IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 May 2024, accepted 17 June 2024, date of publication 19 June 2024, date of current version 27 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3416848

APPLIED RESEARCH

Semi-Automatic Building Layout Generation for Virtual Environments

GABRIEL K. SEPÚLVEDA, NICOLÁS ROMERO, CRISTIAN VIDAL-SILVA, FELIPE BESOAIN, (Member, IEEE), AND NICOLAS A. BARRIGA, (Member, IEEE)
Faculty of Engineering, University of Talca, Talca, Chile

Corresponding author: Nicolas A. Barriga (nbarriga@utalca.cl)

This work was supported in part by the National Agency for Research and Development [Agencia Nacional de Investigación y Desarrollo (ANID)], Sub-Directorate of Applied Research [Subdirección de Investigación Aplicada (SIA)], under Grant ID21|10363.

ABSTRACT Creating floor plan layouts for houses, offices, hotels, or videogame levels is time-consuming. In several applications, the process starts with a bubble diagram or graph derived from a client's requirements, which is then used as a basis for drawing a blueprint to continue the design process. That process requires a tool to preserve consistency between design models and diminish the design time. Hence, we have developed a Unity plugin to graphically represent a building layout using nodes and edges for rooms and doors that also allow defining the size and aspect ratio of spaces. This abstract representation is then automatically converted into a blueprint and optimized for explicit constraints (doors, size, and ratio) and implicit ones (tightness, convexity) by applying stochastic hill-climbing or steepest ascent hill-climbing algorithms. Then, the user can edit that blueprint further. As a final result, the tool generates a navigable 3D representation of the floor plan for the user to explore and go back to edit it if necessary. Results show that both algorithms are computing efficient enough to apply them interactively in the tool while providing functional building layouts. This work can assist established designers or enable building layout tools to be used by non-experts, such as potential homebuyers wishing to communicate a desired layout to an architect or hobbyist game designers working on their first videogame.

INDEX TERMS Mixed-initiative procedural context generation, computer-aided design, videogame-level editor.

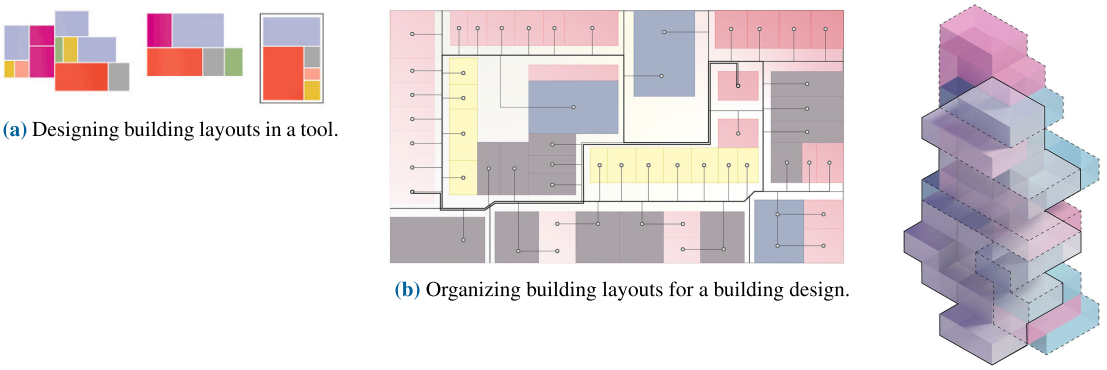
I. INTRODUCTION

Creating floor plan layouts have several applications for the architecture and Videogame industry for the designing and creation of houses, offices, hotels, and videogame levels instances [1].

The design and implementation of layouts are labor-intensive decision-making processes, and external variables must be considered, such as climate, neighborhood, and material [2]. Each phase involves critical decision-making, where seemingly trivial choices can significantly affect the outcome [3]. From the arrangement of spaces in a built structure to the configuration of levels in a virtual environment, creativity, and strategic vision converge with the need to find pragmatic and functional solutions.

In the design process, the influence of external variables that introduce additional complexity levels must be considered [4]. Aspects such as climate emerge as fundamental factors that determine decisions from the selection of materials to spatial distribution. The surrounding environment and the particularities of the terrain are also crucial elements that give shape and direction to the design. In this context, the choice of material constitutes a critical dimension that influences each process stage. In selecting structural components for physical constructions [5] and choosing visual and mechanical elements for video game levels [6], materiality is revealed as a factor that transcends the mere aesthetic to influence functionality, durability, and user experience substantially.

The associate editor coordinating the review of this manuscript and approving it for publication was Andrea Bottino.



(a) Designing building layouts in a tool.

(b) Organizing building layouts for a building design.

(c) Generated building design option.

FIGURE 1. Abstract process of building floor plan design. The image is schematic and referential.

On the other hand, video game-level design tasks do not require external participants to accomplish their main task [7]. Similarly, single-player casual games may need hundreds or thousands of intricately designed levels with escalating difficulties corresponding to player skill advancements [8]. For example, the automatic floor plan creation usually starts with a bubble diagram or data and graph derived from clients' or owners' requirements [1], [9] to draw a blueprint and continue the design process. In the architectural design process, the "floor-plan design" stage is between the "schematic design" and the "design development" phases [10]. Generative layout tools can enhance established digital design phases [9]. Automated approaches allow testing ideas and iteratively generating design options that would be challenging to achieve with manual workflows. In that context, Figure 1a depicts the process of designing small pieces for a building layout, Figure 1b shows how to organize building layout pieces for a building design, and Figure 1c presents a generated building design option.

Regarding videogame-level design, handmade and procedural methods are two distinct approaches for the level design, each offering unique advantages in creating engaging and diverse game environments. Handmade level design involves the manual, artistic creation of game levels by skilled designers [11].

