Procedural Personas as Critics for Dungeon Generation

Antonios Liapis[1], Christoffer Holmgård[2],
Georgios N. Yannakakis[1,2], and Julian Togelius[2]

[1] Institute of Digital Games, University of Malta, Msida, Malta
[2] Center for Computer Games Research, IT University of Copenhagen, Copenhagen, Denmark
antonios.liapis@um.edu.mt, holmgard@itu.dk, georgios.yannakakis@um.edu.mt, juto@itu.dk

Abstract. This paper introduces a constrained optimization method which uses procedural personas to evaluate the playability and quality of evolved dungeon levels. Procedural personas represent archetypical player behaviors, and their controllers have been evolved to maximize a specific utility which drives their decisions. A "baseline" persona evaluates whether a level is playable by testing if it can survive in a worst-case scenario of the playthrough. On the other hand, a Monster Killer persona or a Treasure Collector persona evaluates playable levels based on how many monsters it can kill or how many treasures it can collect, respectively. Results show that the implemented two-population genetic algorithm discovers playable levels quickly and reliably, while the different personas affect the layout, difficulty level and tactical depth of the generated dungeons.

1 Introduction

The generation of dungeons is one of the first instances of procedural content generation (PCG) with *Rogue* (Toy and Wichman 1980). Since then, many games have used algorithms to generate dungeons, e.g. *Diablo* (Blizzard 1996), *Daggerfall* (Bethesda 1996) and *Daylight* (Zombie Studios 2014). Generating dungeons has also been a fertile research topic as summarized by [1]; algorithmic approaches using constraints [2], grammars [3] and genetic algorithms [4] have been successfully applied to this task.

This paper introduces a method where procedural personas act as critics in a search-based procedural content generation (SBPCG) framework [5]. Procedural personas are artificial agents which represent archetypical player behaviors (e.g. rushing to the goal, killing monsters, collecting treasures). In this paper, the personas have been evolved on a set of authored dungeons, according to different fitnesses that match archetypical decisions-making priorities. The testbed game, named MiniDungeons, is a simple turn-based roguelike game; the game has been tested by human users and a close match between procedural persona playstyle and human playstyle was found [6].

Using procedural personas to test the evolving dungeons situates the proposed method as a type of simulation-based SBPCG. However, the persona-critics are used not only to evaluate how appropriate a dungeon is for a particular playstyle, but also whether the dungeon is actually playable. The requirement that a dungeon can be completed by a simple "baseline" persona — despite any stochasticity of the gameplay — adds another constraint to the generative process. This paper uses a two-population genetic algorithm for the purposes of constrained optimization, which evolves both feasible and infeasible dungeons [7]. Dungeons are tested by a "baseline" persona based on whether it can complete a worst-case scenario of the dungeon; this persona also evaluates infeasible dungeons' distance from feasibility. Playable levels are evaluated by a Monster Killer persona or a Treasure Collector persona based on how many monsters it can kill or how many treasures it can collect, respectively.

2 Previous Work

This Section covers the core background material (testbed game, procedural personas and evolutionary level design) on which the presented method is built.

2.1 MiniDungeons game

MiniDungeons is a simple turn-based roguelike puzzle game, implemented as a benchmark problem for modeling decision making styles of human players [8]. MiniDungeons levels are laid out on a grid of 12×12 tiles: tiles can be walls (which obstruct movement), empty, or contain monsters, treasure, the level's entrance or exit. The player has full information of the level except for monsters' damage, as discussed below.

In MiniDungeons, a hero (controlled by the player) starts at the level's entrance and must proceed to the level exit: stepping on the exit tile concludes a level and loads the next one. A hero starts each level with 40 hit points (HP) and dies at 0 HP. The hero can collect treasure by stepping on treasure tiles: treasures have no in-game effect but a treasure counter is shown on the user interface. The hero can drink potions by stepping on potion tiles: potions heal 10 HP, up to the maximum of 40 HP. Finally, the hero can kill monsters by stepping on monster tiles: monsters do not move and only engage the hero if the hero moves onto their tile. Combat is stochastic: a monster deals a random number between 5 HP and 14 HP of damage to the hero and then dies.

For the purposes of collecting player data as well as for evolving procedural personas, ten Minidungeons levels were created in advance (see Fig. 1). These levels were designed in a mixed-initiative fashion [9] and had several patterns which allowed different decision making styles to be exhibited. The authored levels have many branching points, but usually include an easy path (with minimal combat) between the entrance and the exit. Moreover, treasures and potions are often "guarded" by monsters, although some treasures are easily accessible and some monsters do not obstruct any paths. These patterns allow for different ways of traversing the level, as will be seen in Section 2.2.

