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

Cristina Guerrero-Romero, Simon M. Lucas and Diego Perez-Liebana
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, UK
{c.guerreroromero, simon.lucas, diego.perez}@qmul.ac.uk

***Abstract*—General Video Game Playing (GVGP) has become a popular line of research in the past years, leading to 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. It introduces a methodology consisting of 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 that play the same game simultaneously 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 the play-through. Having this methodology available to designers can help them decide if the game or level under analysis fits the initial expectations. Including a *Logging System* can also be used to detect anomalies while the development is still at an early stage. We believe this approach allows the flexibility and portability to be easily applied to games with different characteristics.**

***Index Terms*—methodology, General Artificial Intelligence, automatic testing, game design, team of agents**

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 and need to be tested quickly and efficiently. While using human play-testing is a broad practice, there is no denying 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 cannot be matched by game-specific ones.

This paper proposes a methodology consisting of a *team* of Artificial General Intelligence (AGI) agents with differentiated goals to aid the design and testing processes during the development of a game. Each of the agents forming the team has its own objective (winning, exploring, collecting items, killing NPCs, etc.) and skill level. This set up provides a flexibility that would not be possible using just one. The designer can choose the agents to run and set their expected targets of performance. Each of the specialists selected plays the game under evaluation focusing on its own goal and, as a result, a Logging System and two types of 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 the play-through.

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

*1) **Rules**:* The base of every game. Making any change to the rules can trigger unexpected outcomes and affect other rules in a way the designer did not plan to. General agents are independent of the rules so they do not need to be adjusted when they change to be able to check that everything is working as it should be. It grants the possibility of carrying out 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.

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

*3) **Non-Player 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 analysed 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 proposed methodology works within the game. If using algorithms specific to the game, 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.

The use of General AI does not mean that some game-specific tweaks should not be added 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 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 recent 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 share 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 good performance over all the games [7].

The **GVGAI Framework** has been used for the ongoing GVGAI Competition since it was 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 foresee the possible states originated by taking any of the available actions, the first 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 on 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 an 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, **Project Malmo** is 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 with 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 to create agents capable of learning to achieve high scores when working with a range of both artificial and human partners.

The existence of these (and other) frameworks implies that there is a huge variety of algorithms with different characteristics, weaknesses and strengths, at everyone's disposal. There is a large and active community of researchers 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 no bugs are affecting the gameplay. The Quality Assurance (QA) of the games is usually carried out manually by members of the development team or game testers. 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 much 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 an external reward like the score. S. Roohi *et al.* state how the emerging field of simulated-based game testing looks promising [18]. Being able to use simulated agents instead of human players to provide feedback during the game design process can increase the speed and reduce the costs. These authors review the existing literature on intrinsic motivation in player modelling, focusing on 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.

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 on 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 quickly as possible for the *Runner* or just being able to reach it, for the rest of them.

The authors ran a series of experiments to compare the performance between the evolved personas and the baseline algorithms and to test how different they interact with the environment. The results show how all these evolved personas perform better than baseline UCB1 ones regarding the 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 were affected by the patterns of the level played. They discuss how these personas with differentiated behaviours can be used for level evaluation, either providing feedback to a human game designer or assisting the improvement of automatically 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 they should define the utility functions 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 significant advantages. 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. Despite the results backing their hypothesis, complexity does not necessarily infer quality and, by its own, this approach is not able to provide further information about the game under evaluation.

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 on the winning rate and difference of score. In our case, the

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

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 in the creation and development of VGDL games. It provides a game editing mechanism to add the sprites and rules that form 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 visualisation. 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 a recording of the gameplay to analyse the sequence of events, being able to go forward and backwards in the session. Its limitation is that it is very oriented to VGDL and the GVGAI framework, impeding its application to a wider range of games. Also, although general algorithms are used for evaluation, it is not taking advantage of most of the information that could be extracted from their play-through to provide richer information to the designers. Moreover, 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 plays and behaves differently within the same game. Extracting certain information from their play-through, and having the right tools to interpret it can help the designer. They can check if they are on the right path, or if a change needs to be carried out to get aligned with the expected outcome. In contrast with meta-heuristic approaches, where a heuristic is involved in deciding which type of agent is run depending on the state of the game, in our case, all the agents play the game simultaneous and independently.

