A Self-Adaptive Evolutionary Approach to the Evolution of Aesthetic Maps for a RTS Game

Raúl Lara-Cabrera, Carlos Cotta and Antonio J. Fernández-Leiva [*]

Universidad de Málaga, Andalucía Tech, Departamento de Lenguajes y Ciencias de la Computacin.

Abstract

Procedural content generation (PCG) is a research field on the rise, with numerous papers devoted to this topic. This paper presents a PCG method based on a self-adaptive evolution strategy for the automatic generation of maps for the real-time strategy (RTS) game Planet Wars. These maps are generated in order to fulfill the aesthetic preferences of the user, as implied by her assessment of a collection of maps used as training set. A topological approach is used for the characterization of the maps and their subsequent evaluation: the sphere-of-influence graph (SIG) of each map is built, several graph-theoretic measures are computed on it, and a feature selection method is utilized to determine adequate subsets of measures to capture the class of the map. A multiobjective evolutionary algorithm is subsequently employed to evolve maps, using these feature sets in order to measure distance to good (aesthetic) and bad (non-aesthetic) maps in the training set. The so-obtained results are visually analyzed and compared to the target maps using a Kohonen network.

1 Introduction

Within the field of computational intelligence we find the procedural content generation (PCG) [1] for games, which represents a set of algorithmic techniques used to automatically generate game content, whether it be maps, levels, 3D models, music and/or even rules. The employment of PCG provides several advantages to the development of video games, such as a reduction in the expenses linked to manually creating the content, a reduced amount of used memory and the ability to create infinite games (e.g., new levels/characters/missions can be automatically created to extend the lifetime of the game). PCG has been applied adequately during the development of several successful commercial games such as Borderlands saga or the sandbox games Minecraft and Terraria.

PCG covers various methodologies, being our work focused on the search-based procedural content generation (SBPCG) [2]. This search-based methodology consists in generating a substantial amount of game resources in order to find, through a guided search (e.g., via meta-heuristics) those assets that

[*]Email: {raul, ccottap, afdez}@lcc.uma.es).

meet a certain level of quality. Therefore, it is necessary to define the quality criterion for the generated content. The quality criteria we used in this study was how aesthetic the generated resources were (more specifically, the content we consider are maps for a real-time strategy game). In our earlier work [3, 4], we used two different criteria when generating and evaluating the maps, in particular *balance* (i.e., no player gets an advantage over the other) and *level of dynamism* (measured in terms of resource fluctuations or by evaluating the nature of confrontations between the players). The goal in both cases was to produce interesting games providing atractive challenges to the players. While fulfilling our requirements of balance and dynamism, the generated maps lacked aesthetic (for example, maps with their planets clustered in a small region), an interesting property that a map might has and that may conduct to obtain even more appealing games; for this reason, we now analyze how to incoporate this feature to our generated maps.

The chosen game, *Planet Wars*, is a real-time strategy game developed in the context of the Google AI Challenge competition. This game genre offers a wide variety of artificial intelligence research problems [5] (such as opponent modelling, spatial and temporal reasoning, resource management and real-time planning). Planet Wars is a game about conquering planets and fighting space battles between several players. The goal is to conquer all the planets while trying to eliminate all opponents. These planets, which can hosts ships and are distributed over a 2-dimensional plane, may belong to any player or be neutral (that is, the planet belongs to nobody). Non-neutral maps produce new ships every turn, depending on their size. Players send groups of ships (i.e., fleets) from planets they control to other ones. If the player owns the target planet then the number of arriving ships is added to the number of defending ships on that planet, otherwise a battle takes place in the target planet: ships of both sides destroy each other so the player with the highest number of ships owns the planet (with a number of ships determined by the difference between the initial number of ships). The distance between the planets affects the required time for a fleet to arrive to her destination, which remains fixed during the fleet movement (that is, it is not possible to change the destination of a moving fleet). Although the game is played in turns, players issue their orders at the same time, so it can be considered as a real-time game. Games can end when a player has defeated the other or when the maximum number of turns is reached (the winner is the player with the highest number of controlled planets).

