Map Generation and Balance in the Terra Mystica Board Game Using Particle Swarm and Local Search

Luiz Jonatã Pires de Araújo¹(✉), Alexandr Grichshenko¹, Rodrigo Lankaites Pinheiro², Rommel D. Saraiva³, and Susanna Gimaeva¹

¹ Innopolis University, Innopolis 420500, Russia
                            l.araujo@innopolis.university
² Webroster, Peterborough, UK
³ University of Fortaleza, Fortaleza, Brazil

Abstract. Modern board games offer an interesting opportunity for automatically generating content and models for ensuring balance among players. This paper tackles the problem of generating balanced maps for a popular and sophisticated board game called Terra Mystica. The complexity of the involved requirements coupled with a large search space makes of this a complex combinatorial optimisation problem which has not been investigated in the literature, to the best of the authors’ knowledge. This paper investigates the use of particle swarm optimisation and steepest ascent hill climbing with a random restart for generating maps in accordance with a designed subset of requirements. The results of applying these methods are very encouraging, fully showcasing the potential of search-based metaheuristics in procedural content generation.

Keywords: Combinatorial optimisation · Particle swarm · Procedural content generation · Steepest ascent hill climbing with random restart · Terra Mystica

1 Introduction

Procedural content generation (PCG) occupies an essential role in the development of modern video games and tabletop games [21]. Computation techniques such as optimisation algorithms and machine learning have been used, for example, to generate new maps conveying features such as a balanced distribution of resources and conditions among players [10]. However, there has been limited work assessing different metaheuristics for generating maps in games in the various media. One of the reasons is the complexity and nonlinearity of constraints involved in such games [15]. Moreover, there is a need for quick and online map generation, which prevents the use of computationally expensive approaches [20].

Terra Mystica (TM) has been one of the most popular and complex tabletop Euro game in the market for years. In TM, players from different factions should evolve an economy building engine which is profoundly affected by the initial
distribution of resources in the map, notably the proximity of favourable types of terrain. One of the core elements in TM is, therefore, terraforming, which is an action that converts a different type into another. Players use spades (from 1 to 3 based on the terrain types) to terraform regions. The greater number of required spades corresponds to the greater complexity of the action. The difficulties in achieving map balance for different factions and a long and complicated list of requirements for the map contribute to the limited number (only three) of available maps.

The complexity of the search space of the described problem suggests the use of artificial intelligence techniques. The aim of the paper is to assess population-based metaheuristics applied to the task of generating maps for TM, which also convey balanced features. The compared algorithms are particle swarm optimisation (PSO) and steepest ascent hill climbing with random restart (SAHC). Applications of PSO and SAHC to PCG in such a scale have not been reported to the best of authors’ knowledge.

The remaining part of the paper is organised according to the following structure: Sect. 2 presents a survey of the state-of-the-art of map generation using evolutionary techniques and approaches to ensure map balance. In Sect. 3, we describe a mathematical model for the requirements which the generated maps must meet as well as implementation details of selected algorithms. Subsequently, Sect. 4 provides an overview of the obtained results and the performance analysis of PSO and SAHC. To conclude, the contributions of the paper are evaluated, and suggestions for future research are outlined in Sect. 6.

2 Related Work

2.1 Map Generation Approaches

Map generation is an essential part of PCG that has attracted attention, especially in recent years, with search-based algorithms emerging as the dominant approach [10]. Togelius et al. [21] categorise the algorithms for map generation in two main approaches: constructive methods, which incrementally build complete solutions from a partial solution; and generate-and-test methods, which iteratively examine candidate solutions [20].

One of the first examples of a generate-and-test algorithm applied for map generation was reported in [18] for the Almansur Battlegrounds, a turn-based game. A similar method was also employed to Dune 2, which is a real-time strategy game in which the terrain affects the performance of the player [14]. Both studies succeeded in producing playable maps, although the authors recognise that map balance issues were not completely resolved. Studies applying genetic algorithms and genetic programming for Siphon [17] and Planet Wars [12] achieved slightly better results. However, these studies operated in a relatively small search space, which is not the case for more realistic maps. More recent examples of PCG approaches in map generation include applications of reinforcement learning for Zelda and Sokoban games [9] and quality-diversity
algorithms for *Mario* [8]. Interestingly, there has not been much work demonstrating the use of computation techniques for generating maps in tabletop games. The example of generate-and-test approach with Tabu search have shown the relevance of such methods for balanced map creation in Terra Mystica board game [2].

