Zhu G. and McArthur J.J. (2020). "Automatic generation of architectural layouts using Genetic Algorithms in BIM" In: *Proc. 37th CIB W78 Information Technology for Construction Conference* (CIB W78), São Paulo, Brazil, pp. 17-26. DOI: http://dx.doi.org/10.46421/2706-6568.37.2020.paper002

AUTOMATIC GENERATION OF ARCHITECTURAL LAYOUTS USING GENETIC ALGORITHMS IN BIM

Garbo Zhu[1] and J.J. McArthur [2]

Abstract: Room layout is a complex problem for architects, who must consider adjacencies, room sizes, site constraints, and circulation requirements. This re-search applies genetic and stimulated annealing algorithms to optimize space layout in two dimensions. Visual Programming Languages interfacing with BIM software are used to allow semantic and topological qualities of each room to be imported to develop complete floorplans. A series of case studies adopting different machine learning techniques are presented to demonstrate the relative performance of each approach, using the minimization of unusable space as a performance metric. Previous research has considered purely mathematical solutions to support the automation of these layouts, often resulting in irregular polygons or circular rooms; from an architectural standpoint, this is undesirable and thus a rectangle-packing algorithm has been developed and used. Future research to facilitate the adoption of this approach in architectural practice, particularly for large and complex buildings, is discussed.

Keywords: BIM, automatic floorplan generation, parametric model, genetic algorithms.

1 INTRODUCTION

Space planning is one of the most crucial segments in construction projects, as it directly affects the experiential quality, energy efficiency, and feasibility of the building. The process is often time-consuming and labor-intensive since it re-quires logical senses and previous work experience of the architects involved. However, with the rise of automated design technologies, repetitive tasks such as floor layouts, can now be assigned to computer, leaving creative works to designers. This research focuses on the preliminary phase of a construction project - room planning, intending to generate spatial relationship diagrams as the guidance for future architectural design. In this exercise, the demonstration will be adopting the typology of a small-scaled shopping mall and following the representational programs that are needed. This paper extends upon previous work of this type by eliminating the irregular room shapes developed by approaches such as circle packing and Voronoi, presenting instead a rectangle-packing algorithm developed to optimize floorplans. Further, to permit its application in practice, a BIM integration script is presented to create 3D rooms and map semantic data, resulting in true architectural floorplans.

[1] Research Student, Ryerson University, Toronto, Canada, garbo.zhu@ryerson.ca
[2] Assistant Professor, Ryerson University, Toronto, Canada, jjmcarthur@ryerson.ca

2 Literature Review

With developed technologies, the process of space planning has transformed into a multi-objective optimization problem to not only satisfy the design constraints but also achieve an optimal spatial configuration. There have been numerous studies conducted on the topic of room placement using different mathematical formulas and algorithms, for example (e.g., Merrell et al. 2010; Wong and Chan 2009; Nagy et al. 2017). Recently, parametric approaches have leveraged Genetic Algorithms (GA) (e.g., Rebaudengo and Reorda 1996), Simulated Annealing (SA) (e.g., Yi et al. 2014), and fixed grid/matrix layout (e.g., Sharafi et al. 2017). These have been used with optimization rules for minimum travel distances between room centroids (e.g., Liggett 2000), minimum dead spaces and overlaps (e.g., Baušys and Pankrašovaite 2005), or optimized energy use (e.g., Du et al. 2018).

Two algorithms of interest to this paper are GA and SA. GA (e.g., Tate and Smith 1995; Whitney 1994) is de-signed to speed up the solution-finding process to search for high-quality results under a confined framework. SA techniques (e.g., Van Laarhoven and Aarts 1987) help to eliminate and improve the common hill climbing problems faced by GA. Instead of being a population-based algorithm, SA uses an iterative single solution-based method, making it an exploitative algorithm. In standard result-mining algorithms, the new solution is accepted only if it meets the improved minimum value set by the selection mechanism. In SA, the indicator for an acceptable result can be based on a probability percentage, providing solutions that are more meticulous and closer to the global optima (e.g., Liggett 2000), and its use is recommended at the beginning of the search process (e.g., Liggett 2000; Szykman and Cagan 1997)

A current gap in the literature is the lack of linkage between theories and architectural applications. Most of the researches conducted have only been implemented based on randomized room dimensions within arbitrary boundaries. To accurately examine and improve the usability of space planning optimization techniques, a controlled scenario with defined space programming information is adopted.