2 Background

RTS games have been used many times as a tool to measure the efficiency and performance of artificial intelligence techniques [6]. This kind of games offer several fundamental AI research problems, such as adversarial real-time planning, opponent modelling, spatial and temporal reasoning, decision making under uncertainty and resource management. As in this paper, PCG techniques usually generate maps for differente games, as displayed by the considerable number of papers on this topic [1]. For example, Togelius et al. [7] presented a PCG system that was able to create tracks for a racing game using a parameter vector with a deterministic genotype-to-phenotype mapping. Frade et al. proposed the use of genetic programming to evolve maps for videogames, (i.e., terrain programming), using either subjective human-based feedback [8, 9] or automated quality measures such as accessibility [10] or edge-length [11]. Another example by Mahlmann et al. [12], who suggested a search-based map generator based on the transformation of low resolution matrices into higher resolution maps (using cellular automatas) for a simplified version of the RTS game *Dune II*.

3 Map Characterization

The maps from Planet Wars are made of a certain number of planets n_p distributed over a 2D plane. Every planet is defined by their position on the plane (coordinates (x_i, y_i)), their size s_i and a number of ships w_i. The size s_i sets the rate at which a planet will generate new ships every turn (as long as the planet is controlled by any player) while the remaining parameter, w_i, indicates the number of defending ships. We denoted a map as a list $[\vec{\rho}_1, \vec{\rho}_2, \cdots, \vec{\rho}_{n_p}]$, where each $\vec{\rho}_i$ is a tuple $\langle x_i, y_i, s_i, w_i \rangle$. It is mandatory to specify the initial home planets of the players, which were fixed as the first two planets $\vec{\rho}_1$ and $\vec{\rho}_2$. The number of planets n_p is not fixed and should range between 15 and 30 as set by the rules of the *Google AI Challenge 2010* (this variable number of planets is included in the self-adaptive evolutionary approach described later on).

We characterized every map by computing several measures from the map's spheres-of-influence graph (SIG) [13]. This kind of graph can be used to extract information about a set of points and establishes a relationship between them, which is based on their spatial arrangement and is not affected by rotation, translation and scaling (see Figure 1). SIGs can be applied to many research areas and applications, such as in computer vision (as a low-level operator for object recognition from input dot patterns), data mining (for clustering objects with similar attribute values) and computer graphics (for defining surfaces over point clouds). The procedure for generating the SIG for a certain map is as follows:

1. For each planet p_i placed at (x_i, y_i), compute the distance d_i to the closest planet
$$d_i = \min_j \{\|(x_i - x_j, y_i - y_j)\| \mid j \neq i\} \tag{1}$$
where $\|v\|$ is the Euclidean norm of vector v.

2. For each planet p_i, draw a circle with center (x_i, y_i) and radius d_i

3. Build the SIG as a graph $G(V, E)$, where $V = \{p_1, \cdots, p_{n_p}\}$ and $(u, v) \in E$ iff the circles with centers in u and v intersect.

We gathered two sets of maps, which were used later as a baseline to compare with, one of them containing 100 maps with good aesthetics and the other one including 100 non-aesthetic maps. An expert evaluated the maps and tagged them as aesthetic or non-aesthetic, based on her perceptions and artistic criteria. Then we computed several measures relating to the previously generated SIGs, namely:

- Number of connected components (sub-graphs in which any two vertices are connected by paths).

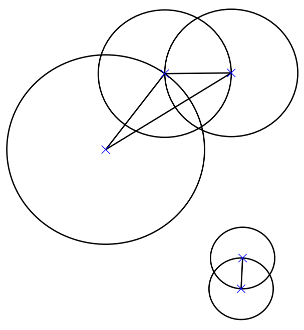
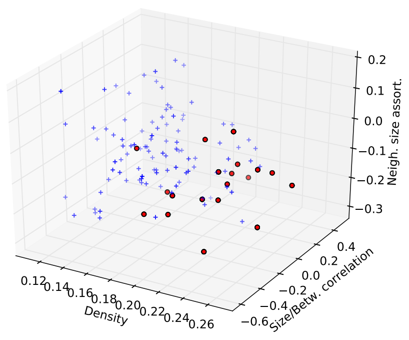
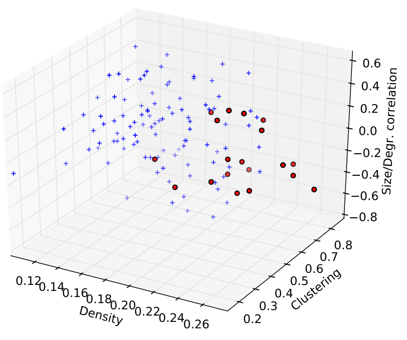


Figure 1: A sphere-of-influence graph.



