Final Project Report:

Using genetic algorithms for settlement planning within the Minecraft Framework.

Aaron Fletcher

Summary

***Project Aim*:** This project addresses town settlement planning aspects for a fictional generative design in Minecraft competition (GDMC) entry[1]. Inspiration for this project was derived from real-world examples of arguably the most prosperous settlements (i.e. ones which support the most inhabitants), such as Tokyo or Delhi. These settlements likely had multiple factors in their initial location choice. However, once selected, they all adopted an iterative development approach, where initial building(s) sites are selected, and buildings are iteratively built from a smaller settlement size to a larger one. Given a search space within the Minecraft Framework, this author aims to create a cluster of building locations based on search space features that resemble an iteratively designed settlement. The success criteria are to create a settlement layout that looks organic (i.e. one that clusters around water sources, utilizes flat landscape and is dense) by evaluating sites using fitness functions (i.e. water distance, flatness and house distance).

Approach: Genetic algorithms (GA) are an artificial intelligence methodology that aligns strongly with an iterative evaluation process, where a population of possible candidates (build sites) are assessed for suitability over time. Iterative processes, such as spacecraft antenna[2] and de novo drug discovery[3], have been mimicked by GA.

The project was coded [4] using Python 3.9 and utilized the Minecraft HTTP framework.

In brief, the project's algorithm:

1. Represented the search space through an undirected graph containing all surface tile information, including x,y,z coordinates and material type.
2. Produced a proposed building location through a GA.
3. Updated the undirected graph with the proposed building location
4. Repeated steps 2-3 until a user-specified number of building locations was produced.

The GA has the following phases:

- **Initialization:** Within user-specified search space, a random *n* sized population of location genomes was initialized, with an *x,z* coordinate and a building radius. Building radius, *r*, determined the building location's space, with an *x,z* coordinate determining the central vector.
- **Assessment:** The population was evaluated using three fitness criteria: feature proximity (water and building) and flatness. The fitness functions calculated feature proximity using Manhattan distance, where a feature was searched for within a specified radius, with a penalty applied if the build space encroached on another feature or if no feature was found within the search radius. Flatness was determined via the standard deviation of a building site. An evaluated value closer to zero correlated with a higher scoring population member in all fitness functions.
- **Selection:** Elitism (retention of *n* fittest population member(s)) occurred throughout generational epochs, and the child population was constituted from a mutated roulette selection of the remaining population[5].
- **Reproduction**: Mutations occurred during selection on the x,z location genomes through an inverse gaussian distribution. No crossover occurred.
- **Termination:** Occurred after a user-specified epoch number.

1. Salge, C., Green, M. C., Canaan, R. & Togelius, J. Generative Design in Minecraft (GDMC), Settlement Generation Competition. *ACM International Conference Proceeding Series* (2018) doi:10.1145/3235765.3235814.
2. Hornby, G. S., Globus, A., Linden, D. S. & Lohn, J. D. Automated antenna design with evolutionary algorithms. *Collection of Technical Papers - Space 2006 Conference* **1**, 445–452 (2006).
3. Spiegel, J. O. & Durrant, J. D. AutoGrow4: an open-source genetic algorithm for de novo drug design and lead optimization. *Journal of Cheminformatics 2020 12:1* **12**, 1–16 (2020).
4. PCrispin/AI-Project. https://github.com/PCrispin/AI-Project.
5. Goldberg, D. E. & Deb, K. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. **1**, 69–93 (1991).

Related Work

Mutlu, H. (2020). Optimization of multi-objective land use model with genetic algorithm. *Journal of Design for Resilience in Architecture and Planning*, *1*(1), 15–32. https://doi.org/10.47818/DRArch.2020.v1i1002[6]

Summary of Source: This paper outlines some significant design challenges presented to town planning, such as spatial relationships between building locations, resources from the locale, transportation and sustainability. It eloquently outlines that settlement planning is a multi-objective optimization problem. The paper subsequently develops a GA that optimizes a fixed land topology (containing prohibited areas, land areas which use would never change). The results demonstrate a well-designed settlement resulting from a GA. It concludes that GAs can help solve settlement planning in a reasonable time frame.

Relation to Project: This paper documents that GAs have been explored to aid with town planning. It provides a non-systematic overview of recent papers relating to GA settlement planning.

It offers potential approaches to solving this problem using GAs within the framework, such as considering a fixed layout topology and how mutation and crossover operators might apply to this project.

The paper offers insight and compelling arguments as to why settlements planned by GAs might be superior to human planning. In the real world, planners generate and evaluate multiple alternatives settlement plants and select the best one. Artificial intelligence techniques, such as GAs, facilitate this approach on a grander scale.

It recommends that domain-specific knowledge be applied to search space. While this is somewhat obvious, it provides clear fitness functions options (i.e. land income, distances between land types, transportation).

