Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

Procedural Content Improvement of Game Bosses with an Evolutionary Algorithm

Daniel Blasco, Jaime Font,
Francisca Pérez and Carlos Cetina

Received: date / Accepted: date

Abstract We present our Evolutionary Boss Improvement (EBI) approach, which receives partially complete bosses as input and generates fully equipped bosses that are complete. Additionally, the evolutionary algorithm and the new genetic operations included in EBI favor genetic improvement, which affects the initial partial content of the incomplete bosses originally provided. We evaluate our approach using Kromaia, a commercial video game released on PlayStation 4 and PC. EBI uses an evolutionary algorithm to evolve a population of bosses guided by duels between the bosses being generated and a simulated player. Our approach evaluates the quality, in terms of game experience, of both the bosses generated and those included in Kromaia using six metrics (Completion, Duration, Uncertainty, Killer Moves, Permanence, and Lead Change) from the literature. The results show that the quality of the bosses created by EBI is comparable to the quality of the original bosses that were manually created by the developers of Kromaia. However, the EBI approach reduces the time required to build the bosses from five months (of elapsed time as opposed to dedicated time) to just 100 minutes of unattended run. EBI enables developers to accelerate the creation of content, such as bosses, which is essential to ensure player engagement.

Keywords Procedural Content Generation · Procedural Content Improvement · Evolutionary Algorithms · Commercial Video Games

1 Introduction

Procedural Content Generation (PCG) is a topic that has become increasingly prominent within the Video Game community [1]. PCG is defined as the automated creation of video game content by means of algorithms [1].

SVIT Research Group, Universidad San Jorge, Zaragoza, Spain
E-mail: {dblasco, jfont, mfperez, ccetina}@usj.es

Within the Software Engineering community, Genetic Programming (GP) [2] tackles the automated generation of software. GP and PCG share the idea of generation (software and video game content, respectively). In the last decade, the Software Engineering community has experienced a surge of interest in a subfield of GP known as Genetic Improvement (GI) [3]. GI improves existing software through an automated search, evolutionary algorithms, and search-based optimization. A recent survey [4] points out the main difference between GP and GI: GP builds a working program from scratch and GI uses an existing program as the starting point. Surveys on PCG [1,5,6] do not identify works that focus on addressing content improvement.

Because the establishment of GI as a subfield of GP has brought novel and positive contributions [3], the goal of our work is to simply recognize the content improvement work in PCG and give that subfield a name: Procedural Content Improvement (PCI). We consider that PCI might be as positive for the Video Game community as GI has been for the Software Engineering community.

In this work, we focus on game boss improvement. It consists in taking partially generated bosses as the starting point and generating complete bosses which are comparable or even better than the bosses completed by human developers in terms of game experience quality, i.e., how interesting the bosses are to players. Bosses are particularly powerful enemies that the player must overcome at the end of a stage or level. In order to help video game developers when they create game bosses, we present our Evolutionary Boss Improvement (EBI) approach. The EBI approach relies on an evolutionary algorithm that is guided by a simulated duel between the boss and the player.

We evaluate our approach using a commercial video game, Kromaia. This video game has been released worldwide in both physical and digital versions for PC and PlayStation 4. To create a boss, the Kromaia development team must perform the following steps: Creative Design, Spatial Organization, Behavior Specification, and Equipment Configuration.

When we apply our EBI approach to Kromaia, it first obtains a partially created boss enemy as input (i.e., Creative Design, Spatial Organization, and Behavior Specification have already been performed) and then performs the last step, Equipment Configuration. The output obtained from our approach is a complete boss.

In the evaluation, we compare the bosses generated by our approach with the five bosses that have been manually created by the development team of Kromaia, using six different indicators of the level of quality achieved by the game. These quality measures, which are taken from the video game research literature [7], are the following: Completion, Duration, Uncertainty, Killer Moves, Permanence, and Lead Change. The results show that the bosses generated by our approach outperformed those designed by the developers.

Additionally, the EBI approach generated those bosses after running unattended for 100 minutes, which is a significant advance in terms of time. The version control system used by the company that created the video game shows

that the developers needed five months[1] to perform the last step (Equipment Configuration) before the final release of Kromaia.

Our approach helps developers to accelerate the creation of content. Content is often released as Downloadable Content (DLC), which is essential to reinforce player engagement and retention. Furthermore, content is also used to refresh existing products when they are released on new platforms.

The contribution of the present work could be summarized in this way:

- This work focuses on addressing the improvement of content, instead of software, and applies the ideas of PCG and GI to the production of complete content, taking as an input incomplete, partially generated content.
- In contrast with other previous works, our approach uses Evolutionary Computation instead of Machine Learning in order not to depend on knowledge bases, since such approaches may require datasets with thousands of examples [8], and those knowledge bases are not always available in the case of industrial products. Furthermore, even if there also exist previous works which generate content from a single sketch or draft of the complete content as a starting point, our work studies the improvement of a content which is received incomplete.
- Our work does not focus on game content such as levels, maps or sprites, which are typically represented by 2D images. In addition, the recent literature calls for works who address more diverse types of content, like characters and their skills. Our work focuses on the generation of final bosses, which are complex entities which are not internally represented as images, and belong to a type of content which has been less studied in comparison.

The paper is organized as follows. Section 2 summarizes related works. Section 3 provides the background. Section 4 presents an overview of our EBI approach. Section 5 presents the evaluation, comparing the results obtained by our approach with the content in the video game case study. Section 6 describes the threats to validity. Finally, Section 7 presents our conclusions.

2 Related Work

Several approaches focus on the generation of new content while others are more focused on the balance of existing content. Both types of approaches use similar methods for assessing and guiding the process [1] (direct fitness, simulations based on intelligent agents, or interactive evaluation with real users). Next, we describe the works belonging to these two categories.

2.1 Generation of new content

Our work leverages search to improve a specific type of non-playable character (NPC): the game boss. The previous surveys [1,5] that cover search-based

[1] This includes testing with real players.

procedural content generation are about ten years old. To cover the years from these surveys, we ran a new search using the following query on Scopus: *TITLE-ABS-KEY ((pcg OR "automatic generation" OR procedural) AND (videogame OR game) AND ("search-based" OR evolutionary OR genetic OR "local search" OR "tabu search" OR "Monte Carlo Tree Search" OR mcts))*. The query returned 237 works. After manual inspection, we identified two works that tackle NPC and consequently are relevant to our work. We considered only those works that tackle NPC building, excluding works that are related to NPC but do not build NPCs (e.g., NPC placement [9] was excluded).

We classified the identified works following the distinction between Procedural Content Generation (PCG) and Procedural content Improvement (PCI) introduced in Section 1: PCG builds content from scratch and PCI uses existing content as the starting point. Two of the identified works belong to PCG as we show below.

Siqueira and Gadelha [10] tackle the generation of NPCs for massive multiplayer online real-time strategy games. They focus on generating heroes, which are the NPCs that manage soldiers in battle formations. Even though their approach takes as input a team of heroes for the defender side, it evolves the heroes of the attacker side. The latter are generated randomly according to a uniform distribution. In other words, the input heroes are not used as a starting point for creating heroes; the input heroes are used to assess the battle performance of the created heroes.

Ashley et al. [11] tackle the mating facet of NPCs in the context of artificial life. More specifically, they focus on a wolf-sheep predation model where they encode the mating partner preferences of each NPC. Their main idea is that a more effective mate selection increases the extinction time of the population. Their results show that NPCs evolve to favor mates who have survival traits. The implementation of their proposed approach in a video game remains as future work. In their work, NPCs are randomly initialized, that is, there is no other NPC used as a starting point.

In contrast, we tackle a different subtype of NPC (game boss) and we follow a PCI approach in which we take partially generated bosses as starting points. Our PCI approach enables developers to significantly accelerate the creation of NPCs (i.e., game bosses).

Even if a systematic literature review of PCI work is out of the scope of this study (actually, it is our future work), we also identify works that tackle the creation of shooter maps in the results of our search query. We focus on this type of content because the our video game case study is commonly described as a shooter game. Four of the identified works qualify as PCG, whereas one qualifies as PCI. Below, we present these works and why they belong to each category.

The work of Cardamone et al. [12] is the first to generate playable maps for an FPS (First Person Shooter). The authors propose four different representations for the levels and use the average fighting time of artificial agent simulations as the fitness function. Then, Lanzi et al. [13] go one step further

by applying a similar approach to evolve shooter maps for match balancing. In other words, while Cardamone et al. aimed to evolve interesting maps (i.e., maps that allow the emergence of interesting game dynamics), Lanzi et al. focus on evolving a map that can improve the match balancing. Loiacono et al. [14][15] were the first to leverage multi-objective evolution for generating shooter maps. All of these works used the same encoding proposed by Cardamone et al. [12], the same static simulation through bots to guide the search, and the same case study (Cube 2). Furthermore, neither of these works reported that they use existing shooter maps as the starting point.

The work of Olsted et al. [16] also tackles the generation of shooter maps using the encoding by Cardamone et al. [12] and the Cube 2 case study. The novelty of this work is that it allows a group of human players to interactively and collectively evolve the shooter maps through voting. The human players guide the evolutionary search towards the map content that they prefer. The approach does not build shooter maps from scratch; their approach builds shooter maps from existing shooter maps that are considered to meet a minimum level of quality, thus preventing human players from receiving particularly bad maps.

It is worth mentioning that all of the above works on shooter maps use a significantly simplified version of Cube 2 maps. This might favor the use of PCG approaches, whereas tackling Cube 2 maps which are not simplified might benefit from PCI approaches. In the case of our work (which tackles game bosses instead of shooter maps), we did not simplify the content in any sense, and PCI enabled us to achieve a significant reduction in development time. Finally, in the manner of interactive approach of Olsted et al. [16], other interactive approaches for other types of content might also use PCI to avoid presenting content to human players if it is not good enough to be considered by them.

In our previous work [17], we tackled the PCG of game bosses of Kromaia. To do this, we leveraged the ideas of Model-Driven Engineering [18], which include the genetic operation of model repair and the use of the model interpreter to guide the evolution of bosses. In this work, we focus on improvement instead of generation from scratch. This work shows that improvement is beneficial in accelerating video game development. In addition, we achieve the positive results of improvement without the model repair operation and the model interpreter. This means that improvement can be used in video games for which the repair operation and the interpreter used as fitness are not available. Improvement allows developers to keep control of some steps of the creation process while they delegate other steps to automation (improvement). The developers of Kromaia acknowledge that some content requires that they keep some level of control, while other content may be completely generated from scratch. This suggests that PCG and PCI approaches can complement each other. In another previous work, we also used Kromaia as a case study for requirement traceability [19]. Traceability is important for video game developers since they are often asked to show how functionality has been implemented. For instance, Nintendo might ask developers to implement game

saves in a specific way. Even though our previous work also uses Kromaia as a case study, the goals are completely different: traceability and improvement, respectively.

