Generating Map Sketches for Strategy Games

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

[1] Center for Computer Games Research, IT University of Copenhagen, Copenhagen, Denmark
[2] Department of Digital Games, University of Malta, Msida, Malta

Abstract. How can a human and an algorithm productively collaborate on generating game content? In this paper, we try to answer this question in the context of generating balanced and interesting low-resolution sketches for game levels. We introduce six important criteria for successful strategy game maps, and present map sketches optimized for one or more of these criteria via a constrained evolutionary algorithm. The sketch-based map representation and the computationally lightweight evaluation methods are geared towards the integration of the evolutionary algorithm within a mixed-initiative tool, allowing for the co-creation of game content by a human and an artificial designer.

1 Introduction

The games industry has often used procedurally generated content to increase the unexpectedness of a game, and thus its replayability value. As games increase in scale, it is becoming common practice for game designers to use procedural content generation tools during development time in order to limit costs and time requirements. In order to assist such design work, computer-assisted design tools should be able to automate the mechanizable parts of content creation (such as ensuring playability and evaluating game balance) and to optimize, on their own, specific gameplay features deemed significant by the designer. Not only would such a tool increase a designer's creative output, it should be able to incite human creativity through the dialogue between the artificial and the human designer.

This paper presents steps towards actualizing such a tool, using maps for strategy games as its test domain. Six important measures of quality for strategy game maps are introduced, inspired by game design patterns [1] popular within strategy games and by previous experiments on map evolution [2]. These criteria are optimized via evolutionary algorithms and their impact on map generation is evaluated. The map generation component is integrated within a mixed-initiative design tool which allows for the co-creation of strategy maps with designers. With the proposed tool we try to overcome some of the limitations of mixed-initiative design as we rely on simple map sketches to counter user fatigue and designer biases reported in the literature [3].

2 Related Work

While procedural content generation has been used in some games since the eighties, recent trends have seen an increasing use of PCG tools such as *SpeedTree*[3] during de-

[3] http://www.speedtree.com

velopment time, to partly automate design work. PCG tools have to balance between expressivity and controllability, with methods capable of producing a wide range of content usually being very hard for designers to work with. Academic interest on more controllable PCG methods is only a few years old [4], but search-based processes [5] such as genetic algorithms are becoming a popular solution to this problem [6–9].

Maps have often been generated algorithmically in games such as *Civilization* (MicroProse, 1991) and *SimCity* (Maxis, 1989); such games usually rely on tightly designed processes to construct playable maps. Within academia, strategy game maps have also been optimized via stochastic search [10, 2] or answer-set solvers [11]; while designers can decide on the objectives or constraints of the generated content prior to the generative process, these projects are hardly interactive. In the authors' previous work, a mixed-initiative tool attempted to address the issues of authorial control and capturing human preferences [3]. The tool failed to achieve its stated goals, primarily due to the requirement of hand-crafting a large-scale initial map, which introduced designer bias and fatigue. In order to counter such limitations, this paper reduces the resolution of the human-authored sketches and allows more user control over the optimized features.

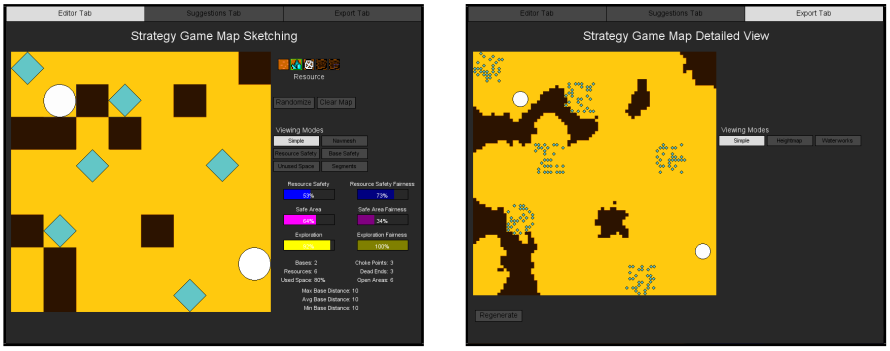
3 Methodology

The tool presented in this paper allows a human or an artificial designer to create low-resolution map sketches. These sketches contain the necessary elements for a simple strategy game, and can be optimized via a genetic algorithm on a number of selected fitness dimensions pertaining to balanced strategic gameplay.