2.2 Achieving Map Balance

One prerequisite for a balanced map is that it must provide equal chances of winning to players of equal skills and that no starting position can guarantee the victory [22]. According to [21], map balance is often achieved by evaluation functions consisting of features and corresponding weights inferred by the game designer. Another method involves the use of artificial agents to playtest several maps, which is a time-consuming process. Examples of evaluation function-based methods for map balance include studies on *StarCraft* [19], *Civilization* [5] and *Ms PacMan* [16] games. A similar methodology was proposed by Ashlock et.al using dynamic programming to design more suitable fitness functions [4].

3 Methodology

This section presents the employed evaluation functions and search space of maps in TM. Moreover, it shows the implementation details of the PSO and SAHC metaheuristics for this particular domain.

3.1 Mathematical Formalisation for TM Map

Maps in TM must conform several requirements regarding the distribution of types of terrains. For this study, the following requirements have been implemented: no land (or non-river) hexagon has a neighbour of the same terrain type (REQ1); each river hexagon has between one and three river neighbours to prevent the occurrences of lakes (REQ2); the number of disconnected river components should be one (REQ3); each land hexagon has at least one neighbour which can be terraformed using only one spade (REQ4). While REQ2 and REQ3 promote the generation of maps that resemble the original map shown in Fig. 1, the remaining requirements (REQ1 and REQ4) promote map balance among the factions.

Equation 1 gives the number of times in which REQ1 is violated \( f_1 \). Let \( h_i \) be the \( i^{th} \) hexagon in the list of 113 hexagons in the map; \( isRiver(h_i) \) is 1 if \( h_i \) is a river hexagon, and 0 otherwise; \( n_j \) is an element of \( nbs(h_i) \), which is the set of hexagons which are neighbours of \( h_i \).

\[
f_1 = \sum_{i=0}^{112} \left( 1 - isRiver(h_i) \right) \sum_{j=0}^{\left| nbs(h_i) \right|} equal(h_i, n_j) \tag{1}
\]
Equation 2 calculates the number of river hexagons that does not meet the requirement REQ2.

\[ f_2 = \sum_{i=0}^{112} (\text{isRiver}(a_i)(1 - \min(1, \sum_{j=0}^{\text{nbs}(a_i)} \text{isRiver}(n_j)))) \]  

Next, let \( \text{connected}(R) \) be the number of river components in the map, which is preferably one (REQ3). A simple graph search algorithm like breadth first search can be used for calculating the number of river components. The number of violations of REQ3 is given in Eq. 3, shown as follows.

\[ f_3 = \text{connected}(R) - 1 \]  

Terraforming, i.e. casting one type terrain into another, is one of the most important actions in the game and has an important role for reaching map balance between different factions. Each faction has a “home terrain” where it can build. In Fig. 2, the Nomads faction has as home terrain desert (yellow) and it would need to spend two spades (indicated in the figure) to convert mountains (grey) into desert. We refer to the number of spades to terraform a terrain into another as the spade-distance between them.
Let \( isOneSpade(h_i, n_j) \) be a function that determines whether it is possible to terraform a hexagon \( n_j \) which is neighbour of the hexagon \( h_i \) using only one spade (i.e. the spade-distance between \( n_j \) and \( h_j \) equals to 1). Equation 4 calculates the number of hexagons that does not conform to the requirement (REQ4).

\[
f_4 = \sum_{i=0}^{112} ((1 - isRiver(h_i))(1 - \min(1, \sum_{j=0}^{\text{nbs}(h_i)} isOneSpade(h_i, n_j))))
\] (4)

Lastly, the minimisation objective function for a map in TM (\( F_{total} \)) is given by Eq. 5. The generated map has not violated requirements REQ1 - REQ4 has \( F_{tot} \) equal to 0.

\[
F_{tot} = f_1 + f_2 + f_3 + f_4
\] (5)

In a TM map, there are 36 river tiles and 11 hexagons of each of the 7 type terrains. This gives rise to a search space \( S \) of approximately \( 3.7 \times 10^{80} \) candidate solutions, which is significantly larger that several other problem domains explored in the PCG literature.