In addition and, in the context of PCG, there is an increasing interest in the application of Machine Learning (ML) and, specifically, Deep Learning (DL) to PCG in video games, as shown by a recent work which explores the evolution of this field of knowledge in the last years [20]. The paper surveys the techniques applied to content generation, and discusses both the limitations of the methods used and potential future research directions. For instance, this work exposes how systems which are autonomous, or merely receive initial parameters from human subjects, have difficulties to create quality content, and, therefore, mixed initiative generation [21], which include initial drafts or specifications given by humans for design assistance purposes [22], is gaining acceptance [20]. Such techniques are mainly used for the generation of platform game stages [23], and maps or 2D stages based on tiles [24][25] [20]. Our work uses an approach which does not receive a draft or sketch of the content generated, but incomplete content, resulting from previous design stages which are not directly related with the step that is necessary to complete the final bosses (Equipment Configuration).

The same survey shows how DL/ML works need datasets with thousands of examples [8], which is something that could be unavailable, for development teams who lack extensive knowledge databases. In the case of the company responsible for the video game case study, Kromaia, it was their first title and, even considering the possibility of generating content for a sequel, the original, finished video game includes few examples of final bosses. Additionally, companies often do not store the data necessary to create proper knowledge bases, a problem which was reported as knowledge vaporization [26], which is a reason for which, in our work, we use evolutionary computation instead of ML in order not to depend on knowledge bases.

The survey also discusses how some works managed to generate content as maps or sprites with DL approaches based on an single draft or sketch [20][27][28][29], or combining stages and levels from games [30][31][32]. However, in contrast with 2D sketches for final image content, our work studies the improvement of bosses, which are received incomplete and without a direct connection between such partial content, related to previous design stages, and the content pending. In addition, for the video game case study, the mix of content from different games is not possible, due to the lack of sequels and prequels, and the unique nature of the the bosses and the game at the time of its original release.

Finally, the survey explains that most of the works focus on the generation of content such as 2D game levels, stages or maps, and sprites, with such content being internally represented by 2D images [33]. Additionally, the article by Liu et al. calls for works who focus on more diverse types of content, such as characters in fighting games, and their skills and characteristics [20]. Our work is focused on the generation of less explored content in comparison with other types, according to the review: bosses, complex entities which are

defined by means of a DSL and not represented as image content; these bosses behave as antagonist characters, and they are characterized by traits like their Equipment Configuration, which is the design stage studied by our approach.

2.2 Balance of content

Some approaches are designed as a companion for the development team in order to be used to gain insights about game balance by gathering information from simulation-based plays of artificial agents. In [34], the authors present an approach for balancing games through the use of restricted play agents. In [35], the authors make use of a multi-objective optimization algorithm to demonstrate the feasibility of an approach for automatic game balancing in the context of the Top Trumps card game. In [36], the authors propose a semi-automatic process for the game balancing of a prototype of a commercial video game (Zombie Village Game by Blue Byte GmbH). In [37], the authors create two different AI agents that are able to play a commercial board game (ticket to ride), generating information that can be used to balance the game.

These approaches are similar to the one presented in this work in the sense that they use artificial agents to gather information about the performance of the individuals being evolved. However, they do not focus on the creation of new content, and the resulting balanced content is not compared with content that has been balanced manually by developers, as our work does.

Other approaches gather the information for balance from the actions and knowledge of real users, and sometimes they adapt the content for those types of users. In [38], an evolutionary algorithm is given an initial population of pre-crafted and random Role-Playing Game (RPG) skills that are then assessed based on how often each skill is used by the players and evolved into new sets of skills. In [39], the approach focuses on informing game development about possible imbalances by means of agents that are trained using a dataset containing six months of plays of 213 human players of the game Aion, a Massive Multiplayer Online RPG. In [40], the authors use a deep-learning surrogate model to generate character classes of an FPS game that is tailored for a specific map. The approach uses a fitness function that takes into account the desired match duration and score and compares them to the values predicted by the surrogate model. The work in [41] applies an evolutionary strategy to generate playing card game decks by using a subset of the cards that are available in Hearthstone using artificial agent matches to evaluate the decks. Another work on the balance of the cards in Hearthstone [42] quantifies the impact of a change in a card on all of the sets of existing decks and game strategies. That work uses search strategies to determine which changes should be selected to balance the game.

These approaches benefit from the knowledge of several users, which can be used to guide the balancing process and even tailor the results for specific users

or styles of play. However, in our work, we only use an artificial agent to guide the process and then compare the resulting content to the (supposedly good) content created originally by the developers. Nevertheless, tailoring the process of boss generation to specific types of players or play styles is something that we will explore in the future (e.g., in Kromaia, since there are different types of ships controlled by the player in the game, suggesting that different archetypes of players are playing the game).

3 Background

In this work, we focus on bosses, which are created in different steps using a Domain-Specific Language (DSL). This section presents the creation steps and the DSL.

3.1 Steps for Creating Bosses

The creation of bosses in video games involves several steps. In our video game case study (Kromaia), the development team followed a four-step process. First, in the creative design step, a general specification of the bosses is provided, determining its structure, anatomical constraints, and visual appearance. Then, in the spatial organization step, the anatomical specification of the bosses is performed, arranging a set of hulls into a specific disposition and specifying relationships and hierarchies among the hulls. The third step is the behavior specification, which determines the artificial intelligence that will drive the boss.

The final step is the Equipment Configuration, which deals with weak point and weapon configuration. Determining the presence of different attack-/defense items and the hulls that they are attached to has a significant impact on both the difficulty associated with the unit and the user's experience. This delimits the power assigned to the boss as well as the valid strategies that the players can use to defeat that boss.

In this work, we focus on the Equipment Configuration step to improve the bosses of Kromaia, the video game case study. This step does not refer to the task of tuning parameters that represent modifications of fixed content: it refers to the addition of procedurally generated content that is classified as Necessary in PCG taxonomy research works [1, 43]. This is because, before the Equipment Configuration step, the bosses lack weapons and weaknesses, and, therefore, they are not suitable for the game.

3.2 Domain-Specific Language for Shooters

This section presents SDML (Shooter Definition Model Language), which is a DSL created by Kraken Empire, the company responsible for the development of the video game Kromaia. SDML covers the definition of every mission,

vehicle, creature, and landscape of the game. The left part of Fig. 1 shows different bosses that are included in the video game case study; the right section of Fig. 1 shows part of the SDML content in `Boss 1: Serpent`. The main SDML concepts that are relevant to the bosses are:

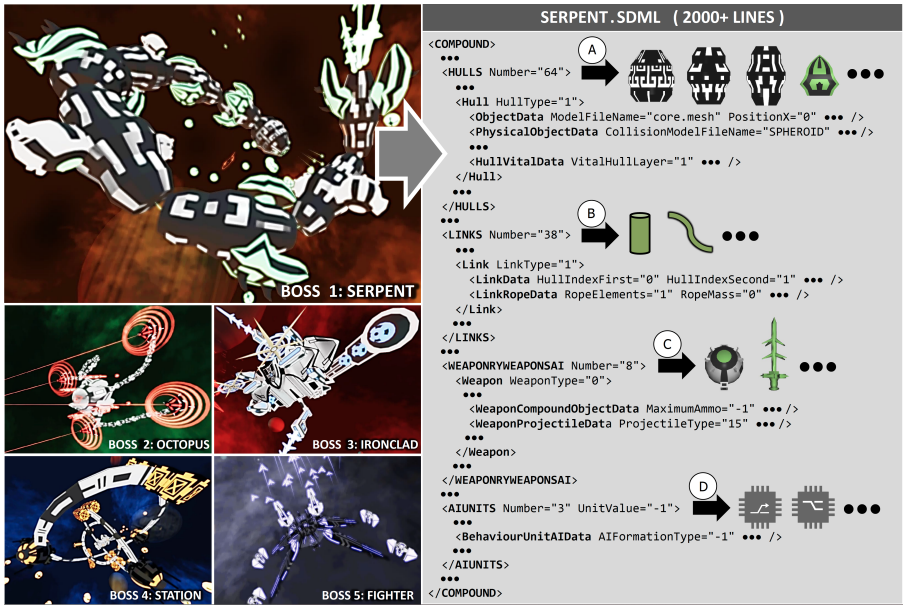

Fig. 1 Bosses in Kromaia, defined with SDML. The content included in this commercial video game (e.g., vehicles, creatures, worlds, and missions) is described in SDML. These include a wide range of data covering geometry, physics, contraptions, and AI, and follow a modular structure.

- **Hulls and Links:** The Hull Module is a collection of solid bodies that are connected through configurable joints or links. This hierarchy defines the anatomy and physics of the boss. Depending on the arrangement and the flexibility of the links used, bosses may have rigid structures, mobile parts, or even segmented tentacles. Fig.1 shows how visual appearance, geometry, and physics-related attributes are defined for hulls (see A) and the varied link types (see B).
- **Weak Points:** These are damageable objects that are attached to certain hulls. They can be organized in layers that are progressively unlocked as an enemy player destroys them. Any opponent trying to defeat a boss must first destroy these weak points. In the beginning, they are the only damageable objects. However, once every weak point has disappeared, the normal hulls become damageable, and, therefore, it is possible to destroy the boss. Fig. 1 includes an example of a weak point (marked as `HullVitalData`) that belongs to a specific layer.

- **Weapons:** These are objects that are capable of inflicting damage by using bullets, launching homing missiles, tracing rays, or affecting the target on direct contact. The bosses included in the commercial releases of Kromaia use these four weapon types. These weapons involve AI automatic gun turret behaviors that make them aim at targets (players). An example of an AI weaponry module and possible parameters for a weapon is shown in Fig. 1(C).
- **AI components:** These concepts define the way that bosses act during a game in terms of artificial intelligence. An AI module can include several AI components for different situations, as shown in Fig. 1(D), and they also can describe flocking behaviors.

In the video game case study, an average boss unit has 64 hulls. Each hull might have, simultaneously, one of the four possible weapon types and a weak point. This is a design feature that was included by the developers, not a decision made during this research study. This turns out to be $2^{64*(4+1)} = 2^{320}$ possibilities for equipment configuration, which shows that testing every single possibility is not feasible. Our EBI approach explores this search space using an evolutionary algorithm.

4 Overview of our EBI Approach

This section presents our EBI approach, the goal of which is to improve bosses. It receives a partially configured boss as input that is used to encode the population of an evolutionary algorithm. Then, the individuals are assessed by a fitness function and evolved through the Improvement Crossover and Improvement Mutation operations. This is repeated until the stop condition is met. Finally, EBI decodes the best individual generated into a completely configured boss.

In the following subsections, we present the evolutionary algorithm, the encoding, the genetic operations, and the fitness for the EBI approach.

4.1 EBI Algorithm Summary

The evolutionary algorithm used by EBI iterates an Equipment Configuration configuration population and makes that population evolve by means of our genetic operations. A fitness operation guides the evolutionary algorithm, which tries to maximize the fitness values of the individuals included in the population. The fittest individuals reproduce and the population size is controlled and remains stable by discarding the lesser promising configurations. Fig. 2 shows the main steps included in our EBI approach.

- Input - Partially Configured Boss (Fig. 2, A): This is a boss, defined by means of SDML that has gone through three of the four creation steps (Creative Design, Spatial Organization and Behavior Specification).