The maps used in this experiment are abstractions of levels used in successful strategy games such as *Starcraft* (Blizzard, 1998). A map is presented to the user as a sketch consisting of a small number of *tiles* (see Fig. 1a). Tiles can be *passable* (light) or *impassable* (dark), and passable tiles can contain *player bases* (circles) or *resources* (rhombi). The map layout assumes that each player starts at a base and gathers resources to produce units; units move through passable tiles to attack the opponent's base.

3.1 Map Design Tool

A graphical interface has been developed to allow a user to manually edit a map sketch (see Fig. 1a). While the sketch is being edited, its scores in several fitness dimensions (see Section 3.2 below) are updated and displayed to the user; the user can also select one or more fitness labels and generate an optimal map on the selected fitness dimensions. At any point, the designer can switch to the final map view (see Fig. 1b), displaying the complete map on which the strategy game can be played. Currently the final map is constructed via random processes and cellular automata to create an organic-looking map which however retains all the properties (chokepoints, passable paths) of the low-resolution sketch. Manual editing, evaluation and evolution (see Section 3.2) are all done on the sketch level, making computations such as pathfinding easier and reducing the required human effort.



(a) Sketching interface while the user generates a rough level sketch.

(b) The sketch in Fig. 1a rendered as a complete map, generated with cellular automata.

Fig. 1: The User Interface for the mixed-initiative level generation tool.

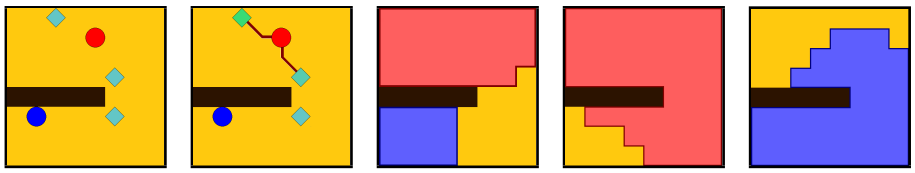
3.2 Evolutionary Optimization

Each map is encoded as an array of integers: each integer represents a tile's type (passable, impassable, base or resource). These parameters are adjusted via constrained optimization — ensuring the playability of feasible maps — carried out by a feasible-infeasible two-population genetic algorithm [12] (FI-2pop GA). FI-2pop GA evolves two populations, one with feasible maps and the other with infeasible maps. Each population selects parents among its own members, but feasible offspring of infeasible parents are moved to the feasible population and vice versa. This interbreeding increases the occurrence of feasible individuals and boosts the feasible population's diversity.

Both populations evolve via fitness-proportionate roulette-wheel selection of parents; parents are recombined using two-point crossover. Mutation can occur on an offspring of two parents (1% chance), or on a single parent (5% chance) in order to create a single-parent offspring. During mutation, 2 to 6 tiles may be transformed: a tile may be swapped with its adjacent (15% chance), an impassable tile can be transformed into passable and vice versa (5% chance) or a passable tile can be transformed into a resource (1% chance). The small number of tiles transformed with each mutation increases the locality (in terms of map structure) of the stochastic search; preliminary tests have shown that when this mutation strategy is combined with the relatively high chance of mutation chosen, the population does not suffer from premature convergence.

The fitnesses used to evolve the feasible population are presented below, while the infeasible fitness is presented at the end of this section.

