ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/272508362

Procedural Content Generation for Real-Time Strategy Games

Article *in* International Journal of Interactive Multimedia and Artificial Intelligence · March 2015
DOI: 10.9781/ijimai.2015.325

CITATIONS
3

READS
302

4 authors, including:

Raul Lara-Cabrera
Universidad Autónoma de Madrid
26 PUBLICATIONS **95** CITATIONS

SEE PROFILE

Carlos Cotta
University of Malaga
323 PUBLICATIONS **3,432** CITATIONS

SEE PROFILE

Antonio J. Fernández-Leiva
University of Malaga
144 PUBLICATIONS **827** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Uncertainty in fitness functions. View project

Bioinspired Algorithms in Complex Ephemeral Environments (EphemeCH) View project

All content following this page was uploaded by Antonio J. Fernández-Leiva on 02 March 2015.

The user has requested enhancement of the downloaded file.

Procedural Content Generation for Real-Time Strategy Games

Raúl Lara-Cabrera, Mariela Nogueira-Collazo, Carlos Cotta and Antonio J. Fernández-Leiva

Lenguajes y Ciencias de la Computación department, Universidad de Málaga

***Abstract* — Videogames are one of the most important and profitable sectors in the industry of entertainment. Nowadays, the creation of a videogame is often a large-scale endeavor and bears many similarities with, e.g., movie production. On the central tasks in the development of a videogame is content generation, namely the definition of maps, terrains, non-player characters (NPCs) and other graphical, musical and AI-related components of the game. Such generation is costly due to its complexity, the great amount of work required and the need of specialized manpower. Hence the relevance of optimizing the process and alleviating costs. In this sense, procedural content generation (PCG) comes in handy as a means of reducing costs by using algorithmic techniques to automatically generate some game contents. PCG also provides advantages in terms of player experience since the contents generated are typically not fixed but can vary in different playing sessions, and can even adapt to the player herself. For this purpose, the underlying algorithmic technique used for PCG must be also flexible and adaptable. This is the case of computational intelligence in general and evolutionary algorithms in particular. In this work we shall provide an overview of the use of evolutionary intelligence for PCG, with special emphasis on its use within the context of real-time strategy games. We shall show how these techniques can address both playability and aesthetics, as well as improving the game AI.**

***Keywords* — Procedural Content Generation, Artificial Intelligence, game strategy, self-learning.**

I. Introduction

SPURRED on by the emergence of the videogame industry as the main component of the entertainment industry has motivated, research on videogames has acquired increasing notoriety during the last years. Such research spans many areas such as marketing and gamification, psychology and player satisfaction, computational intelligence, education and health (serious games) and computer graphics, just to cite a few. This diversification of research areas is largely motivated by a shift in the priorities of the video game industry: while games used to rely heavily on their graphical quality, other features such as the music, the player immersion into the game and interesting storyline have gained enormous importance. To cope with the plethora of new interesting challenges in the area of videogames, artificial and computational intelligence are turning out to be instrumental tools [25].

We recently carried out a mathematical, network-based study of the research community in the field of computational intelligence in video games [22] and obtained conclusive evidence of the vibrant activity of the field, which is steadily gaining momentum (as reflected in the growth patterns of new researchers and new publications). Still, the community of computational intelligence in video games is not yet fully developed, and collaboration links are still forming and improving the cohesion of the community. Besides, the industry is beginning to adopt the techniques and recommendations that academia offers.

Procedural Content Generation (PCG) refers to the algorithmic creation of content for video games, such as maps, levels, terrains, graphic textures, music, rules, quests, narrative, and missions among others possible [33]; traditionally, the creation of NPC (non-player controlled) behavior is not considered as PCG although, in a more global perspective, it is specific content for the game. The advantages of automatically creating video game content are manifold: firstly, it provides a drastic reduction in the cost and time of development as well as the memory used to store game artifacts; secondly, PCG provides a mechanism to inspire human artists to improve their creativity. Therefore, PCG can be considered from many different points of views and raises a high number of challenges from both Academic and Industry [35]. Moreover, the influence of PCG in, at least, other six areas in game programming, namely, NPC behavior learning, search and planning, games as Artificial Intelligence (AI) benchmarks, AI-assisted game design, general game AI, and AI in commercial games, underlines its importance [39, 40].

From the set of genres of videogames, Real-Time Strategy (RTS) games are one of the most exciting sub-genres since they require managing different kind of units and resources in real-time. In addition, they usually involve the participation of multiple players (not all of them necessarily human) that have to deal with incomplete information during the game; it is precisely this combination of resource management, multiplayer context and partial knowledge of the world what makes them an ideal framework to conduct Artificial Intelligence experiments; indeed, many challenging problems, such as resource allocation, adversarial real time planning,

DOI: 10.9781/ijimai.2015.325

spatial and temporal reasoning, opponent modeling, and opponent strategy prediction, just to name a few, can be addressed. As a result, RTS games offer a wide variety of fundamental AI research challenges [20].

In this context, one of the most interesting challenges in the videogame development process is precisely the procedural generation of content for RTS games as the artifact creation can be handled from many different perspectives due to the heterogeneity of the content that can be produced in RTS games, and to the participation of multiple (sometimes hundreds of) players with diverse profiles and skills. This work deals with the application of PCG techniques in RTS games, firstly by providing a brief review on this issue and, consequently, covering specific case studies in which evolutionary search has been employed to produce game components that satisfy certain properties.

