Transdisciplinary Engineering for Resilience: Responding to System Disruptions
L. Newnes et al. (Eds.)

doi:10.3233/ATDE210125

GPU-Enabled Physics-Based Floor Plan Optimization Based on Work Place Analytical Data

Axel NORDIN
Department of Design Sciences, Faculty of Engineering, LTH, Lund University, Sweden

Abstract. This paper presents an approach for optimizing floor plans using data collected from workplaces and a physics-based planning algorithm utilizing GPU-acceleration. The approach is compared to two approaches based on Genetic Algorithms and Particle Swarm Optimization. Common objectives, such as minimizing distance between related departments and daylight usage are optimized using the physics-based method in a case study of an actual plot and company. The results show that the physics-based approach is very efficient at handling large amounts of rooms compared to the other methods tested.

Keywords. Floor planning, architecture, particle swarm optimization, genetic algorithm, physics-based.

Introduction

Designing a residential or office building is a complex task, with many constraints and objectives that need to be fulfilled. The requirements come from technical regulations, economical considerations, and workplace efficiency and experience. As regulations and budgetary concerns are often prioritized, the human factors in floor planning are frequently overlooked. To solve this, and to handle the complexity of the design task, computational tools may help. This paper presents a method for efficiently designing a workplace floor plan based on collected data from employees, and constraints such as the outer boundary of each floor.

Generating and optimizing floor plans has been studied at least since the 1970s [1], however, there have been few, if any, attempts to generate them using a physics-based approach. Physics-based form finding has previously been successful in other areas [2], which is why the approach may be interesting for floor planning as well. Rigid body physics and spring based constraints have many properties that make them interesting from a layout point of view as they deal naturally with the interaction of two and three dimensional objects. First, they efficiently prevent overlapping geometry in two and three dimensions. Second, they efficiently handle location-based objectives, such as preferred locations and preferred neighboring objects/rooms. Third, they can efficiently be run on modern graphical processing units (GPUs) using highly parallelized algorithms.

In this paper, a floor planning approach is presented that is based on Nvidias PhysX Flex platform for particle-based rigid and soft body physics through the Flexhopper plug-in for Grasshopper and Rhinoceros. The optimization is based on actual data collected

on workplace efficiency and experience, such as area and adjacency requirements. This requirement data is handled by the physics-based method, resulting in a time-efficient way of positioning large amounts of offices in parallel on the GPU. The method is compared to two popular algorithms for design exploration: the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). The physics-based method is then applied to a case-study based on actual requirements and building envelopes.

1. Related works

1.1. Floor plan generation

Floor planning has been an interesting optimization and generation problem since at least the 1970s when Eastman [1] did early work on automating floor planning through a rule-based approach for sequentially filling the floor plan. More recent examples include Das et al [3] who use a tree-based partitioning of the boundary to fill grid-based outlines. Flack and Ross [4] use GA and a graph-based approach to evolve rectilinear floor plans. Rodrigues, Gaspar and Gomes [5] study energy efficiency of the floor plan using a two-step approach, where a floor plan is first generated, and then optimized for energy efficiency. Among the approaches that utilize Grasshopper and GA is the work of Boon et al. [6] who use Grasshopper and the plug-in Galapagos to optimize floor plans. They generate floor plans in three dimensions and minimize the distance between the functional areas of the plan.

Within computer graphics, there are also a number of publications that attack the problem, such as Merrel, Schkufza and Koltun [7] and Lopes et al. [8] who use graph-based approaches to plan rectilinear boundaries and Tutenel et al. [9] who use a rule-based algorithm for procedural generation of interiors.

However, none of the above presented works applies a physics-based solver to the floor-planning problem, and none looks at completely arbitrary floor boundaries. While some works discretize an arbitrary boundary using a grid-based system [7][4], the methods do not work directly on the actual outline. Moreover, none of the works utilizes the GPU in finding solutions.

1.2. Genetic Algorithm

First presented by Holland and Goldberg [10][11], GA has been applied to a wide range of applications within building design [12], and has, as mentioned above, been used before in generating floor plans. The genetic algorithm emulates natural evolution by gradually improving the fitness of individuals in a population over several generations. Each individual represents one point in the search space, and is assigned a fitness based on how well it conforms to constraints and objectives. The most fit individuals are then selected to breed the next generation through cross-over and mutation operators. The specific GA implementation used in this paper is Galapagos, which is a Rhino/Grasshopper plug-in.