This paper impacted the project by informing the author about previous attempts to solve this type of problem with this technique. The paper also allowed the author to define what fitness functions would look like within the Minecraft Framework. While the author's implementation differed, this paper provided a good starting point for concepts.

Source Critique: While the paper posits that GAs provide an adequate solution within a reasonable time frame for human planning, this is not adequately evidenced. Furthermore, the paper does not address potential time differences between optimal algorithms (which provide the best solution) and a GA (which *might* provide the best solution).

This author believes the approach outlined in the paper has some flaws, such as using a fixed settlement topology and inadequate outlining of methodology (no clarification for how mutation operators were provided).

Overall this paper is recommended for any practitioner exploring GAs for settlement planning as it explores some key concepts and fitness functions.

6. Mutlu, H. Optimization of multi-objective land use model with genetic algorithm. *Journal of Design for Resilience in Architecture and Planning* **1**, 15–32 (2020).

Illustrative Figure

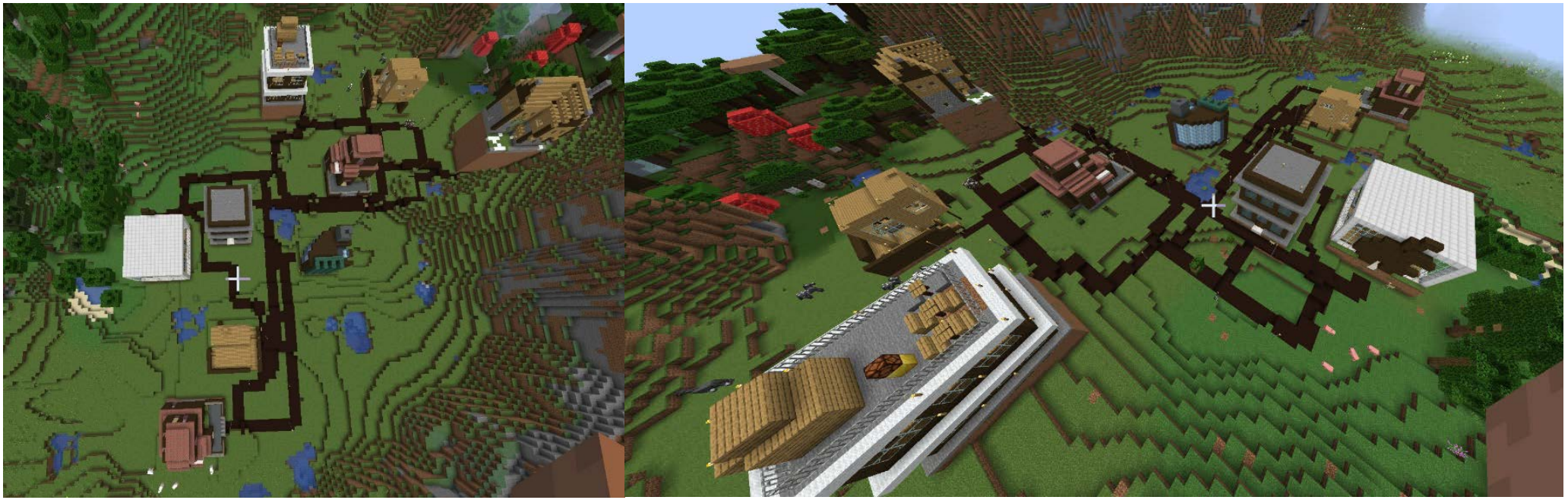


Figure 1 *A bird's eye view of a generated settlement.*

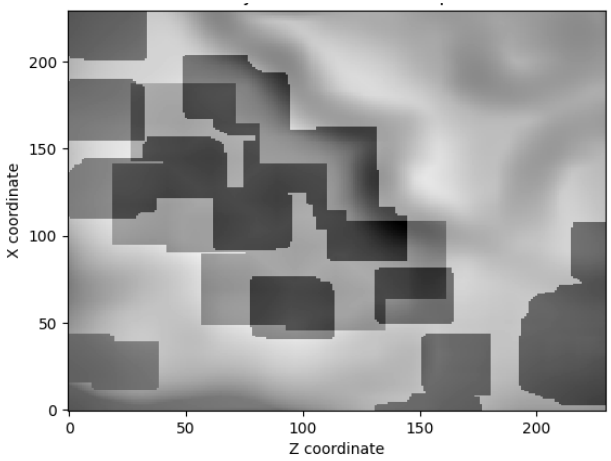
Figure 1 shows a bird's eye view of the completed settlement with the user-specified parameters detailed in *Table 1*. Note that building design was implemented by project partner Philip Crispin, which can drop building locations based on suitability for building type, resulting in nine locations being used rather than the provided ten.

