Geometrical vs Topological Measures for the Evolution of Aesthetic Maps in a RTS Game

R. Lara-Cabrera, C. Cotta, A.J. Fernández-Leiva

Department of "Lenguajes y Ciencias de la Computación"
ETSI Informática, University of Málaga, Campus de Teatinos
29071 Málaga – Spain

Abstract

This paper presents a procedural content generation (PCG) method that is able to generate aesthetic maps for a real-time strategy game. The maps were characterized based on either their geometrical properties or their topological measures (obtained in this latter case from the sphere-of-influence graph induced by each map). Using these features, a distance function between maps can be defined. This function is used in turn to determine how close/far each map generated by the PCG method (a self-adaptive evolutionary algorithm) is to a collection of maps which were taken initially to be aesthetic or non-aesthetic. This correspondence guided a multi-objective evolutionary approach whereby maps close to aesthetic maps and far to non-aesthetic maps are sought. Self-organizing maps are used to ascertain whether the so-generated maps naturally cluster together with aesthetic maps, as well as to provide a qualitative assessment of the ability of each set of features to characterize the latter.

Keywords: Procedural content generation, game aesthetics, computational intelligence, real-time strategy games

1. Introduction

The videogame industry has taken the lead role from the entertainment business, with a total consumer spent of $24.75 billion in 2011 [1] and estimated game revenues of $70.4 billion worldwide in 2013 (which represents a 6% year-on-year increase), according to Newzoo's 2013 Global Games Market Report [2]. Moreover, the number of gamers was expected to surpass 1.2 billion by the end of that year. This situation has motivated the research applied to videogames, which has been acquiring notoriety during the last years, involving many areas such as psychology and player satisfaction, marketing and gamification, computational intelligence, computer graphics, and even education and health (serious

Email addresses: raul@lcc.uma.es (R. Lara-Cabrera), ccottap@lcc.uma.es (C. Cotta), afdez@lcc.uma.es (A.J. Fernández-Leiva)

games). The quality and appealing of video-games used to rely on their graphical quality until the last decade, but now, their attractiveness falls on additional features such as the music, the player immersion into the game and interesting storylines. It is hard to evaluate how amusing a game is because this evaluation depends on each player, nevertheless there is a relationship between player satisfaction and fun [3]. Nowadays, interesting new challenges and goals are emerging within the area of video games, especially in the field of artificial and computational intelligence in games [4].

Procedural Content Generation (PCG) [5, 6] refers to the algorithmic creation of game content, either with human intervention or without it, such as maps, levels, textures, characters, rules and quests, but excluding the behavior of non-playable characters and the game engine itself. The use of PCG has several advantages, including saving memory and disk space, improving human creativity and providing adaptivity to games. These benefits are well known by the industry as demonstrated by the use of PCG techniques during the development of commercial games such as *Borderlands saga* with procedurally generated weapons and items, *Skyrim* (terrains and forests), and *Minecraft* or *Terraria* with procedurally generated worlds. At the moment, there are three main goals [5] of PCG research that are currently not obtainable and it would require significant further research effort: multi-level multi-content PCG (i.e. systems that are able to generate multiple types of quality content at multiple levels of granularity in a coherent fashion while taking game design constraints into consideration), PCG-based game design (i.e. creating games where a PCG algorithm is an essential part of the game instead of being a design tool) and PCG systems that could create complete games including the rules and game engine. There is a subfield in PCG (i.e. Search-based PCG [7, 8]) whose techniques apply a generate and test scheme so the content is firstly generated based on previous evaluations and then evaluated according to some criteria (this paper presents a method that follows this scheme).

The application of the aforementioned PCG techniques for *Planet Wars* involves, in this case, generating the maps on which the game takes place. The particular structure of these maps can lead to games exhibiting specific features. In previous work [9, 10] we focused on making the game more fun to play, achieving games that are balanced (i.e., games in which none of the players strongly dominates her opponent) and dynamic (i.e., games with a high number of battles and changes in the owners of the planets). Despite we were able to accomplish our requirements of balance and dynamism, the generated maps lacked aesthetics (for example, maps with their planets clustered in a small region), which is an interesting property apart from the fun that may lead to increase the player satisfaction; in fact, fun and aesthetics are two complementary means of achieving the same goal [3]. In addition, non-aesthetic maps may confuse the player, reducing her satisfaction or even leading her to stop playing the game. This situation led us to tackle this problem [11]; therein we focused on a way to characterize maps so as to attain a method capable of producing scenarios with good aesthetics. In this paper, we have expanded the aforementioned characterization method including new topological measures that are not affected by rotation, scaling and translation, which is important to prevent two maps being considered different when they are conceptually the same (they are equal according to the gameplay). In order to obtain these measures, we characterized every map as a sphere of influence graph [12], which establishes a relationship between some set of points based on their spatial arrangement. This characterization provides a higher-level of abstraction and paves the way to measure topological properties of maps as well as providing a different perspective to analyze morphological properties of the latter. In the following, we will describe these different characterizations, and study the results obtained when an evolutionary PCG method is deployed on them.