On the other hand, the procedural-level design relies on algorithms and code to generate game levels automatically, offering the advantage of scalability and replayability, as levels can be created on the fly, adapting to player preferences and providing a different experience each time [12]. In recent research on procedural-content generation, genetic algorithms have been applied to evolve cellular automata rules for creating playable mazes in maze-running games [13]. In the same context, reinforcement learning can be applied to train level-designing agents as an approach to procedural content generation in games [14]. Finally, a mixed-initiative procedural content generator plugin for the Unity3D game engine, the Biome Generation Tool (BGT) [15], uses an Evolutionary Algorithm (EA) to succeed in assisting the development of biomes, generating products of acceptable quality while reducing the designer's workload.

In summary, creating layouts is a process that unifies technical skills, artistic intuition, and the ability to adapt to external variables. Although demanding and prolonged, this process allows the generation of virtual spaces that configure different digital products. Research has shown that different approaches, such as fully generated content, mixed initiatives, and automatization, contribute positively to this main area.

A. PROBLEM STATEMENT, GOAL AND CONTRIBUTIONS

Given the current usefulness and experiences of using procedural content generation in building layout generation and existing non-exhaustive optimization algorithms, the main goal of this article is to answer the following research questions.

RQ1 [Applying non-exhaustive algorithms for building layout generation] Can we apply non-exhaustive algorithms for building layout generation? Our experiments look to test and validate the effectiveness of applying non-exhaustive algorithms for building layout generation.

RQ2 [Efficient-enough of non-exhaustive algorithms] Can non-exhaustive algorithms for building layout generation be efficient? Assuming the effectiveness of applying non-exhaustive algorithms, we want to verify their efficiency in performing those tasks.

There are several tools for bridging the gap between professionals from various fields and their clients. We can also find tools of this kind in other areas [16] such as 3D modeling [17] for the digitization and preservation of spaces, or animation for interactive software [18] in other fields. However, an automated tool for integrating architectural experience and knowledge of different disciplines, preserving the consistency between design models during the design process, is highly needed. Two main approaches have been used: generative machine learning techniques [19], [20], and

TABLE 1. Non-exhaustive list and description of optimization algorithms considered.

Algorithm	Description
Hill-climbing	An algorithm that starts with an initial solution and then iteratively makes minor changes based on a neighborhood function to evaluate and improve the current solution quality
Simulated annealing	An algorithm that attempts to avoid local optima in a problem by, in each state, allowing downhill moves with a small probability
Tabu search	An algorithm that attempts to avoid local optima in a problem by, in each state, allowing downhill moves with a small probability
Metropolis Algorithm	An iterative optimization algorithm that stochastically explores the solution space by accepting or rejecting new solutions based on their fitness and a temperature
Evolutionary Algorithm	An optimization algorithm inspired by the process of natural selection, where a population of potential solutions evolves over successive generations through crossover, mutation, and selection

a combination of domain knowledge and experience with optimization methods [21].

The main contribution of this article is an optimization algorithm, using a custom fitness function, integrated into a tool as a plugin of Unity 3D game engine, a videogame development tool, using traditional architectural design items and artificial intelligence assistance. We also present experiments showcasing its performance.

Our solution is part of the tool code named Level Building Sidekick that applies optimization algorithms over a graph of rooms, in which each node has size constraints without a contour to define the building shape. The objective of our tool is a configuration that respects both the size constraints and the adjacency of the rooms represented by the graph edges.

This article is structured as follows: Section II details the main concepts and tools regarding the architecture and videogame-level design. Section III describes our tool proposal's main functions and modules. Section IV details the primary relevant results of applying our solution. Section VI gives this research's main conclusions and future work.

II. BACKGROUND

Architectural floor plans (blueprints, spatial layouts, and interior spatial arrangements) refer to the bi-dimensional spatial arrangement to choose internal building plans [22]. Typically, designers organize a space (a room, a building, a whole city) by capturing the key spatial characteristics in an initial sketch to produce spatial arrangements, floor plans, and technical drawings.

A. MIXED-INITIATIVE

Designers and computer scientists are collaborating to find fresh approaches to automating the generation of spatial solutions. The two main reasons for that collaboration are [22]: firstly, the growth of computing power, accessibility of training data sets, and availability of machine learning research permit increasing their applications and reliability [23], and secondly, the emerging application by designers of machine learning research results in creatively designing structures, cities, and floor plans, among others.

Procedural content generation (PCG) is the programmatic creation of game content utilizing a pseudo-random or random procedure that produces an unpredictably wide range of potential gameplay locations. Textures, objects, and storylines have been created using procedural content generation (PCG) algorithms [24].

Mixed-initiative interaction is when a human user and a machine take the initiative to help solve a problem [25]. In mixed-initiative PCG systems, human users and computing systems ''take the initiative''; human designers and software can co-create content [26]. Tools can range a wide spectrum, from almost automatic with very sparse human input to mostly human-authored content with minor tweaks from the AI. Several of these tools have been built for videogame development [15], [27], [28], [29].

Different optimization algorithms exist in AI to find the best solution for a given problem, like hill-climbing, simulated annealing [30], and tabu search. Table 1 describes those algorithms.

Even though simulated annealing and tabu search can usually find better solutions in search spaces with local optima, hill-climbing usually converges faster [31], [32]. We will apply hill-climbing in this interactive tool.

