Game Corpus," in *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 2016, pp. 1–7.

[37] C. F. Sironi, J. Liu, D. Perez-Liebana, R. D. Gaina, I. Bravi, S. M. Lucas, and M. H. Winands, "Self-adaptive MCTS for General Video Game Playing," in *International Conference on the Applications of Evolutionary Computation*. Springer, 2018, pp. 358–375.

[38] S. M. Lucas, J. Liu, and D. Perez-Liebana, "The N-Tuple Bandit Evolutionary Algorithm for Game Agent Optimisation," *arXiv preprint arXiv:1802.05991*, 2018.

[39] K. Kunanusont, R. D. Gaina, J. Liu, D. Perez-Liebana, and S. M. Lucas, "The N-Tuple Bandit Evolutionary Algorithm for Automatic Game Improvement," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 2201–2208.