II. PROCEDURAL CONTENT GENERATION

Videogames provide a wide range of fundamental problems that are useful for doing research in artificial intelligence. Among these we can cite real-time task planning and decision-making under uncertainty. This is particularly true in the case of Real-Time Strategy (RTS) games, which represent a whole genre of videogames in which the players must manage a collection of units and assets without a definite turn structure, that is, actions are asynchronously taken. Not surprisingly, RTS games have been used as researching tools to study and develop new artificial intelligence techniques, as explained in our paper about RTS games and computational intelligence [20].

The type of content that PCG techniques are able to create is very diverse, being maps and levels the prevailing type, as demonstrated by the large number of papers in the state of the art which are related to automatic level generation [14]. For example, Frade et al. introduced the use of genetic programming for evolving maps for video games, using in this process both human subjective evaluation and quality measures such as maps' accessibility [10] and edge length [11]. Another example of procedural level generation by Lanzi et al. [18] consists of evolving game maps that are specifically designed to improve the balance of the game, so no player has a marked superiority over the opponent (we will return to this issue later on).

Regarding other kind of content, Hastings et al. [12, 13], proposed a PCG algorithm for the game "Galactic Arms Race" in which the weapons available were generated on the fly. In this case, the fitness of the generated weapons was computed based on the amount of time the players used them, hence measuring the player satisfaction without requiring explicit feedback from the players. Onuczko et al. [31] presented a tool prototype for automatically producing specifications for missions and quests for a role-playing game. Font et al. [9] showed initial research towards a system capable of creating the rules for different card games. Collins [5] explored several approaches to procedural music composition.

Focusing on PCG for RTS games, Togelius et al. [36, 37] presented a multi-objective evolutionary algorithm whose objective was to create maps for this kind of games. Mahlmann et al. [26] described a search-based map generator for the game Dune 2, which was able to build playable maps using cellular automata (converting low-resolution matrices into maps fulfilling gameplay requirements). Finally, Ruela and Guimaraes [34] used a coevolutionary evolutionary algorithm aiming to maximize the performance of battle formations for the strategy game Call of Roma. We will also tackle coevolution later in this work.

III. CASE STUDIES

Historically, the success of a video game was directly associated with its graphical quality, but in the last decade this has changed and having good graphics does not necessarily ensure high sales. Players demand video games that show more than just a nice graphical quality and other issues, such as music, the story, or the atmosphere of the game, influence the decision of a player to get a specific game. The question of what it is that attracts the attention of players in a game is easy to answer: fun. How to obtain fun games and whether we can predict if the game will be of interest to players are not so easily answered, though.

There are several theories in the literature on what makes video games fun and why we play games [17], and, according to [4], a game's achievement might be deduced by measuring in advance the quality of the game (which seems however to be a difficult task). The notion of fun is difficult to measure as this depends on each player but it is naturally associated with the notion of player satisfaction: the greater the satisfaction, the greater the fun.

A. Playability-oriented PCG

This subsection focuses on the ability of PCG to engage the player (as commercial games demand) by keeping, during a match, an adequate trade off between the dynamism of the game and the balance between players which, probably, have different skills. More precisely, we aimed to generate maps for the RTS game Planet Wars, focusing on the properties that a priori make it entertaining and appealing to play, ensuring that the games are balanced (i.e., the forces of one of the players are not overwhelmingly larger than those of the other player – see [19]) and dynamic (i.e., action packed, there are battles and changes in the balance of power of the players – see [21]) For this purpose we are going to use evolutionary algorithms (EAs). An EA is a nature-inspired optimization and search method that deals with a set of entities (termed population), which represents a set of possible solutions. These entities, which are called individuals or chromosomes, compete against each other so the fittest individuals prevail over time, evolving towards better solutions. This is an iterative process where each step involves crossing (mixing information from several solutions) and mutating (performing random changes) individuals using genetic operators. Because individuals that represent the most appropriate solutions (as dictated by a so-

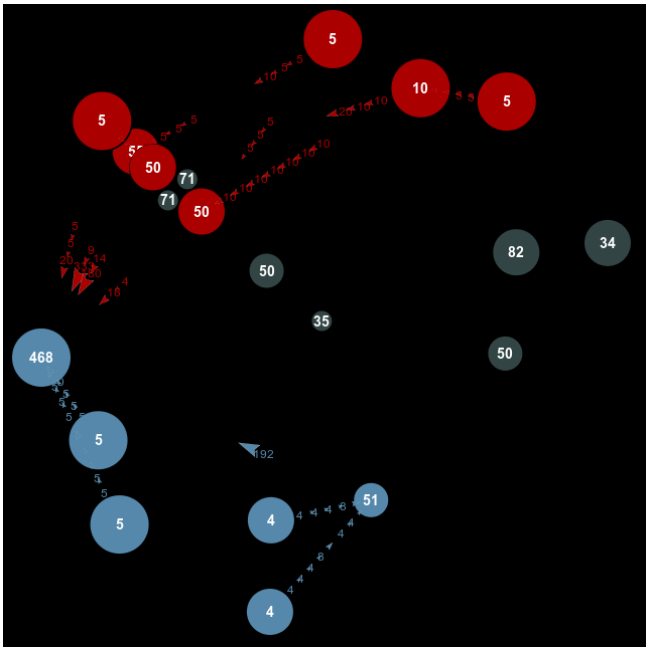


Fig. 1. A game of Planet Wars in progress. The arrows represent moving fleets while the number over the planets shows the number of stationed ships.

called fitness function that measure the goodness of solutions) are more likely to survive, the population gradually improves.