B. UNITY GAME ENGINE

Video game engines play a fundamental role in the development of interactive software, and among them, Unity has emerged as a leading choice in the industry [33]. Unity is a multiplatform engine, enabling developers to create

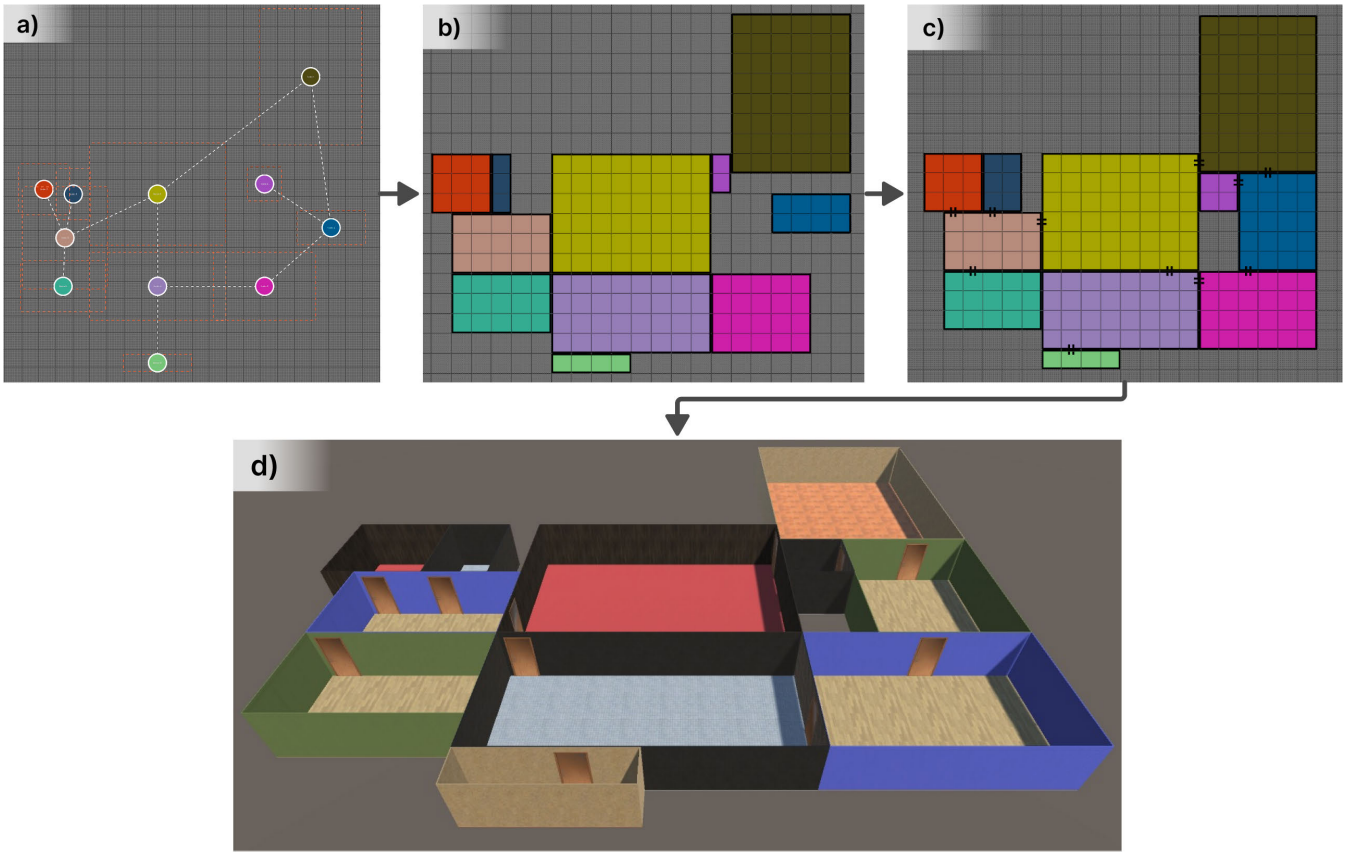


FIGURE 2. Workflow: a) The user places nodes in a graph, representing rooms, annotated with the desired width and height. Then, connections to adjacent rooms are made. b) The system uses a deterministic heuristic function to construct an initial layout from the user information. c) The system optimizes this layout to maximize the fitness function. d) The layout is exported to a navigable 3D representation.

interactive products for computers, consoles, mobile devices, and more. This approach significantly reduces the time and effort required compared to developing specifically for each technology [34]. Furthermore, it allows solutions developed on this engine to be seamlessly transferred across various platforms. The Unity game engine is versatile and robust, attracting teams from diverse areas such as animation [35] and simulation [36], despite their differing primary objectives from game development. Unity is one of the most prevalent engines in the video game industry [37]. Unity's philosophy revolves around modularity [33], facilitating external developers to seamlessly integrate new tools into the engine [38]. This approach also fosters a vibrant community around the engine [27]. Developing solutions using technologies like Unity Engine enables us to leverage its advantages and capabilities, seamlessly translating them into our own solutions.

III. SOLUTION PROPOSAL

As a module of the Level Building Sidekick[1] (LBS) system, we propose a mixed-initiative interaction tool to apply optimization algorithm solutions to produce spatial arrangements for floor plans automatically. The module is written entirely in C# and integrated into the Unity plugin. This tool is aimed at assisting videogame level designers. We have also used it for generating museum gallery layouts in the Maule Virtual Museum [39] and envision it could be used for drafting house layouts by customers, to communicate their ideas to architectural firms in a better way.

The workflow of our solution, shown in Figure 2, starts representing interconnected nodes in graphs, where each node abstractly represents the basic information of a room within the overall design scheme. This information includes the initial position of the room, width, height, and connections with other rooms (arcs in the graph).

This initial representation of nodes and connections is transformed using the basic information described previously into a representation of zones or sectors based on a grid or tile system. Each zone is differentiated from the others with a unique color, corresponding to The different rooms in this step. The sectors only have an initial layout.

Below are the structural rooms after applying the Hill-Climbing algorithm with its respective optimizers, seeking to have the primary conditions described in the graph of nodes, such as the correct distribution, that all the rooms are connected by at least one door panel, and that these respect

[1] Available under a CC BY-NC-SA license at https://github.com/ISILab-Utalca/LevelBuildingSidekick along with the implementation described in this article.