2. Background

Real-time strategy (RTS) games offer a large variety of fundamental AI research problems [13] –such as adversarial real-time planning and decision making under uncertainty among others– have been widely used as a test-bed for AI techniques [14, 15, 16, 17, 18]. PCG techniques are usually employed to generate maps, as exhibited by the large number of papers on this topic [6]. For example, Mahlmann et al. [19] presented a search-based map generator for a simplified version of the RTS game *Dune 2*, which is based on the transformation of low resolution matrices into higher resolution maps by means of cellular automata. Frade et al. introduced the use of genetic programming to evolve maps for videogames (namely terrain programming), using either subjective human-based feedback [20, 21] or automated quality measures such as accessibility [22] or edge-length [23]. There is another paper [24], by Liapis et al., that relies on the human evaluation of the generated content. Togelius et al. [25] designed a PCG system capable of generating tracks for a simple racing game from a parameter vector using a deterministic genotype-to-phenotype mapping. Dormans [26] presented strategies to generate levels for action adventure games using an approach that distinguishes between missions and spaces as two separate structures that need to be generated in two individual steps. There is a study by Ashlock and McGuinness [27] that introduces the so-called landscape automata for searching the space of height maps using an evolutionary algorithm. The designer is able to specify checkpoints that must be mutually accessible in order to control the generated height maps. Another level generator based on an evolutionary algorithm is presented in [28], but in this case, the fitness function depends on the difference between the difficulty curves defined by the designer and calculated for the candidate content, respectively. Some authors [10, 29] presented algorithms capable of generating balanced maps for the RTS games *Starcraft* and *Planet Wars*.

As we stated before, there is a large number of papers devoted to the generation of maps and levels. They all have one thing in common, their algorithms look for carrying out various restrictions related to the gameplay, such as ensuring the accessibility of certain zones, adjusting the difficulty or balancing the gaming level of the players. This increases the fun and thus the player satisfaction. This paper deals with another way of improving this player satisfaction, that is generating maps with good aesthetics, a topic we were not able to find in the state of the art, besides our previous work.

In addition to maps and levels, there are other content that is generated with these methods. For example, Font et al. [30] presented an initial research regarding a system capable of generating novel card games. Collins [31] made an introduction to procedural music in video games, examining different approaches to procedural composition and the control logics that have been used in the past. The authors of [32] have created a prototype of a tool that automatically produce design pattern specifications for missions and quest for the game Neverwinter Nights.

In this work we focus on *Planet Wars*, a RTS game based on *Galcon* and used in the *Google AI Challenge 2010* [33], a competition about creating a computer program that plays the game of Planet Wars as intelligently as possible. Despite players make their orders on a turn-by-turn basis, they issue these orders at the same time, so we can treat this game as a real-time game. The game's objective is to conquer all the planets on the map or destroy all of your opponents ships. A game of Planet Wars takes place in outer space, precisely on a map that contains several planets with some number of ships on it. Each planet may have a different number of ships. The planets may belong to some player or may be neutral. The game has a time limit and it may end earlier if one of the players loses all his ships, implying that the player who has ships remaining becomes the winner. It is considered a draw if both players have the same number of ships when the game ends. On each turn, the player is able to send fleets of ships from any planet she owns to any other planet on the map. She may send as many fleets as she wishes on a single turn as long as she has enough ships to supply them. After sending fleets, each planet owned by a player (i.e. not neutral) will increase the number of defending ships there, according to that planet's growth rate (i.e. size). Different planets have different growth rates. The fleets will then take some number of turns to reach their destination planets, where they will then fight those opposing forces there and, if they win, take ownership of the planet. There is an important restriction: fleets cannot be redirected during travel. Players may continue to send more fleets on later turns even while older fleets are in transit. Next section will detail the structure of maps in this game, and how their properties can be characterized.

3. Methodological issues

As stated before, let us firstly detail how the maps have been represented and characterized in order to get better aesthetics, both from the geometrical and topological point of view; next, we describe the evolutionary algorithm used to generate the maps.