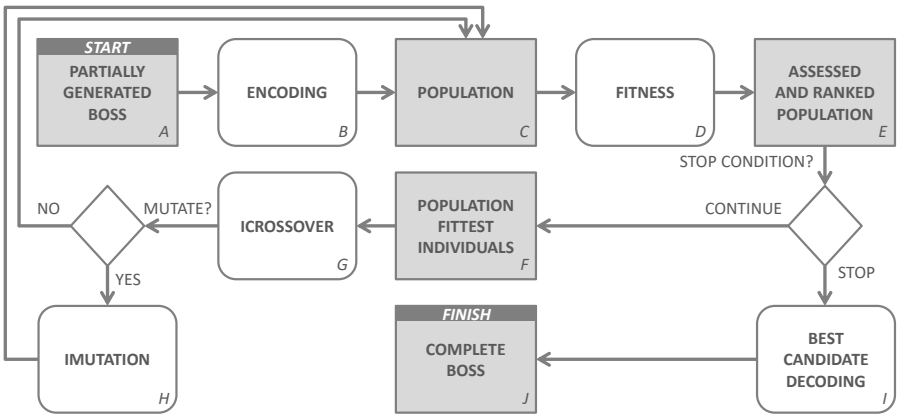

Fig. 2 EBI Approach Overview.

- The Encoding of the possible realizations of the Equipment Configuration (Fig. 2, B) is necessary before creating the initial population which will be evolved by the evolutionary algorithm (Fig. 2, C). These encoded, randomly generated configurations represent the last step in the boss creation process.
- The Fitness step (Fig. 2, D) evaluates the Equipment Configurations by assigning values to them and sorting the population as a ranking (Fig. 2, E). These fitness values depend on the results obtained from the simulation of duels between a human player and the bosses included in the population, which use their Equipment Configurations. The process is over when an Equipment Configuration with a fitness value that is high enough is found or when a time limit is reached.
- Assuming that the process is not over yet, the next step uses improvement focused genetic operations to create a new generation of Equipment Configurations (Fig. 2, G, H). Those configurations with the highest fitness values in the population are selected and allowed to reproduce by means of pairing (Fig. 2, F). Then, new Equipment Configurations are obtained crossing the genetic material of the possible combinations of two potential parents from the selected group. Finally, this step also introduces changes in the new configurations using Mutation, an operation that could make the new Equipment Configurations surpass the fitness values achieved by their parents or get worse than them.
- Output - Completely Configured Boss: This is a complete boss (including the Equipment Configuration) which has been decoded back to SDML (Fig. 2, I, J).

4.2 Boss Equipment Configuration Encoding of the EBI Approach

The Equipment Configurations obtained by our approach are the candidates that are generated for the last configuration stage in the boss creation process. Nevertheless, these configurations must be encoded. In evolutionary algorithms, this is usually done by representing candidates as binary strings or arrays containing values such as true/false or 0/1.

In the EBI approach, the encoding for the Equipment Configurations is as follows: each Equipment Configuration is a binary, bi-dimensional matrix in which columns correspond to the hull collection for the boss studied, and each row represents weak points and the four weapon types used by the bosses. The values in each cell indicate the presence (1) or absence (0) of a certain item type in a hull. The encoding used to represent Equipment Configurations is shown in Fig. 3.

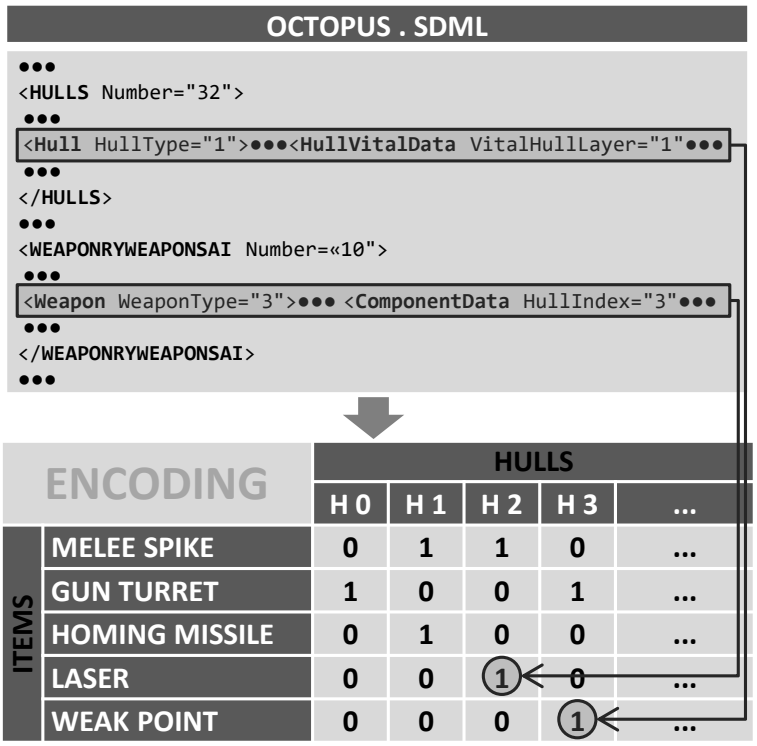

Fig. 3 Example showing the relation between data present in SDML and Equipment Configuration encoding.

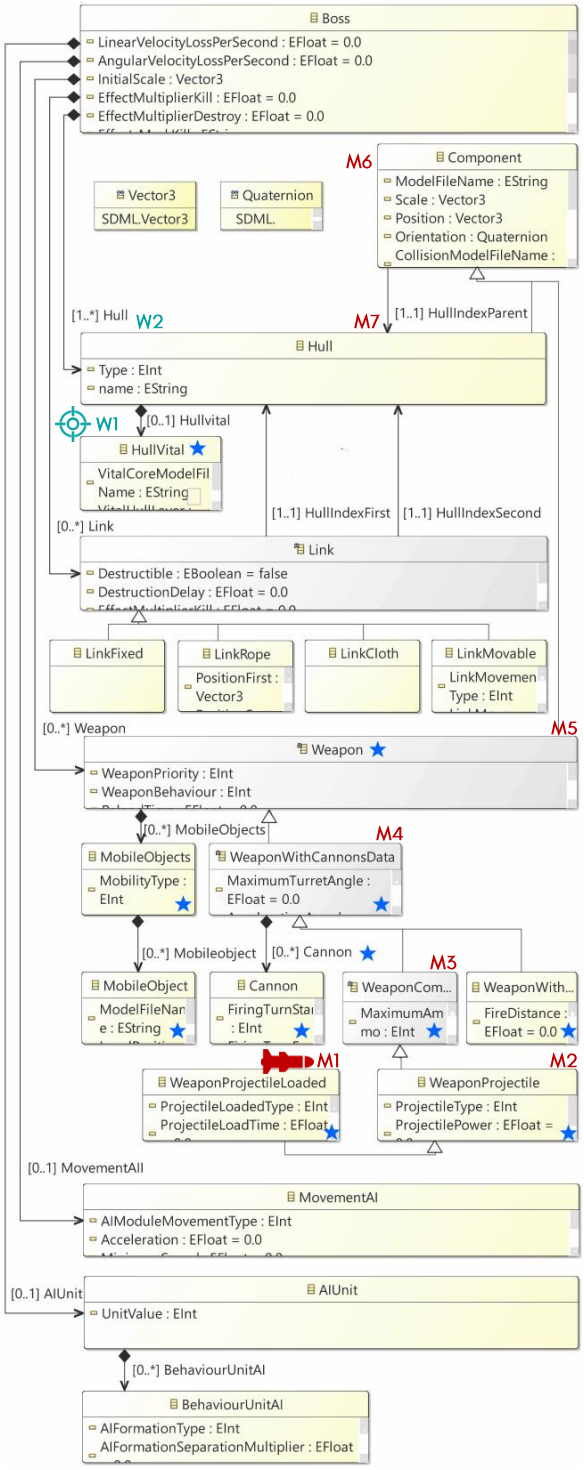

Fig. 4 Fragment of the metamodel which describes the rules for creating bosses in Kromaia. The elements marked with stars indicate the parts of such excerpt which are involved in the Equipment Configuration step. The series of marks M1-M7 and W1-W2 are examples which show the elements involved in terms of support for missiles (a type of weapon) and weak points, respectively, beyond Equipment Configuration.

4.3 Genetic Operations of the EBI Approach

Our work focuses on improvement, and this means that our approach deals with content which has been partially generated previously. In such circumstances, manipulating content by means of evolutionary computation and, therefore, using operations which mix and modify individuals of a given population, imply considerations beyond the changes or additions provided by traditional genetic operations.

Taking into consideration that the complete creation process of a specific content would involve a set of elements, properties, and rules which connect them, our work deals with improvement as the addition, removal, or modification of specific fragments of such set. However, if such modifications only consider those specific fragments, the new or updated content could be invalid and, therefore, not usable, due to the fact that the fragments are not isolated and independent.

In the context of the case study with which our approach is applied, an example of the aforementioned issue would be weak point designation within final bosses. Fig. 3 shows that the encoding used by our approach, which represents elements and properties corresponding to the last step, Equipment Configuration, includes information relative to weak points. In terms of properties and internal definition, weak points are different from the rest of hulls or solid bodies, but the inclusion of hulls in the structure of a final boss corresponds with one of the first stages (Spatial Organization). Therefore, the changes introduced during Equipment Configuration are not independent and isolated.

Recent works have surveyed a compendium of Search-Based approaches, which were developed in the last decades and made use of crossover and mutation operations [44][45]. These works show how methods like single-point crossover and random mutation are the dominant choices. In addition, the literature describes how, within the community of general evolutionary computation research, there exists a significantly high number of different genetic operations, more than 20 and 50 mutation and crossover operations, respectively [46][47][47]. However, the recent surveys do not describe genetic operators which are designed in order to focus on improvement and its implications. Instead of that, the scope of the operators surveyed is not a specific part of the individuals manipulated, and they cross or mutate complete solution candidates.

In this work, we propose new genetic operations which take into account the particularities implied by the notion of improvement. In order to define such improvement-focused operations, we take into account Model-Driven Engineering (MDE) [48] and consider the abstract representation of knowledge, like the content improved by our approach. More specifically, in the context of MDE a metamodel represents the formalization of the characteristics and particularities of the models which will be created in accordance to it [48].The metamodel expresses the relationship between the different elements involved in the definition of an entity, the nature, and cardinality (if applicable) of

such relationships, and the properties included. In the case of video games, the content may be formalized, for instance, by means of tools like Blueprints, in the case of the commercial engine Unreal [49], or the DSL used by the developers in the video game case study of this work. Models may formalize content such as objects, characters, or behaviors [50][51]. Our new operations make use of metamodels in order to focus on the fragments which complete the partially generated content and the binary encoding of such content. The metamodel is also used in order to modify, if it is necessary, the fragment of the partially complete content originally provided, which is not encoded, to keep the complete content coherent and valid.