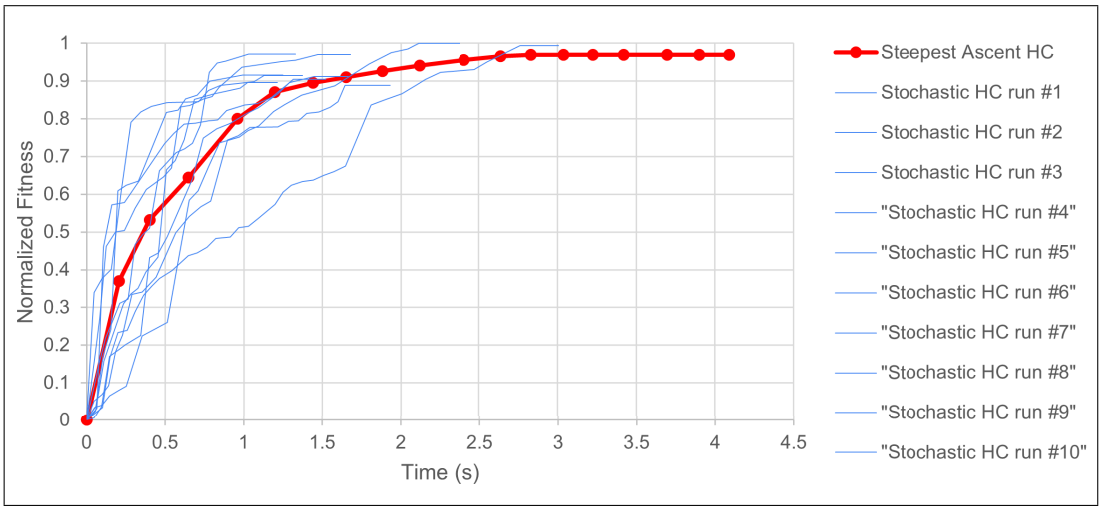


FIGURE 3. In red is the progression of the steepest ascent approach, and in blue is the progression of each run of the stochastic approach. The y-axis is normalized relative to the maximum and minimum fitness results.

the shapes initially described based on their height and width.

Finally, 3D generation methods are applied that use the information from the plot map to generate a 3D inhabitant usable in an experience of this type.

IV. EXPERIMENTS AND RESULTS

Considering Figure 2, the nodes' initial position, width, and height in the graph are extracted to generate a new 2D representation based on a tile map. Tiles are created according to the width and height values of each node. The process begins with the node with the most connections to the other rooms and continues with its neighbors, generating a queue ordered by the number of connections for each node. The process selects the first node from this queue, transforms it into a group of tiles, and calculates the corresponding neighbors. This process aims to prioritize the placement of nodes with more connections, assuming those nodes will be more relevant in the subsequent steps. When there are no more neighbor nodes to transform, the process is considered complete unless the representation contains two or more disconnected graphs. In that case, the system searches for the node with the most connections among all the nodes that have not been transformed yet and continues with the previously described process.

Hill-climbing requires an initial base state corresponding to the tile map. Additionally, it utilizes a fitness function to determine the map's value, a neighborhood function responsible for generating map variations, and a selection function that determines how a new solution is selected.

The optimization process is iterative. The initial tile representation is evaluated using different features, each with a specific orientation and influence weight. These weights determine how much each feature affects the final value or evaluation of the representation. After this stage, a neighborhood of maps or "children" of the original map that exhibit variations is generated. Two modifications generate these variations: wall movements and complete room shifts. The first process involves selecting a wall (row or column of tiles) among all possible walls in the representation and expanding or contracting it by one unit of distance. The second process involves shifting a room's tiles by one unit of distance. The objective of these processes is to generate versions that are similar to the original map without deviating too far from it.

Once the neighborhood has been generated, all the maps are evaluated similarly to the first element. Subsequently, the elements are compared using different selection methods. Steepest ascent hill-climbing selects the highest value neighbor. In contrast, stochastic hill-climbing evaluates neighbors in random order and selects the first one that is better than the current one. After selection, the selected representation is used for the next iteration and as a starting point for generating new variations or a new neighborhood.

$$Fitness = 0.4\Delta + 0.15\Sigma + 0.35\Phi + 0.1\Lambda \quad (1)$$

The fitness function is shown in equation 1. The weights for the features were determined empirically. The features are:

Adjacency (Δ): This function searches for the nearest tiles among the rooms marked as connected in the graph. It evaluates the tile distance between them, assigning a high value when they are close and gradually decreasing as they move apart. The distance is calculated by summing the inverse of the distance between the rooms and dividing it by the number of connections.

Area (Σ): This function compares how different the width and height values of the current tile map are compared to the initial values provided in the previous node graph representation. If the current size is less than half or over 1.5 times the desired value, it assigns a fitness of 0.

TABLE 2. Problem size, Minimum (Min), Maximum (Max), Average($\bar{f}$), and standard deviation (σ) fitnesses f.

	# Rooms	# Connectivity	Min Fitness	Max Fitness	$\bar{f}$ Fitness	$\sigma(f)$ Fitness
Layout 1	11	11	0.469	0.520	0.518	0.001
Layout 2	8	9	0.470	0.537	0.532	0.002
Layout 3	12	12	0.439	0.515	0.509	0.005
Layout 4	11	13	0.412	0.519	0.516	0.003
Layout 5	12	13	0.379	0.500	0.491	0.010
Layout 6	25	28	0.385	0.487	0.477	0.006
Layout 7	10	10	0.424	0.524	0.522	0.001
Layout 8	11	11	0.415	0.519	0.515	0.005
Layout 9	12	15	0.416	0.516	0.513	0.001
Layout 10	10	9	0.433	0.522	0.520	0.001

Empty space (Φ): This function compares the number of tiles in a room with the corresponding rectangle area, taking into account its furthest height and width positions. It evaluates configurations that minimize the number of intermediate empty spaces.

