Procedurally Generating Natural-Looking Villages in Minecraft with Ant Colony Optimization Algorithms

Tobias Deinböck[1,2], Chu-Hsuan Hsueh[1][0000−0001−8888−3116], and Kokolo Ikeda[1]

[1] Japan Advanced Institute of Science and Technology, Nomi, Ishikawa 923-1292, Japan
{hsuehch,kokolo}@jaist.ac.jp
[2] Centrum Wiskunde & Informatica, Science Park 123, 1098 XG Amsterdam, The Netherlands
tobias.deinboeck@cwi.nl

Abstract. This master's thesis shows how ant colony optimization algorithms can be adapted to procedurally generate natural-looking villages in the video game Minecraft by simulating the villagers' lives. Survey results show that the generated villages are perceived as more natural than the default ones, both regarding house placements and path trajectories.

Keywords: Procedural content generation · Multi-agent systems · Ant colony optimization algorithms · Minecraft · Villages · Natural-looking.

1 Introduction

Procedural content generation has been widely deployed to automatically generate digital content with limited or indirect user input [12], especially in video games [8], where villages and cities are important targets. To generate them, this thesis imitates the way villages are formed in the real world [2]. People settle down at some advantageous place and establish paths where they often walk, connecting all houses. New houses are built, and the process repeats. We show how ant colony optimization (ACO) algorithms [3, 4] can be adapted to mimic this behavior and generate natural-looking villages in the video game Minecraft. While village and city generation has been studied before [5, 6, 9, 11], even for Minecraft [1, 7], to our knowledge, we are the first doing it with ACO algorithms.

2 Methodology

We start with placing houses, generate paths to connect them, and repeat the process to grow the village. Houses are stochastically placed in a specified region of a Minecraft world, favoring flat and central areas, and such that every house is

[†] This research was conducted while the first author was affiliated with the Japan Advanced Institute of Science and Technology.

accessible from all sides. Next, villagers inhabiting these houses are represented by multiple ant agents each and simulate life in the village by wandering between houses. After a villager has randomly selected another house, the villager's ants go from one to the other by repeatedly choosing a block in their von Neumann neighborhood and walking to it. Ants are more likely to choose a block that (a) has more pheromones on it, i.e., was traversed more often, (b) is closer to the destination, and (c) is on the same height level if the ant has recently climbed. The last point simulates exhaustion to avoid steep paths.

Once all villagers/ants have found their path, pheromones evaporate, i.e., old paths fade, and new pheromones are secreted on the newly found paths. The shorter and flatter a found path between two houses is, the more pheromones each ant deposits on all blocks it has visited. Selecting houses and wandering between them is repeated multiple times to establish a strong network of pheromone trails, i.e., paths. Repeating all these steps allows the village to naturally grow.

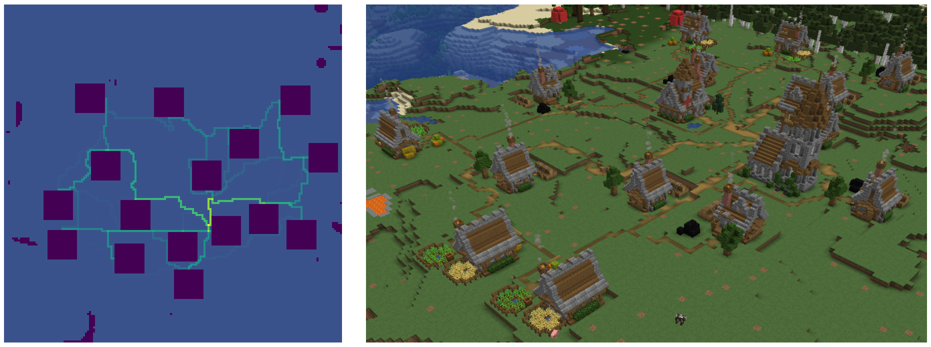
(a) Pheromone heatmap. (b) Village overview.

Fig. 1: A generated village after one growth cycle.

3 Evaluation and Conclusion

In a survey, we asked 45 people to compare the default Minecraft villages and the ones our algorithm generated, and point out which look more natural. The results favor our villages, especially regarding path trajectories, which are seen as more natural and logical. Participants from densely populated East Asian countries criticized our sparse house placements, while ones from Europe preferred it. Furthermore, our algorithm was submitted to the GDMC AI Settlement Generation Challenge [10] 2023, where it placed fourth out of eleven in the adaptability category and was praised for how "the paths mold excellently to the terrain".

In conclusion, we have shown that ACO algorithms can be used in the generation of natural-looking villages in Minecraft. Future work includes a combination with agent-based modeling for a more realistic simulation of the villagers.

References

1. Christiansen, S.S., Scirea, M.: Space segmentation and multiple autonomous agents: a Minecraft settlement generator. In: IEEE CoG. pp. 135–142 (2022)
2. Deinböck, T.: Procedurally Generating Natural-Looking Villages in Minecraft with Ant Colony Optimization Algorithms. Master's thesis, Japan Advanced Institute of Science and Technology (sep 2023)
3. Dorigo, M., Maniezzo, V., Colorni, A.: Ant System: Optimization by a Colony of Cooperating Agents. IEEE Trans. SMC, Part B (Cybern.) **26**(1), 29–41 (1996)
4. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press (jun 2004)
5. Galin, E., et al.: Procedural Generation of Roads. CGF **29**(2), 429–438 (2010)
6. Greuter, S., et al.: Real-time Procedural Generation of 'Pseudo Infinite' Cities. In: Proc. 1st Int. Conf. Comput. Graph. Interact. Tech. Australas. South East Asia. p. 87–ff. GRAPHITE '03 (2003)
7. Iramanesh, A., Kreminski, M.: AgentCraft: An Agent-Based Minecraft Settlement Generator (oct 2021)
8. Liu, J., et al.: Deep learning for procedural content generation. Neural Comput. Appl. **33**(1), 19–37 (jan 2021)
9. Parish, Y.I.H., Müller, P.: Procedural Modeling of Cities. In: Proc. 28th Annu. Conf. Comput. Graph. Interact. Tech. p. 301–308. SIGGRAPH '01 (2001)
10. Salge, C., et al.: Generative Design in Minecraft (GDMC): Settlement Generation Competition. In: Proc. 13th Int. Conf. Found. Digit. Games. FDG '18 (2018)
11. Song, A., Whitehead, J.: TownSim: Agent-based city evolution for naturalistic road network generation. In: Proc. 14th Int. Conf. Found. Digit. Games. FDG '19 (2019)
12. Togelius, J., et al.: What is Procedural Content Generation? Mario on the borderline. In: Proc. 2nd Int. Workshop Proced. Content Gener. Games. PCGames '11 (2011)