In Kromaia, the video game case study, it is necessary to take into account and use certain rules and constraints which must be observed in order to create a suitable final boss. Every step in the production of a boss, in addition to Equipment Configuration, must be completed in accordance to those specifications. The compliance of the bosses generated with them is relevant with regard to the idea of improvement used in our work, and how such improvement affects a complete boss, even if our EBI approach focuses on one of the steps involved in the creation of a boss. This is relevant to the application of the genetic operations of our approach, since the concepts managed and added to the bosses by EBI, which are related to the Equipment Configuration step, could imply changes which affect the concepts corresponding the previous steps. Our approach takes advantage on the fact that the developers of the video game case study used SDML not only to define bosses, but also to formalize the characteristics which the bosses themselves should met in order to be considered as valid. Such formalization is shown in Fig. 4, which includes a fragment of the metamodel used to define the bosses included in the video game case study. This metamodel is used by our approach to correctly conduct improvement, which is driven by the modifications done by the evolutionary algorithm and our genetic operations, which focus on Equipment Configuration.

Our EBI approach generates new Equipment Configurations using some of the existing ones as parents. This process is supported by the new operators which we propose in this work: ICrossover (Improvement Crossover) and IMutation (Improvement Mutation). These genetic operations are based on crossover and mutation, two operators which are widely present in the evolutionary computation research literature [52] and they are used in order to expand populations and introduce variability into them. We created operators based on them in order to work in favor of improvement in the context of our work. The genetic operations are used to add diversity to the population and, eventually, to lead to individuals which are fitter than their ancestors.

First, the operators are adjusted to work with Equipment Configurations which represent possible weak point and weapon distributions in a boss. This configuration is the last step involved in the creation of a complete boss, while the previous steps provide the partially generated bosses to which our approach applies an EA.

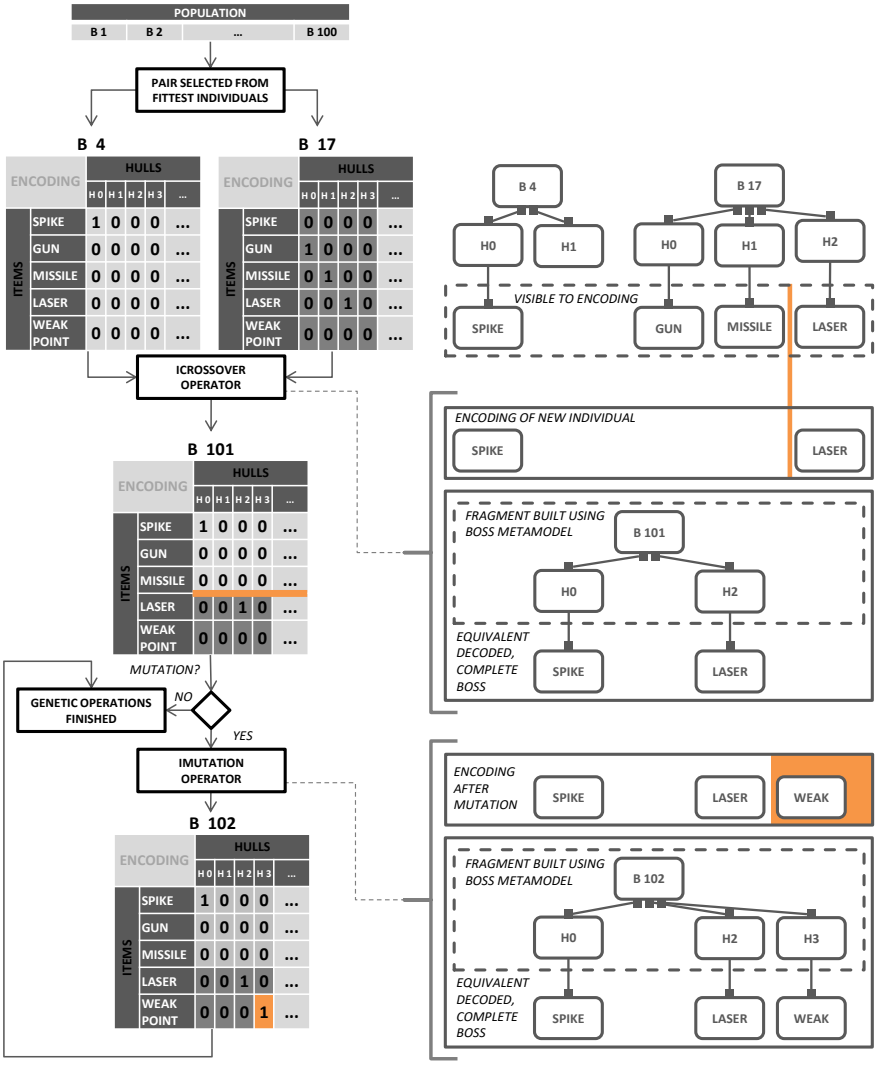

Fig. 5 Genetic Operations overview. The examples show how improvement is driven by the genetic operations, which give priority to Equipment Configuration but influence the partial completion fragment originally provided, taking advantage on the metamodel.

The fittest Equipment Configurations within the population are selected as parents for the new offspring. The fitness value calculation process is described in Subsection 4.4. Once the population is sorted, the best 10% of the Equipment Configurations are selected and pairwise combined to generate the new offspring. Each pair of parents is used to generate a new Equipment Configuration by applying the ICrossover operator. The newly created Equipment Configuration could be randomly designated to undergo mutation, by means

of the IMutation operator. An elitism of 55% is applied, so the best individuals remain unchanged for the next generation.

- **ICrossover:** The ICrossover operation mixes the content of two individuals to create a third, new one. Like the methods used traditionally in evolutionary computation, it is used to increase a population through the combination of the genetic material of two existing individuals. In the context of our approach and the video game case study, the encoding is a bi-dimensional matrix, but the crossover operation processes an Equipment Configuration as if it were a one-dimension array that contains a consecutive collection of rows of binary elements. Therefore, the first element corresponds to the value placed in the first row and the first column, and the last element in the configuration refers to the value present in the last column and the last row. The improvement crossover operation used by our approach gives priority to the concepts introduced by the last step, Equipment Configuration. Those concepts reflect the part of the metamodel related to the use of weak points and weaponry, as shown by the sections highlighted and marked with stars the fragment of the metamodel included in Fig. 4. These are the steps necessary to perform the ICrossover operation:
  - First, the genetic material of both parents is combined following the method known as single-point crossover. Assuming that the encodings of the two parents have the same size, a random position is selected and it marks a reference point for the new individual: the genetic material within the region previous to such point is taken from the first parent, while the rest is inherited from the second parent. The resulting individual could eventually be superior to its progenitors.
    In the case of the application of our approach to Kromaia, the weaponry and weak point elements, which are represented in the binary encoding and correspond with the Equipment Configuration step for both parents, are combined. First, a random value n in the interval [0, S-1] is selected (S is the size of the configuration, interpreted as a one-dimension array). Then, since every Equipment Configuration for a certain boss has the same size S, the new configuration takes its first n array elements from the first parent and the last S - n elements from the second parent. Depending on the fitness value given to a new configuration, it could outperform its parents or be selected among the best in the population to generate new configurations in a later iteration of the evolutionary algorithm.
    The top right part of Fig. 5 shows how the encoded configurations for both the parents and the new individual produced by means of ICrossover only contain information relative to the Equipment Configuration step.
  - Once the new individual has been created, the impact of the mixed genetic material encoded is studied. This study is done in order to determine possible changes required with regard to the fragment not

encoded (but included in the partially completed content which is provided to our approach) of the individual. Such modifications done on decoding are necessary to keep the complete content valid and usable. Therefore, even if the operation is focused on the fragment which would complete the individual, it implicitly drives the improvement since it could lead to necessary modifications on the partially completed fragment of the content which is received as an input by our approach.

In the case of the video game case study, once the genetic material corresponding to the Equipment Configuration step has been mixed, the impact of the elements for such step is calculated making use of the metamodel. The elements included in the encoding for the new individual created determine what other elements, which are necessary to make the complete boss feasible and are not explicitly present in the encoding, are required too. Those elements can be taken from the parents indirectly, thank to the information available in the metamodel. Therefore, the improvement process, which can involve the elements corresponding to different steps of the creation of a boss, is actually driven by the last step, the Equipment Configuration, since the encoding is focused on the elements related with that step. The center right part of Fig. 5 shows how the improvement process could lead to an improved boss in terms of other aspects, like internal structure.

- **IMutation:** The improvement mutation operation is inspired by the mutations found in biology. These mutations make modifications in the genes of individuals that are caused by random factors. The IMutation operation is first focused on producing modifications related to the completion of content which is provided partially. In the context our EBI approach and the video game case study, and similarly to the ICrossover operation, this means that IMutation gives priority to the elements corresponding to the Equipment Configuration step:
  - First, the operation traverses all the elements which belong to the encoding and determines if each of them is mutated according to a certain probability.
    In the case of the bosses from Kromaia a bit string mutation is applied on the new Equipment Configurations, which are created through ICrossover operations.
  - The resulting individual may include changes that would make it not usable or valid unless the part not encoded is redefined.
    With regard to the bosses for Kromaia, the metamodel is used in order to identify the elements which are not included in the encoding and have been indirectly affected by the mutation. The bottom right part of Fig. 5 shows that, for instance, adding weak points could imply structural changes which are not taken into account in the Equipment Configuration step. The right part of Fig. 5 provides a general overview of the implications of crossing and mutating for partially generated

bosses in Kromaia. In addition, Fig. 4 shows how, for example, the ability to launch missiles (Mark series M1-M7), or adding a weak point (W1-W2) implies the presence of certain elements which must be included in order to support those characteristics.

4.4 Fitness of the EBI Approach

The fitness step in our EBI approach determines the value of each Equipment Configuration that is generated. This stage takes an Equipment Configuration population as input and then measures the suitability of each configuration for the problem studied. Once this step is finished, it generates a ranking that sorts the Equipment Configurations according to the fitness values obtained so that the configuration with the best fitness value is ranked first. The evolutionary algorithm in our approach uses this fitness value to select parents for the next generation of configurations and to obtain the highest ranked Equipment Configuration once the search is over.

To obtain the fitness value for an Equipment Configuration, the EBI approach simulates a duel between a boss that uses that configuration and a player. It is an unattended duel in which both contenders are simulated. In that simulation, the player visits the different regions in the boss and tries to destroy the weak points that are available, while the boss uses the weapons in that configuration, which are different in power and range, to try to defeat the player. The simulation uses the information considered to be relevant by the developers to perform a simulated confrontation and includes statistical values regarding weapon accuracy, damage probability, and average player precision. The duel simulation is not deterministic, and it uses player and boss AI simulated agents that are handled in terms of actions, attacks, and damage by a state machine-based system. State machines have been used to describe boss behaviors in previous works [53]. The player agent performs a cyclic itinerary that successively focuses on each of the hulls included in the boss. When the player agent is focused on a hull that contains weak points, it tries to destroy them. The boss agent attacks the player using the weapons present in the different hulls. The weapons that are present in the hull on which the player is currently focused have a higher probability of hitting the player, which decreases with distance. The inherent accuracy and probability of success for the different types of weapons vary as well: Melee weapons that are present in distant hulls are not effective, but distant lasers or missiles have a reduced, yet non-zero probability of success. Both the player and the boss actively try to win the match and do not run away. This avoids draws and ensures that one of the contenders wins. The objective of the player is to destroy the weak points included in the boss, while the attacks performed by the boss aim for the player unit as a whole. Since the duel is not deterministic, it is run 30 times, following the suggestion in [54]. Once the simulation is over, our approach has enough information on relevant events and the progress of the duel to calculate a fitness value.