Room Cut (Λ): For each room, this function selects the first tile, finds its neighbors, and continues summing the number of unique tiles that can be found. When there are no more neighbors, it stops and compares this count with the total number of tiles in the room. If the number of tiles per room equals that room's total number, the room is 100% connected without any divided spaces. The fewer divided rooms in the layout, the lower the assigned value.

Each of these features contributes to determining the overall value of the tile map in the optimization process. For the layout to be considered ''legal'' meaning all rooms are connected as they should, and the room sizes are not too far from the desired values, the fitness value must be at least 0.5.

We developed 10 test layouts of varying room counts and sizes to run the experiments. Each layout was then created in the tool in the form of a graph organized by hand using as a guide the approximate position and distribution that the designer desired for the schema. The graph layout is then converted to a basic schema layout from where the optimization process begins. The optimization process is performed with hill-climbing to compare the results of the steepest ascent vs. a stochastic approach. For the steepest ascent approach, we executed 1 run per layout as it is a deterministic process, while for the stochastic approach, we made 10 runs for each layout.

A sample run can be seen in Figure 3. We can see that, on average, the fitness obtained by the stochastic method is around the same as the steepest ascent approach, but in most runs, the maximum fitness reached was with the stochastic hillclimbing. Moreover, the steepest ascent hill-climbing takes over twice as long as the stochastic approach. Although the stochastic approach is faster and can reach higher fitness values, contrary to the steepest ascent, it is not stable in its evolution. It has the chance of getting trapped in lower local maxima. Different outputs of the stochastic hill-climbing for a single initial layout can be seen in Figure 4.

Table 2 shows the starting and final evaluations for each test layout: the minimum fitness, which is the fitness before the optimization process, and the maximum fitness reached among all optimization runs considering both the steepest ascent and stochastic approach, the average and standard deviation fitnesses.

Figure 5 depicts execution examples of the steepest ascent and stochastic approach progression regarding fitness and required time. We can appreciate that the stochastic approach executions reach high fitness results faster than the steepest ascent. The last figure also shows that the steepest ascent does not reach the top fitness of some stochastic execution.

V. DISCUSSION

Stochastic hill-climbing achieves similar fitness results, standing out by reaching maximum values in most executions. From a practical standpoint, the stochastic approach emerges as a more efficient option in terms of runtime, capable of attaining high fitness values. However, it presents a higher risk of getting trapped in lower local maxima.

The generalization of these results to other contexts involving automatic spatial design generation will depend on the specific characteristics of the domain and design approach. These findings may influence future research in spatial design generation, motivating the development of more specialized tools that leverage the efficiency of the stochastic approach.

Considering the obtained results, both research questions, *RQ1* and *RQ2*, we can explicitly answer that we can apply non-exhaustive optimization algorithm solutions for building layout generation without efficiency issues. In this context, we have presented a building layout generation tool that aids users in defining an abstract representation of a desired floor plan, translates it to a 2D tile-based layout, optimizes it, and finally exports a navigable 3D representation.

A similar tool has been proposed for generating residential building layouts [21]. The authors reported a runtime for the optimization process of 35 seconds for a layout comprised of 14 rooms and 16 connections. For comparison, our layout #6 has 25 rooms and 28 connections, and layout #9 has 12 rooms and 15 connections. All stochastic runs in both layouts finish in less than 30 seconds. In 15 seconds, most of them are

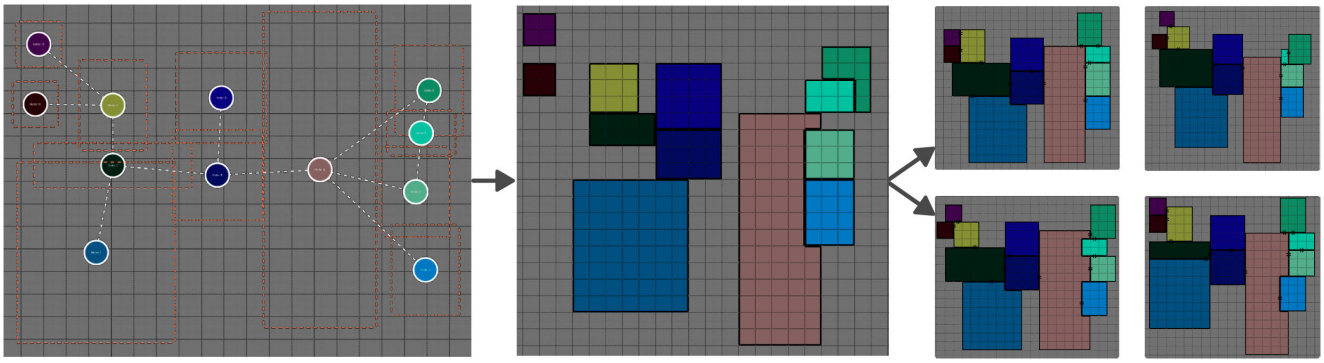


FIGURE 4. Stochastic hill-climbing, starting from the same initial graph and layout, produces a different configuration in each run. Four different runs are shown on the right side of this figure.