2.2 Procedural Personas

The MiniDungeons game was created for two purposes: (a) to investigate how human players enact decision making styles in a simple game, and (b) to construct artificial agents able to represent such decision making styles.

A core assumption of decision theory [10] is that human decision making under risk and uncertainty is shaped by *utility*. A utility function determines the decision maker's willingness to take risks for an expected reward, and is considered idiosyncratic. In digital games, the game's mechanics constitute affordances [11] which are likely to be of utility to the player. Using the MiniDungeons game as a testbed, 38 participants

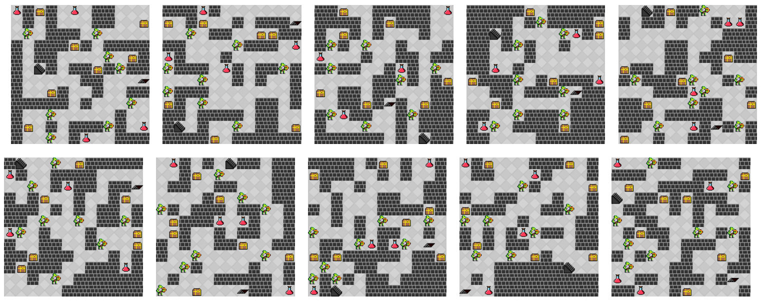
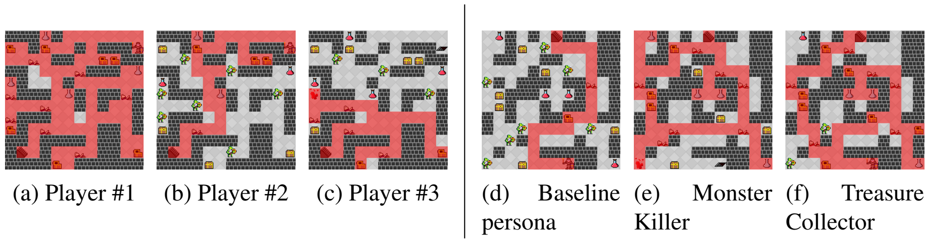


Fig. 1: The levels used for collecting player data and for evolving procedural personas.



(a) Player #1 (b) Player #2 (c) Player #3 (d) Baseline persona (e) Monster Killer (f) Treasure Collector

Fig. 2: Playtraces of human players and evolved procedural personas of MiniDungeons.

played all 10 levels of Fig. 1, as covered in detail in [6]: a few participants managed to collect all the treasures in every level (see Fig. 2a), while others rushed to the exit (see Fig. 2b) or miscalculated the risk of combat and died (see Fig. 2c). Such mechanics (treasure collection, death, reaching the exit) are thus likely sources of utility to players.

Procedural personas are artificial agents which represent archetypical decision making styles. In MiniDungeons, procedural personas consider several gameplay and level elements as sources of utility: killing monsters, collecting treasures, reaching the exit, performing as few actions as possible, or avoiding death. Previous work identified five procedural personas: a Monster Killer, a Treasure Collector, a “baseline” persona, a Speedrunner and a Survivalist, respectively. For the purposes of this paper, generated dungeons will be evaluated by personas evolved on all dungeons of Fig. 1 as per [6]. The controller for each persona is a combination of 7 linear perceptrons, with inputs being the hero’s HP and distance to different elements (e.g. closest potion, closest “safe” treasure) and outputs being the desirability of a strategy (e.g. go to closest potion, go to closest treasure that does not involve combat). The strategy with the highest value is selected by the agent; the decision is re-evaluated in every step rather than upon completion of the strategy. The perceptrons’ weights were evolved via an $(\mu + \lambda)$ evolutionary strategy without self-adaptation. The fitness of each agent was calculated from the utilities collected after all 10 levels were played. Focusing on the personas used in this paper, the baseline persona received a boost to its fitness for every exit it reached; the Treasure Collector received a fitness boost for every treasure collected and a smaller boost for every exit reached; the Monster Killer received a fitness boost for every monster killed and a smaller boost for every exit reached. Optimizing the controllers for these fitnesses resulted in personas exhibiting very different behaviors (see Fig. 2d–2f).