3 Methodology

A case study approach was used to explore room layout options using a six-step process. First, a set of 17 rooms and the sample site of 35m x 35m (1225m^2) are established, along with the required adjacency relationships. Next, a circle packing algorithm is introduced to place all rooms on site. Thirdly, the result generated is optimized through machine learning techniques, including Genetic Algorithms and Simulated Annealing. Then, the optimized output is transformed into realistic square geometries, following the original spatial requirements. To eliminate overlapping boundaries, room outlines are shifted to accommodate the appropriate dimensions. The final step is to define polygons as data points to be input into BIM through Dynamo for Revit.

3.1 Room Datasheet

The proposed workflow starts with a spreadsheet (Table 1) containing spatial requirements and adjacency relationships between rooms as the input. There are three levels of spatial relationships between rooms: 1) adjacency, defining the requirement for rooms to be next to one another; 2) proximity, indicating a preference for rooms to be near one another, and 3) non-adjacency, indicating that rooms cannot share a common wall. To ensure interactivity and flexibility of the result, all information is linked back to the spreadsheet in real time which minimizes lag time and errors caused by delays.

Table 1: Room Datasheet Sample (truncated)

Room ID	Room Name	Area	Radius	Diameter	ADJ	Proximity	Non-ADJ
R0	Entrance	25	2.82	5.64	NA	NA	NA
R1	Service Room	30	3.09	6.18	R16	R6	R9
R2	Electrical Room	30	3.09	6.18	R16	R1	R8

3.2 Room Planning Sequence

From the structured table, the data is then processed through a series of functions (Fig. 1), coded using Python 3.6. The order within the list is determined by the number of adjacent rooms those rooms acquired, arranged in descending order. All adjacency relationships are captured in an ordered dictionary, with the key being rooms that were placed last, and the values representing rooms that should be placed next.

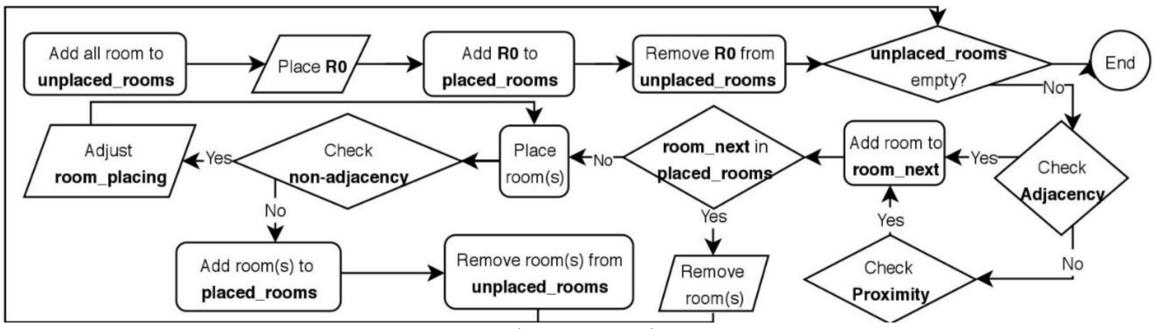

Fig. 1. Room Plotting Order Sequence

3.3 Room Placing Function – Circle Packing Algorithm

Circles are chosen as the primary visual representation due to their flexible quality with unlimited edges, giving more potential for spatial layouts. Stemming from the theory and logic behind circle packing algorithms (e.g., Collins and Stephenson 2003; Castillo et al. 2008), the proposed function introduced room planning sequence that governs the order and constraints of circle-plotting. Naturally, there will be dead spaces in-between each circle due to the curved outlines. Therefore, the input dimensions are smaller by 10% of the actual footprint required. Rooms are separated into three types of arrays for simple identification: array N, rooms that are currently being placed; array P, rooms that were placed prior to N; array D, rooms that are in the placed_rooms list. As each room (N_i) is added to the diagram, there are two rules that the function needs to satisfy. First, N_i cannot exceed the site constraints (**Eq. 1-4**). Where x/y superscript denote the x/y - coordinate of the centroid, r superscript denotes the radius of the room, and i subscript denotes the i^{th} item in the array (Fig. 2).

