# Abstract

This project explores the application of the Firefly Algorithm (FA), a metaheuristic inspired by the behavior of tropical fireflies, to room layout optimization. The problem is transformed into a discrete one with rectangular objects, each defined by its width, depth, rotation, reserved space, and a unique identification. The design decisions include an objective function to maximize the roundness of open areas, conflict resolution through an iterative noise separation technique, and the role of hyperparameters in object movement and rotation. Experimental results indicate that the algorithm performs well in maximizing open space and resolving conflicts, but faces challenges with complex room setups. A working tool for layout optimization is presented and avenues for its enhancement are proposed. The research offers insights into the application of the Firefly Algorithm for room layout optimization.

# Introduction

Optimizing room layout is a complex problem that involves arranging objects within a given space in the most efficient or aesthetically pleasing manner. This problem is multi-faceted, involving considerations such as object dimensions, room size, and user preferences. Traditional methods can be time-consuming and may not always yield the best results. Therefore, there is a need for more efficient and effective solutions.

Metaheuristics, a high-level problem-independent algorithmic framework, offer a promising approach to tackle such optimization problems. They provide strategies to escape local optima and explore the search space efficiently, making them suitable for solving complex optimization problems where the landscape of the solution space is not fully known or is computationally expensive to evaluate.

One such metaheuristic is the Firefly Algorithm (FA), inspired by the natural behavior of fireflies. The FA uses the concept of light intensity and attractiveness to move the ‘fireflies’ (potential solutions) towards the ‘brightest’ one (the best solution so far), thereby exploring the solution space for global optima. This research delves into the application of the Firefly Algorithm for the problem of room layout optimization, offering a novel perspective on leveraging bio-inspired algorithms for spatial design challenges.

# Experimental

## Set-up

The experimental procedure consisted firstly of selecting a real space to approximate and optimize. The space chosen was the laboratory shared by the authors. The dimensions of the selected room and of notable furnishings within it were measured and abstracted to a grid format. The measurer used his best judgement to assign a suitable amount of reserved space to appropriate objects. Table 1 describes the room and furnishings.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Object** | **Width** | **Depth** | **Reserved Space** | **No. in Room** | **Total Area** | **Total Area (Reserved Included)** |
| Room | 40 | 20 | - | - | 800 | - |
| Desk | 5 | 2 | 3 | 16 | 160 | 400 |
| Shelf | 3 | 1 | 4 | 2 | 6 | 30 |
| Cabinet | 3 | 2 | 2 | 3 | 18 | 36 |
| Couch | 6 | 2 | 3 | 1 | 12 | 30 |
| Table | 4 | 2 | 3 | 3 | 24 | 60 |
| Door | 4 | 0 | 4 | 2 | 0 | 32 |

The doors in the last row are located along the lower wall, with their left edges at positions 0 and 14, approximately where they are in the physical room. They are not moveable while the optimizer runs.

The reserved area of the objects makes a great deal of difference: when the actual occupied area is considered, it takes up 27.5% of the room’s space; when the reserved space is included, this increases to 69.5%.

For an empty room of these dimensions, the taxicab distance metric has an objective function value, or score, of 0.02155625, which forms an upper bound on the quality of the solutions the algorithm finds. This value is used to normalize the scores in the rest of the experimental section.

## Tests

The two most significant hyperparameters for the operation of the algorithm are the number of fireflies initialized and the number of iterations. For a simple set-up consisting of 10 desks (with no other furniture or doors), both the number of fireflies and the number of iterations was varied. In each case, the non-varied parameter was held at 10, and alpha and beta were [2, 0.2] and [5, 0.5] respectively. For each hyperparameter test, the final configuration of the room, the value of the objective function for that final value, and the runtime was recorded. The numerical findings are available in Tables 2 and 3. Figure X+1 and X+2 display the final room configuration for the best performing row of Tables 2 and 3 respectively; Figure X is an example of an initial random configuration; the room configurations for the other rows are available in the project’s GitHub, in the Paper/configures directory.

Table 2



Table 3



For 10 desks and 10 iterations, the runtime of the algorithm appears to grow exponentially with the number of fireflies: doubling the fireflies increases the runtime by a factor between 5 and 10. Doubling the fireflies also appears to improve the final solution by approximately 3%; this trend continues until the point at which the runtimes become large enough to render testing impractical. Increasing the number of fireflies corresponds to improving the exploration of the algorithm.