The methodology needs a series of entities to work. The **Designer** is the final user and responsible for 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. Firstly, 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. Lastly, 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 can choose, and optimise, the ones they believe fit 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 focuses 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 picked by the designer are run a certain number of times in the game provided. Each of the agents' gameplay logs a series of metrics and errors triggered to be able to have detailed information about what happened.

*4) **Generating 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 C. Guerrero-Romero *et al.* in [23], and some of the members of the suggested *team* have been inspired by their work.

The inspiration also comes from R. Bartle's *player types* [24]. This work presented four approaches to play MUD games, showing how the same game can be played in various ways based 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. Recently, N. Yee has developed a player motivation profile based on data from more than 250.000 players, coming up with 6 main differentiated clusters of gaming motivations [25].

A *team* of agents focused on different tasks provides a flexibility that would not be possible using just a specific one. The general objectives presented in this section cover different aspects, which could be present, or not, in a game. The designer can accommodate the methodology to adapt their intentions and needs by including the agents to fit its characteristics. The following is a non-exclusive list of agents proposed to form the team, their targets and the information to provide:

*1) **Winner**:* Focused on winning the game; maximising 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 there is more than one option 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 number of different positions of the map visited, the 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 a 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 M. Bellemare *et al.* in [26]. The authors 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 more practically and efficiently.

*4) **Curious**:* Focused on interacting as much as possible with the elements of the game, always prioritising those that have 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) **Record breaker**:* Focused on maximising the score and solving puzzles, without paying attention to the chances of winning the game. The information provided can be the number of points obtained, puzzles solved or game ticks required for these.

*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-Player Characters (NPCs) as possible. The information provided by an agent of this type can be the number of NPCs killed, the 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) **Risk analyst**:* Focused on analysing the level of risk during the play-through and taking actions to maintain it at a certain level chosen by the game designer. A low-risk agent would tend to avoid situations where the chances of losing the game are high, like bumping into a hoard of enemies or complex areas. A high-risk agent would tend to do the opposite and jump into dangerous situations. The information provided by an agent of this type can be the risk percentage predicted at every moment, the number of deaths, NPCs killed, obstacles overcame or game ticks until losing the game.

*10) **Semantic**:* Focused on tasks related to linguistics, as coaching the dialogue of the game or making sure the narration flows and is consistent. The information provided by an agent of this type can be the estimated quality of the dialogues, the number of possible outcomes depending on the choices and the level of consistency of the narrative.

*11) **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 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.

C. Assisting Game Design

Two different types 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 focused on maximising 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

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

a designer plans a game to be easily accessible but challenging 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.

The second type of information retrieved can be easily 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 (number of different positions or states visited, number of elements interacted with, etc.) evolves during the play-through. The designer should be able to extract and conclude interesting information about their game by analysing the shape and evolution of the plotted values. A continuous trend means that the agent is capable of getting information without many impediments, improving uniformly. On the contrary, if the growth is stuck for a period of time, it either means that there is nothing more to be discovered, all targets of the agent have been reached, or that there is 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 a uniform growth to a 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 into 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, elements 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 [27]. In [28], V. Volz *et al.* gather a list of measures envisioned to be included in the GVGAI framework to extract information from the gameplay. They differentiate between agent-based, interpreted and direct and indirect loggable 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 more game states, allowing to trigger errors that would be difficult to catch otherwise.

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 existing methods to be able to arrange a series of algorithms by measuring their performance, used for several competitions and online rankings. The most distinguished ones are the *Bayes Elo system* [29], *Glicko* [30] and *TrueSkill* [31], 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.