The evolved procedural personas were compared to the human playtraces, in terms of persona-player agreement ratio. In every step a human took when playing, the persona was queried “what would be your next action given this game state?”: if the persona’s chosen action matched the human’s, the agreement ratio increased. Summarizing the results of [6], most players had the highest agreement ratio with the Treasure Collector persona, while a smaller number of players matched the Monster Killer persona.

2.3 Constrained Optimization of Game Levels

Previous experiments on the constrained optimization of game levels focused on generating *map sketches*, i.e. low-resolution, high-level abstractions of complete levels [9]. Map sketches contain a small number of tiles which represent the most significant features of a level of a specific genre (e.g. weapon pickups in shooter games, player bases in strategy games). The simplicity of a map sketch allows it to be evolved in a straightforward and computationally lightweight manner. Map sketches of strategy games, roguelike dungeons and first-person shooters have been evolved according to a generic set of objectives which can be customized to the game genre at hand [12]. The constraints of such map sketches revolve around the connectedness between level features: for instance, in a map sketch for a strategy game all bases must be connected (via passable paths) with each other and with all of the map’s resources. In order to ensure constraint satisfaction, evolution has been carried out via a FI-2pop GA [7] which can discover feasible individuals quickly and reliably even in highly constrained spaces [13].

Minidungeons levels differ from map sketches in the fact that, despite a similarly small map size, they are directly playable. This introduces additional constraints on Minidungeons levels in that they must be completable by procedural personas. Moreover, previous experiments optimized map sketches according to hard-coded objectives inspired by game design patterns [14], while Minidungeons levels are evolved according to the play experience of the procedural personas that playtest them. In that regard, the procedural personas act as critics both on the playability and on the quality of the generated level: how this affects the evolutionary process will be explored in Section 4.

3 Methodology

This Section describes the two-population genetic algorithm used to evolve Minidungeons levels, as well as the methods for assessing playability (the infeasible fitness function) and level quality (the feasible fitness function) via procedural personas.

3.1 Evolving Levels for Minidungeons

A Minidungeons level consists of 144 tiles, which can be empty or contain walls, monsters, treasures, potions, the level entrance or the level exit. In the genotype, a Minidungeons level is represented directly as an array of integers: each integer describes the contents of a single tile in the level.

Due to the constraints on playability (discussed in Section 3.2), Minidungeons levels are evolved via a feasible-infeasible two-population genetic algorithm (FI-2pop

GA). The FI-2pop GA separates feasible individuals from infeasible ones (which do not satisfy one or more constraints), placing the former in a feasible population and the latter in an infeasible population [7]. The feasible population evolves to optimize the domain-specific measure of quality, while the infeasible population evolves to minimize its members' distance from the feasible border. As infeasible individuals approach the border of feasibility, the chances that their offspring will be feasible increase. Feasible offspring of infeasible parents migrate to the feasible population, and vice versa: this indirect form of interbreeding may increase the size and diversity of the feasible population. In order to ensure that the feasible population is sufficiently large for efficient optimization, the *offspring boost* mechanism is applied to the FI-2pop GA. The offspring boost is applied in cases where the feasible population is smaller than the infeasible population, and forces both feasible and infeasible populations to produce an equal number of offspring regardless of the number of parents in each population.

In the experiments described in this paper, evolution of Minidungeons levels is driven by asexual mutation alone; preliminary experiments showed that recombination is slower to discover feasible individuals and can result in multiple entrances or exits in the same dungeon. Mutation may transform an empty tile to a wall tile and vice versa, a level feature (non-wall, non-empty tile) may swap places with another level feature chosen randomly, or any tile may swap places with an adjacent one. Every offspring has 5% to 20% of its tiles (chosen randomly) mutated in the above fashion. By evolving content solely via this mutation scheme, an offspring is ensured to contain the same number of monsters, treasures, potions, level entrances and level exits as its parent. Parents are chosen via fitness-proportionate roulette wheel selection; the same parent may be chosen multiple times to generate offspring. In each population (feasible and infeasible), the best individual is transferred to the next generation unchanged.