### 3.2 Particle Swarm Optimisation

Particle swarm optimisation (PSO) is a paradigm for optimising non-linear functions, inspired by the evident collective intelligence in several natural systems such as bird flock, bee swarm, among others [1]. For example, several studies confirm that different kinds of animals often avoid predators more effectively when in a group than individually [11]. PSO starts initialising the population of particles with random position and velocities. Each particle tracks its best position (pbest) according to the evaluation function. In addition, the best position across all particles (gbest) is tracked. The positions of the particles are updated at each step, as well as their velocities based on random variables and predefined parameters [7].

The PSO was implemented as follows. Because a feasible solution must have a fixed number of each type of tiles, we first created a list \( \Gamma \) containing the correct amount of each type of tiles, e.g. 11 tiles of each colour and 36 river tiles. We represented the solution as the vector \( \Delta \), the same size as \( \Gamma \), where each position \( \Delta_i \) can be an integer number between 0 and \( |\Gamma| \) corresponding to an element of \( \Gamma_\Delta \) and a coordinate of a TM map. Thus, the content of an element of the solution vector is a number which maps to an element of \( \Gamma \).

However, that representation would not allow repeated entries in \( \Delta \), creating regions of infeasibility for the algorithm. In order to eliminate repeated entries (and limit the search to the feasible solution space), before converting \( \Delta \) to a TM map and evaluating its fitness, for every \( \Delta_i \), starting from \( \Delta_1 \), if \( \Gamma_\Delta_i \) is not marked, then mark \( \Gamma_\Delta_i \). Now, while \( \Gamma_\Delta_i \) is already marked, \( \Delta_i = \{ \Delta_i + 1 \text{ if } \Delta_i < |\Gamma| \text{ or } 0 \text{ otherwise} \} \). Because \( |\Delta| = |\Gamma| \), there is always a non-marked index for \( \Delta_i \).
As for the PSO, we utilise a standard implementation where each particle’s position is represented by a vector of real numbers. The numbers are truncated to integer numbers when converting to a valid solution.

3.3 Steepest Ascent Hill Climbing with Random Restart

Steepest ascent hill climbing with random restart (SAHC) is a greedy search-based metaheuristic that, unlike traditional hill climbing (HC), is capable of escaping from areas of local optimum and provides a better chance of locating the best global solution [13]. SAHC has several applications in both industry and academia, such as 3D printing and packing [3] and protein structure prediction [6]. The key distinction between HC and SAHC is how both of them handle situations when candidate solutions fail to produce better fitness value. Such case is the stopping condition for HC, and thus the final solution is equivalent to the first encountered local optimum. On the contrary, the stopping condition for SAHC is the run time, and upon encountering the local optimum, the algorithm randomises the current solution and starts again in hopes of locating a better local or even global solution.

Taking into account the design of TM map, the SAHC was implemented as follows. At the very beginning, the initial map is obtained by randomly assigning terrains to each hexagon. Subsequently, at each iteration, the algorithm generates candidate solutions by choosing a random hexagon and swapping its terrain with a terrain of one of the neighbours. The above procedure is done for each of the neighbours of the chosen hexagon. Thus the range of candidate solutions to evaluate is [2; 6]. Furthermore, the important input parameter to specify is the maximum possible run time of SAHC. The initial test has pointed out that the average time of convergence is 270 s. In the subsequent tests, the fitness value is captured every second.

4 Results

4.1 Results of Particle Swarm Optimisation

First, a hyperparameter optimisation was performed considering three of the main PSO parameters: inertia weight, which is the key feature in balancing exploration and exploitation, determining the contribution of particle’s previous velocity to its current one; cognitive and social scaling, which are also weights used in computing particle’s velocity at current step. The values that were assigned to each of these parameters are 0.01, 0.33, 0.66, 0.99. For each hyperparameter, PSO was executed 30 times using a timeout limit of five minutes as the stopping criterion. Figure 5 presents the mean score and the mean time to find the solution with the best score of each hyperparameter.
Figure 3 presents the comparative performance of the hyperparameters, which were evaluated with respect to the mean score of the proposed minimization function and the mean time to converge to a solution. It should be clarified that since we use timeout as stopping criterion, the horizontal axis in Fig. 5 refers to the elapsed time until the algorithm stopped improving the score. Two hyperparameters can be easily distinguished in Fig. 5: HP1, which results in the worst mean score, and HP63, which has the best mean score. The mean scores for each hyperparameter are detailed in the appendix. Although other hyperparameters result in comparable mean scores, they require higher mean time to find the best solution (Fig. 3).