For 10 desks and 10 fireflies, the runtime of the algorithm appears to grow linearly with the number of iterations. Improvement is initially substantial compared to doubling the number of fireflies, 6% rather than 3%. However, increasing the number of iterations corresponds to improving the exploitation of the algorithm, and this improvement tapers off as the number of iterations grows and local minima are found. Indeed, the performance of last row is worse than the preceding row.

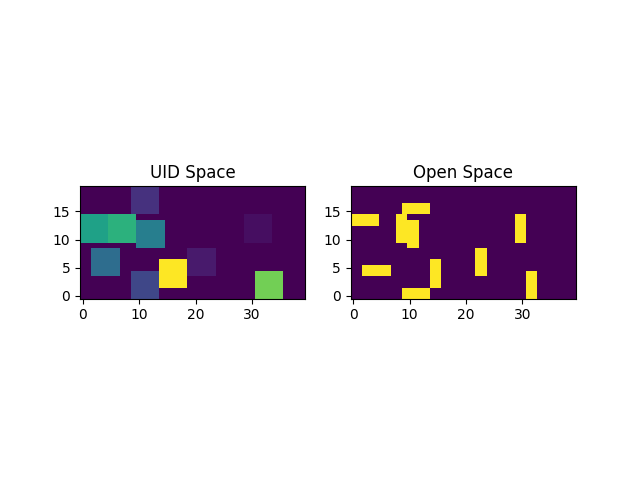


Figure X. Initial random configuration. This figure has been changed.

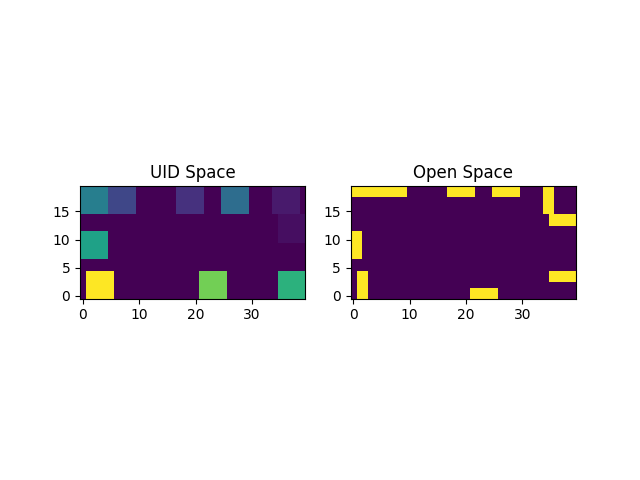


Figure X+1. Final configuration for 10 desks, 160 fireflies, and 10 iterations.

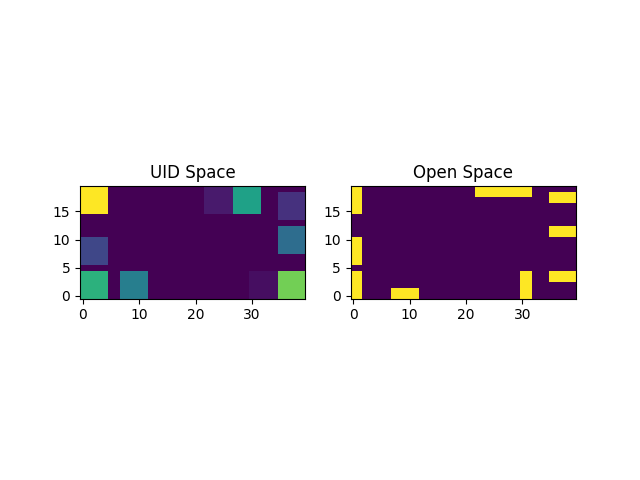


Figure X+2. Final configuration for 10 desks, 10 fireflies, and 640 iterations.

For each figure, UID Space displays the reserved space of every object and assigns each object a distinct color. Open Space displays the true space taken up by each object.

To give the reader a sense of what a change in the score means visually, the normalized score for the random configuration of Figure X is 0.122934184. The improvement between Figure X and Figures X+1 and X+2 corresponds to an approximately 67% increase in the score.

The algorithm adopts and exploits the strategy of moving objects to the walls; for these tests, the number of objects is not large enough for this naïve strategy to break down.