The developers provided two main measures that the bosses in the video game should maximize in order to prove their suitability for commercial releases: the player victory percentage ($F_{Victory}$), and the player optimal health percentage after a victory (F_{Health}). The fitness measures in this work (and the quality measures described in Section 5) use the following function:

$$clamp_{[0,1]}(x) = max(0, min(x, 1)) \tag{1}$$

Our approach calculates the $F_{Victory}$ criterion as a measure of the difference between the number of victories achieved by the player (V_P) and the desired, optimal number of victories ($V_{Optimal}$) (33%, according to the criteria used by the developers):

$$F_{Victory} = clamp_{[0,1]}\Big(1 - \frac{|V_{Optimal} - V_P|}{V_{Optimal}}\Big) \tag{2}$$

The F_{Health} criterion (for duels won by the player) is the average difference between the health percentage of the player at the end of the duel (Θ_P) and the optimal life level that the player should ideally keep at that point ($\Theta_{Optimal}$, 20%, according to the developers). This criterion focuses on how adequate the victories achieved by the player are, and it considers the health level of the player after defeating a boss in those duels:

$$F_{Health} = clamp_{[0,1]}\Big(1 - \frac{\sum_{d=1}^{V_P} \frac{|\Theta_{Optimal} - \Theta_P|}{\Theta_{Optimal}}}{V_P}\Big) \tag{3}$$

$F_{Overall}$ is a direct measure (as intended by the developers) that calculates an average fitness value for an Equipment Configuration, including every specific fitness criterion studied:

$$F_{Overall} = \frac{F_{Victory} + F_{Health}}{2} \tag{4}$$

In the end, $F_{Overall}$ is a value in the interval [0, 1] that allows the EBI approach to create an Equipment Configuration ranking.

4.5 Situating the approach

The present work applies an EA to partially completed game bosses in order to automatically produce complete bosses that have equal or better quality (in terms of six metrics) than the bosses that were originally completed by the developers. According to available taxonomies for procedural content generation [1,5,55,43], our work can be classified as follows. The content type generated is the equipment configuration of boss enemies, and the method

of PCG used is an Evolutionary Algorithm that manipulates SDML (Shooter Definition Model Language) models. With regard to the nature of the content, the multiplicity is single instance, since the case study is a single player game; the content generated is necessary and the type of derivation is built-in (since the elements generated are part of the game). With regard to the generation process, the mode is offline since it is performed as part of the development of the game. The degree of parameterization can be classified as parameter vectors (the input provided is a partially built boss). The nature of the process is stochastic and the EA follows a 'generate and test' constructiveness. The generation is non-personalized for different players; the generation is not controlled by the player. Finally, with regard to the game dependence, Kromaia belongs to the 3D space shooter genre, within the entertainment industry of commercial video games.

5 Evaluation

This section presents the evaluation of our approach: the research questions, the quality measurement, the experimental setup, the implementation details, and the results obtained.

5.1 Research Questions

The following questions address the evaluation of our EBI approach taking into account its results and the time necessary to obtain them:

RQ$_1$: *Does our EBI approach provide completely configured bosses that are comparable (or even better) in quality to those designed by the developers of the video game case study?*

RQ$_2$: *Does our EBI approach reduce the time necessary to configure good quality bosses in the video game case study?*

5.2 Quality Measures for the Configurations

The bosses included in the commercial release of the video game case study use Equipment Configurations that were approved after studying, adjusting, or even discarding several alternatives. The development company provided information regarding the version control system to measure the time invested in the bosses. The revision log showed that, after completing the Creative Design, Spatial Organization, and Behavior Specification steps, the Equipment Configuration step for each of the bosses required a month before the results were satisfactory. Thanks to the feedback given by the players, the developers determined that the users liked these boss units and found them to be enjoyable.

In order to compare the results obtained by the developers and our approach, we use criteria that measure the quality of the bosses. As previous

works describe, in the context of video game development, quality refers to the probability of a game experience being considered interesting by users [7] in terms of intellectually challenging content. These works also state that, in general, players can express whether or not they consider a game experience to be enjoyable; however, they usually find it difficult to express the reasons precisely.

There are measurable indicators that have been studied in the past and are considered to be relevant. Tension [56], Decisiveness [57], Interestingness [58], Interaction [59], and Uncertainty [60] were described in previous research works as being fundamental game quality indicators. More recently, Browne et al. showed through experiments with users that there is a set of criteria that stand out and are the most important: Lead Change, Permanence, Killer Moves, Uncertainty, Duration, and Completion [7]. The evaluation included in our approach gives each of these criteria a value in the interval [0, 1]. In our approach, the quality measures are calculated using data that is retrieved from a simulation, which involves 30 non-deterministic confrontations ($Duels$). For the video game case study, the developers determined through tests and questionnaires with players that concentration and engagement for an average boss reach their peak at approximately 10 minutes ($T_{Optimal}$), whereas the maximum accepted time was estimated to be $2 * T_{Optimal}$ (20 minutes).

– **Completion (Viability):** The criterion $Q_{Completion}$ calculates a ratio of conclusions over total duel count:

$$Q_{Completion} = \frac{Conclusions}{Duels} \tag{5}$$

$Duels$ = *Total number of duels*
$Conclusions$ = *Number of duels won by either side*

A game against a boss unit should end with more conclusions (victories for either the player or the boss) than draws. It is necessary to explain that the concept of draw in Kromaia refers to a situation in which the game exceeds the maximum acceptable duration without concluding since with enough time one of the contenders will eventually win.

– **Duration (Viability):** Our approach calculates the criterion $Q_{Duration}$ as a measure of the average difference between the duration of each duel and the desired, optimal duration:

$$Q_{Duration} = clamp_{[0,1]}\Big(1 - \frac{\sum_{d=1}^{Duels} \frac{|T_{Optimal} - T_d|}{T_{Optimal}}}{Duels}\Big) \quad (6)$$

$Q_{Duration}$ is close to 1 when the duration of the duels is similar to an optimal value

T_d = *Duration of the d-th Duel*
$T_{Optimal}$ = *Optimal duration of a duel*

The duration of duels between players and boss units is expected to be around a certain optimal value. Significant deviations from that reference value are good design-flaw indicators: remarkably short games are probably too easy, thus minimizing the challenge offered by the game experience; duels that go on a lot longer than expected tend to make players lose interest. Duration is a simple and consistent criterion that is effective at determining essential game design proficiency and is one of the criteria used by Cardamone [12].

– **Uncertainty (Quality):** For each duel, $Q_{Uncertainty}$ measures the average deviation between the time at which it is detected that one of the contenders is on the verge of defeat and the time corresponding to the duration of the duel.

$$Q_{Uncertainty} = clamp_{[0,1]}\Big(1 - \frac{\sum_{d=1}^{Duels} \frac{T_d - min\,(P_d, B_d)}{T_d}}{Duels}\Big) \quad (7)$$

$Q_{Uncertainty}$ is high when the contenders keep safe in terms of health as long as possible during the duels

P_d = *Time until Player health is critical in the d-th duel*
B_d = *Time until Boss health is critical in the d-th duel*

In order to keep players engaged with a duel, neither the player nor the boss unit should get extremely close to victory or defeat too early before the duel is settled, with (T_d) being its duration. Therefore, a duel is considered to be more uncertain the longer the time until the health levels of the player or the boss unit reach a dangerous/critical status (P_d and B_d, respectively).

– **Killer Moves:** Q_{KMoves} measures the proportion of killer moves by any contender, taking into account the moves done by both the player and the boss unit that are considered remarkable highlights but that are less important than killer moves.

$$Q_{KMoves} = clamp_{[0,1]}\Big(1 - \frac{\sum_{d=1}^{Duels} \frac{K_d}{H_d}}{Duels}\Big) \tag{8}$$

Q_{KMoves} reaches a high value when the percentage of highlights of duels with contenders that are not close is low

K_d = Number of killer moves detected in the d-th duel
H_d = Number of highlights detected in the d-th duel

This criterion is related to the fact that some events allow one of the contenders in a duel to make a remarkable impact in terms of power balance. They are the result of actions carried out on purpose that cause such an important effect, but that is not decisive enough to end the duel. In this video game, the developers considered that a highlight move occurs when either the boss unit or the player experiences a health decrease, but that killer moves are those that make the health difference between the contenders reach 30%. The health level of a boss depends on the number of weak points left in a specific unit, whereas a player can absorb five impacts.

– **Permanence:** The criterion $Q_{Permanence}$ is measured as follows:

$$Q_{Permanence} = clamp_{[0,1]}\Big(1 - \frac{\sum_{d=1}^{Duels} \frac{R_d}{H_d + K_d}}{Duels}\Big) \tag{9}$$

$Q_{Permanence}$ is higher when the advantages provided by killer moves or highlights are not canceled often

R_d = Number of recovery moves detected in the d-th duel

Duels with a high permanence value are games in which the advantage given by significant actions or moves by one of the contenders are unlikely to be immediately reverted by the opponent in terms of dominance. In the video game case study, the developers considered every highlight move and killer move to be meaningful actions. Another move considered by the developers is the recovery move (R). This move quickly cancels the advantage given by other previous killer or highlight moves.

– **Lead Change:** This criterion is measured taking into account those highlights or killer moves that cause the lead to change during the course of a duel:

$$Q_{LChange} = clamp_{[0,1]}\Big(\frac{\sum_{d=1}^{Duels} \frac{L_d}{H_d + K_d}}{Duels}\Big) \tag{10}$$

$Q_{LChange}$ gets closer to 1 as the number of relevant events which cause lead changes is higher in the duels

L_d = Number of lead changes detected in the d-th duel

In the video game case study, the lead is defined at any given moment by determining the contender with the highest health level, and a low number of lead changes indicates low dramatic value.

Our approach evaluated these six criteria for each boss unit included in the commercial release of the video game case study in order to obtain a quality threshold that is useful for verifying whether the results obtained by our approach reach the same quality levels.

5.3 Experimental Setup

The main goal of the evaluation was to measure the performance achieved by our EBI approach compared to the completely configured bossed included in the commercial release of Kromaia in terms of the six quality metrics proposed. To do this, we follow four steps:

The **first step** is the extraction of bosses from Kromaia, the video game case study. Using their version control system, the developers provided the following: the five partially configured bosses that will be used as input for the EBI approach; the five completely configured versions of the bosses that will be used in the comparison in terms of quality; the time spent in the configuration of the five original bosses.

The **second step** is the execution of the EBI approach. It receives a partially configured boss and produces one completely configured boss. This is repeated for the five partially configured bosses previously extracted from Kromaia. Given the stochastic nature of the approach, we perform 30 independent runs of the approach for each boss (as suggested in [54]), to homogenize the results and ensure that the evolutionary algorithm produces results that are consistent and repeatable.

The **third step** is the comparison of the completely configured bosses produced by the EBI approach and the bosses obtained from Kromaia. Therefore, the six quality measures are calculated for both sets of bosses. To do this, 30 duel simulations between the AI player and each of the bosses are performed

and the quality measures are calculated. The results are gathered, averaged, and compared using box plots.