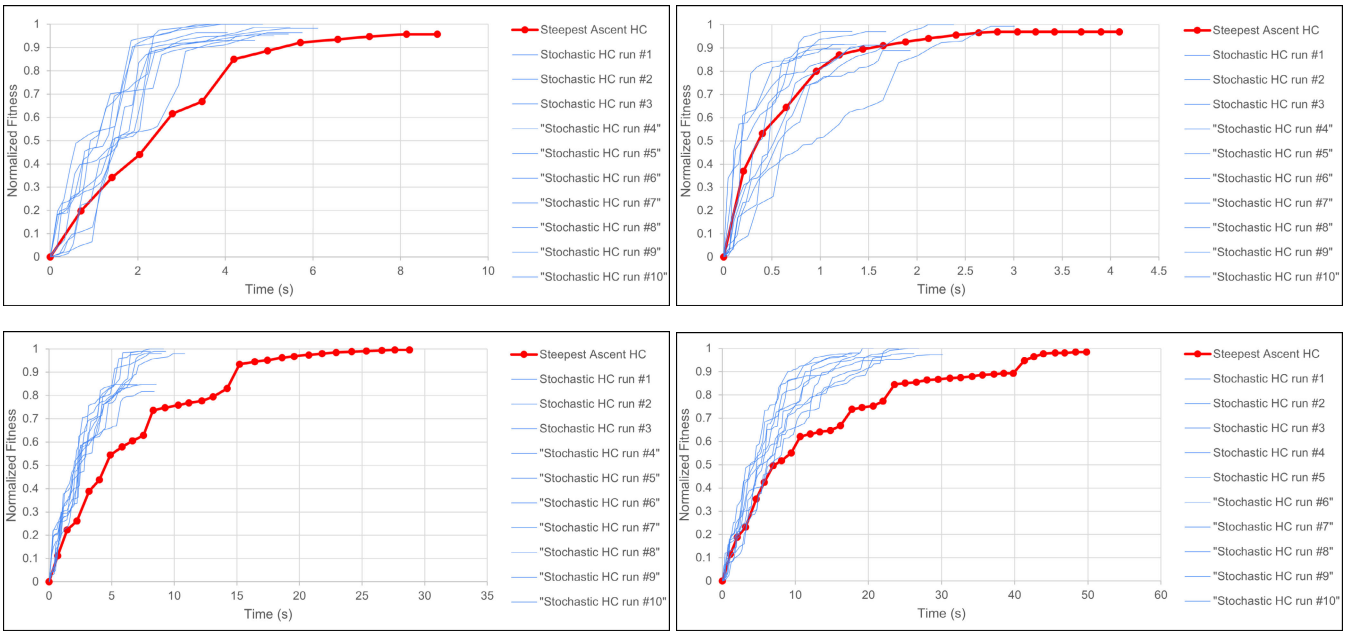


FIGURE 5. Steepest ascent (red) and stochastic (blues) approaches execution examples. The y-axis is normalized relative to the maximum and minimum fitness results.

within 25% of the max fitness in layout #6 and within 10% in layout #9. Beware that this is not a direct comparison on the same hardware, but it gives an idea of the overall performance.

A. LIMITATIONS

Some limitations of the present work are as follows:

- The evaluation of only 10 test layouts in the development process is a limited number that may not have encompassed all possible details in a map, leaving out cases that could influence the results. This quantity may be insufficient to gain a comprehensive perspective on the effectiveness of the employed algorithms. Furthermore, the restriction of generating maps in a tile format constrains the flexibility of the results, hindering the representation of more complex forms that could be relevant in practical applications. It would be ideal to extend the analysis to other techniques not considered in the study to enrich the understanding of the relative effectiveness of different floor plan generation approaches.
- The nature of the results obtained at the laboratory level. Implementing these results in real-world environments and gathering direct qualitative feedback from end-users, such as developers and designers, would be highly beneficial. The absence of this direct interaction may limit the applicability of the results to practical contexts. Concerning internal validity, the measurement of execution times and fitness, as well as the evaluation of systems with diverse architectures, were relevant considerations. However, it is necessary to examine external factors that could influence future research to understand the results' generalization fully.

VI. CONCLUSION

We have presented a semi-automatic building layout generation tool that can be used for designing videogame levels and other virtual spaces. It leverages simple hill climbing and a

custom fitness function to reach high-quality solutions fast enough to be suitable for interactive use.

A. FUTURE WORK

We are working on three main avenues of improvement: the current neighbor creation process doesn't allow the algorithm to explore the search space fully; the fitness function doesn't take into account all of the features a user cares about; and the optimization algorithm itself sometimes gets trapped in local optima.

- When it comes to the algorithm, while the stochastic approach is faster and can lead to better results, it is unreliable by itself. Nevertheless, different techniques cover these shortcomings, such as random restarts [40]. Considering the time and fitness constraints of the problems we wish to solve, we could either rely on the steepest ascent with more stable results and higher execution times, search for stochastic approaches with an aggregate that helps to ensure high-fitness results, or consider another kind of algorithms altogether, such as simulated annealing or tabu search, among others.
- It is worth noting that the evaluation and neighbor search process can be customized depending on the context of the use of the buildings. The context of use can vary greatly from architectural projects to the creation of structures for interactive software. Still, we can be even more specific, discerning between houses, museums, or attraction spots such as parks and mazes. This opens the possibility to study which options are better for each use case to offer the final user preset configurations suited for the specific context they will be working on.
- Optimizing algorithms to avoid getting trapped in local maxima is a challenge to improve performance. When faced with multiple local optima, the task is to maximize the ability of the algorithm to converge toward the global maximum. Exploration strategies beyond local maxima will be studied; for example, genetic algorithms are presented as promising approaches to enhancing the ability of algorithms to reach more optimal solutions in complex contexts.

ACKNOWLEDGMENT

The authors want to thank their industry partners, Abstract Digital, Cienart Studios, Altair Films, and Frisson Games, for their support throughout this project.