Another noteworthy and probably more relevant observation concerns the scores of the best map obtained by each hyperparameter in PSO, shown in Fig. 4. In addition to the best mean score, hyperparameter HP3 also found the map (see Appendix) with the best overall score 3.0. The observation of the hyperparameters sorted in ascending order of score (from the best to the worst) provides the insight that some parameter values seem to favour better results like, for example, all eight highest results have *inertia weight* value set to 0.66.

### 4.2 Results of Steepest Ascent Hill Climbing with Random Restart

Figure 5 illustrates how the mean score of the maps found by the SAHC consistently improves through the run time, stabilising after 270 s. The mean score from the SAHC after 30 executions was 24.9 with a standard deviation of 2.05. The best overall score was 21 (see Appendix).
5 Discussion

Some PSO hyperparameters present a mean score which is comparable to the best hyperparameter (HP63), as shown in Fig. 4. Those hyperparameters, however, present a considerably higher mean time for finding the best solution. However, the same does not occur for different media like in video-games which require the on-demand generation of content. The findings are shown in Sect. 4 suggest that local search methods like SAHC are more suitable to generate good-enough configurations in a reasonable time. For example, while PSO using HP63 reached mean score 10 in mean time of approximately 100 s (Fig. 4), SAHC can reach the score of 22 in half of the time (Fig. 5) with a trade-off of resulting in worst scores over a long period of time.

Runtime is usually a critical performance indicator considering in hyperparameter optimisation and algorithm selection. However, in procedural content generation for tabletop games, this indicator becomes less important as the execution of the algorithms occurs in early design stages, prior to the manufacturing. In an offline tabletop game manufacturing pipeline, the relevance of algorithm runtime can be often ignored. The comparison of the results of the

![Fig. 4. Best fitness for each PSO parameter set.](image1)

![Fig. 5. Mean score and time for SAHC.](image2)
PSO and SAHC metaheuristics demonstrates that PSO is consistently more efficient in finding maps with the best scores. In other words, PSO can find maps that violate the smallest number of requirements (Eqs. 1–5). Interestingly, some requirements are often more disregarded than others, as shown in Table 1.

Table 1. Number of times each requirement is violated.

| Map           | Score ($F_{tot}$) | REQ1 | REQ2 | REQ3 | REQ4 |
|---------------|-------------------|------|------|------|------|
| PSO HP3       | 3                 | 0    | 0    | 0    | 3    |
| SAHC          | 21                | 4    | 8    | 3    | 6    |
| Original map  | 9                 | 0    | 3    | 0    | 6    |
| Fire and ice  | 12                | 0    | 3    | 0    | 9    |
| Fjords        | 9                 | 0    | 0    | 0    | 9    |

As shown in Table 1, the number of violated requirements of the best map obtained by SAHC suggest that this metaheuristic is not the most suitable algorithm to be used in early stages of offline manufacturing. Interestingly, the existing maps and the best map obtained by PSO present no violations of requirements REQ1 and REQ3. The reasons for such seem to have root on the nature of the perturb operator of the tested metaheuristics, which randomly select a hexagon and swap by one of its neighbours. More complex operators designed to address REQ4 requirement can contribute to the generation of more balanced maps.

PCG using metaheuristics can answer the demand from the gaming community for new maps in TM that satisfy complex requirements that, if attended, contribute to more balanced and enjoyable games. A user could play with generated maps in online platforms like ‘TM AI’\(^1\). Moreover, the algorithms offer new material that can be commercialised in future expansions. Only one expansion containing two maps has been released to the date.