The **fourth step** is the statistical analysis of the results (following the guidelines in [61]), which provides quantitative evidence of the impact of the results and shows whether this impact is significant. Since our data does not follow a normal distribution, our analysis requires the use of non-parametric techniques. We carry out a Quade Test [62] followed by a Holm's post hoc to determine if the differences between pairs of bosses (one from EBI and one from Kromaia) are significant enough to be considered different. Then, we apply Vargha and Delaney's effect size [63] to determine to what degree the generated boss is better than the original boss for each of the six quality measures applied.

5.4 Implementation Details

To implement the approach of this work, we used the TinyXML parser to process SDML models. In addition, the specifications of the computer used in the evaluation process are the following: Toshiba Satellite Pro L830 laptop, with a processor Intel® Core™ i5-3317U with 4GB RAM and Windows 8 64bit.

The parameter settings used in our approach are detailed in Table I. The size of the population is limited to 100 individuals. Each generation, an elitism of 55 individuals is applied. The best 10 parents (μ) are selected by truncation to generate offspring of 45 new individuals through crossover and mutation of pairwise combinations of the parents. The probability of mutation (p_m) depends on the size of the individual (`1/(Number of Hulls)`

All of the parameters displayed in Table 1 were obtained from different tests. These tests showed that the settings currently applied reached better solutions in less time. In this work, we do not focus on tuning the parameters to achieve higher performance numbers for specific problems. In the context of testing, the default values used to measure the performance of search-based techniques are good enough, as suggested by Arcuri and Fraser [54]. Nevertheless, we intend to evaluate the parameters of our EBI approach in future works.

Table 1 EBI Approach Parameters

Parameter description	**Value**
$Size$: Population size	100
μ: Number of parents	10
λ: Number of offspring from μ parents	45
p_m: Mutation probability	1 / (Number of Hulls)

In general, there are two (atomic) performance measures used in search algorithms: a measure for speed or search effort, and a quality measure. In this paper, after running some prior convergence tests, we established a certain amount of wall clock time for each of the runs of our EBI approach (1200 seconds). Then, we focused on the solution quality.

For replication purposes, the CSV files used as input to report the results and the statistical analysis are available at: `https://svit.usj.es/bosses-kromaia-vs-ebi/`. There is a CSV file per boss in Kromaia where each file includes the results of each quality measure. There is also a CSV per boss configured in EBI where each file includes the mean results of the 30 runs from the best *Size* (100) configurations for each quality measure.

5.5 Results

This subsection presents five completely configured bosses obtained from our approach and compares them to the five bosses included in the commercial release of Kromaia using six quality measures studied in the video game research literature [7] and statistical methods.

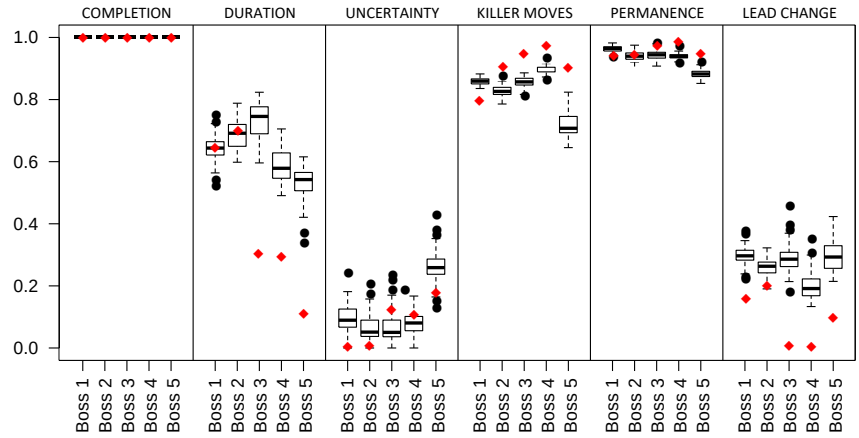

Fig. 6 Results for the bosses generated by our approach for each quality measure. The columns show red diamonds representing the values obtained by the bosses originally included in the video game case study.

Fig. 6 shows the results in different box plot groups representing the study of each quality measure. For every quality measure, the values belong to the interval [0,1] and the box plots represent the average results obtained (after 30 runs, due to non-deterministic factors) by each of the completely configured bosses generated by our approach (`Boss1..5`). In addition, every box plot includes a red diamond showing the average quality results achieved by the

corresponding boss included in the commercial release of Kromaia for each quality measure.

Table 2 shows the *p-Values* of Holm's post hoc analysis for each boss and measure. A *p-Value* under 0.05 is statistically significant as accepted by the research community [61]. Each row of the table shows the results of each pair-wise comparison between a boss from Kromaia and a boss from our EBI approach, whereas the columns of the table show the Holm's post hoc *p-Values*. For example, the row "K1 vs E1" shows the *p-Values* of Holm's post hoc analysis for each measure that correspond to the pair-wise comparison between Boss 1 of Kromaia (K1) and Boss 1 of our EBI approach (E1). Table 3 shows $\hat{A}_{12}$ values for each boss and measure. For example, the row "K1 vs E1" shows $\hat{A}_{12}$ values for each measure that correspond to the pair-wise comparison between K1 and E1.

Table 2 Holm's post hoc *p-Values* for each boss and measure

	Holm's post hoc *p-Values*					
	Completion	Duration	Uncertainty	Killer Moves	Permanence	Lead Change
K1 vs E1	-	0.53	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$
K2 vs E2	-	0.21	0.36	$\ll 2 \times 10^{-16}$	2.8×10^{-11}	$\ll 2 \times 10^{-16}$
K3 vs E3	-	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$
K4 vs E4	-	$\ll 2 \times 10^{-16}$	1.3×10^{-9}	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$
K5 vs E5	-	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$	$\ll 2 \times 10^{-16}$

Table 3 $\hat{A}_{12}$ effect size for each boss and measure

	Completion	Duration	Uncertainty	Killer Moves	Permanence	Lead Change
K1 vs E1	0.5	0.48	0.01	0	0.04	0
K2 vs E2	0.5	0.52	0.59	1	0.76	0.02
K3 vs E3	0.5	0	0.91	1	1	0
K4 vs E4	0.5	0	0.77	1	1	0
K5 vs E5	0.5	0	0.06	1	1	0

Table 4 shows the results per quality measure by taking into account the mean for that quality measure in all bosses. Each row of the table corresponds to a quality measure. Column 2 shows the standard deviations of the mean results of Kromaia bosses for each measure, whereas Column 3 shows the standard deviations of the mean result of the bosses of our EBI approach. Column 4 (Holm's post hoc *p-Values*) and Column 5 ($\hat{A}_{12}$) show the results of the pair-wise comparison between the mean of the Kromaia bosses and the mean of the EBI bosses for each quality measure. Each of the bosses produced by our approach is obtained by using the partially generated version of one of the original bosses included in Kromaia as a starting point. This partial generation initially constrains the characteristics of the bosses produced by our

approach to a certain extent. For example, it determines the number of hulls used, which is not modified via encoding during the Equipment Configuration stage. The restrictions resulting from the partial generation do not determine the quality of the bosses, but characteristics like anatomy or behavior are affected. Therefore, the pair-wise comparison is used in order to study the differences between bosses in terms of the Equipment Configuration stage. The mean quality values for both the bosses included in Kromaia and those produced by our approach are also summarized in the radar chart included in Fig. 7.

Table 4 For each measure, mean results and standard deviations, Holm's post hoc $p\text{-}Values$ and $\hat{A}_{12}$ effect size

	Mean $\pm$ (σ)		Kromaia vs EBI Bosses	
	Kromaia Bosses	EBI Bosses	Holm's post hoc $p\text{-}Values$	$\hat{A}_{12}$
Completion	1 ± 0	1 ± 0	-	0.5
Duration	0.41 ± 0.25	0.64 ± 0.02	$\ll 2 \times 10^{-16}$	0
Uncertainty	0.10 ± 0.07	0.11 ± 0.02	1.2×10^{-10}	0.26
Killer Moves	0.91 ± 0.07	0.83 ± 0.01	$\ll 2 \times 10^{-16}$	1
Permanence	0.97 ± 0.02	0.93 ± 0.01	$\ll 2 \times 10^{-16}$	1
Lead Change	0.11 ± 0.07	0.27 ± 0.02	$\ll 2 \times 10^{-16}$	0

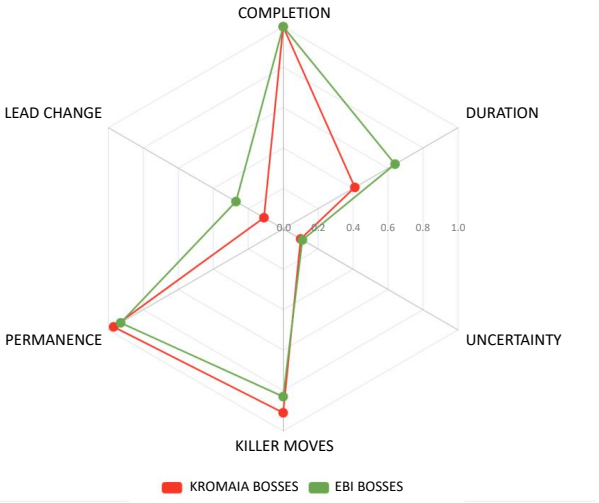

Fig. 7 Quality measure mean values obtained by the bosses originally included in Kromaia, the video game case study, and those produced by our EBI approach.

5.6 Research Question 1

Does our EBI approach provide completely configured bosses that are comparable (or even better) in quality to those designed by the developers of the video game case study?

To answer the first research question, we compared the quality values achieved by the completely configured bosses obtained from our approach and the original bosses in the commercial release of the video game case study:

For **Completion** (the first column in Fig. 6), both the bosses from our approach and the original bosses obtained good values close to the maximum. These high values for $Q_{Completion}$ come from the characteristics of the battles between players and bosses in Kromaia: the effectiveness of the weapons used and the difficulty for an average player to avoid combat maximize the duel conclusions within acceptable duration limits.

For **Duration**, the more similar the duration of duels between players and bosses to the optimal value estimated, the higher the quality. The second column in Fig. 6 shows that the bosses generated by our approach outperformed the bosses in the case study. Monitoring the evolution of the Equipment Configuration population showed that it took few generations for the EA in our EBI approach to obtain $Q_{Duration}$ values that surpassed those achieved by the original bosses. The Version Control System used by the development company shows that the different Equipment Configurations tested varied little in terms of weapon/weak point sizes. This shows that, for $Q_{Duration}$, it is necessary to explore a larger Equipment Configuration search space than the one provided by the developers in order to obtain weapon and weak point distributions that achieve high values.

For **Uncertainty** (the third column in Fig. 6), four of the bosses generated by our approach achieved mean values similar to those obtained by the original bosses. High $Q_{Uncertainty}$ values would denote duels for which the time between when one of the contenders is about to be defeated and the conclusion is very low. In comparison to other video games, the design of Kromaia does not favor high uncertainty values. As duels advance, the bosses become more damaged and the weak point set is reduced. The distance in time between hits also progressively increases. For players, each time a health level decrease occurs, there is a safety time period during which the player is immune. The column displaying data related to Uncertainty in Fig. 6 shows that the fifth boss generated by our approach (`Boss5`) achieved a higher value than its corresponding boss. The most remarkable restriction in that fifth boss for both the commercial release of the video game case study and our approach is the fact that it is the boss that has the simplest geometry in terms of hull size (11) in comparison to the rest of the bosses, which have an average of 47 hulls. Therefore, considering the search spaces for the corresponding encoding sizes ($2^{11*5} = 2^{55}$ and $2^{47*5} = 2^{235}$, respectively), our algorithm managed to obtain a significantly higher value for `Boss5` with the time budget assigned.