3.1. Representation and characterization

Game maps are sets with a certain number of planets n_p located on a 2D plane. These planets are 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 defines the rate at which a planet will produce new ships every turn (as long as the planet is controlled by any player) while the remaining parameter, w_i, indicates the number of ships that are defending that planet. Hence, we can denote a map as a list $[\vec{\rho}_1, \vec{\rho}_2, \cdots, \vec{\rho}_{n_p}]$, where each $\vec{\rho}_i$ is a tuple $\langle x_i, y_i, s_i, w_i \rangle$. A playable map needs to specify the initial home planets of the players, which have been fixed as the first two planets $\vec{\rho}_1$ and $\vec{\rho}_2$ for the sake of simplicity. The number of planets n_p is not fixed and should range between 15 and 30 as specified by the rules of the *Google AI Challenge 2010*. This variable number of planets is also a part of the self-adaptive evolutionary approach described later on.

In order to evaluate the aesthetics of maps, we need to characterize its features and compare these with those of other maps known to be aesthetic/non-aesthetic. For this purpose we have considered different kinds of properties. From a general point of view, we draw a distinction between geometrical features (based on the spatial geometry of the map, namely coordinates and distances), and topological features (based on higher-level qualitative features of the map which are unaffected by simple geometrical transformations). In addition, we also take into account morphological features (based on individual planet properties, such as size or initial number of ships) in combination with the other two, leading to two collections of features for classification purposes which –by virtue of simplicity– we will refer to as *geometrical* and *topological* respectively. Beginning with the former, the list of features considered is listed below:

- Spatial distribution of planets: Let $\vec{p}_i = (x_i, y_i)$ be the coordinates of the i-th planet and n_p the total number of planets. The average distance between planets μ_d and the standard deviation of these distances σ_d is defined as follows:

$$\mu_d = \frac{1}{n_p^2} \sum_{i=1}^{n_p} \sum_{j=1}^{n_p} \|\vec{p}_i - \vec{p}_j\|, \qquad \sigma_d = \sqrt{\frac{1}{n_p^2} \sum_{i=1}^{n_p} \sum_{j=1}^{n_p} (\|\vec{p}_i - \vec{p}_j\| - \mu_d)^2}$$

- Planet features: Let s_i and w_i be the size (i.e. growth rate) and number of ships, respectively, of the i-th planet. The average and standard deviation of these sizes (μ_s and σ_s respectively) and the Pearson's correlation ρ between the planet's size and the number of ships on it are defined as:

$$\mu_s = \frac{1}{n_p} \sum_{i=1}^{n_p} s_i, \qquad \sigma_s = \sqrt{\frac{1}{n_p} \sum_{i=1}^{n_p} (s_i - \mu_s)^2}$$

$$\rho = \frac{\sum_{i=1}^{n_p} s_i w_i - n_p \mu_s \mu_w}{n_p \sigma_s \sigma_w}$$

  where μ_w and σ_w are the average and standard deviation of ships, respectively.

These geometrical measures have been applied to compare the likelihood between maps in the following way: each map is characterized by a tuple

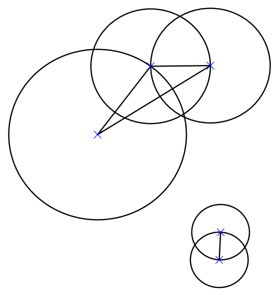


Figure 1: A sphere-of-influence graph. There is an edge between two points only if their spheres of influence intersects. These spheres are centered on the points and their radius are the minimal distance from the center point to the others.

$\langle \mu_d, \sigma_d, \mu_s, \sigma_s, \rho \rangle$, then the euclidean distance between these tuples defined the similarity among the planets they represented. Additionally, we specified two sets of maps, one of them containing 10 maps with good aesthetics and the other one including 10 non-aesthetic maps; there was an expert that reviewed and tagged as aesthetic/non-aesthetic several maps from the Google AI Challenge and those produced by our previous algorithms. These sets provide a reference to compare with in a way that the goal 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 necessary to insert diversity into the set of generated maps in order to avoid the generation of maps that were very similar to the aesthetic ones.

As to topological features, these emerge from the sphere-of-influence graph (SIG) of each map. This graph establishes a relationship between some set of points, based on their spatial arrangement (see Figure 1). This relationship is not affected by scaling, translation and rotation, which is a desirable feature as stated in section 1. SIGs have been applied to many applications and research areas, such as data mining for clustering depending on their attributes, computer vision for object recognition from input dot patterns, and computer graphics, for defining surfaces over point clouds. In this case, we have computed the SIG for a certain map executing the following procedure:

1. For each planet p_i placed at (x_i, y_i), compute the distance d_i to the closest planet as $d_i = \min_j \{\|(x_i - x_j, y_i - y_j)\| \mid j \neq i\}$ 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.

Once the SIG is available, we have calculated several measures to characterize them, namely:

- Number of connected components (sub-graphs in which any two vertices are connected by paths).
- 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 $\delta = 2m/(n_p(n_p - 1))$, with n_p and m being the number of nodes (i.e. planets) 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 centrality. 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.