Another possible extension can be to take into consideration the information retrieved by all agents as a whole and study the correlations between them. The designer can choose which agents' information combine to obtain greater levels of granularity.

IV. Limitations

The approach proposed in this paper has a clear strength, but there are a series of limitations, presented in this section.

The time needed to perform the evaluations should be taken into consideration to arrange enough time to analyse the reports and to plan the actions to be taken as a result. The more complex the game is, the more time the evaluation would take, as the agents would need more time to run and finish the play-through to provide feedback. A feasible solution would be presenting the game split into 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 in solving complicated environments.

The methodology presented here would obviously be strengthened if these limitations would not exist or should they be minimised. 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. They must explore the environment, having to decide between exploitation and exploration as it learns which actions lead to rewards [32]. The more complex the game is, the more they struggle as more the rewards are delayed in time.

There have been clear advances in RL methods, showing good performance in well-defined problems. An example is *AlphaGo* mastering *Go* [33]. Although the rules of the game are simple, it has certain characteristics that impeded AI to master it 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 video games, come from the work done by Mnih. *et al.* [6] (see Section II) and the research on the *VizDoom* platform [34].

Despite this progress, RL has not yet provided world winning approaches for more complex games, such as *Starcraft* [35]. 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 optimisation discussed in the next section) and therefore have a quick set-up. However, they require a forward model to be able to simulate possible future states to choose the best action available. Hence, there exists the challenge of having to create a forward model from scratch to include it in the game or working with abstract 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 [36], 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 on 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 [37] ran a series of experiments through 62 games that form 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 Optimisation

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 optimised. 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.* compared how the winning rate of the RHEA was influenced by the size of the population and individual length [15], so the optimisation of the parameters is important indeed.

Optimising 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 providing misleading reports. The optimisation has usually been done off-line to provide enough time to reach a certain level of performance. However, there has been some recent progress, and an online adaptive parameter tuning mechanism for MCTS has been implemented in GVGP, with promising results [38].

The N-Tuple Bandit Evolutionary Algorithm (NTBEA) shows ways to mitigate some of the limitations presented by this parameter optimisation. Lucas *et al.* describe the NTBEA as a simple, informative and efficient model capable of being applied to numerous optimisation-related problems [39]. In the referenced work they show how to apply this approach to optimise the parameters of RHEA. In [40] 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 any game-specific information to guide them. Because of the difficulties of the problem, several approaches have been created and are being investigated to tackle it. Thus, General AI is an ongoing research. Even considering the latest improvements, the results of the *GVGAI Competition*[3] show how it is still not good enough to generalise to every kind of game. 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 of the problems to tackle increases the complexity of the generalisation.

V. Conclusion

This paper proposes a new methodology using General AI for assisting game design and testing and explains its features. It presents a series of differentiated goals to be applied to the general agents to play the game in different ways. Having agents focusing on targets that go beyond simply winning the game leads to specialists with distinct gameplay styles to use to extract information. The two type of reports and logging system generated can help the designer to check if their game under evaluation fulfils the expectations. The information retrieved can be used to detect bugs, balance the game or tweak its parameters. 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 major modifications when it is extended or modified.

This proposed methodology is rooted in previous work on the field of general game AI, automatic playtesting and AI-assisted game design. It takes into account the needs of the games industry for efficient and accurate game testing and highlights interesting areas of future research. Several extensions in the methodology are possible, as including agents with different levels of skill, players tackling multiple objectives or adding collaborative and social-oriented profiles that can fit multi-player games. Also, it can create reports considering the objectives of the different specialists at once, combining the results obtained to analyse the information of multiple agents outputs, study their correlations and provide a greater level of granularity.

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.

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

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*. IEEE, 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] N. Yee, "The Gamer Motivation Profile: What We Learned from 250,000 Gamers," in *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*. ACM, 2016, pp. 2–2.

[26] 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.

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

[28] 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.

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

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

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.