(a)



(b)

Figure 2: Distribution of aesthetic (circles) and non-aesthetic (crosses) maps according to the values of the variables from group A (a) and B (b)

- Average node's degree (number of edges incident to the node).
- Density of the graph, which measures the ratio between number of edges and nodes (specifically for undirected graphs $\delta = 2m/(n(n-1))$, with n and m being the number of nodes (i.e. n_p) and edges respectively).
- Average clustering coefficient. The clustering coefficient c_i of a node quantifies how interconnected (or grouped) with his neighbors it is. Mathematically, $c_i = 2E_i/\left[k_i(k_i-1)\right]$ where E_i is the number of edges connecting the immediate neighbors of node i and k_i is the degree of node i.
- Pearson correlation between the size of the nodes and their betweenness. The latter is a measure of the importance as intermediate gateway of a node in a graph, as is computed as the average fraction of shortest paths between any two nodes that go through a third certain node.
- Pearson correlation between the size of the nodes and their degree.

- Size assortativity, i.e., Pearson correlation coefficient between the size of nodes connected in the graph.

Subsequently, we created a Random Forest classifier for each possible combination of the above measures and we evaluated the performance of the classifiers by a Leave-one-out (LOO) cross-validation. Each classifier had 50 estimators (i.e., decision trees), bootstrap sampling and the Gini impurity criterion as the function to measure the quality of the splits. After analyzing the AUC (Area under ROC curve) values of the different combinations, we decided to use the following groups of variables as the map characterization (see Figure 2):

- **Group A:** Graph's density (δ), correlation between node size and betweenness (ρ_{SB}), and size assortativity (ρ_{Size}) (i.e., Pearson correlation coefficient between the size of connected planets). ($AUC = 0.9916$)
- **Group B:** Graph's density (δ), clustering coefficient ($\overline{c_i}$) and Pearson's correlation coefficient between planets' sizes and their degree (ρ_{SG}). ($AUC = 0.9984$)

We compared the similarity between maps using these two groups of features in the following way: each map is characterized by a tuple $\langle \delta, \rho_{SB}, \rho_{Size} \rangle$ and $\langle \delta, \bar{c_i}, \rho_{SG} \rangle$; then the Euclidean distance between these tuples defined the similarity among the planets they represented. The previously defined sets of aesthetic and non-aesthetic maps made up a baseline to compare with so the problem of generating aesthetic maps turned into an optimization problem about minimizing the distance between generated and aesthetics maps while maximizing their distance to non-aesthetic maps. The latter was essential in order to insert diversity into the set of generated maps, hence avoiding the creation of maps that were very similar to the aesthetic ones.

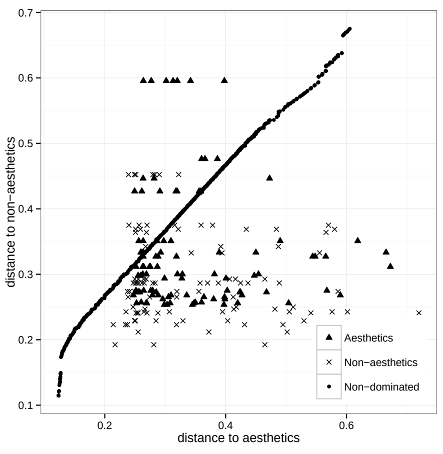
4 Procedural Map Generation

We used a NSGA-II multi-objective self-adaptive (μ+λ) evolution strategy (with $\mu = 10$ and $\lambda = 100$) as the map generator. The objectives, as described in section 3, were to increase and reduce the distance between the generated maps and non-aesthetic and aesthetic maps, respectively. The solutions are represented as mixed real-integer vectors of planets, planet's coordinates (x_i, y_i) are real-valued numbers but sizes s_i and initial number of ships w_i are positive integers. To overcome this situation we considered a hybrid mutation operator with different methods for parameters of either type. For real-valued parameters, it used a Gaussian mutation; as for integer variables, it used a method that generates suitable mutations [14, 15].