1.3. Particle Swarm Optimization

PSO has been used for a large number of applications [13][14] including floor planning [15]. PSO is inspired by the motion of, for instance, birds, bees or fish and their ability to work intelligently as a swarm, even though each individual may only be acting according to simple rules. In PSO, each individual, or agent, adapts its motion through the search space according to its history, fitness and neighboring agents. By attractive forces to more optimal regions of the search space, the agents collectively move towards a global optimum. The specific PSO implementation used in this paper is Silvereye, which is a Rhino/Grasshopper plug-in.

2. Description of the implementation

2.1. Workplace efficiency and experience

In order for the results of the floor plan generator to be meaningful, real-life objectives and constraints are required. One way of collecting these is by analyzing an existing workplace through interviews, surveys, location based trackers, video recording, etc. Such data can be analyzed and converted into physical requirements, such as number, size and type of rooms, which functions need to be close to each other, etc. Other requirements in terms of building codes, such as minimum daylight intake and accessibility can also be added. In this paper, the following requirements have been implemented and are taken into account by the algorithms presented:

- Area of room
- Type of room
- Number of rooms of a certain type
- Distance to daylight
- Distance to preferred neighboring functions
- Accessibility
- No overlapping rooms
- No rooms outside the perimeter of the floor

2.2. Physics-based requirements

Some requirements can be fulfilled simply through generating rigid bodies with the correct geometry. However, other requirements need to be represented as forces to enable a physics-based optimization algorithm. Forces include, but are not limited to, user-defined force fields such as point sources, wind, gravity; contact forces between moving and static objects; frictional forces; and forces generated by springs. A summary of how the different requirements were represented is shown in **Table 1**.

Table 1. List of requirements and their physics-based representation

Requirement	Representation
Area of room	Geometry
Type of room	Geometry
Number of rooms of a certain type	Geometry
Distance to daylight	Force field
Distance to preferred neighboring functions	Springs

Accessibility	Contact forces and force field
No overlapping rooms	Contact forces
No rooms outside the perimeter of the floor	Contact forces

2.3. Time-dependent constraints

As applying all force-based requirements at once would result in a sub-optimal solution, where objects do not have enough time to find their optimal positions before being entangled or forced into sub-optimal positions by other objects, a time-dependent approach is needed. The time-dependent approach applies physics-based requirements one at a time. For instance, during the initial stage the rooms are allowed to move freely on the floor plan and are only constrained by the neighboring springs, the floor surface and gravity. This enables the rooms to find their neighbors without any pre-positioning, other than a random scatter at the beginning. The time-line of the approach is shown in Table 2. After a requirement has been activated, it remains active throughout the time-line.

Table 2. Time-line of the requirements

Requirement	Sequence
Area of room	0
Type of room	0
Number of rooms of a certain type	0
Distance to preferred neighboring functions	1
No overlapping rooms	1
Distance to daylight	2
No rooms outside the perimeter of the floor	2
Accessibility	3

2.4. Platform

The rigid body physics simulation is implemented through the Nvidia PhysX Flex software library, which enables rigid body physics to be simulated on Nvidia GPUs. This approach enables massive parallelization of the simulation and orders of magnitude faster simulation times compared to CPUs at a comparable price point. The PhysX library was interfaced through the Flexhopper plug-in for Grasshopper, which, in turn, is a scripting plug-in for the CAD-tool Rhinoceros by McNeel. This enables the definition of geometry such as floor perimeter and office rooms in Rhinoceros' standard CAD interface, and the algorithmic scripting of Flexhopper through Grasshopper. As Rhino provides a familiar CAD interface, this platform is intuitive to use for an Architect.

2.5. Set-up

The scale of the model is in meters, to better work with the default physics settings and numerical limits of the calculations. An artificially high gravity of 350 m/s^2 was used to ensure that the blocks representing the rooms remained in contact with the floor during the simulation. A particle radius of 0.4 m was used. The same radius was used for the Solid Rest Distance and Collision Distance setting, which determines the minimum distance between solids. A Particle and Shape collision margin of 0.5 m was used. A maximum speed of 3.4028e38 m/s and a maximum acceleration of 1000 m/s^2. All the remaining rigid body physics settings were set to 0, except Relaxation Mode, which was

left at the default value of True, and Relaxation Factor, which was left at the default value of 1.