3.2 Assessing Playability with Personas

In order for a MiniDungeons level to be playable, a number of constraints need to be satisfied: (a) the level must contain a specific number of tiles of certain types, e.g. one entrance and one exit, (b) all features of the level (monsters, potions, treasures, exit) must be accessible via passable paths to the hero, and (c) the hero must be able to reach the exit without dying. Constraints of type (a) are automatically satisfied by seeding the initial population with levels containing the desired number of level features: since mutation does not add or remove features, the number of features in the initial population will remain constant throughout the evolutionary process. Constraints of type (b) require that a passable path exists between the level entrance and all other features in the level: levels that fail this constraint are evaluated based on how many features are inaccessible. Finally, constraints of type (c) require that an agent simulates a playthrough of the level. In order to ensure that the level can be completed regardless of the stochasticity of combat, a 'worst-case' scenario is constructed by assigning maximum damage (14 HP) to all monsters of the level. The agent chosen to perform the playthrough is the baseline persona, whose affordance is only to reach the exit: this persona does not get "distracted" by treasure or monsters, and is likely to finish the level quickly. If the baseline persona dies then this constraint is failed: however, an additional check for the number of tiles explored by the persona is performed. This additional constraint was added after preliminary experiments in order to ensure that the entrance and exit are not close to each other, so that even speedrunners face at least a minimal challenge. If a baseline persona completes the level having explored less than 12 tiles, the level fails to satisfy the constraint of type (c) and is evaluated based on how many tiles the baseline persona explored, or a worse score if the baseline persona died.

Combining constraints (b) and (c) into a fitness measure for infeasible content, the distance to feasibility is calculated via d_{inf} of eq. (1). The infeasible population evolves to minimize d_{inf}, which increases the chances of feasible content being discovered. Observing d_{inf}, there is a clear priority between constraints: levels that fail constraints of type (b) automatically fail constraint (c) and assume that the baseline persona died without even testing for it. Moreover, if a baseline persona dies then the level receives a much worse score than if it completes the level, even within a very small number of steps. This aims to guide infeasible content towards first becoming well-formed (with all features accessible to the hero), then minimally playable for the baseline persona.

$$d_{inf} = \begin{cases} 1 + \frac{u_N}{N} & \text{if } u_N > 0 \\ 1 & \text{if baseline persona died} \\ \frac{1}{2}(1 - \frac{s_B}{C_s}) & \text{if baseline persona completed the level with } s_B < C_s \end{cases} \quad (1)$$

where N is the total number of level features (monsters, potions, treasures, exit) and u_N is the number of features which are not accessible from the level entrance; s_B is the number of tiles explored by the baseline persona in the worst-case scenario (all monsters dealing maximum damage) and C_s is the minimum number of explored tiles for a level to be considered feasible ($C_s = 12$ in this study).

3.3 Assessing Level Quality with Personas

The main contribution of the procedural personas is towards the evaluation of feasible, playable game levels. However, it is not obvious what a persona (or indeed the human players it represents) looks for in a level. Granted that the decisions of procedural personas are shaped by their own utility functions, only events which affect their utility should be considered. This paper will consider the two most dominant (and distinct) procedural personas of past experiments: the Monster Killer (with a utility for killing monsters and reaching the exit) and the Treasure Collector (with a utility for collecting treasure and reaching the exit). Most playtraces of the 38 human players who tested MiniDungeons matched the Treasure Collector persona (86%), while the Monster Killer was second (8%). When evaluating a level it has just finished playing (either by reaching the exit or by dying), the Monster Killer assigns the score of eq. (2) while the Treasure Collector assigns the score of eq. (3). The values of C_m, C_t and C_r are taken directly from the fitness function which guided the evolution of each persona's controller[1]; the persona was evolved on 10 authored levels (see Fig. 1) and was evaluated on how it represents an archetypical decision making style (a Monster Killer that

[1] The fitness function of all personas' controllers included a penalty for taking extraneous actions. Since this penalty was a control mechanism to avoid playthroughs taking too long rather than an explicit utility, it is omitted for the purposes of level evaluation.

kills most monsters, a Treasure Collector that collects most treasure) [6]. Inversely, the scores of eq. (2) and (3) evaluate whether the level provides the desired utilities to personas that play optimally towards attaining them.

$$S_{MK} = \frac{(d_m C_m + C_r r)}{(N_m C_m + C_r)} \tag{2}$$

$$S_{TC} = \frac{(d_t C_t + C_r r)}{(N_t C_t + C_r)} \tag{3}$$

where N_m and N_t is the number of monsters and treasures in the level respectively; d_m and d_t is the number of dead monsters and collected treasures respectively; r is 1 if the hero reached the exit and 0 if not; C_m, C_t and C_r are constants expressing the priority of monsters, treasures and level completion (respectively) in each persona's utility; for these personas $C_m = C_t = 1$ and $C_r = 0.5$. The denominator normalizes the score of eq. (2) and (3) between 0 (no affordances acquired) and 1 (all affordances acquired).