Feasible Fitnesses: A feasible map sketch is evaluated on six fitness dimensions (see (3) to (8) below) which are inspired by game design patterns [1] suitable for strategic gameplay and geared towards game pace and player balance in terms of starting conditions. Game pace is affected by the *area control* afforded by a player's starting location — including control of *strategic resources* — and by the challenge for enemies to discover this location via *exploration*. If a player has an easily controllable, resource-rich



(a) (b) (c) (d) (e)

Fig. 2: Visualization of resource safety, safe areas and exploration on a test map (Fig. 2a) with impassable tiles (dark), resources (rhombi), base 1 at the map's top and base 2 at the bottom. Resources at the map's top are much closer to base 1 than to base 2, and therefore have large $s_{t,1}$ values (connected in Fig. 2b). The bottom-most resource has an equal distance between the two bases, therefore it is contested ($s_{t,1}{\approx}s_{t,2}{\approx}0$). Figure 2c shows the areas around the two bases with safety values over C_s (A_1 at the top, A_2 at the bottom). By applying flood fill from base 1 until base 2 is covered (Fig. 2d) and from base 2 until base 1 is covered (Fig. 2e), we calculate $E_{1\to2}$ and $E_{2\to1}$ respectively.

area around their base and their base is difficult to reach by enemies, then defensive play is favored and game pace is slow. If a base is within the enemy's reach, and resources are in contested and difficult to control areas, then fast-paced aggressive play is favored. On the other hand, *player balance* is a universal design pattern for any multi-player game [1], and is captured in this paper as the *symmetry* in affordances for game pacing among players. The concepts of safety, fairness and path overlap presented in this paper have been covered in previous work [2], but they are evaluated differently: while the safety metric in (1) is similar, base safety and exploration are only loosely captured in [2] by *base space* and *base distance*, respectively; more emphasis is also placed on balance, with three fitnesses rather than the single *resource fairness* of [2].

In order to evaluate area control and exploration, two heuristics for *safety* and *exploration* are used in the calculation of the fitnesses in (3)–(8). The safety metric ($s_{t,i}$) in (1) evaluates a tile t according to its safety with respect to a player base i; the closer the tile is to base i compared to any other base, the larger its safety value. The exploration metric (E_i) evaluates the effort required to discover all other player bases from base i; it uses a simple flood fill algorithm to simulate random exploration of the map, which ends once one base is discovered and runs again for every other base. It is calculated as per (2), and has high values for distant bases and if open areas exist between bases.

$$s_{t,i} = \min_{\substack{1\le j\le N_B \\ j\neq i}} \left\{ max\left\{0, \frac{d_{t,j} - d_{t,i}}{d_{t,j} + d_{t,i}}\right\}\right\} \tag{1}$$

$$E_i = \frac{1}{N_B - 1} \sum_{\substack{j=1 \\ j\neq i}}^{N_B} \frac{E_{i\to j}}{w_m h_m - N_I} \tag{2}$$

where N_B is the number of bases; $d_{t,i}$ is the distance from tile t to base i (using A* pathfinding); w_m and h_m is the map's width and height, respectively; N_I is the number

of impassable tiles and $E_{i \to j}$ is the map coverage when a four-direction flood fill is applied starting from base i and stopping once base j has been found (see Fig. 2).

The *resource safety* fitness (f_{res}) in (3) uses the safety metric from (1) to calculate the safety of the map's resources. Low scores in this fitness correspond to maps with resources in contested areas, equally accessible to two or more player bases. The *base safety* fitness (f_{saf}) in (4) calculates the safe areas around every player's base. Low scores in this fitness correspond to maps with insecure bases, since many areas around them are easily accessible by at least one enemy base. The *exploration* fitness (f_{exp}) in (5) uses the exploration metric from (2) to simulate the difficulty in finding other bases from each player's base.

$$f_{res} = \frac{1}{N_R} \sum_{j=1}^{N_R} \max_{1 \leq i \leq N_B} \{s_{t_j,i}\} \tag{3}$$

$$f_{saf} = \frac{1}{w_m h_m - N_I} \sum_{i=1}^{N_B} A_i \tag{4}$$

$$f_{exp} = \frac{1}{N_B} \sum_{i=1}^{N_B} E_i \tag{5}$$

$s_{t_j,i}$ is the safety metric of resource j (located at tile t_j) to base i; A_i is the map coverage of safe tiles for base i and E_i is the exploration metric for base i. A tile t is safe for base i if its $s_{t,i} < C_s$; the constant $C_s = 0.35$ throughout this paper, as it amounts to a good ratio of contested areas in most maps (see Fig. 2c).