These parameters characterize several features of the maps, which are detailed below. The number of connected components gives a measure of how planets are clustered, precisely the number of interconnected group of planets, while the average clustering coefficient assesses the shape of the cluster. The average node's degree and graph's density quantify how regularly distributed are the planets, because, according to the definition of a SIG, if the planets are highly interconnected then the distances between them are quite similar. Finally, the correlations introduce the size of the planets into the map characterization process, establishing relationships between this parameter and other planet features such as betweenness centrality, node degree and the size of the neighbor planets.

In order to select a subset of the above variables, we have made a Random Forest classifier (50 estimators each and bootstrap sampling) for each possible combination of the above measures and we have evaluated the performance of the classifiers using a Leave-one-out (LOO) cross-validation. AUC (Area under ROC curve) values of the different combinations lead us to choose the following variables as the map characterization: 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 planets connected in the graph), with an $AUC = 0.9916$.

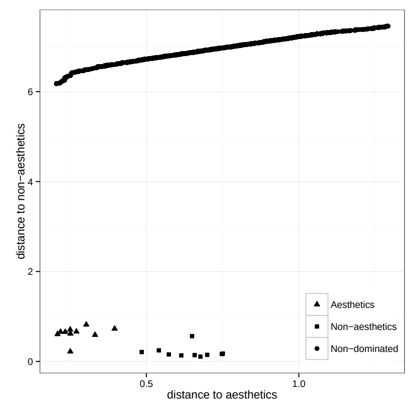
As in the aforementioned map characterization using geometrical measures, we have featured each map by a tuple $\langle \delta, \rho_{SB}, \rho_{Size} \rangle$. Hence the Euclidean distance between these tuples defines the similarity among the maps they represent.

3.2. Evolutionary map generation

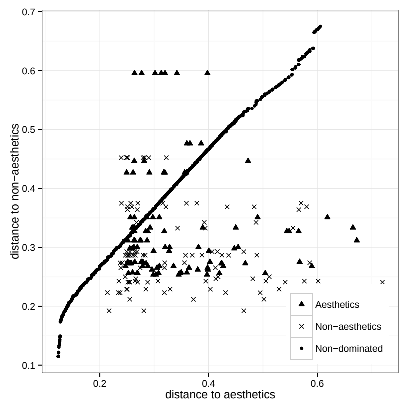
Evolutionary algorithms [34] are population-based metaheuristic optimization algorithms whose operation is inspired by biological evolution. There is a set of possible solutions to the optimization problem that act as individuals in a population and whose quality (fitness) is measured by an evaluation function. The algorithm mutates and recombines the solutions to create new ones, while a process of selection dependent on the fitness of the individuals is performed to determine which of them survive to the next generation. This iterative process results in an increase in the quality of the solutions. There is a subset of these algorithms that are capable of optimize multiple functions simultaneously, the so-called multi-objective evolutionary algorithms; in this case, the fitness of individuals are vectors with many components as optimization objectives. The best solutions are those found in the set of non-dominated solutions (i.e. if none of the objective functions can be improved in value without degrading some of the other objective values).

We used a similar approach for the map generator as in previous work [10, 11], that is, a multi-objective NSGA-II [35] self-adaptive $(\mu + \lambda)$ evolution strategy (with $\mu = 10$ and $\lambda = 100$) with two objectives: reduce (resp. increase) the distance between the created maps and those from the training set that were considered as aesthetic (resp. non-aesthetic). The solutions were mixed real-integer vectors: planet's coordinates (x_i, y_i) were real-valued numbers but sizes s_i and initial number of ships w_i were positive integers; hence we used two different mutation operators, one for each value type (Gaussian mutation for real-valued parameters and a specific mutator for integers that uses the difference of two geometrically distributed random variables to produce the perturbation – see [10] for a full description). Regarding the recombination, we considered a "cut and splice" operator that selects one cut point for each individual and then exchanges these pieces, getting two new individuals with a different number of planets in relation to their parents and ensuring the maximum mix-up of the genetic information during the recombination. As with the mutation operator, which includes the mutation steps as a part of the solutions, the recombination operator endows the algorithm with self-adaptation capacities, hence affecting the complexity of the maps, i.e., the number of planets in the solutions.

As described in section 3.1, we characterized every map as both tuples of five (geometrical) and three (topological) elements, respectively, so the euclidean distance between these vectors measures the likelihood between them, hence the fitness function used to evaluate the individuals 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 that are within the training set.