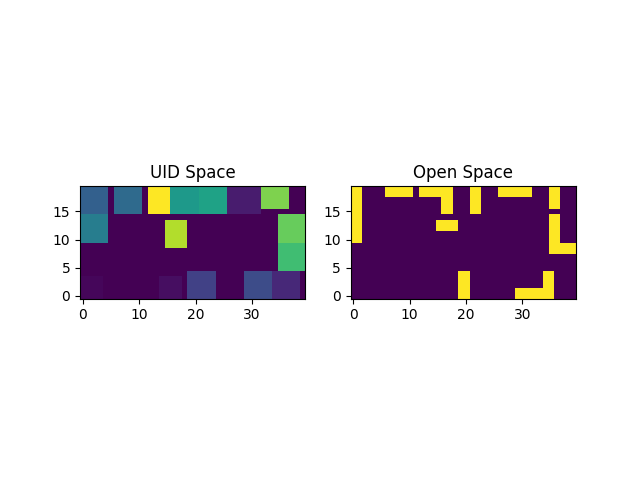
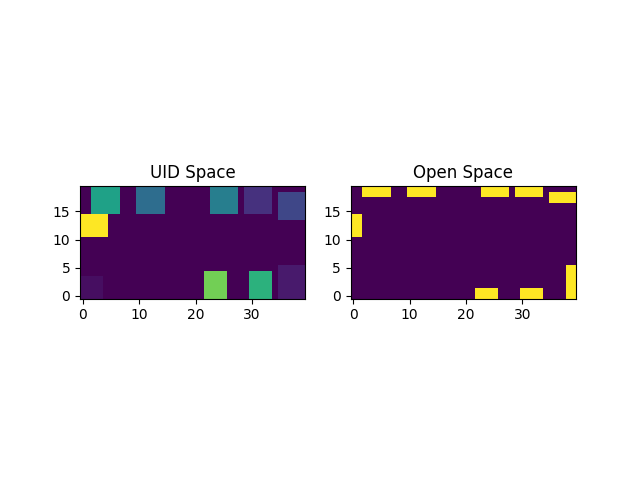
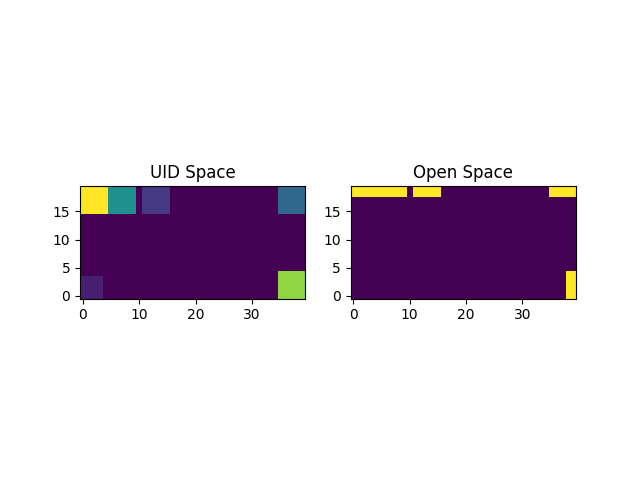
However, there are significant spatial problems present in these solutions: in Figure X+1, the top right corner of the room is cut off; in Figure X+2, the lower right corner is not cut-off, but the route to it is circuitous. More subtly, the algorithm does not appear to always favor rotating objects with their broadside along the wall, which would be considered more optimal by the high-level roundness metric discussed in part III.A.

The next task involves evaluating the performance of the algorithm as the complexity or “busyness” of the room increases. This is more difficult to test objectively than a simple increase in parameters; Table 4 illustrates the strategy adopted in this paper.

Table 4. Each entry indicates how many of its column’s furnishing was present in its row’s complexity test.