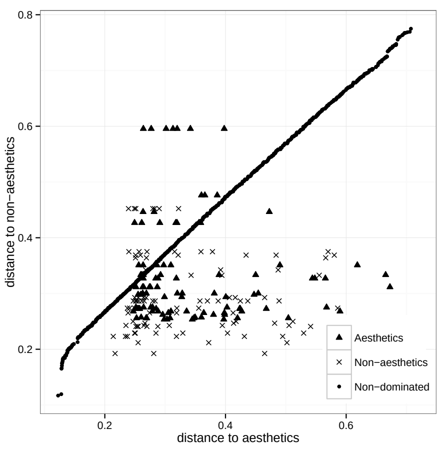
The method for integer variables is similar to the mutation of real values but, in this case, the difference of two geometrically distributed random variables produces the perturbation (instead of the normal distributed random variables used by the Gaussian mutation). Real-valued parameters $\langle r_1, ..., r_n \rangle$ were extended with n step sizes, one for each parameter (thus providing the means for self-adaptation), resulting in $\langle r_1, ..., r_n, \sigma_1, ..., \sigma_n \rangle$. The mutation method is specified as follows:

$$\sigma_i' = \sigma_i \cdot e^{\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)} \quad (2)$$

$$r_i' = r_i + \sigma_i \cdot N_i(0,1) \quad (3)$$



(a)



(b)

Figure 3: Cumulative set of non-dominated generated solutions (circle) for group of variables A (a) and B (b) and maps from aesthetic (triangle) and non-aesthetic (square) baseline sets.

where $\tau' \propto 1/\sqrt{2n}$, and $\tau \propto 1/\sqrt{2\sqrt{n}}$. A boundary rule is applied to step-sizes to forbid standard deviations very close to zero: $\sigma'_i < \epsilon_0 \Rightarrow \sigma'_i = \epsilon_0$ (in this algorithm, σ_0 comprises a 1% of the parameter's range).

Regarding integer-valued parameters $\langle z_1, ..., z_m \rangle$ they were extended in a similar way as real-valued parameters, resulting in $\langle z_1, ..., z_m, \varsigma_1, ..., \varsigma_m \rangle$. The following equations define the mutation mechanism:

$$\varsigma'_i = \max(1, \varsigma_i \cdot e^{\tau \cdot N(0,1) + \tau' \cdot N(0,1)}) \tag{4}$$

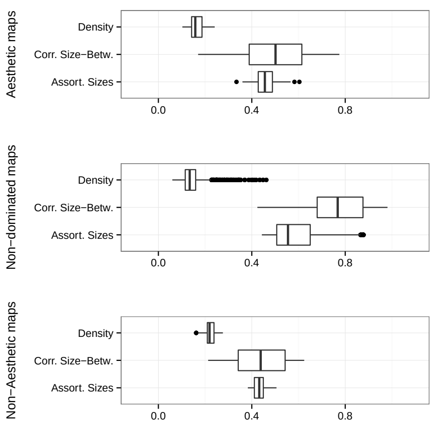
$$\psi_i = 1 - (\varsigma'_i/m)\left(1 + \sqrt{1 + \left(\frac{\varsigma'_i}{m}\right)^2}\right)^{-1} \tag{5}$$

$$z'_i = z_i + \left\lfloor \frac{ln(1 - U(0,1))}{ln(1 - \psi_i)} \right\rfloor - \left\lfloor \frac{ln(1 - U(0,1))}{ln(1 - \psi_i)} \right\rfloor \tag{6}$$

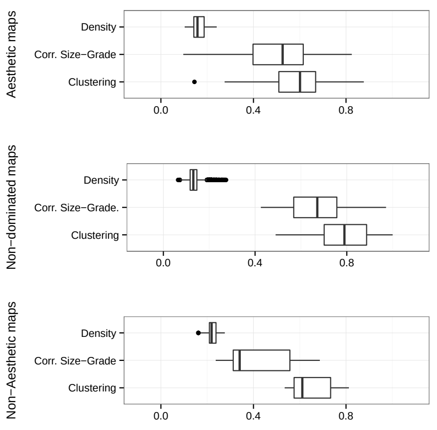
where $\tau = 1/\sqrt{2m}$ and $\tau' = 1/\sqrt{2\sqrt{m}}$.