In order to use an EA, it is necessary to define several parameters: the individual's representation, the genetic operators, the size of the population and the number of generations the algorithm will be running.

Firstly, we had to consider how the solutions were to be represented and evaluated. A map for Planet Wars (see [24] for a description of the game) is defined as a collection of n_p planets distributed over a bi-dimensional plane. Each planet is characterized by its coordinates (x_i, y_i), its size s_i (determining the rate at which this planet produces new ships once captured by one of the players) and an initial number of ships w_i (determining the forces required to conquer the planet for the first time). As a result, a map can be described as a list $[\rho_1, \rho_2, \ldots, \rho_n]$, where each ρ_i is a tuple $\langle x_i, y_i, s_i, w_i \rangle$.

Two of the planets (the first two for simplicity) are initially marked as home planets of the players. From the point of view of the EA, the he number of planets n_p need not be fixed, and can range between an upper and a lower limit (15 and 30 in our experiments, see, e.g., Figure 1). In fact, one of the features of the evolutionary approach discussed later on is the ability to self-adapt to not only search parameters but also to the complexity (i.e., number of planets) of the map.

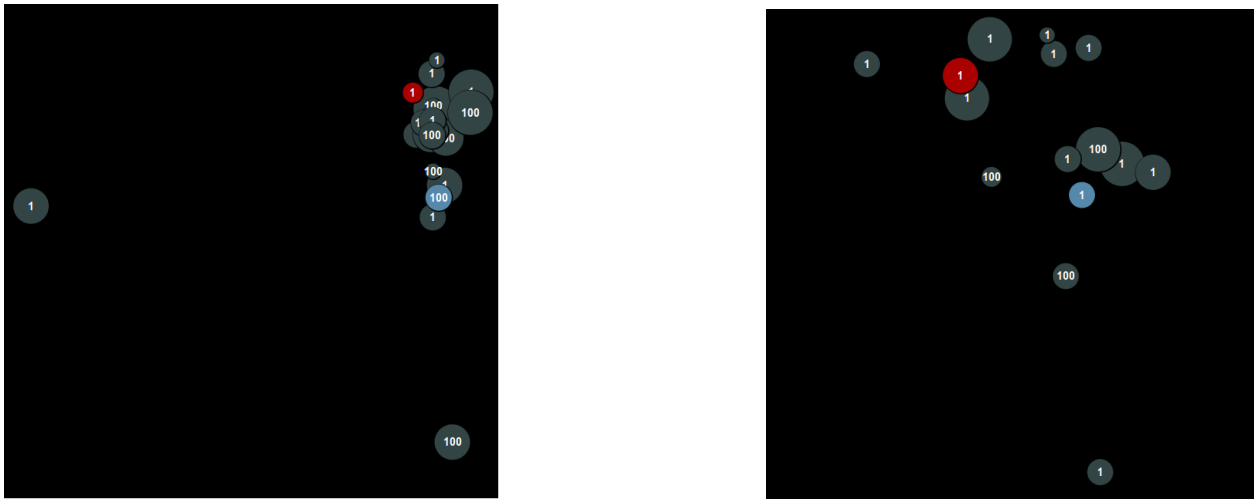
Regarding the evaluation of a map's playability features, we defined a tournament system which runs several games between an arbitrary number of pre-defined bots. Then, the tournament system analyzed some statistics gathered from each game in order to compute and quantify how balanced and dynamic the game was. Precisely, the system collects the following information from the *i*-th game (out of the total number of N_g games played in the tournament):

- *Territorial imbalance*: this is defined as the average imbalance in conquered planets throughout the game (the difference between the percentage of planets conquered by each player at each turn, averaged for all turns).
- *Growth imbalance:* this is measured analogously to the territorial imbalance, but considering the combined ship production capacity rather than the number of planets conquered (a player may have conquered many planets but these may be small, whereas other player may only dominated a few large planets).
- *Ship imbalance:* the same ideas sketched above are in this case applied to the number of ships (notice that a player can accumulate a large number of ships by following a passive strategy and not getting involved in fights and vice versa).
- *Game length:* this is just the percentage of the maximum number of turns played in the current game. Short games are imbalanced because it is implied that one of the player quickly destroys the fleet of its opponent.
- *Conquering rate:* this is the percentage of plantes conquered at the end of the game. If it is high, it means that the players have actively engaged in expanding their territories rather than sitting in their home planets.
- *Reconquering rate:* related to the previous measure, this is the average percentage of plantes whose ownership changes during the game (a high rate indicates that the players are actively fighting each other).
- *Peak difference:* this is actually a collection of variables, each of them measuring the maximal amplitude of the variation in any of the resources accounted for, in this case planets, combined size and ships.

These variables are subsequently averaged across the N_g games comprised in the tournament. In order to evaluate the actual balance and dynamism of a map we define a fuzzy rule base that captures some expert characterization of these features. For example, in order to account for balance we can use:

1) **if** territorial imbalance **is** LOW **and** growth imbalance **is** LOW **then** balance is HIGH
2) **if** territorial imbalance **is** HIGH **and** growth imbalance **is** low and ship imbalance **is** LOW **then** balance *is* MEDIUM
3) **if** (territorial imbalance **is** LOW **and** growth imbalance **is** HIGH) or game length **is** LOW **then** balance **is** LOW

Intuitively, we consider that a map has high balance if the average imbalance in planets and growth is low during the game. If one of the players has material advantage in terms of planets and ships, even if the combined growth is similar, the we deem the map to have medium balance. Finally, if both players manage to conquer a similar number of planets but their sizes are disparate or the game length is short, the map has low balance. Of course, there is some room for refining this characterization of balance by considering other combinations of the variables, but the above serves as an illustrative example. Similarly, we can define fuzzy rules for dynamism. For example, we can state that



(a) (b)

Fig. 2. Two examples of maps that have been generated by the algorithm. Planet's colors denote whether it is conquered by some player (red/blue) or remains neutral. The number shows how many ships are defending each planet.