(a)



(b)

Figure 2: Cumulative set of non-dominated generated solutions (circle) and maps from aesthetic (triangle) and non-aesthetic (square) training sets for both geometrical (a) and topological (b) approaches

4. Experimental Results

We have used the DEAP (Distributed Evolutionary Algorithms in Python) library [36] to implement the aforementioned algorithm. We have run two sets of 20 executions of the algorithm during 100 generations, using both characterization approaches. We have also computed the cumulative non-dominated set of solutions from every approach and execution – see Figure 2. As we can see, there is a linear relationship between both distances in the middle range of the front, no matter which characterization we use. This hints at the density of the search space and the feasibility of linearly trading increasing distance to good maps by increasing distance to bad maps.

Figure 3 shows how are distributed the values of the different variables that make up the characterization vectors of a map. Note that there are some variables that are similar in both aesthetics and non-aesthetic maps, such as σ_d, σ_s and δ. However, some variables such as ρ_{SB} and ρ_{Size} are higher in the case of the non-dominated maps, which should explain the high distance between many solutions in the front and the training maps, as seen in Figure 2. Another interesting observation is the highly distributed values of μ_d in the non-dominated maps, which probably means that this variable has an uncertain effect over the fitness and hence the search space for this variable is wider with respect to other variables.

In order to analyze qualitatively the generated maps, we created two self-organizing map (SOM) [37] with 32×32 process units over a non-toroidal rectangular layout, one for each characterization approach. As we can see in figure[1]

[1]There is a color version of the figure at `http://dx.doi.org/10.6084/m9.figshare.`

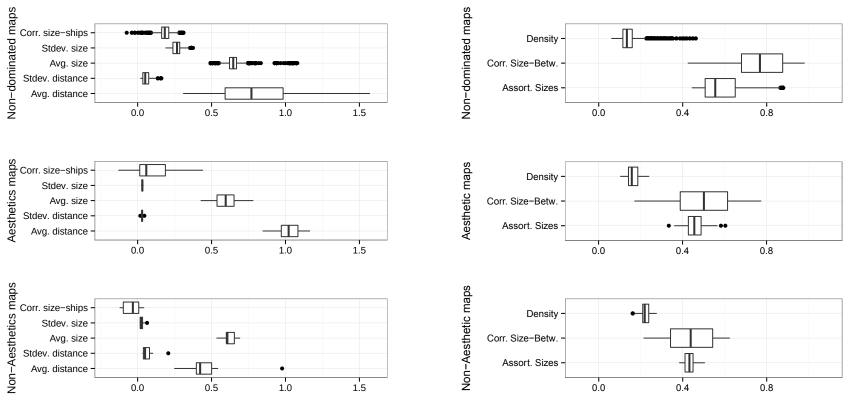


Figure 3: Characterization variables for both non-dominated maps and training maps: spatial distribution and SIG approaches

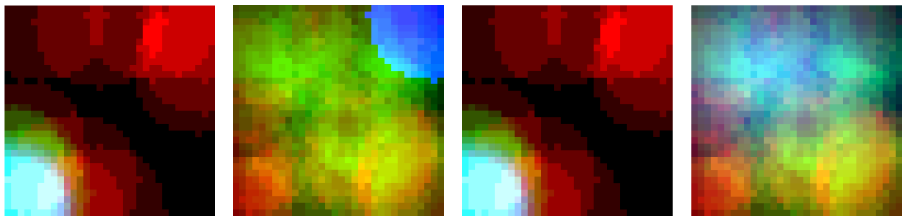


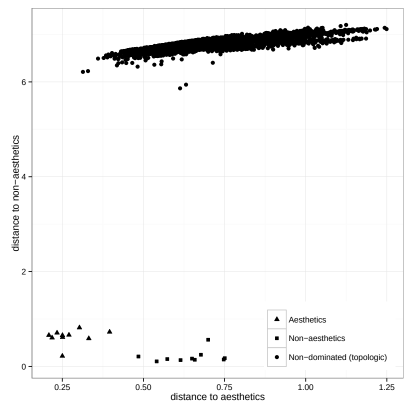
Figure 4: Map's distribution over the SOM for both geometric (a) and topological (b) approaches. Red for non-aesthetic, green for aesthetic and blue for non-dominated. (c) and (d) show the topological approach solution projected over the geometric approach SOM and vice versa.

4, the SOM of the geometrical approach established a separation between non-aesthetic (red zones, upper-right) and aesthetic maps (green zones, lower-left). Moreover, generated maps (blue zones) share the same region as aesthetic maps, hence they should be considered aesthetic as well. However, regarding the topological approach, it was not able to establish a clear distinction between aesthetic and non-aesthetic maps (note the overlapped zones).