In the case of the recombination, we chose a "cut and splice" operator that recombines two individuals by swapping cut pieces with different sizes. It selects one cut point for each individual and then swaps these pieces, getting two new individuals with a different number of planets in relation to their parents (affecting the complexity of the maps, i.e., the number of planets in the solutions, and enabling the algorithm with additional self-adaptation features).

As described in section 3, vectors of three components characterized every map, so the Euclidean distance between these vectors measures the likelihood between the maps they represent. The fitness function is, precisely, the median Euclidean distance from the individual to every map from the set of aesthetics (minimization objective) and non-aesthetics (maximization objective) maps.



(a)



(b)

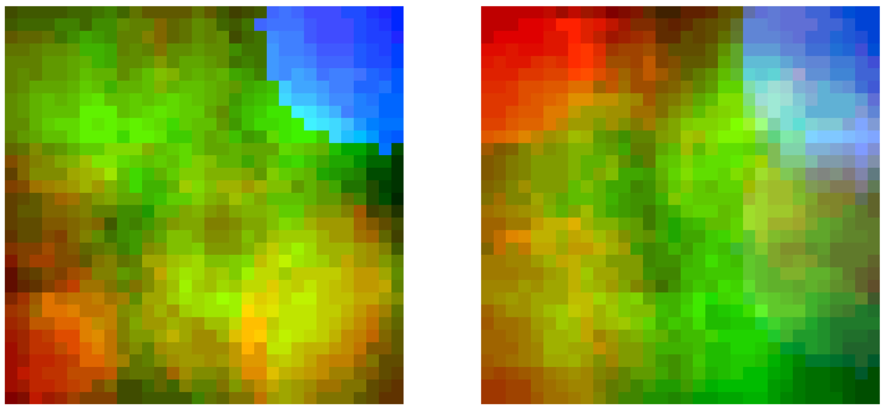
Figure 4: Distribution of the variables from each group, group A (a) and group B(b), for both non-dominated solutions and baseline maps.

5 Experimental Results

We conducted two sets of experiments, one for each group of variables. Each set of experiments consisted of 20 runs of the evolutionary algorithm described previously, with 100 generations per run, obtaining a set of non-dominated solutions in each. Then we calculated the cumulative set of non-dominated solutions for each group (see Figure 3). As we can see, the relationship between both distances is linear regardless of the group of variables used, specifically in the middle range of the Pareto front. This suggests that it is hard to increase the distance of the solutions to non-aesthetic maps without also increasing it toward the aesthetic maps, in addition to the high density of the search space. Furthermore, we should observe that there are aesthetic maps from the baseline with a higher mean difference to non-aesthetics maps than some non-aesthetic ones, and vice versa (i.e., there are aesthetic maps that are more similar to non-aesthetic maps than other non-aesthetic ones, and vice versa). This hints at the subjectivity of the expert during the process of map classification.

Regarding how the characterization values of baseline and generated maps are distributed (see Figure 4), we can observe that there are not high differences between the variables from aesthetic and non-aesthetic maps. However, some values from non-dominated maps, such as ρ_{SB} and ρ_{Size} from group A, are higher than their corresponding values from baseline maps, which should explain the high distance between some of the non-dominated solutions and baseline maps (as seen in Figure 3).

We created two self-organizing maps (SOMs), one for each group of variables, so that we could be sure about the validity of the generated maps and we could make a comparison between both groups of variables. These maps are artificial neural networks that are trained using unsupervised learning to generate a discretized representation of the input space [16]. In this case, we built both SOMs with 32×32 process units over a non-toroidal rectangular layout,



(a) (b)

Figure 5: Map's distribution over the SOMs (groups A (a) and B (b)). Red for non-aesthetic, green for aesthetic and blue for non-dominated.

using our baseline maps as the training set, and then, we projected the non-dominated solutions onto the SOM (see Figure 5). As we can see in both cases, they were not able to establish a clear distinction between aesthetic (green) and non-aesthetic (red) maps (note the overlapped zones). Regarding the projected solutions, they occupy a distinct region in the case of group A, while in group B the region is blurred (even some solutions are located in a different area: specifically the lower right area). Besides, the red component of the solutions from group B seems to be greater than the group A red component, which may indicate that the results obtained from the group B variables are lower quality.