Fitnesses in (3)–(5) do not differentiate between players; for instance, high scores in f_{res} can correspond to a map where all resources are safe for only one base. Since competitive strategy games favor equivalent starting conditions for each player, the fitnesses in (6)–(8) are evaluated with regards to player balance. Thus, resource balance (b_{res}), base safety balance (b_{saf}), and exploration balance (b_{exp}) are calculated as follows:

$$b_{res} = 1 - \frac{1}{N_R N_B (N_B - 1)} \sum_{k=1}^{N_R} \sum_{i=1}^{N_B} \sum_{\substack{j=1 \\ j \neq i}}^{N_B} |s_{t_k,i} - s_{t_k,j}| \tag{6}$$

$$b_{saf} = 1 - \frac{1}{N_B (N_B - 1)} \sum_{i=1}^{N_B} \sum_{\substack{j=1 \\ j \neq i}}^{N_B} \frac{|A_i - A_j|}{max\{A_i, A_j\}} \tag{7}$$

$$b_{exp} = 1 - \frac{1}{N_B (N_B - 1)} \sum_{i=1}^{N_B} \sum_{\substack{j=1 \\ j \neq i}}^{N_B} \frac{|E_i - E_j|}{max\{E_i, E_j\}} \tag{8}$$

Infeasible Fitness: Infeasible maps fail to satisfy playability constraints (having unreachable bases or resources) or designer specifications regarding the number of map features. In order to increase the chances of infeasible parents creating feasible offspring, the infeasible population must shift its members towards the border with feasibility [13]. The infeasible population optimizes its members according to the infeasible

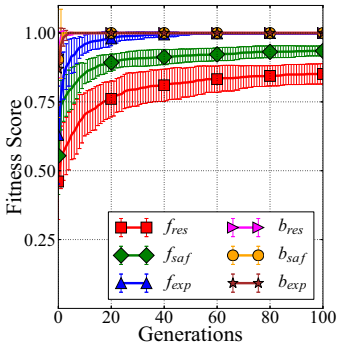
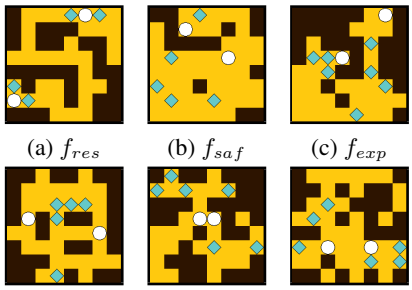


Fig. 3: Optimization of the populations' maximum fitness scores when evolving strategy maps on a single fitness. Error bars represent standard deviation across 20 runs.



(a) f_{res} (b) f_{saf} (c) f_{exp}

(d) b_{res} (e) b_{saf} (f) b_{exp}

Fig. 4: Best final individuals among 20 evolutionary runs, optimized for a single fitness dimension displayed in each map's caption.

fitness (f_{inf}) in (9), which aims to minimize the distance from feasibility. This distance from feasibility has four components, equal to the number of constraints for feasible maps: a) fidelity with designer-specified number of bases, b) fidelity with designer-specified number of resources, c) passable paths between bases and d) passable paths between resources and bases.

$$f_{inf} = 1 - \left[\frac{1}{4}|N_B - N_{B,d}| + \frac{1}{4}|N_R - N_{R,d}| + \frac{1}{4}\frac{2d_b}{N_B(N_B-1)} + \frac{1}{4}\frac{d_r}{N_R N_B}\right] \quad (9)$$

where $N_{B,d}$ and $N_{R,d}$ is the designer-specified allowed number of bases and resources, respectively, and d_b and d_r is the number of base pairs and base-resource pairs that are not connected, respectively.

4 Experiments

Several experiments were conducted to test the efficiency of the genetic algorithm used to optimize the rough map sketches of the mixed-initiative tool. These experiments were conducted without any user interaction or human-authored sketches, and assess the algorithm's ability to optimize one or more fitness dimensions. All experiments presented below run for 100 generations, on a population of 100 individuals including both feasible and infeasible genes; the number of individuals is large enough for the simultaneous optimization of two populations while allowing for sufficiently fast evolutionary runs as demanded by a responsive mixed-initiative tool. Evolving maps have 64 tiles (with equal width and height of 8 tiles), 2 bases and anywhere between 4 and 10 resources.