Intuitively, a persona prefers levels that allow it to maximize its utility function: i.e. a Monster Killer prefers levels that allow it to kill all monsters and a Treasure Collector prefers levels that allow it to collect all treasure. Due to the stochastic nature of combat, the same level is played by a persona multiple times (R=10 in this paper) with damage for each monster randomized in each playthrough. When maximizing the level's utility for a persona, the simulations' S_{MK} and S_{TC} scores are averaged in the fitness of eq. (4) for a Monster Killer, and eq. (5) for a Treasure Collector, respectively.

$$F_{MK} = \frac{1}{R}\sum_{i=1}^{R} S_{MK}(i) \tag{4}$$

$$F_{TC} = \frac{1}{R}\sum_{i=1}^{R} S_{TC}(i) \tag{5}$$

Maximizing the utility function of a persona, however, may be somewhat naive considering the decisions taken within MiniDungeons. Maximizing the utility of a Treasure Collector, for instance, can be trivially solved by placing all the treasure in a straight path between the level entrance and the level exit. In such cases, the player does not take a decision at any point during play; there is no risk/reward where the idiosyncratic utility function would shape the decision. In order to provide an element of risk, and thus require that the persona makes *meaningful* decisions, the level can be evaluated on how different a playthrough is from the next. Due to the randomness of combat, different playthroughs by the same persona may result in a premature death, in more or fewer treasures collected or monsters killed. Using the standard deviation of S_{MK} and S_{TC} among the 10 simulations, eq. (6) (for a Monster Killer) and eq. (7) (for a Treasure Collector) aim to maximize the levels' risk involved in personas' decisions.

$$D_{MK} = \sqrt{\frac{1}{R-1}\sum_{i=1}^{R}(S_{MK}(i) - F_{MK})} \tag{6}$$

$$D_{TC} = \sqrt{\frac{1}{R-1}\sum_{i=1}^{R}(S_{TC}(i) - F_{TC})} \tag{7}$$

4 Experiments

The experiments described in this section test how the different procedural personas (Monster Killer and Treasure Collector) and different fitness functions of eq. (4)-(7) affect the evolutionary process and the final generated dungeons. Dungeons generated in this paper have the same properties as those of Fig. 1: a 12×12 tile grid containing one entrance, one exit, 8 monsters, 7 treasures and 4 potions (21 level features in total). All experiments in this paper were performed with a population size of 20 (including feasible and infeasible levels), and evolution runs for 100 generations; results were averaged from 20 independent evolutionary runs and each level is evaluated by a procedural persona via 10 playthroughs.