REFERENCES

[1] S. Liu, L. Chaoran, L. Yue, M. Heng, H. Xiao, S. Yiming, W. Licong, C. Ze, G. Xianghao, L. Hengtong, D. Yu, and T. Qinting, "Automatic generation of tower defense levels using PCG," in *Proc. 14th Int. Conf. Found. Digit. Games*, Aug. 2019, pp. 1–9.

[2] D. Ma, X. Li, B. Lin, Y. Zhu, and S. Yue, "A dynamic intelligent building retrofit decision-making model in response to climate change," *Energy Buildings*, vol. 284, Apr. 2023, Art. no. 112832. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778823000622

[3] S. Moghtadernejad, L. E. Chouinard, and M. S. Mirza, "Design strategies using multi-criteria decision-making tools to enhance the performance of building façades," *J. Building Eng.*, vol. 30, Jul. 2020, Art. no. 101274. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S235271021931959X

[4] G. H. Vidal and J. R. C. Hernández, "Complexity in manufacturing systems: A literature review," *Prod. Eng.*, vol. 15, nos. 3–4, pp. 321–333, Jun. 2021.

[5] L. Zhang and C. Kim, "Chromatics in urban landscapes: Integrating interactive genetic algorithms for sustainable color design in marine cities," *Appl. Sci.*, vol. 13, no. 18, p. 10306, Sep. 2023.

[6] H. Ahmadi, S. Zadtootaghaj, F. Pakdaman, M. R. Hashemi, and S. Shirmohammadi, "A skill-based visual attention model for cloud gaming," *IEEE Access*, vol. 9, pp. 12332–12347, 2021.

[7] A. Khalifa, M. C. Green, G. Barros, and J. Togelius, "Intentional computational level design," in *Proc. Genetic Evol. Comput. Conf.* New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 796–803, doi: 10.1145/3321707.3321849.

[8] J. Zhu and S. Ontañón, "Experience management in multi-player games," in *Proc. IEEE Conf. Games (CoG)*, Aug. 2019, pp. 1–6.

[9] R. E. Weber, C. Mueller, and C. Reinhart, "Automated floorplan generation in architectural design: A review of methods and applications," *Autom. Construct.*, vol. 140, Aug. 2022, Art. no. 104385. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580522002588

[10] D. Lobos and D. Donath, "The problem of space layout in architecture: A survey and reflections," *Arquitetura Revista*, vol. 6, no. 2, pp. 136–161, Dec. 2010.

[11] T. Short and T. Adams, *Procedural Generation in Game Design*. Boca Raton, FL, USA: CRC Press, 2017.

[12] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games*, 1st ed. Cham, Switzerland: Springer, 2016.

[13] C. Adams and S. Louis, "Procedural maze level generation with evolutionary cellular automata," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2017, pp. 1–8.

[14] A. Khalifa, P. Bontrager, S. Earle, and J. Togelius, "PCGRL: Procedural content generation via reinforcement learning," in *Proc. AAAI Conf. Artif. Intell. Interact. Digital Entertainment*, 2020, vol. 16, no. 1, pp. 95–101.

[15] G. K. Sepúlveda, F. Besoain, S. von Brand, and N. A. Barriga, "Biome generation tool: A mixed-initiative software for the procedural generation of biomes," *Appl. Sci.*, vol. 13, no. 14, p. 8070, Jul. 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/14/8070

[16] N. Grandon, H. Peldoza, and F. Besoain, "Automatizing the generation of a virtual tour of an architecture model through an information system," in *Proc. IEEE Biennial Congr. Argentina (ARGENCON)*, Argentina, Jun. 2018, pp. 1–7.

[17] E. Turner, P. Cheng, and A. Zakhor, "Fast, automated, scalable generation of textured 3D models of indoor environments," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 3, pp. 409–421, Apr. 2015.

[18] G. Lai, F. F. Leymarie, and W. Latham, "On mixed-initiative content creation for video games," *IEEE Trans. Games*, vol. 14, no. 4, pp. 543–557, Dec. 2022.

[19] R. Hu, Z. Huang, Y. Tang, O. Van Kaick, H. Zhang, and H. Huang, "Graph2Plan: Learning floorplan generation from layout graphs," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 1–118, Aug. 2020.

[20] W. Wu, X.-M. Fu, R. Tang, Y. Wang, Y.-H. Qi, and L. Liu, "Data-driven interior plan generation for residential buildings," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–12, Dec. 2019.

[21] P. Merrell, E. Schkufza, and V. Koltun, "Computer-generated residential building layouts," in *Proc. ACM SIGGRAPH Asia Papers SIGGRAPH ASIA*, 2010, pp. 1–12.

[22] S. Carta, "Self-organizing floor plans," *Harvard Data Sci. Rev.*, vol. 3, pp. 1–39, Jul. 2021.

[23] A. Dhondse, S. Kulkarni, K. Khadilkar, I. Kane, S. Chavan, and R. Barhate, "Generative adversarial networks as an advancement in 2D to 3D reconstruction techniques," in *Data Management, Analytics and Innovation: Proceedings of ICDMAI 2019*, vol. 2. Cham, Switzerland: Springer, 2020, pp. 343–364.

[24] N. A. Barriga, "A short introduction to procedural content generation algorithms for videogames," *Int. J. Artif. Intell. Tools*, vol. 28, no. 2, Mar. 2019, Art. no. 1930001.

[25] A. Liapis, G. Smith, and N. Shaker, "Mixed-initiative content creation," in *Procedural Content Generation in Games*. Cham, Switzerland: Springer, 2016, pp. 195–214.

[26] A. Alvarez, J. Font, and J. Togelius, "Story designer: Towards a mixed-initiative tool to create narrative structures," in *Proc. 17th Int. Conf. Found. Digit. Games*. New York, NY, USA: Association for Computing Machinery, Sep. 2022, pp. 1–9, doi: 10.1145/3555858.3555929.