For **Killer Moves** (Q_{KMoves}), the average values obtained by four of the bosses generated with our approach (`Boss1..4`) are comparable to those

achieved by the five original bosses, as shown in Fig. 6. However, the fifth boss obtained by our approach (`Boss5`) achieved a lower mean value than the original boss. Due to the anatomical restrictions on hull size for both our approach and the original bosses, the version of the fifth boss generated by our approach obtained a higher percentage of uncertain duels, as described above. As shown in Fig. 6, there is an inverse relationship between Uncertainty and Killer moves: bosses with higher $Q_{Uncertainty}$ values obtain lower values for Q_{KMoves}. This is because a high number of killer moves (highlighted events that mean a significant health gap between contenders) is incompatible with high uncertainty.

For **Permanence** ($Q_{Permanence}$), the fifth column in Fig. 6 shows that the average values obtained from the bosses provided by our approach and those in the video game case study are comparable (even more so when `Boss5` is not considered). The reason behind that specific case is that Permanence measures the persistence in time of an advantage achieved in terms of health level gaps, so there is an inverse relationship between Q_{KMoves} and $Q_{Permanence}$.

For **Lead Change** ($Q_{LChange}$), the sixth column in Fig. 6 shows that the bosses generated by our EBI approach obtained better values than the original bosses included in Kromaia. Due to the specific design of the video game case study, it is difficult to achieve a high proportion of lead-change events in relation to highlights (health decrease events) or killer moves (highlights resulting in a considerable health gap between contenders). In Kromaia, a player can absorb up to five impacts before being defeated. However, the bosses generated by our approach and those included in the commercial release of Kromaia include a considerably higher number of weak points in order to reach acceptable values in the rest of the quality measures. Therefore, an impact received by a player is likely to take several weak points in the boss (each of which involves a highlight event) in order to revert the lead status. Since our approach obtains bosses with better $Q_{Duration}$ values, those bosses favor duels that are closer to the optimal duration and that have more lead changes. This minimizes the possibility of duels that are too long or too short, which occur due to passiveness or a significant skill difference between the contenders, respectively.

The *p-Values* of Holm's post hoc analysis of Table 2 show that the differences in the bosses from Kromaia and the bosses from our EBI approach are statistically significant (under 0.05), except for the following: for Completion, the results are equivalent in all pair-wise comparisons of bosses; for Duration, the results are not significant when Bosses 1 and 2 are compared; and for Uncertainty, the results are not is not significant when Bosses 2 are compared. With regard to the $\hat{A}_{12}$ values of Table 3, Boss 1 from our EBI approach outperforms Boss 1 from Kromaia in all quality measures (except for Completion, which is equivalent), as row "K1 vs E1" of the table shows. The highest differences correspond to Uncertainty, Killer Moves, and Lead Change where Boss 1 of our EBI approach outperforms Boss 1 of Kromaia in almost all of the runs, respectively. This is especially relevant because Kromaia developers acknowledged that Boss 1 is the boss that they have devoted the most

development effort to since Boss 1 is the most played and defeated boss in the video game (up to 6 times more than Boss 5 according to the global gameplay statistics of Steam).

When the data of bosses is aggregated by quality measure, the *p-Values* of Holm's post hoc analysis of Table 4 show that the differences are statistically significant (under 0.05), except for Completion where the results are equivalent. The $\hat{A}_{12}$ values of Table 4 show that EBI bosses outperform Kromaia bosses in the majority of the runs for Duration, Uncertainty, and Lead Change. In Killer Moves and Permanence, Kromaia bosses outperform EBI bosses in all of the runs. Although it may appear that EBI bosses do not achieve good results for Killer Moves and Permanence, these results are comparable to the results achieved by the original Kromaia bosses. The difference between the mean of Killer Moves in Kromaia and Killer Moves in EBI is only 0.08; the difference between the mean of Permanence in Kromaia and Permanence in EBI is only 0.04. We showed these results to two Kromaia developers and they confirmed that the results obtained by our EBI approach are comparable (or even better) in quality to the results in Kromaia, so they will use our EBI approach to produce new, completely configured bosses in Kromaia through downloadable content.

5.7 Research Question 2

Does our EBI approach reduce the time necessary to configure good quality bosses in the video game case study?

This research question takes into account the time spent by the developers and our approach to perform the last configuration stage (Equipment Configuration) to obtain completely configured bosses that achieve good quality values.

To evaluate this last configuration stage, it was necessary to study the Version Control System of Kromaia used by the developers. The log history registry shows that the sum of the Equipment Configuration stages that led to the original bosses that were commercially released was five months. For our approach, the stop condition for the evolutionary algorithm in EBI was restricted only by time (20 minutes). Therefore, since it was necessary to generate five completely configured bosses, this task was completed in 1 hour and 40 minutes.

These results show that our approach generated good-quality, completely configured bosses about 99% faster than the developers. In addition, both the evolutionary algorithm and the duels that guide it and confront simulated contenders allow our approach to run unattended. Therefore, with better equipment, it would be possible to launch five separate instances of EBI (one per boss and CPU) in order to divide the time required by 5.

5.8 Fitness Progress in the Evolutionary Algorithm

With regard to the progress shown by the evolutionary algorithm used by EBI, Table 5 shows the average fitness values obtained by the best candidate of each of the five bosses generated by EBI at the end of the evolution. The fitness values of the five bosses from Kromaia (depicted as red diamonds in Fig. 6) are also included for comparison. The five EBI bosses reached mean fitness values of around 0.9. The fitness values shown by the original bosses included in Kromaia are low in comparison. Only the original K2 obtained values of 0.6 for both $F_{Victory}$ and F_{Health}. With regard to the convergence of the search performed by EBI, the fourth column in Table 5 shows the percentage of the time budget needed to find the best boss candidate. Interestingly, the improvement of the randomly-equipped original population used by the evolutionary algorithm is an average of 20% for three of the bosses and 10% for two of them. The last column of Table 5 shows the number of bosses explored during the time budget allocated (20 minutes). Additionally, doubling the time budget from 20 to 40 minutes did not affect the maximum fitness value reached).

Table 5 Average fitness values of the best candidate at the end of each evolution, percentage of the time budget required and number of bosses explored. Kromaia bosses are also included for comparison.

	Fitness			Time budget	
	$F_{Victory}$	F_{Health}	$F_{Overall}$	% needed	Bosses explored
E1	0.91	0.86	0.88	10%	305,650
K1	0.08	0.1	0.09		
E2	0.99	0.86	0.92	35%	566,640
K2	0.6	0.6	0.6		
E3	0.99	0.83	0.91	11%	523,855
K3	0.09	0.11	0.10		
E4	0.98	0.83	0.91	3%	311,140
K4	0.08	0.08	0.08		
E5	0.99	0.86	0.93	0.1%	1,420,165
K5	0.08	0.09	0.08		

6 Threats to Validity

In this section, we present how we addressed or mitigated the possible threats to validity regarding our approach. To identify the threats to be considered in this work, we use the guidelines described by De Oliveira et al. [64] and classify the threats into different groups.

Conclusion validity threats are concerned with the statistical relationships between data treatment and outcome. We have identified the following two threats in this category. The first one is not accounting for random variation. We addressed this threat by performing 30 runs for each of the bosses to be configured with our approach. The second one is the lack of a meaningful comparison baseline. To address this threat, we compared the results obtained from our approach with those generated by the developers for the commercial release of the video game case study.

Internal validity threats involve non-causal relationships between treatment and outcome. We have identified the following two threats in this category. The first one is the lack of clarity of data collection tools and procedures. Since it is difficult to calculate fitness values based on tests with players (due to their considerable time length), our approach simulates duels between the bosses and an AI player. We used the data provided by the SDMLs of the contenders to perform the simulation, and we used two main indicators provided by the developers to value configurations: victory percentage and health level. The second threat is the lack of real problem instances. To address this, the evaluation performed in our work was applied to an industrial video game case study, and the problem artifacts (partially and completely configured bosses) were directly obtained from the video game industry.

Construct validity threats are concerned with the relations between observations and theory. We have identified the following threats in this category. The first one is not assessing the validity of cost measures. In order to perform a fair comparison between the completely configured bosses included in Kromaia and the bosses generated by our approach, we studied the time spent by the developers and our algorithm to obtain the results (see Section 5.7). The second threat is not assessing the validity of effectiveness measures. We addressed this by using quality measures presented in the video game research literature [7] and performing a statistical analysis of the results (see Sections 5.2 and 5.5).

External validity threats deal with the generalization of the results obtained in a larger population, which is outside the experiment. We have identified one threat in this category, the lack of a clear definition of target instances. To address this threat, we have provided as much detail as possible regarding the DSL used by the video game case study (SDML, see Section 3.2). Nevertheless, EBI should be applied to other video games before assuring its generalization.

7 Conclusion

The creation of video game content is relevant to user retention and engagement. Our work shows that it is possible to accelerate video game content creation by means of Procedural Content Improvement. In this work, we focus on game bosses and the improvement of their quality through procedural content generation using our EBI approach. It works with an evolutionary al-

gorithm that is guided by the simulation of duels between a player and the game bosses produced. The evolutionary algorithm and the genetic operations which we propose in our approach give priority to the content fragment which must be added to the initial fragment in order to consider such content complete. However, the improvement is not limited to that last fragment of the content, and these algorithm and operations drive a genetic improvement of the complete content, including the fragment that is provided originally as partially complete content.

We evaluate our approach in the context of Kromaia, a commercial PC and PlayStation 4 video game. In the application of the EBI to this game, our approach receives partially created bosses and produces complete bosses by automatically performing the last step in the creation process followed by human developers: the Equipment Configuration stage. This stage adds weapon and weak point content and defines the characteristics of a boss in terms of attack/defense items and weak point distribution. In the evaluation of our approach, we use six quality indicators from the video game research literature in order to give the game bosses a quality level value: Completion, Duration, Uncertainty, Killer Moves, Permanence, and Lead Change. We use these measures to evaluate the quality of both the bosses produced by our approach and those originally created by the developers of Kromaia.

The results show that our EBI approach provides completely configured bosses that are comparable (or even better) in quality to those designed by the developers of Kromaia. Our EBI approach improves video game content in less time than the developers of a commercial video game. The improvement of this content is an essential issue that concerns developers due to the nature of life cycles in commercial video games. This includes renovating products through downloadable content. For future work, we are planning to explore the potential benefits of EBI for level improvement.

Declarations

Funding. This work was supported in part by the Ministry of Economy and Competitiveness (MINECO) through the Spanish National R+D+i Plan and ERDF funds under the Project VARIATIVA under Grant PID2021-128695OB-I00, and in part by the Gobierno de Aragón (Spain) (Research Group S05_20D).