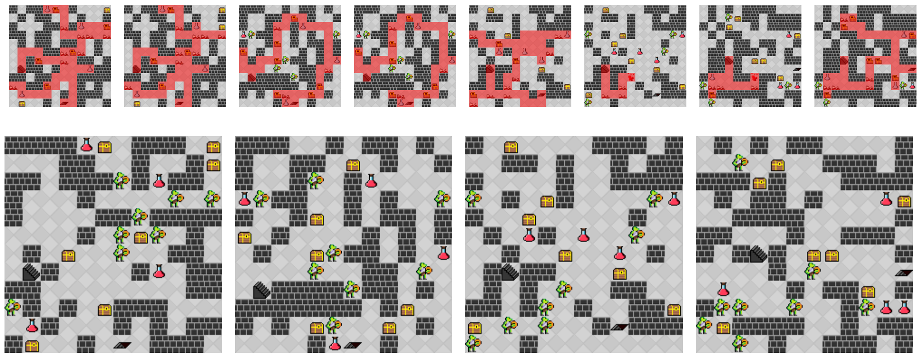
4.1 Discovery of feasible content

Despite the small map size of MiniDungeons, the constraints of connectivity of 21 level features and that of baseline persona survival were expected to make discovery of feasible individuals by random chance highly unlikely. Out of 10^6 randomly initialized levels, 360 were feasible (for all constraints) and 958 satisfied the constraints of connectivity, i.e. $u_N = 0$ in eq. (1). Evolving infeasible individuals allowed the FI-2pop GA to discover playable levels quickly despite the limited population size: the first feasible individual was discovered on average after 14.39 generations[2] (standard error: 1.40). This performance of the FI-2pop GA can be compared with a single population approach which handles infeasible individuals by applying the death penalty (i.e. fitness of 0). Using the same parameters as the FI-2pop GA and performing 20 evolutionary runs with each of eq. (4)–(7) (80 runs in total), the single population approach did not discover any feasible individuals in 21 of 80 runs (while all runs of the FI-2pop GA discovered playable levels). Moreover, among those runs where feasible individuals were found when using the death penalty, discovery of playable levels occurred after 35.42 generations (standard error: 0.84). As the difference in generation of discovery between FI-2pop GA and single-population GA is statistically significant ($p < 10^{-6}$ via two-tailed Student's t-test assuming unequal variances), it is clear that the FI-2pop GA can discover playable Minidungeons levels faster and more reliably.

4.2 Quality of feasible content

Figure 3 displays the best final evolved levels of 20 evolutionary runs, for each fitness function of eq. (4)–(7). To better demonstrate the levels' gameplay, each level is accompanied by a visualization of different playthroughs of the persona that evaluates it. Levels evolved towards F_{MK} tend to allow access from the entrance to the exit as well as to most potions (i.e. no monsters guard those level features); therefore it is the players' decision to pursue combat without it being forced upon them. Levels evolved towards F_{TC} tend to leave most treasures unguarded (in Fig. 3b only one treasure, near the exit, is guarded by a monster) and therefore collecting all treasures is not a risky choice for

[2] Since the infeasible fitness (d_{inf}) is the same for all experiments, discovery of the first feasible individual is calculated based on all four sets of experiments (F_{MK}, F_{TC}, D_{MK}, D_{TC}).



(a) F_{MK} (b) F_{TC} (c) D_{MK} (d) D_{TC}

Fig. 3: Best evolved levels for the different fitness functions of eq. (4)-(7). The levels shown have the highest fitness among 20 independent runs. Above each level are two playthroughs of the persona for which the level is evolved (Monster Killer or Treasure Collector), with randomized damage values for each monster.

the player. Levels evolved towards D_{MK} tend to place more monsters at chokepoints, therefore guarding many of the level's features such as the exit, potions and treasure: in Fig. 3c the hero must face a minimum of two monsters in order to reach the exit, and a minimum of three monsters to reach the treasures in the middle of the map (the exit tile can not be crossed as it ends the level). Levels evolved towards D_{TC} similarly place monsters at chokepoints: in Fig. 3d two monsters must be fought to reach the exit as well as the treasures in the middle of the map. While the Treasure Collector persona could theoretically have fought those two monsters and gained access to the 6 otherwise unguarded treasures, it opted to go for the bottom right treasure which often caused it to die. This odd decision demonstrates the bias introduced by the representation of the personas' controllers (the Treasure Collector went for the closest guarded treasure in this case) and by the levels they were evolved on (which rarely had so many monsters clustered in a map corner). This issue will be further discussed in Section 5.

To evaluate the quality of a generated level, the utility function of its persona-critic is a straightforward performance metric. Expanding on that, the quality of the persona's playthroughs in each level can be captured by other gameplay metrics, such as number of tiles explored, actions taken or times the persona died. Table 1 contains the gameplay metrics of the best final evolved levels as evaluated via 10 playthroughs of its persona-critic. Values in parentheses represent the deviation between playthroughs of the same level (rather than deviation between levels). For comparative purposes, Table 1 includes the gameplay metrics of the authored levels of Fig. 1, on which the personas were evolved. Observing Table 1, there is a clear difference between Monster Killer personas and Treasure Collector personas: Monster Killers kill far more monsters (unsurprisingly), drink more potions, die far more often and take much more damage than Treasure Collectors. Comparing between levels evolved towards F_{MK} and D_{MK}, the former can be played by a Monster Killer persona more efficiently: more monsters are killed, more potions drunk and less deaths occur than with D_{MK}. The high death ratio