$$N_i^{\,x} - N_i^{\,r} > \emptyset\,;\;\; N_i^{\,x} + N_i^{\,r} < \text{site}_{\text{width}};\;\; N_i^{\,y} - N_i^{\,r} > \emptyset\,;\; N_i^{\,y} + N_i^{\,r} < \text{site}_{\text{length}} \#(1)-(4)$$

Second, N_i cannot overlap with any existing circles in the array D. The logic adopts the Pythagorean theorem to determine whether two circles collide (**Eq. 5**).

$$\sqrt{(N_i^{\,x} - D_i^{\,x})^2 + (N_i^{\,y} - D_i^{\,y})^2} < (N_i^{\,r} - D_i^{\,r}) \#(5)$$

The positioning for N_i is dependent on the location of P_i (**Eq. 6 & 7**), where $\boldsymbol{\alpha}$ represents the angle between the centroid of N_i and P_i.

$$N_i^{\,x} = \cos\alpha\,(N_i^{\,r} + P_i^{\,r}) + P_i^{\,x} \#(6)$$

$$N_i^{\,y} = \sin\alpha\,(N_i^{\,r} + P_i^{\,r}) + P_i^{\,y} \#(7)$$

At the beginning of the room plotting function, placement angle α is set equal to zero, resulting in $N_i^{\,y}$being set equal to $P_i^{\,y}$. To ensure that each N_i fulfils the two constraints listed above, an increment of 0.01 (in radians) is added to α, causing N_i to rotate around P_i until both rules are satisfied.

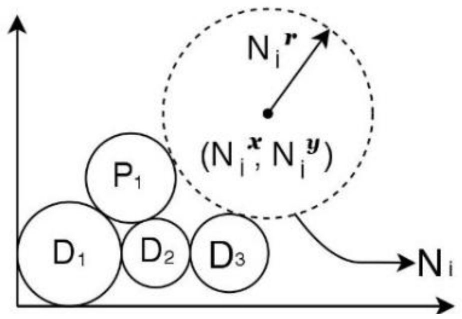

Fig. 2. Room plotting sequence sample

3.4 Machine Learning Algorithms for Space Optimization

To optimize the output and to transform the results to be more spatially efficient, theories of machine learning was applied to determine the optimal placement angle.

The method adopted is a common approach to Genetic Algorithm, where chromosomes of two parents are selected to produce children containing genes from both parties. To start off the search, a population {- π<**α**<π} is defined. In this scenario, the genes are represented by all **α** that were implemented in the plotting function and are stored in an array that can be mutated in increments of +/- 0.01. Since the goal is to achieve the smallest footprint while placing all rooms within the site, the fitness calculation is based on the bounding area that the current configuration. The smaller the area is, the higher the fitting score, and the more likely it is for its genes to be copied down. Conversely, if the total bounded area is larger than the previous generation, it is a less ideal candidate for reproduction.

To refine the mating pool, Elitism (e.g., Deb et al. 2007) is performed, meaning that the top 10% of the fittest population **α** goes to the next generation. Then, two parents are generated, through both selecting 50% each from the previous array randomly. The same logic applies to mating, where the child adopts 50% of the array from each parent randomly to produce the next generation. However, this optimized algorithm was stuck in the local minimum between two iterations, one that has excess dead space, and the other having overlapping polygons.

To achieve global optima efficiently, SA is introduced to improve the current circumstances. For a finite set of iterations, the value of bounding area is used to compare with its previous iteration. The objective is consistence with GA listed above, if the fitness score is higher, meaning that the bounding area is smaller, the value of the previous array **α** will take on the current array **α'**, which is chosen randomly. Inversely, if the fitness score is lower, whether α will take on the value of **α'** is dependent on the rate of probability **p**, which is calculated through Boltzmann probability factor (e.g., Ingber 1993). The benchmark for acceptance probability p is also randomly generated between ∅<**p**<1.0 to avoid termination at local minima (e.g., Ingber 1993).

3.5 Room Squaring

Circle diagrams are the schematic representations for room layouts. To generate a workable floor plan, the graphic must be transformed through shape-finding functions. The goal is to create realistic geometries that are pragmatic for construction. Since the centroids of each room, all required dimensions and adjacency relationships are stored in a central database, the process of producing straight-edged spaces is straightforward.