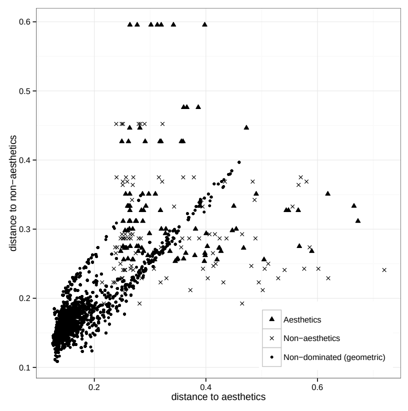
We performed a cross analysis between the non-dominated solutions (i.e. maps) from the geometric and topological approaches to compare both characterizations. As we can see in Figure 5, the solutions from the topological approach are distributed over the same region of the fitness space than the solutions from the geometrical approach, manifesting the similarity between the quality of both sets of maps, according to the geometrical gauge. Similarly, we

1091371

obtained analogous results after projecting the solutions from the topological and geometrical approaches over the latter SOM – see Figure 4c. Regarding the solutions from the geometrical approach, they are mostly distributed over the lower-left region of the topological fitness space, indicating that these maps are nearer to the aesthetic and non-aesthetic maps of the training set than many maps from the topological approach. If we look at the projection of the former over the topological SOM (Figure 4), we can see that these geometric solutions are more scattered than the corresponding topological solutions, although they share the same area as the aesthetic maps, exposing their higher diversity with respect to the solutions from the topological approach, even though both approaches produce suitable maps.



(a)



(b)

Figure 5: Cumulative non-dominated topological solutions in the fitness space of the geometric solutions (a) and geometric solutions in the fitness space of the topological solutions (b).

5. Conclusions

We have performed an initial approach towards the procedural aesthetic map generation for the RTS game *Planet Wars*. We have defined two methods of map characterization in order to evaluate how aesthetic a map is. One of them is based on several of its maps' geometrical properties, the other one depends on topological measures extracted from the spheres-of-influence graph, which has been built for each map. We have used two sets of maps (aesthetics and non-aesthetics) as a baseline to compare with, and an evolution strategy whose objectives are to minimize/maximize the distance of the generated maps to aesthetics/non-aesthetics maps in the training set (note some map samples in Figure 6). The solutions have been further analyzed with a self-organizing clustering method (SOM) which was able to make a separation between the aesthetic and non-aesthetic maps that were procedurally generated using the