6 Conclusion

This paper demonstrates how PSO and SAHC metaheuristics can be used for generating new maps for the popular game Terra Mystica. Such maps address the need from a large gaming community eager for new maps. Moreover, the generated maps comply with requirements for map balance, supporting equal initial conditions for players. These features provide the opportunity for commercialising expansions with new material and maps.

One interesting research direction to be addressed is to expand the set of requirements contributing to balance with the use of integer programming. It is also essential to interact with the gaming community as well as the creators of

\(^1\) [https://lodev.org/tmai/](https://lodev.org/tmai/).
Terra Mystica in order to receive play-testing feedback. Moreover, it is necessary to consider more sophisticated and refined evolutionary algorithms for this combinatorial optimisation problem that are able to produce balanced maps according to the expanded set of rules.

Appendix

(See Fig. 6 and Table 2).

Fig. 6. Maps generated by PSO and SAHC (Color figure online)
Table 2. Best and mean score for each PSO hyperparameter after 30 executions.

| Hyperparameter | Cognitive | Social | Inertia | Best score | Mean score | Standard deviation |
|----------------|-----------|--------|---------|------------|------------|-------------------|
| 3              | 0.01      | 0.01   | 0.66    | 3.0        | 10.0       | 3.710             |
| 47             | 0.66      | 0.99   | 0.66    | 3.0        | 10.5       | 3.947             |
| 63             | 0.99      | 0.99   | 0.66    | 4.0        | 9.3        | 2.620             |
| 43             | 0.66      | 0.66   | 0.66    | 5.0        | 12.1       | 4.080             |
| 59             | 0.99      | 0.66   | 0.66    | 6.0        | 11.0       | 3.337             |
| 35             | 0.66      | 0.01   | 0.66    | 6.0        | 20.0       | 5.089             |
| 55             | 0.99      | 0.33   | 0.66    | 8.0        | 13.7       | 3.386             |
| 39             | 0.66      | 0.33   | 0.66    | 8.0        | 14.6       | 3.746             |
| 62             | 0.99      | 0.99   | 0.33    | 9.0        | 14.7       | 3.119             |
| 46             | 0.66      | 0.99   | 0.33    | 9.0        | 18.7       | 4.532             |
| 53             | 0.99      | 0.33   | 0.01    | 9.0        | 22.8       | 5.011             |
| 54             | 0.99      | 0.33   | 0.33    | 10.0       | 18.9       | 3.850             |
| 51             | 0.99      | 0.01   | 0.66    | 10.0       | 19.0       | 5.174             |
| 50             | 0.99      | 0.01   | 0.33    | 10.0       | 23.3       | 5.738             |
| 58             | 0.99      | 0.66   | 0.33    | 11.0       | 17.4       | 4.224             |
| 4              | 0.01      | 0.01   | 0.99    | 11.0       | 21.4       | 4.483             |
| 45             | 0.66      | 0.99   | 0.01    | 11.0       | 21.8       | 4.387             |
| 23             | 0.33      | 0.33   | 0.66    | 12.0       | 18.7       | 4.748             |
| 57             | 0.99      | 0.66   | 0.01    | 12.0       | 19.9       | 3.191             |
| 41             | 0.66      | 0.66   | 0.01    | 12.0       | 20.1       | 3.609             |
| 61             | 0.99      | 0.99   | 0.01    | 13.0       | 17.1       | 2.792             |
| 42             | 0.66      | 0.66   | 0.33    | 13.0       | 19.1       | 3.794             |
| 38             | 0.66      | 0.33   | 0.33    | 13.0       | 20.7       | 3.752             |
| 37             | 0.66      | 0.33   | 0.01    | 13.0       | 20.7       | 4.123             |
| 27             | 0.33      | 0.66   | 0.66    | 13.0       | 24.7       | 5.014             |
| 20             | 0.33      | 0.01   | 0.99    | 13.0       | 28.6       | 4.773             |