For each room N_i, the existing centroid is maintained and the bottom, top, left, and right edges of the room polygon $\{N_\mathrm{i}{}^{B} = N_\mathrm{i}{}^{y}, N_\mathrm{i}{}^{T}, N_\mathrm{i}{}^{L}, N_\mathrm{i}{}^{R}\}$ are defined by adding or subtracting the radius $N_\mathrm{i}{}^{r}$ from the centroid as appropriate, for example (**Eq. 8**). This increases the room area but provides flexibility when overlapping walls are resolved in the next step. Overlapped edges are detected by looping through all rooms N_i and comparing to the remaining rooms N_j, where j=i+1 to the total number of rooms and checking whether top right, top left, bottom right, or bottom left corners overlap with any of the other rooms N_j. The rules (**Eq. 9a & b**) and resolution (**Eq. 10**) for a clashing top (of N_i) and bottom (of N_j) edges are shown below; identification and resolution of all other overlaps follow this logic. Finally, once all overlaps have been resolved, each room is tested (Eq. 11) to ensure that the area is maintained within acceptable tolerance.

.

$$N_\mathrm{i}{}^{B} = N_\mathrm{i}{}^{y} - N_i{}^{r} \#(8)$$

$$N_\mathrm{j}{}^{T} > N_i{}^{B} > N_\mathrm{j}{}^{B} \#(9)$$

$$N_\mathrm{j}{}^{\mathrm{R}} > N_i{}^{L} > N_\mathrm{j}{}^{L} \#(9b)$$

$$N_\mathrm{j}{}^{T'} = N_\mathrm{i}{}^{B'} = \frac{N_\mathrm{i}{}^{B} + N_\mathrm{j}{}^{T}}{2} \#(10)$$

$$1.05 N_\mathrm{i}{}^{Area} \geq (N_\mathrm{i}{}^{T'} - N_\mathrm{i}{}^{B'}) * (N_\mathrm{i}{}^{R} - N_\mathrm{i}{}^{L}) \geq 0.95 N_\mathrm{i}{}^{Area} \#(11)$$

Finally, rooms not achieving the area tolerance were adjusted (Eq. 10).

3.6 Room Import to BIM

To transform the room data developed above to architectural floor plans, visual programming language, Dynamo, was used to allow users to access the results through BIM software. Due to the fundamental deviation in programming languages – Dynamo uses IronPython and cannot recognize the more standard CPython – the polygons were extracted as a list that can then be read by Dynamo.

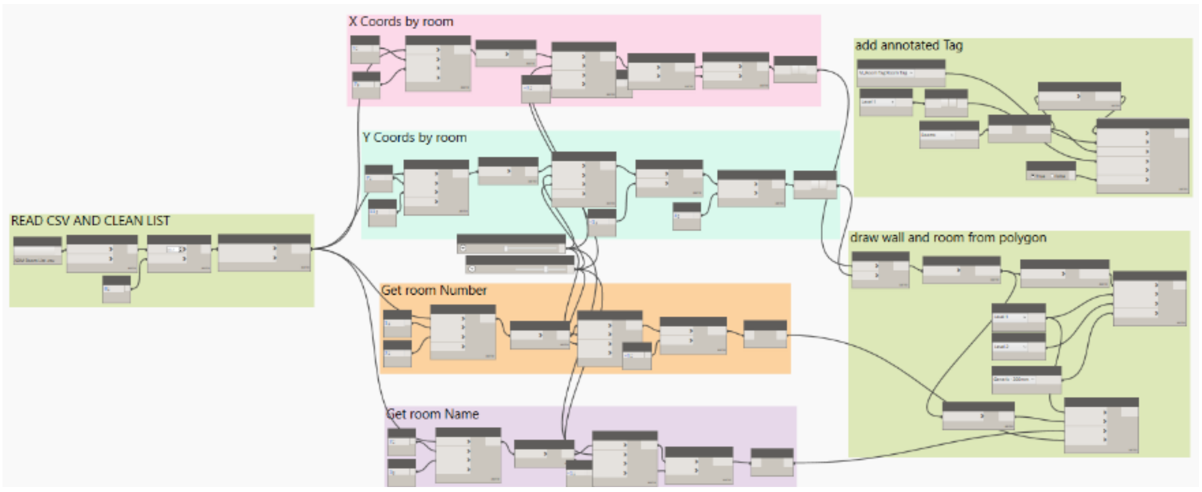

Fig. 3. Creation of rooms and generated polygons and mapping of room numbers using Dynamo.