1) **if** conquering rate **is** HIGH **and** reconquering rate **is** HIGH **then** *dyn* **is** HIGH
2) **if** all peak differences **is** HIGH **then** *dyn* **is** HIGH

i.e., if there are many planets being conquered and reconquered, or the peak differences in all three resources is high, then dynamism is high (there are battles and action).

Likewise if it turns out that one of the peak differences is high but any of the other two is not, the dynamism can be said to be intermediate (this would be captured in a family of three rules). Finally, if all peak differences are low or the conquering and reconquering rates are not high and the game length is very short, the dynamism of the map is considered to be low.

The procedural map generator used a self-adaptive evolutionary approach with the solutions encoded as mixed real-integer vectors. The parameters governing mutation were also a part of the solutions, thus providing the means for self-adapting them (see [24] for a full explanation of the evolutionary algorithm and its parameters and operators). The players of the tournament system used to assess the quality of the maps during the evaluation phase were three bots submitted to the *Google AI Challenge 2010*, namely *Manwe*, *Flagscapper's bot* and *fglider's bot*. All of them ranked in the top 100 (there were over 4600 participants) and their source code was available – see [24] for the URLs.

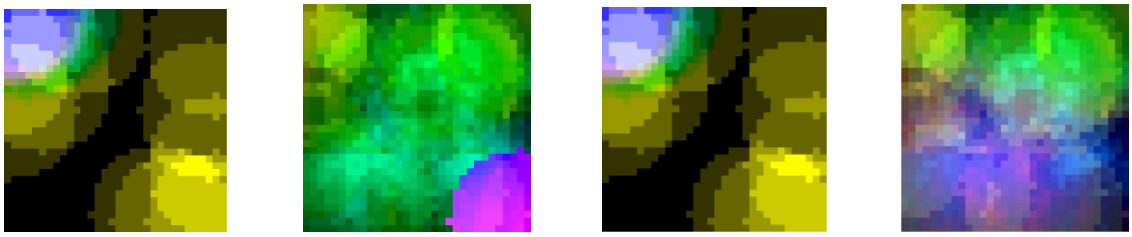
Experiments focusing separately in either of the two properties point at the higher difficulty of attaining dynamism with respect to balance. Figure 2 shows an example of the maps obtained[6]. Of course it is possible to optimize both properties at the same time following a multi-objective approach (the Non-dominated Sorting Genetic Algorithm II – *NSGA-II*– in our case). By doing so, we can obtain a collection of solutions representing different tradeoffs between balance and dynamism (ranging from highly balanced and lowly dynamic to highly dynamic and poorly balanced, with different intermediate scenarios in which an increase in one the properties is traded by a decrease in the other). Note in this sense that a single-objective approach cans easily exploit the first objective (i.e. balance), providing maps that achieve perfect balance due to the complete inaction of the players. However, the situation is different from the point of view of dynamism, since according to our definition a very unbalanced game is likely going to be short or feature less alternation between the players, hence resulting to be non-dynamic as well. For this reason, the multiobjective approach yields a graceful degradation of dynamism when balance is increased, eventually exhibiting an abrupt reduction of the dynamism upon reaching the high end of balance. Further studies show that, in general, dynamic games seem to be related to maps featuring a larger number of planets, widely scattered on the map and whose sizes are positively correlated to the initial number of ships.

[6] It is possible to watch a game on these maps at http://www.lcc.uma.es/~raul/maps/maps.html

B. Introducing Aesthetics

In Section **¡Error! No se encuentra el origen de la referencia.**III-A we focused on making the game more fun to play, obtaining games that are balanced and dynamic. However, the generated maps lacked aesthetics (for example, maps with all their planets clustered in a small region, see Figure 2), which is an interesting feature apart from the fun that may lead to increase the player satisfaction. It turns out that fun and aesthetics are two complementary ways of achieving the same goal [27]. Moreover, non- aesthetic maps may confuse the player, reducing his/her satisfaction or even leading him/her to stop playing the game.

Following a similar evolutionary scheme and representation of the solutions for the automatic generation of balanced and dynamic maps, we considered different properties in order to evaluate the aesthetics of maps. We establish a separation between geometrical features (based on the spatial properties of the map, namely coordinates and distances), and topological features (based on qualitative



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

Fig. 3. Map's distribution over the SOM for both geometric (a) and topological (b) approaches. Yellow for non-aesthetic, cyan for aesthetic and magenta for non-dominated. (c) and (d) show the topological approach solution projected over the geometric approach SOM and vice versa.

relationships among planets invariant under geometrical transformations such as rotation, translation or scaling). We also take into account morphological features based on individual planet properties, such as size or initial number of ships.

These are the geometrical measures:

- *Spatial distribution of planets*: given planet coordinates we compute the average distance between planets μ_d and the standard deviation of these distances σ_d.
- *Planet features*: given the sizes and initial number of ships of each planet, we compute the average and standard deviation of sizes (μ_s and σ_s respectively) and Pearson's correlation ϱ between sizes and number of ships.