	Monsters	Treasures	Potions	Explored	Actions	Death Ratio	Damage
Monster Killer							
Auth.	7.47 (0.47)	0.95 (0.21)	3.77 (0.15)	45.19 (4.57)	70.64 (10.34)	0.68 (0.42)	68.64 (5.25)
F_{MK}	7.91 (0.23)	2.90 (0.28)	3.99 (0.05)	44.34 (2.32)	78.81 (9.11)	0.43 (0.47)	72.63 (5.64)
D_{MK}	6.16 (1.15)	2.39 (0.45)	3.58 (0.51)	35.66 (6.09)	53.70 (12.75)	0.92 (0.13)	54.23 (6.16)
Treasure Collector							
Auth.	5.93 (0.42)	6.52 (0.61)	2.47 (0.49)	52.03 (6.12)	84.25 (14.42)	0.29 (0.42)	55.72 (4.18)
F_{TC}	2.68 (0.03)	7.00 (0.00)	3.57 (0.02)	43.57 (0.15)	75.89 (0.68)	0.00 (0.00)	25.29 (4.97)
D_{TC}	3.30 (0.83)	4.59 (2.07)	2.34 (1.29)	30.63 (12.39)	41.97 (21.55)	0.17 (0.19)	30.84 (9.26)

Table 1: Metrics of the best final levels, derived from simulations with procedural personas. Each level is simulated 10 times, and the value in the table represents the average of those 10 simulations, averaged again across the 20 independent runs of the GA. The value in parentheses represents the standard deviation of that metric within the 10 simulations (on the same level), and is also averaged across the 20 runs of the GA. Included are the gameplay metrics of the authored levels of Fig. 1: the values are averaged from 10 simulations, with deviation between simulations (on the same level) in parentheses.

of D_{MK} is a direct result of the fitness computation: the most straightforward way to achieve a larger deviation in monster kills is by dying prematurely. This is achieved in the map design by "hiding" potions behind multiple monsters, whereas maps evolved towards F_{MK} allow the hero to heal at any time (see Fig. 3). Comparing between levels evolved towards F_{TC} and D_{TC}, it is obvious that the former present minimal challenge to the Treasure Collector persona: with F_{TC}, all 7 rewards are always collected — without the hero ever dying — in every simulation and in every best final level. In contrast, with D_{TC} the hero collects less treasure with a high deviation in treasure collected between playthroughs, and has some chance of dying. Interestingly, the chance that the Treasure Collector dies is lower for maps evolved towards D_{TC} than for authored maps on which it was evolved; this is different than with maps evolved towards D_{MK}, where the death ratio is higher than for authored maps. Observing the Treasure Collector's actions in maps evolved for D_{TC}, its cautious tactics (compared to the Monster Killer) led it to rush to the exit when at low HP, since unguarded treasures were rarely available.

5 Discussion

The experiments in Section 4 demonstrated the impact of the FI-2pop GA in the swift and reliable discovery of playable Minidungeons levels. Moreover, the influence of the persona-critic was shown in the evolved dungeons' design patterns: levels evolved according to a Monster Killer had many unguarded potions while levels evolved according to a Treasure collector had many unguarded treasures. However, optimizing for most monsters killed or treasures collected resulted in Minidungeons levels of limited interest, especially for F_{TC} where there was no risk of dying when collecting all treasure. In contrast, maps evolved towards deviations between monsters killed (D_{MK}) or treasures collected (D_{TC}) featured a higher chance of dying for either persona, and therefore interesting risk/reward decisions. Maps evolved towards either D_{MK} or D_{TC} are superficially similar, as both fitnesses result in levels with more monsters guarding potions and treasure; the difference in gameplay metrics, therefore, is introduced by the different decisions and utility functions of the personas playtesting them. It may be worthwhile in future work to explore the potential of evolving maps based on how different the playthroughs between these two personas are.

However, the design patterns of evolved levels were biased by the personas' architecture as well as the levels that they were evolved on. Using two inputs for estimating the utility of treasure (closest treasure and closest unguarded treasure) works well for the authored levels the personas were evolved on (which had several unguarded treasures) but fell short when all treasures were guarded e.g. in Fig. 3d. Additionally, following a strategy such as "collect closest treasure" should avoid monsters when possible by using more sophisticated planning approaches than the ones currently in place. Finally, future work can explore how dungeons can be evolved according to personas with more elaborate utilities (e.g. a completionist persona targeting both monsters and treasure), or according to clones of human players, i.e. artificial agents evolved to match the decisions of a specific human player [15], thus providing personalized dungeons.