The springs between the rooms were connected to the center of gravity for each room and had a spring stiffness of 0.05 and a rest length of 0 m.

The force field that attracts bodies to the outside perimeter was represented as a number of point attractors spread out at equal distances along the perimeter with a strength linearly changing from 100 to -100 throughout the simulation time-line, with a positive value indicating a repulsive force and a negative an attractive force.

To ensure accessibility, a wall offset from the outside perimeter by the width of the rooms and with the thickness of the required corridor was introduced at the last stage of the simulation to force the rooms to leave a void. The corridor was introduced from below the floor to lift up and displace the offending rooms. The corridor initially had a repulsive force field attached to it to force rooms away from it. During the last stage of the simulation, the repulsive force was replaced with an attractive force to pull the rooms to it.

3. Benchmark

The physics-based method was benchmarked against two commonly used methods for design optimization, GA and PSO. The benchmark consisted of a simple outer boundary and a number of rooms to be placed. The objectives were to keep rooms inside the outer boundary, to avoid overlapping rooms, and to place the rooms as closely to each other as possible. To compare how the methods perform with the complexity of the problem, two configurations were run: one with 10 rooms, and one with 100 rooms.

3.1. Set up

The rooms are set up as rectangles with random widths and lengths between 1 and 2 m. Each room is given a random initial position, but within the outer boundary.

The parameters for the GA and PSO methods are the *x*/*y* offsets from the initial position, represented as floating point numbers. The initial value for each parameter is set to 0 m. The parameter intervals are set to -30 m to 30 m, which allows the rooms to be moved just outside the perimeter.

All methods were constrained to a run-time of 1.5 minutes for the 10-room benchmark, and 15 minutes for the 100-room benchmark. While this does not ensure convergence of the result, the difference in efficiency between the methods should be possible to determine.

As Silvereye and Galapagos do not support separate constraints or multiple objectives, the goal functions are weighted into a single objective function that is to be minimized.

Rooms outside of the outer boundary are penalized by adding the number of rooms that are outside the outer boundary, $n_{outside}$, and normalizing the value by dividing by the total number of rooms, n, as described by Eq. 1.

$$f_1 = \frac{n_{outside}}{n} \quad \text{Eq. 1}$$

The number of overlapping rooms is penalized by counting the number of overlapping rooms, $n_{overlapping}$, and normalizing the value by dividing it by the total number of rooms, as described by Eq. 2.

$$f_2 = \frac{n_{overlapping}}{n} \quad \text{Eq. 2}$$

The objective to bring the rooms as close together as possible is measured by the total distance between the rooms and normalizing the value by dividing it by a reasonable estimate of the maximum possible distance between the rooms and the number of distance comparisons, as described by Eq. 3.

$$f_3 = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sqrt{(X_j - X_i)^2 + (Y_j - Y_i)^2}}{\sqrt{(X_{max} - X_{min})^2 + (Y_{max} - Y_{min})^2}\, n(n-1)/2} \quad \text{Eq. 3}$$

The weighted objective value used by the GA and PSO methods, $f_{weighted}$, is simply the sum of all objective functions divided by the number of them, as described by Eq. 4.

$$f_{weighted} = \frac{f_1 + f_2 + f_3}{3} \quad \text{Eq. 4}$$

PSO settings: Iterations 100, max velocity 0.200, swarm size 20.

GA settings: Max stagnant 50, Population 50, Initial boost 2, Maintain 5%, Inbreeding 75%.

3.2. Results

The results for the benchmark run with 10 and 100 rooms are presented in **Figure 1** and **Figure 2** respectively. The fitness reported in the figures is based on $f_{weighted}$ from Eq. 4. A visual representation of the results for the methods is presented in **Figure 4**. It is evident that the physics-based method converges on a result very quickly compared to the two other methods. The difference in convergence rate is even more evident in the 100-room benchmark. The PSO method seems to be more suited to this particular problem than GA.