Table 1 *Parameters used to generate settlement.*

Parameter	**Value**
Search Space	(0,0,256,256)
Building Number	10
Max building radius	7
Minimum building radius	13
Epochs	100
Population size	20
Mutation Rate	(1/6)

Data

Plot 1: A 2-dimensional layered fitness map



Plot 2: A 3-dimensional layered fitness map

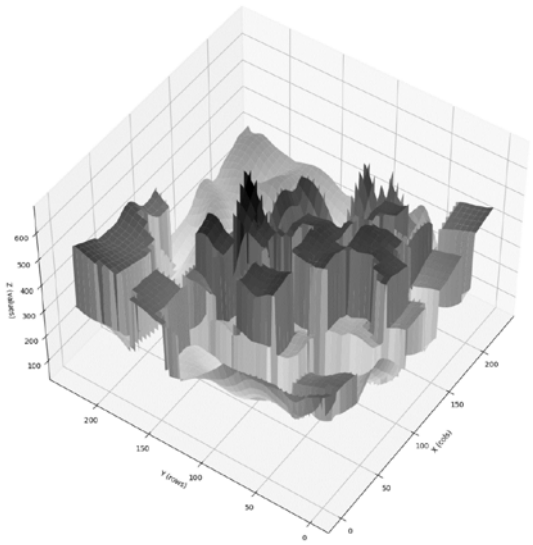


***Figure 2** A 2d combined evaluated fitness map (left), and a 3d combined evaluated fitness map (right)*

***Analysis:** Figure 2* shows two different dimensional views of the same dataset. The GA requested ten building locations based on *Table 1*'s parameters. After producing all building sites, each grid location was evaluated using all fitness functions (i.e. water distance, building distance, and flatness). Subsequently, each grid location was evaluated, with its fitness score being scaled and summed to produce the above plots. It is important to note that a lower scored value represents a fitter proposed building location, denoted as a lighter area in *Plot 1* or a lower value in *Plot 2*. This figure is relevant to the project's aims as higher evaluated building locations will be located in these minima. Initial iterations during the project design process alerted the author of poorly designed fitness functions, such as house distance producing flat peaks. The figures also demonstrate the complexity of solving this problem.

This figure demonstrates:

- A fitness slope was generated, enabling GAs to find better solutions more easily[7].
- Proposed building locations were preferentially clustered and were proximal to water sources.
- Existing building structures modify the combined fitness map.

7. Poli, R. & Vanneschi, L. Fitness-proportional negative slope coefficient as a hardness measure for genetic algorithms. *Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference* 1335–1342 (2007) doi:10.1145/1276958.1277209.

Conclusion

Conclusion: This project was somewhat successful in regards to the success criteria. The building locations generated were subjectively organic, and the GA produced layouts where building distances from each other and water surfaces were minimized.

This author has concluded the following from implementing this project:

1. ***Distance metrics are not required for all search space vectors.***

An early key optimization issue faced was determining the distance of a building location to another space feature (i.e. water or building). Initially, a fitness was calculated from all other vectors, with the smallest distance selected for fitness calculation. Exhaustive searching resulted in prohibitory significant GA run times. An effective implemented solution was to have a search radius for each location and flood fill search to determine if a feature was present, then calculate the distance from the feature. A penalty was applied if a feature was not present in the search radius.

2. ***The initially selected build location is influential.***

The results of the GA implementation suffered from water distance fitness not being weighted to water body size. In the initial first location selection, only flatness and water distance was evaluated (due to no other buildings being present to calculate distance). The lack of other fitness functions means the GA might produce a small building site next to a single water surface vector, resulting in settlements built around it. Further iterations of the GA could modify the water distance function to scale based on the water body size to generate better fitness scores next to larger water bodies.

3. ***GAs provide a rich source for humanistic design are faster than an extensive search space evaluation for optimal results.***

GAs present an excellent technique for a GDMC entry. Evaluation time of the search space using a GA was faster than evaluating the entire search space. While outside the scope of this project's aim, in producing the final figures for this report, it was noted that the evaluation time for the final search space was many times magnitude longer than the entirety of the GA. While this author accepts that GAs *might* produce good but not optimal results, this approach is deemed beneficial, as the aim of the GDMC (which this project idea is based upon) is to produce "good looking settlements" within a predetermined time. Firstly, the GA implementation consistently outputs large numbers of sites in under 5 minutes; secondly, a non-optimal approach reflects better human-designed settlements ("realism"), where settlements are not conceived in total prior to building. While the GDMC competition aims to produce settlements with artificial intelligence techniques, it aims to ensure that settlements do not look like they have been.

4. ***GAs are easily extendable.***

As additional fitness functions are discovered or standardized, extending the code to accommodate this is trivial. For future iterations, sources of additional fitness functions could be:

- Resource abundance (i.e. stone, wood cubes).
- Ability to defend the building site (i.e. preference for building settlements next to cliff faces).
- Future expansion of the settlement (i.e. if town population increases).
- Improving land flatness fitness function through evaluation of block displacement.
- Improving water distance metric to consider water body size.