| 19             | 0.33      | 0.01   | 0.66    | 14.0       | 22.0       | 3.573             |
| 17             | 0.33      | 0.01   | 0.01    | 14.0       | 22.9       | 4.230             |
| 21             | 0.33      | 0.33   | 0.01    | 14.0       | 25.9       | 4.654             |
| 22             | 0.33      | 0.33   | 0.33    | 16.0       | 23.5       | 4.544             |
| 34             | 0.66      | 0.01   | 0.33    | 16.0       | 23.7       | 4.036             |
| 18             | 0.33      | 0.01   | 0.33    | 16.0       | 24.4       | 4.579             |
| 49             | 0.99      | 0.01   | 0.01    | 16.0       | 24.8       | 4.377             |
| 31             | 0.33      | 0.99   | 0.66    | 18.0       | 27.4       | 3.535             |
| 24             | 0.33      | 0.33   | 0.99    | 18.0       | 29.3       | 3.216             |
| 33             | 0.66      | 0.01   | 0.01    | 19.0       | 24.9       | 3.509             |
| 15             | 0.01      | 0.99   | 0.66    | 19.0       | 27.1       | 3.130             |
| 29             | 0.33      | 0.99   | 0.01    | 19.0       | 31.1       | 2.741             |
| 26             | 0.33      | 0.66   | 0.33    | 22.0       | 30.2       | 3.184             |
| 56             | 0.99      | 0.33   | 0.99    | 22.0       | 30.2       | 3.059             |
| 44             | 0.66      | 0.66   | 0.99    | 22.0       | 30.8       | 2.565             |
| 25             | 0.33      | 0.66   | 0.01    | 22.0       | 32.0       | 2.633             |
| 11             | 0.01      | 0.66   | 0.66    | 23.0       | 30.0       | 2.345             |
| 64             | 0.99      | 0.99   | 0.99    | 23.0       | 30.8       | 2.565             |
| 36             | 0.66      | 0.01   | 0.99    | 24.0       | 30.4       | 3.029             |
| 7              | 0.01      | 0.33   | 0.66    | 24.0       | 30.5       | 2.680             |
| 40             | 0.66      | 0.33   | 0.99    | 24.0       | 30.8       | 2.522             |
| 16             | 0.01      | 0.99   | 0.99    | 24.0       | 31.0       | 2.066             |
| Hyperparameter | Cognitive | Social | Inertia  | Best score | Mean score | Standard deviation |
|----------------|-----------|--------|----------|------------|------------|--------------------|
| 14             | 0.01      | 0.99   | 0.33     | 25.0       | 30.8       | 2.891              |
| 48             | 0.66      | 0.99   | 0.99     | 25.0       | 30.8       | 2.315              |
| 52             | 0.99      | 0.01   | 0.99     | 25.0       | 31.7       | 3.226              |
| 10             | 0.01      | 0.66   | 0.33     | 25.0       | 31.9       | 3.304              |
| 6              | 0.01      | 0.33   | 0.33     | 25.0       | 32.5       | 3.159              |
| 30             | 0.33      | 0.99   | 0.33     | 26.0       | 29.9       | 2.077              |
| 60             | 0.99      | 0.66   | 0.99     | 26.0       | 31.1       | 2.294              |
| 32             | 0.33      | 0.99   | 0.99     | 27.0       | 30.8       | 1.621              |
| 28             | 0.33      | 0.66   | 0.99     | 27.0       | 31.3       | 1.750              |
| 2              | 0.01      | 0.01   | 0.33     | 27.0       | 33.5       | 2.540              |
| 12             | 0.01      | 0.66   | 0.99     | 28.0       | 30.9       | 1.746              |
| 8              | 0.01      | 0.33   | 0.99     | 28.0       | 31.0       | 1.449              |
| 5              | 0.01      | 0.33   | 0.01     | 28.0       | 34.6       | 2.246              |
| 13             | 0.01      | 0.99   | 0.01     | 29.0       | 34.0       | 2.543              |
| 9              | 0.01      | 0.66   | 0.01     | 31.0       | 35.0       | 2.137              |
| 1              | 0.01      | 0.01   | 0.01     | 31.0       | 35.6       | 2.057              |