Competing interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. J. Togelius, G.N. Yannakakis, K.O. Stanley, C. Browne, IEEE Trans. Comput. Intellig. and AI in Games **3**(3), 172 (2011). URL `http://dblp.uni-trier.de/db/journals/tciaig/tciaig3.html#TogeliusYSB11`
2. J.R. Koza, Statistics and computing **4**(2), 87 (1994)

3. W.B. Langdon, *Genetically Improved Software* (Springer International Publishing, Cham, 2015), pp. 181–220. DOI 10.1007/978-3-319-20883-1_8. URL https://doi.org/10.1007/978-3-319-20883-1_8
4. J. Petke, S.O. Haraldsson, M. Harman, W.B. Langdon, D.R. White, J.R. Woodward, IEEE Trans. Evol. Comput. **22**(3), 415 (2018). DOI 10.1109/TEVC.2017.2693219. URL https://doi.org/10.1109/TEVC.2017.2693219
5. M. Hendrikx, S. Meijer, J. Van Der Velden, A. Iosup, ACM Trans. Multimedia Comput. Commun. Appl. **9**(1) (2013). DOI 10.1145/2422956.2422957. URL https://doi.org/10.1145/2422956.2422957
6. A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A.K. Hoover, A. Isaksen, A. Nealen, J. Togelius, IEEE Trans. Games **10**(3), 257 (2018). DOI 10.1109/TG.2018.2846639. URL https://doi.org/10.1109/TG.2018.2846639
7. C. Browne, F. Maire, IEEE Trans. Comput. Intellig. and AI in Games **2**(1), 1 (2010). DOI 10.1109/TCIAIG.2010.2041928. URL https://doi.org/10.1109/TCIAIG.2010.2041928
8. Z. Liu, P. Luo, X. Wang, X. Tang, in *Proceedings of the IEEE international conference on computer vision* (2015), pp. 3730–3738
9. J.A. Brown, D. Ashlock, S. Houghten, A. Romualdo, in *2020 IEEE Congress on Evolutionary Computation (CEC)* (2020), pp. 1–8. DOI 10.1109/CEC48606.2020.9185601
10. A. Ruela, F.G. Guimarães, Soft Computing **21**(23), 7005 (2017). DOI 10.1007/s00500-016-2238-3. URL https://doi.org/10.1007/s00500-016-2238-3
11. D.R. Ashley, V. Chockalingam, B. Kuzma, V. Bulitko, in *2019 IEEE Conference on Games (CoG)* (2019), pp. 1–8
12. L. Cardamone, G.N. Yannakakis, J. Togelius, P.L. Lanzi, in *European Conference on the Applications of Evolutionary Computation* (Springer, 2011), pp. 63–72
13. P.L. Lanzi, D. Loiacono, R. Stucchi, in *2014 IEEE Conference on Computational Intelligence and Games* (IEEE, 2014), pp. 1–8
14. D. Loiacono, L. Arnaboldi, in *2017 IEEE Conference on Computational Intelligence and Games (CIG)* (2017), pp. 199–206. DOI 10.1109/CIG.2017.8080436
15. D. Loiacono, L. Arnaboldi, IEEE Transactions on Games **11**(1), 36 (2019). DOI 10.1109/TG.2018.2830746
16. P.T. Ølsted, B. Ma, S. Risi, in *2015 IEEE Congress on Evolutionary Computation (CEC)* (2015), pp. 1527–1534. DOI 10.1109/CEC.2015.7257069
17. D. Blasco, J. Font, M. Zamorano, C. Cetina, Journal of Systems and Software **171**, 110804 (2021). DOI https://doi.org/10.1016/j.jss.2020.110804. URL http://www.sciencedirect.com/science/article/pii/S0164121220302089
18. S. Kent, in *Integrated Formal Methods*, ed. by M. Butler, L. Petre, K. Sere (Springer Berlin Heidelberg, Berlin, Heidelberg, 2002), pp. 286–298
19. D. Blasco, C. Cetina, O. Pastor, Inf. Softw. Technol. **119** (2020). DOI 10.1016/j.infsof.2019.106235. URL https://doi.org/10.1016/j.infsof.2019.106235
20. J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G.N. Yannakakis, J. Togelius, Neural Computing and Applications **33**(1), 19 (2021)
21. G.N. Yannakakis, A. Liapis, C. Alexopoulos, (2014)
22. D. Ha, D. Eck, arXiv preprint arXiv:1704.03477 (2017)
23. M. Guzdial, N. Liao, M. Riedl, arXiv preprint arXiv:1809.09420 (2018)
24. A. Liapis, G.N. Yannakakis, J. Togelius, in *In Proceedings of ACM Conference on Foundations of Digital Games, 2013. In Print*
25. O. Delarosa, H. Dong, M. Ruan, A. Khalifa, J. Togelius, in *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)* (Springer, 2021), pp. 412–426
26. J.S. van der Ven, A.G.J. Jansen, J.A.G. Nijhuis, J. Bosch, (2006), pp. 4–5. DOI 10.1007/978-3-540-30998-7_16
27. B. Yoo, K.J. Kim, in *2016 IEEE Conference on Computational Intelligence and Games (CIG)* (IEEE, 2016), pp. 1–2
28. A. Liapis, G.N. Yannakakis, J. Togelius, in *International Conference on Evolutionary and Biologically Inspired Music and Art* (Springer, 2013), pp. 180–191
29. Y.R. Serpa, M.A.F. Rodrigues, in *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)* (IEEE, 2019), pp. 182–191

30. S. Snodgrass, A. Sarkar, in *International Conference on the Foundations of Digital Games* (2020), pp. 1–11
31. A. Sarkar, S. Cooper, in *AIIDE Workshops* (2018)
32. A. Sarkar, Z. Yang, S. Cooper, arXiv preprint arXiv:2002.11869 (2020)
33. K. Park, B.W. Mott, W. Min, K.E. Boyer, E.N. Wiebe, J.C. Lester, in *2019 IEEE Conference on Games (CoG)* (IEEE, 2019), pp. 1–8
34. A. Jaffe, A. Miller, E. Andersen, Y.E. Liu, A. Karlin, Z. Popovic, in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 8 (2012), vol. 8
35. V. Volz, G. Rudolph, B. Naujoks, in *Proceedings of the Genetic and Evolutionary Computation Conference 2016* (2016), pp. 269–276
36. M. Beyer, A. Agureikin, A. Anokhin, C. Laenger, F. Nolte, J. Winterberg, M. Renka, M. Rieger, N. Pflanzl, M. Preuss, et al., in *2016 IEEE Conference on Computational Intelligence and Games (CIG)* (IEEE, 2016), pp. 1–8
37. F. de Mesentier Silva, S. Lee, J. Togelius, A. Nealen, in *AAAI Workshops* (2017)
38. A. Pantaleev, in *Proceedings of the The third workshop on Procedural Content Generation in Games* (2012), pp. 1–5
39. J. Pfau, A. Liapis, G. Volkmar, G.N. Yannakakis, R. Malaka, in *2020 IEEE Conference on Games (CoG)* (IEEE, 2020), pp. 431–438
40. D. Karavolos, A. Liapis, G.N. Yannakakis, in *Proceedings of the 13th international conference on the Foundations of digital games* (2018), pp. 1–10
41. A. Bhatt, S. Lee, F. de Mesentier Silva, C.W. Watson, J. Togelius, A.K. Hoover, in *Proceedings of the 13th International Conference on the Foundations of Digital Games* (2018), pp. 1–10
42. F. de Mesentier Silva, R. Canaan, S. Lee, M.C. Fontaine, J. Togelius, A.K. Hoover, in *2019 IEEE Conference on Games (CoG)* (IEEE, 2019), pp. 1–8
43. G.N. Yannakakis, J. Togelius, *Artificial intelligence and games*, vol. 2 (Springer, 2018)
44. F. Pérez, T. Ziadi, C. Cetina, IEEE Transactions on Software Engineering (2020)
45. I. Boussaïd, P. Siarry, M. Ahmed-Nacer, Automated Software Engineering **24**(2), 233 (2017)
46. S.M. Lim, A.B.M. Sultan, M.N. Sulaiman, A. Mustapha, K.Y. Leong, International journal of machine learning and computing **7**(1), 9 (2017)
47. G. Pavai, T. Geetha, ACM Computing Surveys (CSUR) **49**(4), 1 (2016)
48. S. Kent, in *International conference on integrated formal methods* (Springer, 2002), pp. 286–298
49. E. Games. Unreal engine, version 2018.3.9 (1998). URL `http://www.unrealengine.com/`
50. E.M. Reyno, J.Á. Carsí Cubel, Computers in Entertainment (CIE) **7**(2), 1 (2009)
51. S. Tang, M. Hanneghan, in *2010 Developments in E-systems Engineering* (IEEE, 2010), pp. 95–100
52. A.E. Eiben, J.E. Smith, et al., *Introduction to evolutionary computing*, vol. 53 (Springer, 2003)
53. K. Siu, E. Butler, A. Zook, in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 12 (2016), vol. 12
54. A. Arcuri, G. Fraser, Empirical Software Engineering **18**(3), 594 (2013). DOI 10.1007/s10664-013-9249-9. URL `http://dx.doi.org/10.1007/s10664-013-9249-9`
55. N. Shaker, J. Togelius, M.J. Nelson, *Procedural content generation in games* (Springer, 2016)
56. W. Kramer, The Games Journal (2000). URL `http://www.thegamesjournal.com/articles/WhatMakesaGame.shtml`
57. J.M. Thompson, The Games Journal (2000). URL `http://www.thegamesjournal.com/articles/DefiningtheAbstract.shtml`
58. I. Althöfer, Technical Report, Friedrich Schiller Universität Jena (2003). URL `https://www.minet.uni-jena.de/preprints/althoefer_03/CAGI.pdf`
59. C. Browne, *Connection Games: Variations on a Theme* (AK Peters, Natick, Massachussetts, 2005)
60. H. Iida, K. Takahara, J. Nagashima, Y. Kajihara, T. Hashimoto, in *Entertainment Computing - ICEC 2004, Third International Conference, Eindhoven, The Netherlands, September 1-3, 2004, Proceedings* (2004), pp. 333–338. DOI 10.1007/978-3-540-28643-1_41. URL `https://doi.org/10.1007/978-3-540-28643-1_41`

61. A. Arcuri, L. Briand, Softw. Test. Verif. Reliab. **24**(3), 219 (2014). DOI 10.1002/stvr.1486. URL http://dx.doi.org/10.1002/stvr.1486
62. S. García, A. Fernández, J. Luengo, F. Herrera, Information Sciences **180**(10), 2044 (2010). DOI https://doi.org/10.1016/j.ins.2009.12.010. URL https://www.sciencedirect.com/science/article/pii/S0020025509005404. Special Issue on Intelligent Distributed Information Systems
63. A. Vargha, H.D. Delaney, Journal of Educational and Behavioral Statistics **25**(2), 101 (2000). DOI 10.3102/10769986025002101. URL http://jeb.sagepub.com/content/25/2/101.abstract
64. M. De Oliveira Barros, A.C. Dias-Neto, RelaTe-DIA **5**(1) (2011)