Thus, we can characterize a map by a 5-tuple $\langle \mu_d, \sigma_d, \mu_s, \sigma_s, \varrho \rangle$, and use some distance measure (e.g., Euclidean distance) to determine the geometrical distance among two maps.

As to the topological features, these are extracted from the sphere-of-influence graph (SIG) of each map, which sets a relationship between some set of points based on their spatial arrangement [38] (defining a planet's radius of influence as the shortest distance of any other planet, and defining a graph in which vertex is a planet and edges are defined between planets whose distance is less or equal to the sum of their respective radii of influence). Using this SIG we can compute:

- *Number of connected components*: number of maximal sub-graphs in which any two vertices are connected by at least one path.
- *Average node's degree*: average number of edges incident to each node.
- *Density of the graph*: ratio between the number of edges of the graph and that of a complete graph with the same number of vertices.
- *Average clustering coefficient*: average percentage of each node's neighbors which are neighbors of each other too.
- *Pearson correlation between the size of the nodes and their betweenness centrality*. Betweenness is a measure of the importance of each node as an intermediate gateway in the paths between any other two nodes. We measure is highly central nodes are also large planets.
- *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 (i.e., the extent to which plantes are linked to other planets of larger or smaller size)

As with geometrical measures, these topological measures can be used to characterize a map and define a distance metric among them. However, some of these measures turn out to be somewhat redundant. By considering a collection of 20 maps (10 with good aesthetics and 10 with bad aesthetics as tagged by a human expert) and using a Random Forest classifier to determine which measures are useful for classification purposes we obtain that graph's density, correlation between node size and betweenness and size assortativity are the most relevant ones – see [23] for further details.

If we run an EA using distance to aesthetic maps (to be minimized) and to non-aesthetic maps (to be maximized) in a multi-objective approach, we observe that there is a smooth, linear transition between these two objectives. More qualitatively, we created two self-organizing map (SOM) [16] with 32×32 process units over a non-toroidal rectangular layout, one for each characterization approach (geometrical and topological). As we can see in Figure 3, the SOM of the geometrical approach set a separation between non-aesthetic (yellow zones) and aesthetic maps (cyan zones), as well as generated maps (magenta zones) share the same region as aesthetic maps. Thus, they can be considered aesthetic as well. Regarding the topological approach, the distinction between aesthetic and non-aesthetic maps is not so clear though, as shown by the overlapped areas.

C. Self-learning of RTS strategies

As another branch of PCG, the search of game strategies via computational intelligence (CI) emerges as an important sub-field. RTS games are specifically distinguished for imposing the players the control of many different resources during the game. For this reason the procedural generation of game strategies should be backed up by methods allowing a significant reduction of the computational time involved in the exploration of the large search spaces implied. We are here specifically concerned with the use of techniques providing continuous, autonomous learning capabilities for the artificial intelligence embedded in a RTS game. We consider coevolution for this purpose.

Coevolution is a model inspired in the principles of natural evolutionary theory. It is based on the interaction between different species and can take two forms: one based in the collaboration and other one based on competition. Cooperative approaches simulate a symbiotic relationship, used for finding

Fig. 4. Screenshot of RobotWars game.

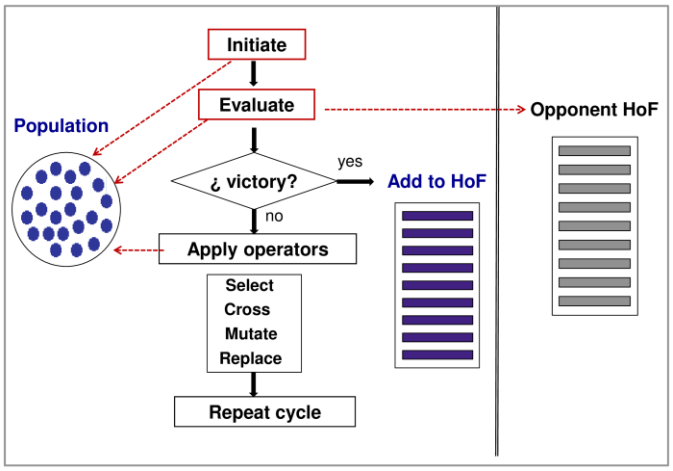


Fig. 5. Basic coevolutionary cycle which uses the HoF during the evaluation process.

a solution through the collaboration between many possible solution components; on the other hand, competitive approaches establish a competition between individuals much like a predator/prey environment. The goal is to trigger an "arms race" in which the improvement of some individuals stimulates the improvement in the opponents, and vice versa. This last approach is usually used for solving optimization problems in inherently competitive contexts like games.

Several experiments have showed significant results in the application of coevolutionary models as a mechanism of self-learning in a RTS. For example, different variants of competitive coevolutionary (CC) algorithms [3], [15], [1] have been proposed to find optimal strategies for the Tempo game. Also, the authors of [2] analyzed the employment of coevolution for creating a tactical controller for small groups of game entities in a real-time capture-the-flag game. The proposal described in [6] explores several methods for automatically shaping the coevolutionary process by modifying the fitness function as well as the environment during evolution.