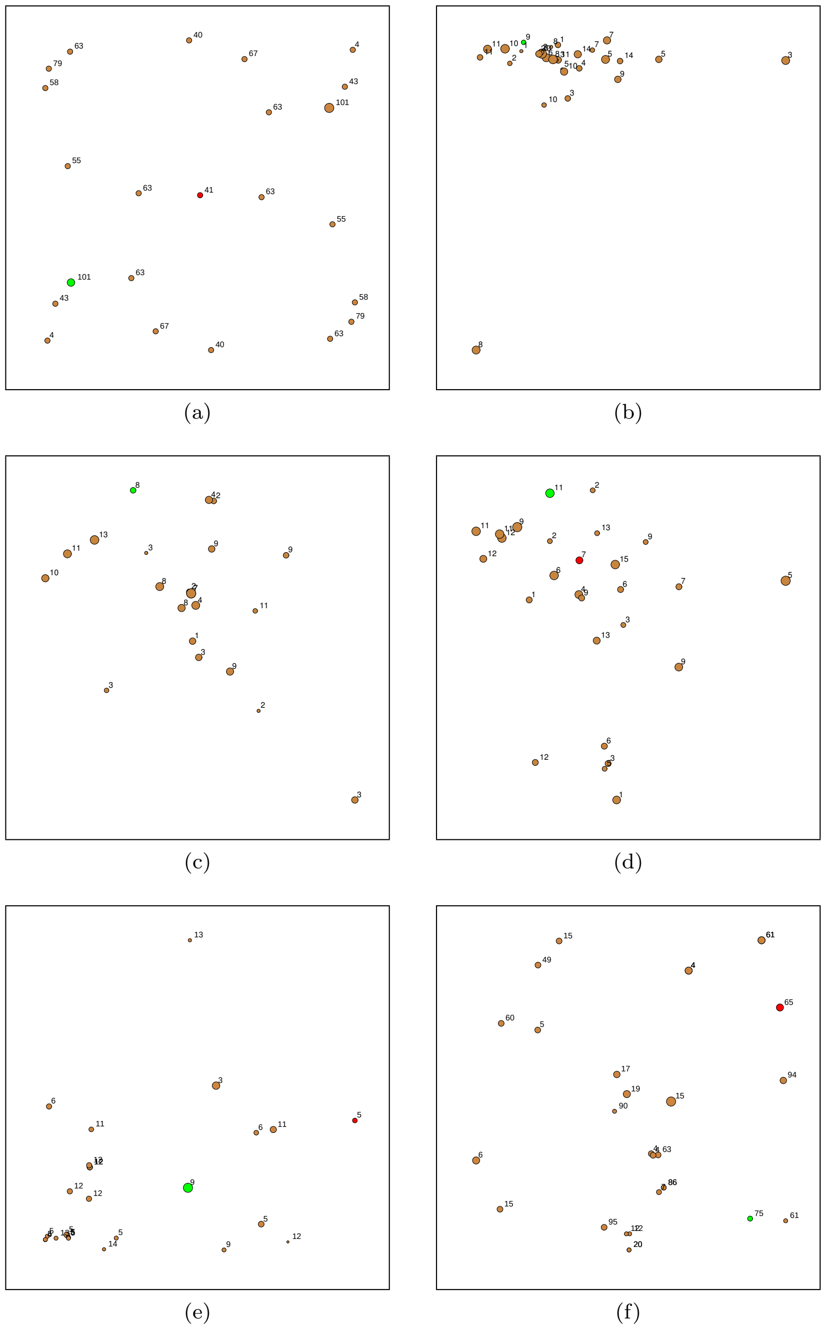


Figure 6: Map samples: maps from training set with good (a) and bad (b) aesthetics and maps generated using geometrical (c) (d) and topological (e) (f) approaches. Planets' color represents who owns the planet: red and green for player 1 and 2, respectively, and brown for unoccupied planets. The number next to each planet represents the parameter "w", that is, the number of defensive ships.

geometrical map characterization (i.e. created maps shared the same region as the aesthetic maps). Nevertheless, the separation between both types of maps that were created with the topological characterization is not so crisp as the previous one. Finally, we have compared both approaches through a cross analysis of their solutions, concluding that the maps from the geometric approach are still considered aesthetic even if they are characterized using the topological approach, and vice versa.

Lines for future developments are manifold. Firstly, a combined geometrical/topological approach emerges as the next natural step. Such a hybrid approach can be then augmented with information dynamically supplied by a human user, thus providing a more refined depiction of the subjective notion of aesthetics. Finally, playability can be included as an additional measure, exploring the tradeoffs between what constitutes an aesthetically pleasant map and what provides a satisfactory playing experience.

References

[1] Entertainment Software Association, Essential facts about the computer and video game industry (2012).
URL `http://www.theesa.com/facts/pdfs/esa_ef_2012.pdf`

[2] T. Hagoort, P. Warman, 2013 global games market report, Tech. rep., Newzoo, accessed 20 Jan 2014 (2013).
URL `http://www.globalgamesmarket.com`

[3] M. Nogueira, C. Cotta, A. J. Fernández-Leiva, On modeling, evaluating and increasing players' satisfaction quantitatively: Steps towards a taxonomy, in: C. D. Chio, et al. (Eds.), Applications of Evolutionary Computation, Vol. 7248 of Lecture Notes in Computer Science, Springer-Verlag, Málaga, Spain, 2012, pp. 245–254.

[4] S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, J. Togelius (Eds.), Artificial and Computational Intelligence in Games, Vol. 6 of Dagstuhl Follow-Ups, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[5] J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, K. O. Stanley, Procedural Content Generation: Goals, Challenges and Actionable Steps, in: S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, J. Togelius (Eds.), Artificial and Computational Intelligence in Games, Vol. 6 of Dagstuhl Follow-Ups, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2013, pp. 61–75.

[6] M. Hendrikx, S. Meijer, J. Van Der Velden, A. Iosup, Procedural content generation for games: A survey, ACM Trans. Multimedia Comput. Commun. Appl. 9 (1) (2013) 1:1–1:22.

[7] J. Togelius, G. Yannakakis, K. Stanley, C. Browne, Search-based procedural content generation, in: C. Chio, S. Cagnoni, C. Cotta, M. Ebner,

A. Ekrt, A. Esparcia-Alcazar, C.-K. Goh, J. Merelo, F. Neri, M. Preu, J. Togelius, G. Yannakakis (Eds.), Applications of Evolutionary Computation, Vol. 6024 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 141–150.

[8] J. Togelius, G. N. Yannakakis, K. O. Stanley, C. Browne, Search-based procedural content generation: A taxonomy and survey, IEEE Transactions on Computational Intelligence and AI in Games 3 (3) (2011) 172–186.

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

[10] R. Lara-Cabrera, C. Cotta, A. J. Fernández-Leiva, A procedural balanced map generator with self-adaptive complexity for the real-time strategy game planet wars, in: A. Esparcia-Alcázar, et al. (Eds.), Applications of Evolutionary Computation, Springer-Verlag, Berlin Heidelberg, 2013, pp. 274–283.