4 Results

The sequence of images shown (Fig. 4) highlights the results of each step in the room placement sequence. The result for attempt #1 (Fig. 4a) has the following errors: overlapping polygons, dead spaces, and circles that have exceeded the site boundaries (square in red). The total footprint of the bounded area is 792m^2. The optimization algorithm achieved a footprint of 730m^2 while satisfying both requirements of within site boundaries and having zero overlapping circles. Initially, only GA was for optimization, however, the proposed function was trapped in two local minima between two iterations: one with excess dead space, and the other having over-lapping polygons. To resolve this, SA was introduced, and its logical randomization approach assisted the searching process to be more accurate and efficient and resulted in a fully resolved and optimized layout (Fig. 4b). During squaring (Fig. 4c), significant overlaps were noted between all rooms and had to be resolved. Applying Eq. 8-11, 15 of 17 rooms achieved the 5% area tolerance, with the remaining two rooms at 6.50% and 7.25% above the required area, which is within the 10% tolerance typically used in practice. This resulted in a final, fully resolved layout (Fig. 4d) with a total room area of only 2% above the required area presented in the room datasheet.

The final floor plan (Fig. 5), implemented in Revit achieved all desired adjacencies, as it follows the adjacency matrix and size constraints of the original data input yet remaining a buildable geometry.

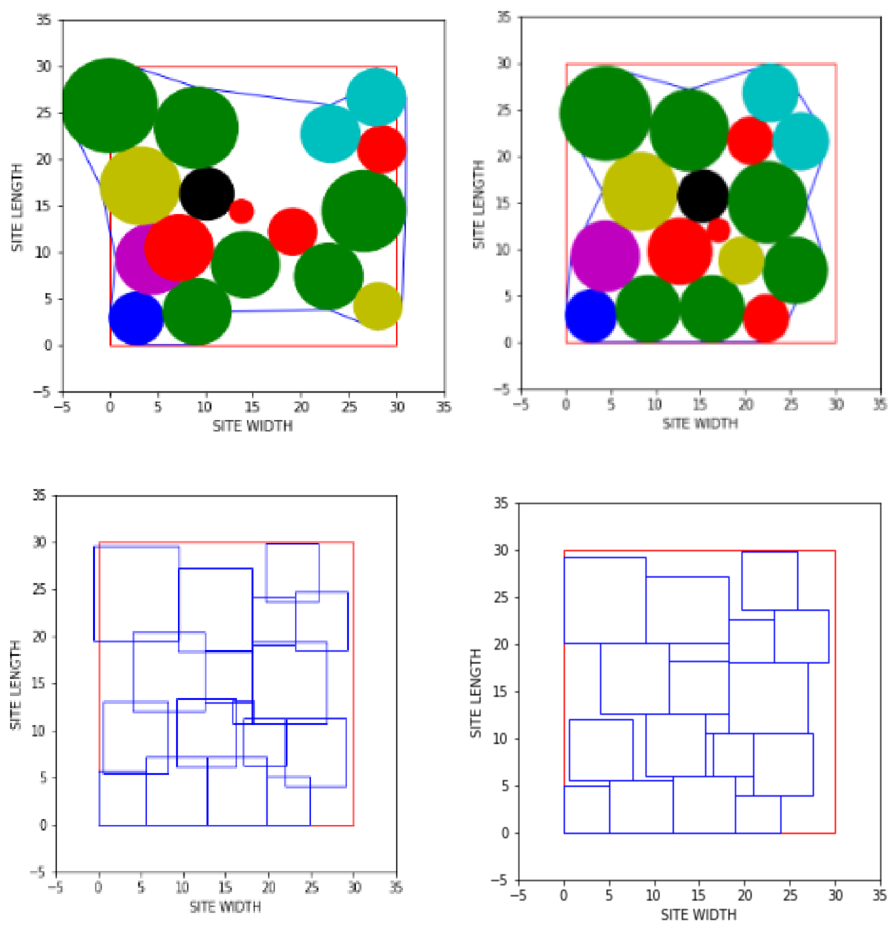

Fig. 4. Room placement and optimization sequence: Circle packing before (a; top left) and after optimization (b; top right); squared rooms with (c; bottom left) and without overlap (d; bottom right)