The success of the application of coevolutionary approaches is out of question but coevolution has also its own intrinsic problems – see [8], [7]. In particular the evaluation mechanism is a key point in a coevolutionary model because it guides the arms race that emerges from the interactions between individuals. For this reason several evaluation approaches have been proposed in the literature to alleviate some of the coevolutionary pathologies. In this line of work we have already explored the use of the Hall-of-Fame (HoF) [32] based mechanism as an archive method to memorize the successful solutions to guide the search process for generating game strategies in RTS games. This mechanism is used to provide a long term memory of the coevolutionary process, avoiding that some good strategies are forgotten due to lack of selective pressure.

Our first works [28],[29] were conducted in the context of the RTS game *RobotWars*[7]. The main goal was generating game strategies to control the behavior of an army. RobotWars is a self-developed game for testing game AI strategies (i.e. bots), and hence human players do not have place here. It is a two player's game, in which two different armies fight in a 3D scenario with many obstacles (Figure 4 shows a screenshot of this game). Each army has different units and one general; if an army wipes out the enemy general them they will be the winners of the game.

Using this RTS game five variants of a CC algorithm using HoF as a memory mechanism to keep the winning strategies were tested. In our model the individual was represented as a matrix of actions that allows to control, deterministically, the behavior of an army during the game. The basic coevolutionary schema implemented is showed in Figure 5. It is based in coevolutionary turns of multiple strategies for each army. The goal is to find a winning strategy which is then put in that player's HoF. That HoF is then used in the evolution of strategies for the other army until a new winning strategy is found, placed in the corresponding HoF, and the roles are reversed again. If at the end of the coevolutionary turn no solution is obtained, a new turn starts again until a champion is found or until the maximum number of cycles is reached.

During experiments in RobotWars we analyzed how the diversity and growth of the HoF can influence the quality of the solutions obtained by HoF-based CC algorithms. In this sense we studied the performance of eleven algorithms based on different mechanisms for maintaining and updating the champions' memory during the evaluation process. This was aimed to reduce the size of the HoF (hence reducing computational time) but doing so in an intelligent way, without losing the beneficial contribution of the long term memory. A *diversity* indicator based on the contribution to each champion to the diversity of the HoF diversity showed a good performance (i.e., the HoF was reducing by removing similar champions which did not contribute much to the coevolutionary learning). We also detected that manipulating the size of the HoF has a direct influence on the quality of the search result due to the loss of transitivity (a solution *A* beating another solution *B* which in turns beats *C* which can however beat *A*), so this should be done carefully.

That previous work was extended in [30] proposing a different evaluation mechanism to exploit the potential offered

[7] http://www.lcc.uma.es/~afdez/robotWars

by archive methods to maintain transitivity between the solutions; we considered a new RTS game –*Planet Wars*, described before– allowing a deeper experimental analysis and more consistent conclusions. This time we added novel strength indicators that were independent from the fitness function with the objective of avoiding the appearance of cycling (strategies being forgotten and re-discovered over and over again). The novelty of this last aspect consisted of incorporating into our prime CC algorithm which used the HoF as shown in Figure 5, an additional archive (termed call-of-celebrities, HoC) that contained a team of experienced virtual players. These were used to evaluate how strong a candidate was. The combined use of both halls (HoF and HoC) with the (possibly combined) utilization of diversity and quality metrics helped the optimization to obtain competitive bots that self-adapt to beat their (co)evolved enemies.

IV. CONCLUSION

Procedural Content Generation (PCG) is one of the corner stones of the modern video game industry. Throughout this paper we have described three case studies that are part of our work in the area of PCG for real-time strategy video games. In the first place, we have presented and compared several methods for generating maps for the game Planet Wars; such maps are firstly oriented to fulfill the requirements of the player in terms of playability, that is, providing an interesting and enjoyable experience as to what the game mechanics regards. This has been done characterizing some positive features a game should have such as balance (having an opponent with similar skills as the player, as reflected in the achievements of the former in the game with respect to those of the latter) and dynamism (delivering an existing game in which numerous events unfold and there are changes in the balance of power between the two players). It has been shown how maps with these features can be accomplished by using an evolutionary approach for their automatic generation. Subsequently, we have considered the aesthetics perspective. Given the highly subjective nature of this endeavor, the input of an expert is required in order to provide samples of aesthetic/non-aesthetic maps, which can be in turn used by an evolutionary algorithm as reference to reproduce features of good maps, and avoid features of bad maps. Such features admit different characterizations; we have described the use of both geometrical (based on the spatial distribution of map components), morphological (based on the individual properties of map components) and topological (based on properties of the maps which are invariant under simple geometrical transformations). By using an unsupervised learning method we can infer that an evolutionary approach based on these characterizations is capable of producing aesthetic maps.

Afterwards, we have extended the classical view of PCG by considering game AI as game content; in particular, we have considered NPC behavior and we have briefly described a self-learning approach that we employed on two RTS games with significant success. To do so, we used co-evolutionary techniques to lead the search process in a competitive context; we have also shown that our algorithmic proposals were based on the concept of Hall-of-fame (HoF) that basically represents a memory that allows to store the best candidates that are further employed in the evaluation phases to improve the optimization process. A number of different structures and mechanisms to select the champions to be stored in the HoF can be defined and this selection can have drastic influence in the results.