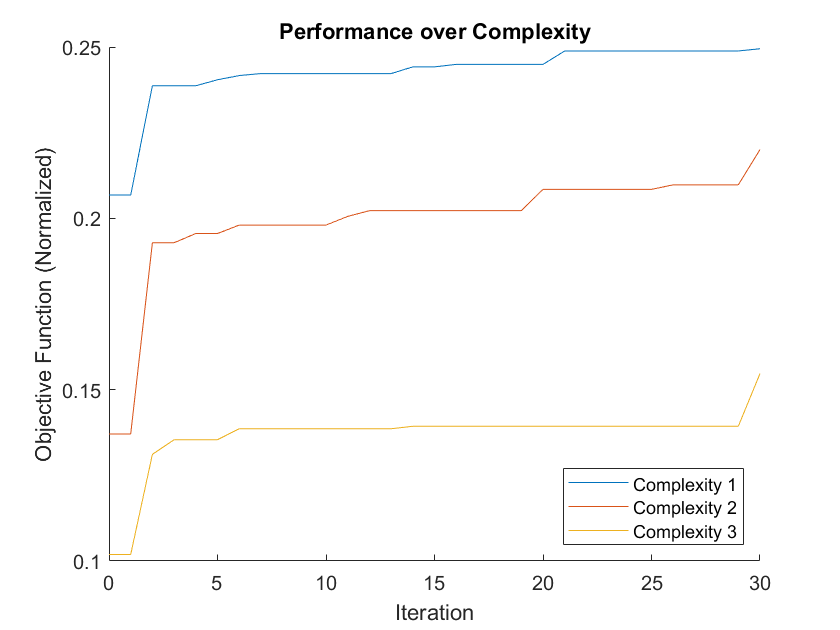
For each complexity test, 20 fireflies were initialized and the algorithm ran for 30 iterations. Figures Y, Y+1, and Y+2 display the final room configuration of each complexity test.



The number of furnishings in Complexity 3 was large enough that the Algorithm abandoned the strategy of simply moving all objects against a wall.

The Cabinet and Shelf columns in Table 4 are conspicuous by the 0s present in all rows, and the number of desks never rises higher than 16. When the complexity of the room rose above the level of Complexity 3, the algorithm tended to stall out in the initialization stage. This is due to the increasing difficulty of finding random configurations which do not suffer from conflicts.

Figure Y+3 visualizes the change in solution quality as the algorithm runs.



A final and more general experiment was performed to assess the algorithm as a whole. The results are summarized in Table 5. The furniture consisted of 10 desks, a couch, and the two doors for these tests.

Table 5.



The expectation is that a combined approach, in which both the number of fireflies and iterations is above 10, should perform better than those tests performed at the beginning of this section. However, the normalized score for the last two columns of Table 5 are significantly lower than those of Tables 2 and 3. This is explained by the increased complexity of the room setup in this set of experiments: the score of an initial randomized configuration for this setup is only 0.107712, considerably lower than that of the original run of experiments.

# Conclusion

In this paper, the problems of furniture arrangement and algorithmic searches were explored; in the intersection of the two, a tool for finding near-optimal room layouts was developed. The Firefly Algorithm (FA) was selected as the central algorithm due to its large degree of exploration (vital for the vast solution space of a room layout), its ability to escape local minima, and its advanced ability to be tuned for specific problems. The FA requires that solutions have a well-defined distance from one another, and an ability to move towards each other with varying step sizes; classes with sophisticated methods were crafted to meet these requirements, and to ensure that the resultant solutions were physically realizable. Finally, the performance and characteristics of the developed system were evaluated.

The inherent subjectivity of the problem makes an overall judgement of the technique difficult. Different readers may reasonably differ in their assessment of the solutions found in the experimental section. Probably no one would deny that there is room for improvement. The authors do not deny this; however, they do contend that the system shows significant promise. Even in its current rough state, the system can find solutions that might not be obvious to an initial inspection—and while those solutions might have obvious flaws, they are still close enough to at least locally optimal solutions that the task of correcting those flaws is fairly simple, rather than intractable.

The current system could be immediately improved upon by fixing its tendency to stall when given particularly full rooms. However, a method for doing this is not trivial: the FA requires random initial solutions in order to explore the search space adequately, but random approaches are inherently tricky in systems in which conflicts arise so easily. One avenue worth exploring would be a physics-inspired approach, drawing on the way buoyant objects distribute themselves when dropped into water.

It would also be useful to evaluate other objective functions, such as the open distance metric first described in section III.A., and other configurations of the hyperparameters alpha and beta. The open distance metric might motivate the algorithm to flatten objects against the wall more effectively.

More complex and comprehensive constraints and object types would also be beneficial, such as a way of preventing non-contiguous and difficult to navigate open space, and objects such as whiteboards or bookshelves which are movable but are held to the walls.

Finally, a more nuanced algorithm could provide better results. Adaptive hyperparameters could be useful, allowing for more rotation when distance between fireflies is small, and for greater exploitation later in the search. One flaw of FA is that the same or similar solutions tend to be explored multiple times, which is wasteful for problems with global information. A hybrid Genetic/Firefly Algorithm could address this concern, particularly one which encouraged speciation.

# Contributions of Team Members

Randall – algorithm development, algorithm implementation

Conor – algorithm development, experimentation