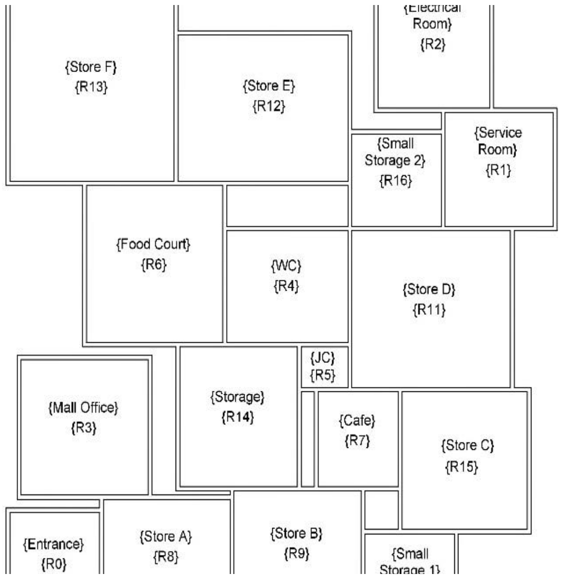

Fig. 5. Creation of rooms from polygons generated and mapping of room numbers using Dynamo

5 Discussion and Conclusions

From room datasheet to BIM, this researched has achieved a streamlined approach to room-planning under the assistance of visual programming language and two automated optimization algorithms. The current gap in existing studies is the disconnection between research and applications. Most of the functions have only been tested within the research team, since examples such as circle-packing and GA/SA optimization algorithms were initially developed to solved mathematically problems. Without considering room-planning in a real-life setting, the logic developed cannot be proven to be reliable and beneficial within the industry. In this paper, techniques were refined and tested using a small-scaled architectural prototype. The results obtained have shown the realistic performance of the approaches adopted, displaying the adaptability of this research and the great potential it acquires.

By combining the circle room plotting sequence with machine-learning algorithms, the intention to achieve the most compact configuration was realized. Through the room datasheet, users had absolute control over the required rooms, dimensions, and the adjacency matrix, while retaining the benefits of an algorithm-assisted decision-making design. The room plotting sequence provides architects with an accurate visual representation of the most spatially efficient layout within a shorter timeframe, particularly when designing large-scaled projects with repetitive unit types. With the assistance of automated design tools, laborious tasks can be relinquished to computer programs, freeing designers to focus on creative ventures.

A limitation of this approach is that it is not yet seamless. This is primarily be-cause machine learning packages are not usable within Dynamo and required the use of Python and export of polygon coordinates to for import into Dynamo. A second limitation is the rigidity of the fitting function. Since the bounding area is the only measure of success, outcomes of the optimization had omitted other aspects of the space. Finally, consistent with the majority of previous studies, this algorithm considered only a single level. The extension of this work to multiple floors is desirable and should be developed in future research. Future research should also focus on holistic approaches when applying generative design tools for construction projects, such as incorporating circulation spaces and building code compliance evaluation. Alternative fitting functions and multi-objective should be considered to suit different needs depending on the functionality of space, for example, energy optimization and/or minimized travel distances.

The practical implications of this work are significant. For architects, layout automation tools can increase the efficiency in the preliminary stage of design. Traditionally, during massing studies, where building footprints are constantly under revision, hours of works were poured into the monotonous routine of mapping out all required programs within a confined space. With the assistance of optimized room-planning technique, the most compact floor layouts can be created automatically, and ready to be used in BIM right away. With the significant decrease in time spent on repetitive tasks, architects can explore a broader range of potential building footprints and floorplates to optimize energy performance, and focus on the occupant experience of the space, thus enhancing overall design quality.

6 References

Baušys, R. and Pankrašovaite, I. (2005). Optimization of architectural layout by the improved genetic algorithm. *Journal of Civil Engineering and Management.*, 11(1), pp. 13-21. Available at: https://doi.org/10.1080/13923730.2005.9636328 [Accessed 8 Oct. 2019].

Bergmann, A., Fritz, G. and Glatter, O. (2000). Solving the generalized indirect Fourier transformation (GIFT) by Boltzmann simplex simulated annealing (BSSA). *Journal of applied crystallography*, 33(5), pp. 1212-1216. Available at: https://doi.org/10.1107/S0021889800008372 [Accessed 16 Oct. 2019].