Many lines remain open; for instance, in order to accelerate the creation process (and as consequence, to minimize development costs), the industry demands the automatic generation of diverse content at the same time; moreover, there are artifacts that surely influence the creation of other class of elements, and vice versa. This basically means that PCG should be defined to enable the generation of contents (of distinct nature) at the same time with the goal of producing compound components. Our next step follows precisely this line of research and it consists of designing PCG methods to co-evolve graphical content (e.g., maps/levels) and game AI. In addition, obtaining correct quality metrics is an area that deserves more research; the evolutionary search directed to find high quality content heavily depends on the fitness functions that guide the optimization process, and it is not easy to evaluate the goodness of these; moreover, content creation is directly related to human creativity and, therefore, humans (both developers and players) are required to be involved in the evolution process: in this sense, designing correct user-centric interaction evolutionary models is also another line of exciting research.

REFERENCES

[1] P. Avery et al., "Coevolving a computer player for resource allocation games: using the game of tempo as a test space." Ph.D. dissertation, School of Computer Science University of Adelaide, 2008.

[2] [2] P. Avery and S. J. Louis, "Coevolving team tactics for a real-time strategy game," in IEEE Congress on Evolutionary Computation. Barcelona, Spain: IEEE, 2010, pp. 1–8.

[3] P. M. Avery and Z. Michalewicz, "Static experts and dynamic enemies in coevolutionary games," in IEEE Congress on Evolutionary Computation. IEEE, 2007, pp. 4035–4042.

[4] C. Browne and F. Maire, "Evolutionary game design," IEEE Trans. Comput. Intellig. and AI in Games, vol. 2, no. 1, pp. 1–16, 2010.

[5] [5] K. Collins, "An introduction to procedural music in video games," Contemporary Music Review, vol. 28, no. 1, pp. 5–15, 2009.

[6] A. Dziuk and R. Miikkulainen, "Creating intelligent agents through shaping of coevolution," in IEEE Congress on Evolutionary Computation. New Orleans, LA, USA: IEEE, 2011, pp. 1077–1083.

[7] M. Ebner, R. A. Watson, and J. Alexander, "Coevolutionary Dynamics of Interacting Species," in Applications of Evolutionary Computation, EvoApplicatons 2010 (EvoApplications (1)), ser. Lecture Notes in Computer Science, C. D. Chio et al., Eds., vol. 6024. Istanbul, Turkey: Springer, 2010, pp. 1–10.

[8] S. G. Ficici and A. Bucci, "Advanced tutorial on coevolution," in 2007 GECCO Conference Companion on Genetic and Evolutionary Computation. New York, USA: ACM, 2007, pp. 3172–3204.

[9] J. Font, T. Mahlmann, D. Manrique, and J. Togelius, "A card game description language," in Applications of Evolutionary Computation, ser. Lecture Notes in Computer Science, A. Esparcia-Alcázar, Ed. Springer Berlin Heidelberg, 2013, vol. 7835, pp. 254–263.