6 Conclusion

In this paper we have presented a method for automatically generating maps for the RTS game Planet Wars, based on evolution strategies. The quality of these generated maps was based on their aesthetics, specially on measures calculated from topological properties of the sphere-of-influence graph computed for each map. We conducted a study in order to discover which measures are the most significant, using for this purpose random forest classifiers, obtaining two sets of three measures each (groups A and B). We tested our method using both sets of measures, projecting the solutions (altogether with the baseline maps) on a self-organizing map. Experiments showed that there is a linear relationship between the objectives of the algorithm, i.e., it is difficult for the method to generate maps that seem to aesthetic maps without seeming also to non-aesthetic. Furthermore, observing the SOM, it seems that the results obtained using the first group of variables (graph's density, correlation between node size and betweenness, and size assortativity) had better quality than those obtained by the second group (density, clustering coefficient and Pearson's correlation coefficient between planets' sizes and their degree). In any case, the method

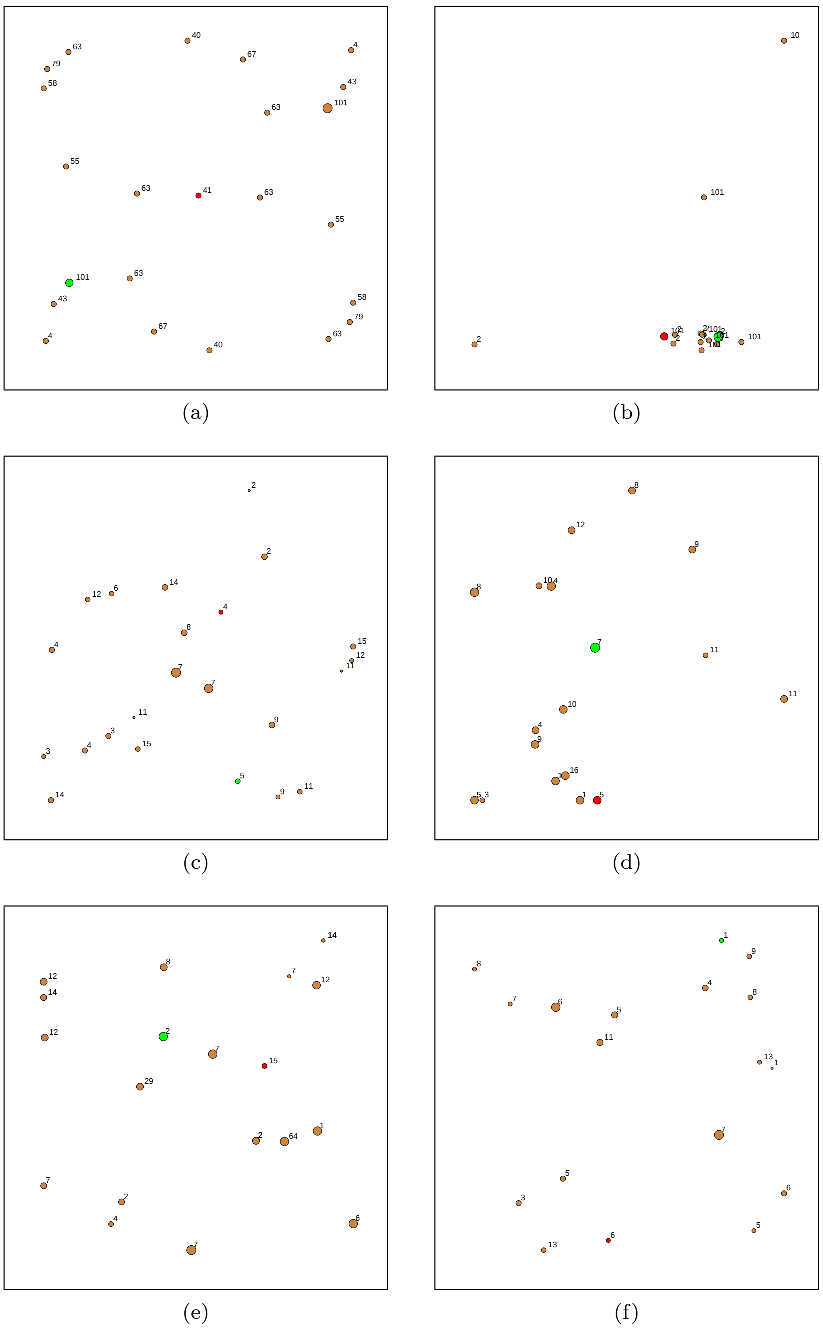


Figure 6: Map samples: maps from baseline with good (a) and bad (b) aesthetics and maps generated using the variables from groups A (c) (d) and B (e) (f)

is capable of generating fully playable aesthetic maps for the aforementioned real-time strategy game.