4.1 Optimizing a single fitness dimension

Using a single fitness dimension as the objective function, the genetic algorithm aims to optimize a single gameplay feature; the generated maps are, thus, expected to be

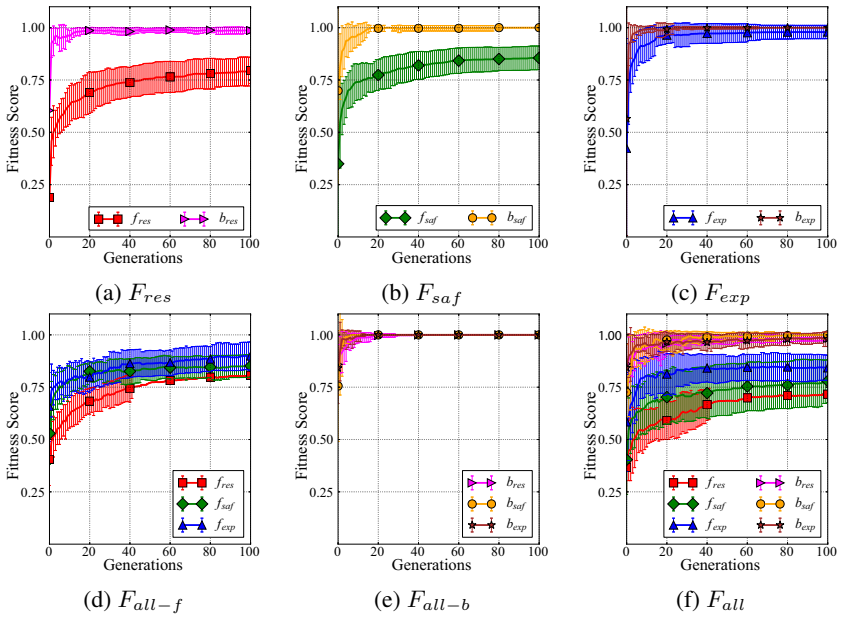
one-sided and lack the necessary features for competitive strategy play. Figure 3 shows the evolutionary progress of the maximum fitness in the population; displayed values are averaged from 20 independent runs, with the standard deviation shown as error bars. The highest scoring final individual among the 20 runs for each fitness dimension is shown in Fig. 4. Results indicate that optimal fitness scores for balance (b_{res}, b_{saf}, b_{exp}) are easily attainable, since high-scoring individuals exist even in the random initial populations and evolution quickly finds optimal solutions for these fitness dimensions within few generations. Observing the maps in Fig. 4, optimal maps in b_{res} have symmetrical resources between players; resources are often far from player bases, so that differences between their distances from each base remain relatively small. Optimal maps in b_{saf} have bases near each other and safe areas are small but equal between the bases. Optimal maps in b_{exp} often have bases near each other (even adjacent), and finding the other base is equally effortless for either player. Among the other dimensions, f_{res} is the most difficult to optimize, mainly because of how the safety metric is calculated: based on (1), $s_{t,i}$ cannot reach its optimal value (1.0), but approaches it if $d_{t,j} \gg d_{t,i}$ for all bases $j \neq i$. Granted that the small size of the maps cannot allow such disparities between distances, it is expected that even in the best circumstances $s_{t,i}$ and thus f_{res} will be considerably lower than 1.0. Similarly, f_{saf} cannot reach optimal values since there will be at least one tile in the map at equal distance from both bases (and thus not safe). The best maps in f_{res} have resources adjacent to bases, while the bases are far from each other. The best maps in f_{saf} have numerous impassable regions between bases; in Fig. 4b, one base is hidden behind impassable tiles while the other base has safe access to the rest of the map. The best maps in f_{exp} have bases far from each other, with impassable regions between them to make navigation more difficult.