[10] M. Frade, F. F. de Vega, and C. Cotta, "Breeding terrains with genetic terrain programming: The evolution of terrain generators," International Journal of Computer Games Technology, vol. 2009.
[11] ------, "Evolution of artificial terrains for video games based on obstacles edge length," in IEEE Congress on Evolutionary Computation. IEEE, 2010, pp. 1–8.
[12] E. J. Hastings, R. K. Guha, and K. O. Stanley, "Automatic content generation in the Galactic Arms Race video game," Computational Intelligence and AI in Games, IEEE Transactions on, vol. 1, no. 4, pp. 245–263, 2009.
[13] ------, "Evolving content in the Galactic Arms Race video game," in Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on. IEEE, 2009, pp. 241–248.
[14] 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.
[15] R. Johnson, M. Melich, Z. Michalewicz, and M. Schmidt, "Coevolutionary Tempo game," in Evolutionary Computation. CEC'04. Congress on, vol. 2, 2004, pp. 1610–1617.
[16] T. Kohonen, "The self-organizing map," Proceedings of the IEEE, vol. 78, no. 9, pp. 1464–1480, 1990.
[17] R. Koster, A theory of fun for game design. Paraglyph Press, 2004.
[18] P. L. Lanzi, D. Loiacono, and R. Stucchi, "Evolving maps for match balancing in first person shooters," in 2014 IEEE Conference on Computational Intelligence and Games, CIG 2014, Dortmund, Germany, August 26-29, 2014, 2014, pp. 1–8.
[19] R. Lara-Cabrera, C. Cotta, and A. J. Fernández-Leiva, "A procedural balanced map generator with self-adaptive complexity for the real-time strategy game planet wars," in Applications of Evolutionary Computation 2013, ser. Lecture Notes in Computer Science, A. Esparcia-Alcázar et al., Eds., vol. 7835.Berlin Heidelberg: Springer-Verlag, 2013, pp. 274–283.
[20] ------, "A review of computational intelligence in RTS games," in IEEE Symposium on Foundations of Computational Intelligence, FOCI 2013, Singapore, Singapore, April 16-19, 2013, 2013, pp. 114–121.
[21] ------, "Using self-adaptive evolutionary algorithms to evolve dynamism-oriented maps for a real time strategy game," in Large-Scale Scientific Computing - 9th International Conference, LSSC 2013, Sozopol, Bulgaria, June 3-7, 2013. Revised Selected Papers, 2013, pp. 256–263.
[22] ------, "An analysis of the structure and evolution of the scientific collaboration network of computer intelligence in games," Physica A: Statistical Mechanics and its Applications, vol. 395, no. 0, pp. 523 – 536, 2014.
[23] ------, "Geometrical vs topological measures for the evolution of aesthetic maps in a rts game," Entertainment Computing, vol. 5, no. 4, pp. 251–258, 2014.
[24] ------, "On balance and dynamism in procedural content generation with self-adaptive evolutionary algorithms," Natural Computing, vol. 13, no. 2, pp. 157–168, 2014.
[25] S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, Eds., Artificial and Computational Intelligence in Games, ser. Dagstuhl Follow-Ups, vol. 6. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
[26] T. Mahlmann, J. Togelius, and G. N. Yannakakis, "Spicing up map generation," in EvoApplications, ser. Lecture Notes in Computer Science, C. D. Chio et al., Eds., vol. 7248. Springer, 2012, pp. 224–233.
[27] M. Nogueira, C. Cotta, and A. J. Fernández-Leiva, "On modeling, evaluating and increasing players' satisfaction quantitatively: Steps towards a taxonomy," 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. 245–254.
[28] M. Nogueira, J. Gálvez, C. Cotta, and A. J. Fernández-Leiva, "Hall of Fame based competitive coevolutionary algorithms for optimizing opponent strategies in a new RTS game," in 13th annual European conference on simulation and AI in computer games (GAME-ON 2012), A. F.-L. et al., Ed. Málaga, Spain: Eurosis, November 2012, pp. 71–78.
[29] M. Nogueira Collazo, C. Cotta, and A. J. Fernández Leiva, "An analysis of hall-of-fame strategies in competitive coevolutionary algorithms for self-learning in RTS games," in Learning and Intelligent Optimization - 7th International Conference, LION 7, Catania, Italy, January 7-11, Revised Selected Papers, ser. Lecture Notes in Computer Science, G. Nicosia and P. M. Pardalos, Eds., vol. 7997. Springer, 2013, pp. 174–188.
[30] ------, "Virtual player design using self-learning via competitive coevolutionary algorithms," Natural Computing, vol. 13, no. 2, pp. 131–144, 2014.
[31] C. Onuczko, D. Szafron, J. Schaeffer, M. Cutumisu, J. Siegel, K. Waugh, and A. Schumacher, "Automatic story generation for computer role-playing games." in AIIDE, 2006, pp. 147–148.
[32] C. Rosin and R. Belew, "New methods for competitive coevolution," Evolutionary Computation, vol. 5, no. 1, pp. 1–29, 1997.
[33] N. Shaker, J. Togelius, and M. J. Nelson, Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer, 2014.
[34] A. Siqueira Ruela and F. Gadelha Guimaraes, "Coevolutionary procedural generation of battle formations in massively multiplayer online strategy games," in Computer Games and Digital Entertainment (SBGAMES), 2014 Brazilian Symposium on, Nov 2014, pp. 89–98.
[35] J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. O. Stanley, "Procedural content generation: Goals, challenges and actionable steps," in Artificial and Computational Intelligence in Games, ser. Dagstuhl Follow-Ups, S. M. Lucas, et al., Eds. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013, vol. 6, pp. 61–75. [Online].
[36] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelback, and G. N. Yannakakis, "Multiobjective exploration of the Starcraft map space," in Computational Intelligence and Games (CIG), 2010 IEEE Symposium on. IEEE, 2010, pp. 265–272.
[37] J. Togelius, M. Preuss, and G. N. Yannakakis, "Towards multiobjective procedural map generation," in Proceedings of the 2010 Workshop on Procedural Content Generation in Games. ACM, 2010, p. 3.
[38] G. T. Toussaint, "A graph-theoretic primal sketch," Computational Morphology, pp. 229–260, 1988.
[39] G. Yannakakis and J. Togelius, "A panorama of artificial and computational intelligence in games," Computational Intelligence and AI in Games, IEEE Transactions on, vol. PP, no. 99, pp. 1–1, 2014.
[40] Broncano, C. J., C. Pinilla, R. Gonzalez-Crespo, and A. Castillo-Sanz, "Relative Radiometric Normalization of Multitemporal images", International Journal of Artificial Intelligence and Interactive Multimedia, vol. 1, issue A Direct Path to Intelligent Tools, no. 3, pp. 53-58, 12/2010

Raúl Lara-Cabrera received the Bachelor's degree in computer science and the Master's degree in software engineering and artificial intelligence from the University of Málaga (UMA), Spain, where he is currently pursuing the Ph.D., focusing his research on fields related to artificial intelligence and video games. He is currently a researcher with the Lenguajes y Ciencias de la Computación department.

Mariela Nogueira-Collazo obtained the Bachelors degree in Computer Science by the University of Computer Science (UCI) of The Havana, Cuba; and the Master's degree in Software Engineering and Artificial Intelligence from the University of Málaga (UMA), Spain. Currently she is pursuing the Ph.D. in UMA and her research cover the application of artificial intelligence in videogames context.

Carlos Cotta obtained his MSc and PhD in Computer Science from the University of Málaga (UMA), Spain in 1994 and 1998 respectively. He holds a tenured Professorship in the Department of Lenguajes y Ciencias de la Computación of this University since 2001. His main research areas involve metaheuristic optimization -in particular hybrid and memetic approaches- with a focus on both algorithmic and applied aspects (particularly combinatorial optimization) as well as complex systems.

Antonio J. Fernández-Leiva received, in 2002, the PhD degree in Computer Science from the University of Málaga (UMA), where he is currently associate professor in the Lenguajes y Ciencias de la Computación department. In the past, he worked in private companies as computer engineer. His main areas of research involve both the application of metaheuristics techniques to combinatorial optimization and the employment of Computational Intelligence in Games. He also leads a Master on Design and Programming of Videogames at UMA.

View publication stats