[27] C. Aliaga, C. Vidal, G. K. Sepulveda, N. Romero, F. González, and N. A. Barriga, "Level building sidekick: An AI-assisted level editor package for unity," in *Proc. 19th AAAI Conf. Artif. Intell. Interact. Digit. Entertainment (AIIDE)*), 2023, pp. 392–399.
[28] D. Gravina, A. Khalifa, A. Liapis, J. Togelius, and G. N. Yannakakis, "Procedural content generation through quality diversity," in *Proc. IEEE Conf. Games (CoG)*, Aug. 2019, pp. 1–8.
[29] A. Liapis, G. N. Yannakakis, and J. Togelius, "Sentient sketchbook: Computer-assisted game level authoring," in *Proc. 8th Int.Conf. Found. Digit. Games*, 2013, pp. 213–220.
[30] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, May 1983.
[31] V. Jately, B. Azzopardi, J. Joshi, V. B. Venkateswaran, A. Sharma, and S. Arora, "Experimental analysis of hill-climbing MPPT algorithms under low irradiance levels," *Renew. Sustain. Energy Rev.*, vol. 150, Oct. 2021, Art. no. 111467.
[32] K. Musiał, "Implementing local search algorithms to multi-series production task," in *Proc. Int. Conf. Intell. Syst. Prod. Eng. Maintenance*. Cham, Switzerland: Springer, 2023, pp. 764–775.
[33] A. M. Barczak and H. Woźniak, "Comparative study on game engines," in *Studia Informatica. Systems and Information Technology. Systemy I Technologie Informacyjne*, nos. 1–2. Wrocław, Poland: Springer, 2019.
[34] S. Blackman, *Beginning 3D Game Development With Unity: All-in-One, Multi-Platform Game Development*. New York, NY, USA: Apress, 2011.
[35] L. Ecole, W.-T. Kim, and J.-S. Yoon, "Unity: A powerful tool for 3D computer animation production," *J. Korea Comput. Graph. Soc.*, vol. 29, no. 3, pp. 45–57, Jul. 2023.
[36] A. Hussein, F. García, and C. Olaverri-Monreal, "ROS and unity based framework for intelligent vehicles control and simulation," in *Proc. IEEE Int. Conf. Veh. Electron. Saf. (ICVES)*, Sep. 2018, pp. 1–6.
[37] M. Toftedahl. (Sep. 2019). *Which Are the Most Commonly Used Game Engines*. [Online]. Available: https://www.gamedeveloper.com/production/which-are-the-most-commonly-used-game-engines-#close-modal
[38] I. Carmosino, F. Bellotti, R. Berta, A. De Gloria, and N. Secco, "A game engine plug-in for efficient development of investigation mechanics in serious games," *Entertainment Comput.*, vol. 19, pp. 1–11, Mar. 2017.
[39] F. Besoain, L. Jego, and I. Gallardo, "Developing a virtual museum: Experience from the design and creation process," *Information*, vol. 12, no. 6, p. 244, Jun. 2021. [Online]. Available: https://www.mdpi.com/2078-2489/12/6/244
[40] S. H. Jacobson and E. Yücesan, "Analyzing the performance of generalized Hill climbing algorithms," *J. Heuristics*, vol. 10, no. 4, pp. 387–405, Jul. 2004.

GABRIEL K. SEPÚLVEDA received the degree in videogame development and virtual reality engineering from the University of Talca, where he is currently pursuing the Ph.D. degree in engineering systems. He is a Programmer and a Developer with the ISI Laboratory Research Team. His research interests include AI algorithms and the applications to real-life problems.

NICOLÁS ROMERO received the degree in game development and virtual reality engineering from the University of Talca, in 2022, where he is currently pursuing the Postgraduate degree in engineering systems, solidifying his commitment to the forefront of artificial intelligence and its practical application in innovative solutions. He is a Developer with the ISI Laboratory, focuses on a tool utilizing artificial intelligence algorithms.

CRISTIAN VIDAL-SILVA received the B.S. degree in information engineering from the Faculty of Engineering, Catholic University of Maule, Talca, Chile, in 2003, the first M.S. degree in computer science from the University of Concepcion, Concepción, Chile, in 2007, the second M.S. degree in computer science from Michigan State University, East-Lansing, MI, USA, and the Ph.D. degree in software engineering from the University of Seville, Seville, Spain. He was a Fulbright Scholar with Michigan State University. He is a Professor with the Faculty of Engineering, Videogame Development and Virtual Reality Engineering School, University of Talca, Chile. He is the principal author of more than 30 articles in research areas, such as feature-oriented and aspect-oriented software engineering, machine learning, digital circuits, and programming.

FELIPE BESOAIN (Member, IEEE) received the B.S. degree in bioinformatic engineering from the Faculty of Engineering, University of Talca, Talca, Chile, in 2010, and the M.S. degree in free software and the Ph.D. degree in network information technologies from the Universitat Oberta de Catalunya, Barcelona, Spain, in 2012 and 2018, respectively. In 2021, he started a postdoctoral research in social psychology with the University of Talca, combining app development with current attitude change theory. He has been a researcher in various research and development projects with demonstrable experience in the development of mobile, immersive technologies (virtual reality) in industry in health, agronomy, tourism, culture, and heritage. His current research and interests include the application of persuasive technologies in intelligent contexts for the promotion of health; tourism and cultural heritage; and the use of ubiquitous computing and mobile devices, gamification, and immersive technologies for the form/change of attitudes.

NICOLAS A. BARRIGA (Member, IEEE) received the Ph.D. degree from the University of Alberta, Canada, for his work on state and action abstraction mechanisms for RTS games. He is a Professor with the Department of Interactive Visualization and Virtual Reality, University of Talca, Chile. His current research interests include procedural content generation for videogames and the broad area of video game AI.