Castillo, I., Kampas, F. and Pintér, J., (2008). Solving circle packing problems by global optimization: numerical results and industrial applications. *European Journal of Operational Research*, 191(3), pp. 786-802. Available at: https://doi.org/10.1016/j.ejor.2007.01.054 [Accessed 26 Oct. 2019].

Collins, C. and Stephenson, K. (2003). A circle packing algorithm. *Computational Geometry*, 25(3), pp. 233-256. Available at: https://doi.org/10.1016/S0925-7721(02)00099-8 [Accessed 20 Oct. 2019].

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2), pp. 182-197. Available at: https://doi.org/10.1109/4235.996017 [Accessed 6 Nov. 2019].

Du, T., Turrin, M., Jansen, S., Van Den Dobbelsteen, A. and Biloria, N. (2018). A Review on Automatic Generation of Architectural Space Layouts with Energy Performance Optimization. *Int. Conf. Build. Energy, Environ.,* pp. 856-861. Available at: http://www.cobee2018.net/assets/pdf/p/283.pdf [Accessed 2 Nov. 2019].

Ingber, L. (1993). Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11), pp. 29-57. Available at: https://doi.org/10.1016/0895-7177(93)90204-C [Accessed 10 Nov. 2019].

Liggett, R. S. (2000). Automated facilities layout: past, present and future. *Automation in Construction*, 9(2), pp. 197-215. Available at: https://doi.org/10.1016/S0926-5805(99)00005-9 [Accessed 20 Oct. 2019].

Merrell, P., Schkufza, E. and Koltun, V. (2010). Computer-generated residential building layouts. In *ACM SIGGRAPH Asia 2010*, pp. 1-12. Available at: https://doi.org/10.1145/1866158.1866203 [Accessed 25 Oct. 2019]

Nagy, D., Lau, D., Locke, J., Stoddart, J., Villaggi, L., Wang, R., Zhao, D. and Benjamin, D. (2017). Project Discover: An application of generative design for architectural space planning. In *Proceedings of the Symposium on Simulation for Architecture and Urban Design,* pp. 1-8. Society for Computer Simulation International. Available at: https://doi.org/10.5555/3289787.3289794 [Accessed 15 Nov. 2019]

Rebaudengo, M. and Reorda, M. (1996). GALLO: A genetic algorithm for floorplan area optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(8), pp. 943-951. Available at: https://doi.org/10.1109/43.511573 [Accessed 20 Nov. 2019]

Sharafi, P., Samali, B., Ronagh, H. and Ghodrat, M. (2017). Automated spatial design of multi-story modular buildings using a unified matrix method. *Automation in Construction,* 82, pp. 31-42. Available at: https://doi.org/10.1016/j.autcon.2017.06.025 [Accessed 12 Oct. 2019]

Szykman, S. and Cagan, J. (1997). Constrained three-dimensional component layout using simulated annealing. *Journal of Mechanical Design,* 119(1), pp. 28-35. Available at : https://doi.org/10.1115/1.2828785 [Accessed 23 Oct. 2019]

Tate, D. and Smith, A. (1995). A genetic approach to the quadratic assignment problem. *Computers & Operations Research*, 22(1), pp. 73-83. Available at: https://doi.org/10.1016/0305-0548(93)E0020-T [Accessed 23 Oct. 2019]

Van Laarhoven, P. and Aarts, E. (1987). Simulated annealing. In *Simulated annealing: Theory and applications,* pp. 7-15. Springer, Dordrecht.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing*, 4(2), pp. 65-85. Available at: https://doi.org/10.1007/BF00175354 [Accessed 21 Oct. 2019]

Wong, S. and Chan, K. (2009). EvoArch: An evolutionary algorithm for architectural layout design. *Computer-Aided Design*, 41(9), pp. 649-667. Available at: https://doi.org/10.1016/j.cad.2009.04.005 [Accessed 23 Nov. 2019]

Yi, H., Yi, Y. and Chan, T. (2014). Performance Based Architectural design optimization: Automated 3D space Layout using simulated annealing. In *Proceedings of the 2014 ASHRAE/IBPSA-USA Building Simulation Conference*, pp. 10-14. Available at: https://www.researchgate.net/publication/288790548_Performance_based_architectural_design_optimization_Automated_3D_space_layout_using_simulated_annealing [Accessed 12 Nov. 2019]