4.2 Optimizing multiple fitness dimensions

Experiments in Section 4.1 showed that maps optimized for a single dimension usually have interesting traits, but lack the necessary features needed for competitive play. Combining multiple fitness dimensions into a weighted sum and using it as the objective function for the genetic algorithm is expected to generate better designed maps. Experiments in this section will assess the combined optimization of two or more fitness dimensions. For space considerations, the following representative fitness function combinations are explored in this paper:

- $F_{res} = \frac{1}{2} f_{res} + \frac{1}{2} b_{res}$
- $F_{saf} = \frac{1}{2} f_{saf} + \frac{1}{2} b_{saf}$
- $F_{exp} = \frac{1}{2} f_{exp} + \frac{1}{2} b_{exp}$
- $F_{all-f} = \frac{1}{3} f_{res} + \frac{1}{3} f_{saf} + \frac{1}{3} f_{exp}$
- $F_{all-b} = \frac{1}{3} b_{res} + \frac{1}{3} b_{saf} + \frac{1}{3} b_{exp}$
- $F_{all} = \frac{1}{6} f_{res} + \frac{1}{6} f_{saf} + \frac{1}{6} f_{exp} + \frac{1}{6} b_{res} + \frac{1}{6} b_{saf} + \frac{1}{6} b_{exp}$

Figure 5 shows the evolutionary progress of the contributing fitness scores for the fittest individuals in the different fitness combinations; displayed values are averaged from 20 independent runs, with the standard deviation values depicted as error bars. The highest scoring final individual among the 20 runs for each fitness combination is

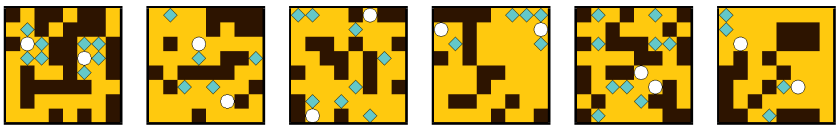


(a) F_{res} (b) F_{saf} (c) F_{exp}

(d) F_{all-f} (e) F_{all-b} (f) F_{all}

Fig. 5: Optimization of the contributing fitness dimensions in the fittest individuals in the population, when evolving strategy maps for multiple fitness dimensions. Error bars represent standard deviation across 20 runs.

shown in Fig. 6. Results indicate that, for F_{res}, F_{saf} and F_{exp}, optimization is dominated by the fitness dimension of balance, which is optimized quickly and largely determines the selection of parents; since f_{res} and f_{saf} are slower to optimize, they do not achieve as high scores as when optimized individually. Their best maps in Fig. 6 are, however, of good quality: F_{res} has an equal number of resources adjacent to each base, while F_{saf} has a single chokepoint in the map, with areas on either side of the chokepoint being of equal size; finally, F_{exp} has bases far away from each other, hidden behind large impassable regions. On the more complex fitness combinations, F_{all-b} easily finds optimal maps in all the contributing fitnesses; F_{all-f}, on the other hand, does not achieve as high scores as when each dimension is optimized on its own, but it does not seem to be dominated by any dimension. Finally, the optimization of F_{all} unsurprisingly shows difficulties in reaching high fitness scores in all contributing fitnesses; fitness dimensions of balance dominate f_{res}, f_{saf} and f_{exp}, which are harder to optimize and show higher sensitivity with respect to their convergence (as depicted by their large standard deviation values). The best maps for F_{all-f} have all the features of the contributing fitness dimensions, but are very unbalanced. The best maps for F_{all-b} have bases adjacent to each other, since that is often optimal both for b_{exp} and b_{saf}; such maps are not generally playable in a strategy game. Despite the slow and asymmetrical optimization of F_{all}, its best maps are probably the most appropriate for use in a strategy game. Even better maps may be possible with other combinations of criteria, such as minimizing f_{saf} and f_{res} to create maps suited for aggressive gameplay.



(a) F_{res} (b) F_{saf} (c) F_{exp} (d) F_{all-f} (e) F_{all-b} (f) F_{all}

Fig. 6: Best final individuals among 20 evolutionary runs, optimized for a combination of two or more fitness dimensions displayed in each map's caption.