Future work will try to explore the connection between aesthetics and playability (as defined in [3, 4]), defining augmented fitness functions to take the latter into accout. We also plan to introduce the user in the loop in order to refine the automated aesthetic criterion.

Acknoledgements

This work is partially supported by Spanish MICINN under project ANYSELF (TIN2011-28627-C04-01), Junta de Andalucía under project P10-TIC-6083 (DNEMESIS) and Universidad de Málaga. Campus de Excelencia Internacional Andalucía Tech.

References

[1] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 9, no. 1, pp. 1:1–1:22, Feb. 2013. [Online]. Available: http://doi.acm.org/10.1145/2422956.2422957

[2] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.

[3] R. Lara-Cabrera, C. Cotta, and A. J. Fernández-Leiva, "Procedural map generation for a RTS game," in *13th International GAME-ON Conference on Intelligent Games and Simulation*, A. F. Leiva *et al.*, Eds. Malaga (Spain): Eurosis, 2012, pp. 53–58.

[4] ——, "A procedural balanced map generator with self-adaptive complexity for the real-time strategy game planet wars," in *Applications of Evolutionary Computation*, A. Esparcia-Alcázar *et al.*, Eds. Berlin Heidelberg: Springer-Verlag, 2013, pp. 274–283.

[5] M. Buro, "RTS games and real-time AI research," in *Behavior Representation in Modeling and Simulation Conference*, vol. 1. Curran Associates, Inc., 2004.

[6] R. Lara-Cabrera, C. Cotta, and A. J. Fernández-Leiva, "A Review of Computational Intelligence in RTS Games," in *2013 IEEE Symposium on Foundations of Computational Intelligence*, M. Ojeda, C. Cotta, and L. Franco, Eds., 2013, pp. 114–121.

[7] J. Togelius, R. De Nardi, and S. Lucas, "Towards automatic personalised content creation for racing games," in *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, 2007, pp. 252–259.

[8] M. Frade, F. F. de Vega, and C. Cotta, "Modelling video games' landscapes by means of genetic terrain programming - a new approach for improving

users' experience," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science, M. Giacobini *et al.*, Eds., vol. 4974. Berlin Heidelberg: Springer-Verlag, 2008, pp. 485–490.

[9] ——, "Breeding terrains with genetic terrain programming: The evolution of terrain generators," *International Journal of Computer Games Technology*, vol. 2009, 2009.

[10] M. Frade, F. de Vega, and C. Cotta, "Evolution of artificial terrains for video games based on accessibility," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, C. Di Chio *et al.*, Eds. Berlin Heidelberg: Springer-Verlag, 2010, vol. 6024, pp. 90–99.

[11] M. Frade, F. F. de Vega, and C. Cotta, "Evolution of artificial terrains for video games based on obstacles edge length," in *IEEE Congress on Evolutionary Computation.* IEEE, 2010, pp. 1–8.

[12] T. Mahlmann, J. Togelius, and G. N. Yannakakis, "Spicing up map generation," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, C. D. Chio *et al.*, Eds., vol. 7248. Málaga, Spain: Springer-Verlag, 2012, pp. 224–233.

[13] G. T. Toussaint, "A graph-theoretic primal sketch," *Computational Morphology*, pp. 229–260, 1988.

[14] R. Li, "Mixed-integer evolution strategies for parameter optimization and their applications to medical image analysis," Ph.D. dissertation, Leiden University, 2009.

[15] G. Rudolph, "An evolutionary algorithm for integer programming," in *Parallel Problem Solving from Nature III*, ser. Lecture Notes in Computer Science, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Jerusalem, Israel: Springer-Verlag, 1994, vol. 866, pp. 139–148.

[16] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.