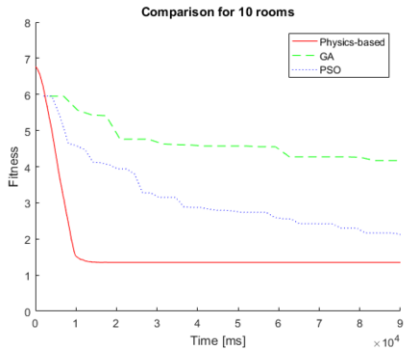

Figure 1. Fitness over time for the three methods while optimizing the placement of 10 rooms.

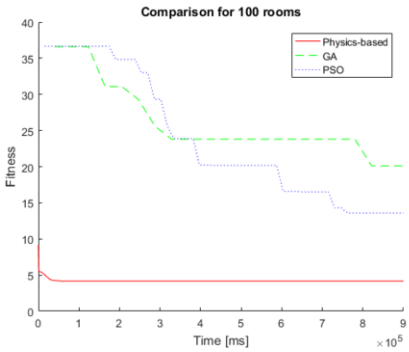

Figure 2. Fitness over time for the three methods while optimizing the placement of 100 rooms.

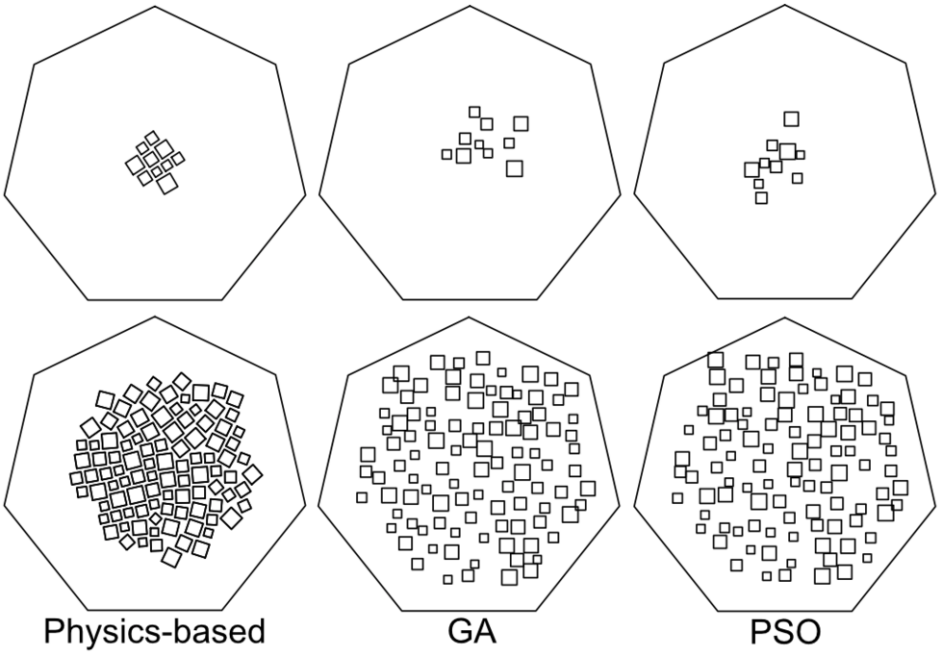

Figure 3. A visual representation of the optimization results for the three methods.

4. Case study

A floor plan for an office space in the H+ area of Helsingborg, Sweden was used as the starting point, see **Figure 4**. Workplace data from WPA for an anonymous company was used as a basis for the requirements. Based on this data, geometries for the different rooms and their preferred neighboring functions were input.

Figure 4. Plan for the area specified for the case study, with the building footprint marked in red (courtesy of Helsingborg Stad and Superlab)

Enabling requirements one after another, the results shown in **Figure 5** are achieved. Initially, only the accessibility, perimeter and no overlap requirements are enabled, as shown in **Figure 5**a. In the **Figure 5**b, the proximity requirements are enabled, resulting in the two groups of rooms to cluster together. In **Figure 5**c, the proximity to the outer perimeter requirement is enabled. The algorithm was then applied to several floors of the building, with the results presented in **Figure 6**.

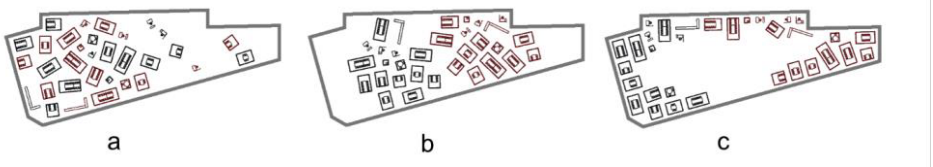

Figure 5. Rooms placed within the boundary. The rooms are color-coded based on the department they belong to.