5 Conclusion

This paper presented the concept of map sketches which are appropriate for a mixed-initiative tool allowing for the collaborative creation of maps for strategy games. With this tool, a computer can evaluate the human-authored map in real-time, and the user can request improved maps in several fitness dimensions. Since maps are represented as low-resolution sketches, computational time for map evaluation and optimization is minimized, while designer fatigue and fixation are also expected to be reduced. Experiments conducted on random initial populations, with sufficient time to evolve, demonstrated that small two-player maps of high quality can be easily optimized by the genetic algorithm on one or more fitness dimensions. While the combination of more fitness dimensions into a weighted sum limited the efficiency of optimization somewhat, the evolved maps are more appropriate for use in strategy games than those optimized on a single fitness dimension. Aggregating multiple objectives into a single fitness has its caveats, although the fitness dimensions combined in this paper are not particularly conflicting. While multi-objective evolutionary algorithms such as those used in [2] could potentially lead to better results, such algorithms are generally more computationally demanding and thus inappropriate for the fast feedback mechanisms of a mixed-initiative tool. Preliminary tests show that, while slower to evolve, the same processes can optimize, to a high quality, larger maps with more bases and resources.

Future work includes using the evolutionary methods described in this paper to optimize a user-created map via the map editor. The experiments presented in this paper were strictly offline, starting from large random populations and without any time constraint. Online evolution while a user edits the map may have its own challenges, particularly since the general form of the user-created map must be retained. Additionally, while the fitness scores are fast to calculate, they are built on design decisions that may not be accurate enough: for instance, exploration is measured via flood fill (which is not how players explore maps in strategy games), while the safety metric underestimates the impact of chokepoints. Future work should refine the existing fitnesses and possibly add new ones; however, the more fitnesses are added, the more difficult their simultaneous optimization will become. Finally, while the user may enjoy controlling which fitness dimension is being optimized, it may also become cumbersome and unintuitive to guess which combination of fitness dimensions are needed for the designer's purposes. This can be addressed by indirectly modelling the designer's intentions based on their history of content authoring and content selection through *choice-based inter-*

active evolution, which has been used in previous work to automatically adapt models of aesthetic preferences [14].

Acknowledgements

The research is supported, in part, by the FP7 ICT project SIREN (project no: 258453) and by the FP7 ICT project C2Learn (project no: 318480).

References

1. Bjork, S., Holopainen, J.: Patterns in Game Design. Charles River Media (2004)
2. Togelius, J., Preuss, M., Beume, N., Wessing, S., Hagelback, J., Yannakakis, G.N.: Multiobjective exploration of the StarCraft map space. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games. (2010) 265–272
3. Liapis, A., Yannakakis, G.N., Togelius, J.: Limitations of choice-based interactive evolution for game level design. In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference. (2012)
4. Yannakakis, G.N.: Game AI revisited. In: Proceedings of the ACM Computing Frontiers Conference. (2012) 285–292
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 (99) (2011) 172–186
6. Browne, C., Maire, F.: Evolutionary game design. IEEE Transactions on Computational Intelligence and AI in Games **2**(1) (2010) 1–16
7. Sorenson, N., Pasquier, P., DiPaola, S.: A generic approach to challenge modeling for the procedural creation of video game levels. IEEE Transactions on Computational Intelligence and AI in Games **3**(3) (2011) 229–244
8. Hastings, E.J., Guha, R.K., Stanley, K.O.: Evolving content in the galactic arms race video game. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games. (2009) 241–248
9. Liapis, A., Yannakakis, G.N., Togelius, J.: Neuroevolutionary constrained optimization for content creation. In: Proceedings of the IEEE Conference on Computational Intelligence and Games. (2011) 71–78
10. Frade, M., de Vega, F.F., Cotta, C.: Evolution of artificial terrains for video games based on accessibility. In: Proceedings of EvoApplications. Volume 6024, LNCS., Springer (2010) 90–99
11. Smith, A., Mateas, M.: Answer set programming for procedural content generation: A design space approach. IEEE Transactions on Computational Intelligence and AI in Games **3**(3) (2011) 187–200
12. 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
13. Schoenauer, M., Michalewicz, Z.: Evolutionary computation at the edge of feasibility. In: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, Springer-Verlag (1996) 245–254
14. Liapis, A., Yannakakis, G.N., Togelius, J.: Adapting models of visual aesthetics for personalized content creation. IEEE Transactions on Computational Intelligence and AI in Games **4**(3) (2012) 213–228