References

1. Alam, M.N.: Particle swarm optimization: algorithm and its codes in MATLAB, pp. 1–10. ResearchGate (2016)
2. Grichshenko, A., Jonata, L., de Araújo, P., Gimaeva, S., Brown, J.A.: Using Tabu search algorithm for map generation in the Terra Mystica tabletop game (2020)
3. Araújo, L.J., Özcan, E., Atkin, J.A., Baumers, M.: A part complexity measurement method supporting 3D printing. In: NIP and Digital Fabrication Conference, vol. 2016, pp. 329–334. Society for Imaging Science and Technology (2016)
4. Ashlock, D., Lee, C., McGuinness, C.: Search-based procedural generation of maze-like levels. IEEE Trans. Comput. Intell. AI Games 3(3), 260–273 (2011)
5. Barros, G.A., Togelius, J.: Balanced civilization map generation based on open data. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 1482–1489. IEEE (2015)
6. Chira, C., Horvath, D., Dumitrescu, D.: An evolutionary model based on hill-climbing search operators for protein structure prediction. In: Pizzuti, C., Ritchie, M.D., Giacobini, M. (eds.) EvoBIO 2010. LNCS, vol. 6023, pp. 38–49. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12211-8_4
7. Eberhart, R., Kennedy, J.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. Citeseer (1995)
8. Gravina, D., Khalifa, A., Liapis, A., Togelius, J., Yannakakis, G.N.: Procedural content generation through quality diversity. In: 2019 IEEE Conference on Games (CoG), pp. 1–8. IEEE (2019)
9. Khalifa, A., Bontrager, P., Earle, S., Togelius, J.: PCGRL: procedural content generation via reinforcement learning. arXiv preprint arXiv:2001.09212 (2020)
10. Khalifa, A., Fayek, M.: Literature review of procedural content generation in puzzle games (2015)
11. Krause, J., Ruxton, G.D., Ruxton, G.D., Ruxton, I.G., et al.: Living in Groups. Oxford University Press, Oxford (2002)
12. Lara-Cabrera, R., Cotta, C., Fernández-Leiva, A.J.: A procedural balanced map generator with self-adaptive complexity for the real-time strategy game planet wars. In: Esparcia-Alcázar, A. I. (ed.) EvoApplications 2013. LNCS, vol. 7835, pp. 274–283. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37192-9_28

13. Lin, S.W., Ying, K.C., Lu, C.C., Gupta, J.N.: Applying multi-start simulated annealing to schedule a flowline manufacturing cell with sequence dependent family setup times. Int. J. Prod. Econ. 130(2), 246–254 (2011)

14. Mahlmann, T., Togelius, J., Yannakakis, G.N.: Spicing up map generation. In: Di Chio, C., et al. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 224–233. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29178-4_23

15. de Mesentier Silva, F., Lee, S., Togelius, J., Nealen, A.: Ai-based playtesting of contemporary board games. In: Proceedings of the 12th International Conference on the Foundations of Digital Games, p. 13. ACM (2017)

16. Morosan, M., Poli, R.: Automated game balancing in MS PacMan and StarCraft using evolutionary algorithms. In: Squillero, G., Sim, K. (eds.) EvoApplications 2017. LNCS, vol. 10199, pp. 377–392. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55849-3_25

17. Nielsen, J.J., Scirea, M.: Balanced map generation using genetic algorithms in the siphon board-game. In: Ciancarini, P., Mazzara, M., Messina, A., Sillitti, A., Succi, G. (eds.) SEDA 2018. AISC, vol. 925, pp. 221–231. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-14687-0_20

18. Pereira, G., Santos, P.A., Prada, R.: Self-adapting dynamically generated maps for turn-based strategic multiplayer browser games. In: Proceedings of the International Conference on Advances in Computer Entertainment Technology, pp. 353–356. ACM (2009)

19. Togelius, J., Preuss, M., Beume, N., Wessing, S., Hagelbäck, J., Yannakakis, G.N.: Multiobjective exploration of the StarCraft map space. In: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, pp. 265–272. IEEE (2010)

20. Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation. In: Di Chio, C., et al. (eds.) EvoApplications 2010. LNCS, vol. 6024, pp. 141–150. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12239-2_15

21. Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation: a taxonomy and survey. IEEE Trans. Comput. Intell. AI Games 3(3), 172–186 (2011)

22. Uriarte, A., Ontanón, S.: PSMAGE: balanced map generation for StarCraft. In: 2013 IEEE Conference on Computational Intelligence in Games (CIG), pp. 1–8. IEEE (2013)