The algorithms covered in this paper can be applied to any problem that includes search in constrained spaces using simulations to evaluate content quality. Within games, procedurally generated content usually has to satisfy certain constraints; such constraints can be tested via planning [16], ensuring that a "perfect" or "worst-case" player can finish the game. However, in games with high stochasticity (e.g. roguelike games), with emerging tactics (e.g. multi-player strategy games) or where players don't always play optimally (e.g. sandbox games), simulations using one or more artificial agents to test the game can be useful both for playability checks (assuming more human-like perception, cognitive load and response times) and for evaluating the quality of completed playthroughs. Beyond games, constrained optimization is extensively applied in evolutionary industrial design [17] where simulations are often used to test robot locomotion or the performance of a machine part. The results of these simulations can act as constraints (e.g. minimal distance covered by a robot or lifetime of a machine part) in order to divide the search space into feasible and infeasible, allowing the FI-2pop GA to explore it using simulation-based fitnesses on the feasible and infeasible population.

6 Conclusion

This paper described a method for using procedural personas to evaluate the playability and quality of generated levels for the MiniDungeons game. Playability is determined by a "baseline" persona playing through a worst-case scenario of the level, with monsters dealing maximum damage. Using a two-population genetic algorithm to distinguish between feasible and infeasible content, discovery of playable levels is fast and reliable despite the highly constrained search space. To test the level's quality, a procedural persona simulates multiple playthroughs: a good level may require that the persona maximizes its utility or that the decisions taken by the persona affect its utility significantly. This paper tested two procedural personas, the Monster Killer and the Treasure Collector, and the final evolved levels demonstrated different map designs appropriate for each. Future work aims to improve the persona-critics, explore other simulation-based level evaluations, and increase the complexity of MiniDungeons.

Acknowledgements

The research was supported, in part, by the FP7 ICT project C2Learn (project no: 318480) and by the FP7 Marie Curie CIG project AutoGameDesign (project no: 630665).

References

1. van der Linden, R., Lopes, R., Bidarra, R.: Procedural generation of dungeons. IEEE Transactions on Computational Intelligence and AI in Games **6**(1) (2013) 78 – 89
2. Roden, T., Parberry, I.: From artistry to automation: a structured methodology for procedural content creation. In: Proceedings of the International Conference on Entertainment Computing. (2004) 151–156
3. Dormans, J.: Adventures in level design: generating missions and spaces for action adventure games. In: Workshop on Procedural Content Generation in Games. (2010)
4. Hartsook, K., Zook, A., Das, S., Riedl, M.: Toward supporting stories with procedurally generated game worlds. In: Proceedings of the IEEE Conference on Computational Intelligence and Games. (2011) 297–304
5. Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games **3**(3) (2011) 172–186
6. Holmgård, C., Liapis, A., Togelius, J., Yannakakis, G.N.: Evolving personas for player decision modeling. In: Proceedings of the IEEE Conference on Computational Intelligence and Games. (2014)
7. Kimbrough, S.O., Koehler, G.J., Lu, M., Wood, D.H.: On a feasible-infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. European Journal of Operational Research **190**(2) (2008) 310–327
8. Holmgård, C., Liapis, A., Togelius, J., Yannakakis, G.N.: Generative agents for player decision modeling in games. In: Poster Proceedings of the 9th Conference on the Foundations of Digital Games. (2014)
9. Liapis, A., Yannakakis, G., Togelius, J.: Sentient sketchbook: Computer-aided game level authoring. In: Proceedings of the ACM Conference on Foundations of Digital Games. (2013)
10. Kahneman, D., Tversky, A.: Prospect theory: An analysis of decision under risk. Econometrica: Journal of the Econometric Society **47** (1979) 263–291
11. Gibson, J.J.: The theory of affordances. Perceiving, Acting, and Knowing (1977) 67–82
12. Liapis, A., Yannakakis, G.N., Togelius, J.: Towards a generic method of evaluating game levels. In: Proceedings of the AAAI Artificial Intelligence for Interactive Digital Entertainment Conference. (2013)
13. Liapis, A., Yannakakis, G.N., Togelius, J.: Generating map sketches for strategy games. In: Proceedings of Applications of Evolutionary Computation. Volume 7835, LNCS., Springer (2013) 264–273
14. Björk, S., Holopainen, J.: Patterns in Game Design. Charles River Media (2004)
15. Holmgård, C., Liapis, A., Togelius, J., Yannakakis, G.N.: Personas versus clones for player decision modeling. In: Proceedings of the International Conference on Entertainment Computing. (2014)
16. Horswill, I., Foged, L.: Fast procedural level population with playability constraints. In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference. (2012)
17. Michalewicz, Z., Dasgupta, D., Le Riche, R., Schoenauer, M.: Evolutionary algorithms for constrained engineering problems. Computers & Industrial Engineering Journal **30** (1996) 851–870