[11] R. Lara-Cabrera, C. Cotta, A. J. Fernández-Leiva, Evolving aesthetic maps for a real time strategy game, in: P. A. G. Calero, M. A. G. Martín (Eds.), 1st Spanish Symposium on Entertainment Computing, Universidad Complutense de Madrid, Madrid (Spain), 2013, pp. 61–71.

[12] G. T. Toussaint, A graph-theoretic primal sketch, Computational Morphology (1988) 229–260.

[13] M. Buro, RTS games and real-time AI research, in: Behavior Representation in Modeling and Simulation Conference, Vol. 1, Curran Associates, Inc., 2004, pp. 53–60.

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

[15] M. Preuss, N. Beume, H. Danielsiek, T. Hein, B. Naujoks, N. Piatkowski, R. Stür, A. Thom, S. Wessing, Towards intelligent team composition and maneuvering in real-time strategy games, IEEE Transactions on Computational Intelligence and AI in Games 2 (2) (2010) 82–98.

[16] D. Keaveney, C. O'Riordan, Evolving coordination for real-time strategy games, IEEE Transactions on Computational Intelligence and AI in Games 3 (2) (2011) 155–167.

[17] A. Mora, A. Fernández-Ares, J.-J. Merelo, P. García-Sánchez, C. Fernandes, Effect of noisy fitness in real-time strategy games player behaviour optimisation using evolutionary algorithms, Journal of Computer Science and Technology 27 (5) (2012) 1007–1023.

[18] A. Fernández-Ares, P. García-Sánchez, A. Mora, J.-J. Merelo, Adaptive bots for real-time strategy games via map characterization, in: Computational Intelligence and Games (CIG), 2012 IEEE Conference on, 2012, pp. 417–721.

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

[20] M. Frade, F. F. de Vega, C. Cotta, Modelling video games' landscapes by means of genetic terrain programming - a new approach for improving users' experience, in: M. Giacobini, et al. (Eds.), Applications of Evolutionary Computing, Vol. 4974 of Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 2008, pp. 485–490.

[21] M. Frade, F. F. de Vega, C. Cotta, Breeding terrains with genetic terrain programming: The evolution of terrain generators, International Journal of Computer Games Technology 2009.

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

[23] M. Frade, F. F. de Vega, 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.

[24] A. Liapis, H. Martinez, J. Togelius, G. Yannakakis, Adaptive game level creation through rank-based interactive evolution, in: Computational Intelligence in Games (CIG), 2013 IEEE Conference on, 2013, pp. 1–8. doi:10.1109/CIG.2013.6633651.

[25] J. Togelius, R. De Nardi, 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.

[26] J. Dormans, Adventures in level design: Generating missions and spaces for action adventure games, in: Proceedings of the 2010 Workshop on Procedural Content Generation in Games, PCGames '10, ACM, New York, NY, USA, 2010, pp. 1:1–1:8.

[27] D. Ashlock, C. McGuinness, Landscape automata for search based procedural content generation, in: Computational Intelligence in Games (CIG), 2013 IEEE Conference on, 2013, pp. 1–8. doi:10.1109/CIG.2013.6633619.

[28] H. Diaz-Furlong, A. Solis-Gonzalez Cosio, An approach to level design using procedural content generation and difficulty curves, in: Computational

Intelligence in Games (CIG), 2013 IEEE Conference on, 2013, pp. 1–8. doi:10.1109/CIG.2013.6633640.

[29] A. Uriarte, S. Ontanon, Psmage: Balanced map generation for starcraft, in: Computational Intelligence in Games (CIG), 2013 IEEE Conference on, 2013, pp. 1–8.

[30] J. Font, T. Mahlmann, D. Manrique, J. Togelius, A card game description language, in: A. Esparcia-Alczar (Ed.), Applications of Evolutionary Computation, Vol. 7835 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 254–263.

[31] K. Collins, An introduction to procedural music in video games, Contemporary Music Review 28 (1) (2009) 5–15.

[32] C. Onuczko, D. Szafron, J. Schaeffer, M. Cutumisu, J. Siegel, K. Waugh, A. Schumacher, Automatic story generation for computer role-playing games., in: AIIDE, 2006, pp. 147–148.

[33] Google AI challenge, accesed: 2014-07-01.
URL `http://planetwars.aichallenge.org/`

[34] A. E. Eiben, J. E. Smith, Introduction to evolutionary computing, Springer, 2003.

[35] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, Evolutionary Computation, IEEE Transactions on 6 (2) (2002) 182–197.

[36] F.-A. Fortin, F.-M. D. Rainville, M.-A. Gardner, M. Parizeau, C. Gagné, DEAP: Evolutionary algorithms made easy, Journal of Machine Learning Research 13 (2012) 2171–2175.

[37] T. Kohonen, The self-organizing map, Proceedings of the IEEE 78 (9) (1990) 1464–1480.