Figure 6. The physics-based algorithm applied to several floors in a building.

5. Conclusions

This paper has demonstrated that physics-based floor planning is a viable method for quickly and efficiently determining the optimal position of rooms within an outer boundary. With the given constraints and objectives described in this paper, the physics-based method outperformed popular methods such as GA and PSO. It should be noted, however, that the convergence rate of stochastic methods like GA and PSO will vary slightly depending on the initial random seed and more independent runs would be needed to statistically compare the methods.

While physics-based optimization is promising in its efficiency, there are limitations to its usability. Mainly, the challenge lies in being able to represent objectives and constraints as physical forces.

Acknowledgements

This work was carried out with support from Familjen Kamprads stiftelse, grant number 20200099, and Region Skåne within the project What Matter_S. Information and data has been provided by Superlab and Helsingborg Stad.

References

[1] C.M. Eastman, Automated Space Planning, *Artificial Intelligence*, 1973, 4 (1): 41–64.
[2] D. Piker, Kangaroo: Form Finding with Computational Physics, Architectural Design, 2013, Vol. 83 (2): 136–37. https://doi.org/10.1002/ad.1569.
[3] S. Das, C. Day, A. Hauck, J. Haymaker, and D. Davis, Space Plan Generator: Rapid Generation & Evaluation of Floor Plan Design Options to Inform Decision Making, *Proceedings of the 36th Annual Conference of the Association for Computer Aided Design in Architecture*, 2016, 106–15.
[4] R.W.J. Flack and B.J. Ross.. Evolution of Architectural Floor Plans, *Lecture Notes in Computer Science*, 6625 LNCS, 2011, pp. 313–322.
[5] E. Rodrigues, A. Rodrigues Gaspar and Á. Gomes. Improving Thermal Performance of Automatically Generated Floor Plans Using a Geometric Variable Sequential Optimization Procedure, 2014, *Applied Energy*, Vol. 132, pp. 200–215.
[6] C. Boon, G. Corey, N. Papaefthimious, J. Ross, and K. Storey, Optimizing Spatial Adjacencies Using Evolutionary Parametric Tools: Using Grasshopper and Galapagos. *Perkins + Will ResearcH JournaL* 2015, 07 (02): 25–37
[7] P. Merrell, E. Schkufza and V. Koltun. Computer-Generated Residential Building Layouts, *ACM Transactions on Graphics*, 2010, 29:1, https://doi.org/10.1145/1866158.1866203.
[8] R. Lopes, T. Tutenel, R.M. Smelik, K.J. de Kraker and R. Bidarra, A Constrained Growth Method for Procedural Floor Plan Generation, *11th International Conference on Intelligent Games and Simulation, GAME-ON 2010*, pp. 13–20.
[9] T. Tutenel, R. Bidarra, R.M. Smelik and K.J. De Kraker. Rule-Based Layout Solving and Its Application to Procedural Interior Generation. *Proceedings of the CASA Workshop on 3D Advanced Media in Gaming and Simulation (3AMIGAS)*, 2009, https://graphics.tudelft.nl/Publications-new/2009/TSBD09a/TBSD09aRB.pdf, Accessed July 1, 2021.
[10] J.H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.
[11] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, 1988.
[12] A.T. Nguyen, S. Reiter and P. Rigo, A Review on Simulation-Based Optimization Methods Applied to Building Performance Analysis, *Applied Energy*, 2014, Vol. 113, pp. 1043-1058.
[13] R. Poli, Analysis of the Publications on the Applications of Particle Swarm Optimisation, *Journal of Artificial Evolution and Applications*, 2008, pp. 1–10. https://doi.org/10.1155/2008/685175.

[14] J. Kennedy and R.C. Eberhart, Swarm Intelligence, *IEEE Technology and Society Magazine*. 2002, Vol. 21, Issue 1, pp. 9, https://doi.org/10.1109/MTAS.2002.993594.
[15] T.Y. Sun, S.T. Hsieh, H.M. Wang and C.W. Lin, Floorplanning Based on Particle Swarm Optimization, *Proceedings - IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures 2006*, 2006, pp. 7–11. https://doi.org/10.1109/ISVLSI.2006.46.