Non-Linear Machine-Learning Based Differential Evolution Optimization of Self-Shaded Curved facades

Mahmoud Mohamed AbdelRahman A0192070M

PhD. Student - National University of Singapore

Abstract

This study aims to improve energy efficiency and thermal performance of a curved faade office building located in Singapore by adopting self-shading building envelope by maximizing shaded area. The optimization process took place by adopting two different approaches. Firstly: irradiance-based optimization approach and secondly: image-processing based optimization approach. Both approaches were conducted using Radiance simulation engine developed by Ward. For the irradiance-based optimization, the irradiance values are measured at different test points mounted near the opening, while for the image-processing optimization, a projection of the facade is captured at each iteration, and then using OpenCV (Computer vision library for python), the ratio between shaded to non-shaded area of the facade was calculated and then maximized. The optimization model that is used in this project is non-linear differential evolution and the optimization variables are the extension length of each room perpendicular to the faade surface. Due to the high noisy of the results, a machine learning algorithm coupled with the differential evolution algorithm is then used to find the global minimum solution among st all the instances. a large data-set of instances was used, that is: 50,000 instances for the irradiance-based optimization and 100,000+ for the image-processing optimization. Results showed that deploying machine-learning alongside with differential evolution proved good results for optimization.

Keywords: Evolutionary computation, Differential Evolution, Machine Learning, self-shading, faade optimization.

Contents

List of Figures

List of Tables

1. Introduction

Solar radiation accounts for the highest portion of cooling loads in buildings, this radiation could be transmitted into the building through the building envelop (exterior walls and glazing) and the roof Chan and Chow [2]. When talking about the envelope thermal transfer (ETT) into buildings, there are four main contributors, namely, building orientation, walls materials, the glazing materials, and the window to wall ratio (WWL). The following equation shows the role of each term in the thermal transfer into buildings through the envelop Chou et al. [3], Chua and Chou [4]:

$$ETTV = 12(1 - WWR)U_W + 3.4(WWR)U_f + WWR(CF)(SC) \quad (1)$$

Where $ETTV$ is the Envelop Thermal Transfer Value, WWR is the window to wall ratio, CF is a correction factor for solar heat gain through fenestration and SC is the shading coefficient which depends on the material of the openings, and the geometry, orientation and shading of the windows. From the previous equation, thermal transfer into building could be in a form of long wave radiations (through walls and windows) or in a form of shortwave radiation i.e. direct solar radiation through openings. In the latter case, shading devices plays the major role in preventing solar radiation from reaching the building envelop or glazing, Thus, by optimizing shading methods, this could significantly reduce cooling loads. For the Singaporean context, the Building and Construction Authority (BCA) indicated that the maximum permissible $ETTV$ value must be 50 W/m2 Singapore [5].

Self-shading envelope has been studied and applied in different buildings with different shapes e.g. the inclined facades and extended facades as shown in Figure.1 which are applied to The City Hall in London and The Diamond Building in Malaysia as shown in Figure 2. However, for the cases where facades are inclined, this could lead to a very wide roof area more than that of the extended facade.

In this project, a self-shaded envelop optimization is carried out for a curved façade using nonlinear programming, namely, differential evolution coupled with machine learning regression model for more efficient prediction.

1.1. Radiance

Developed by Ward and J. [6], Radiance is an extensively validated backward ray tracing and scene radiance-based model. It is considered the most frequently used model among both lighting simulation research and computer

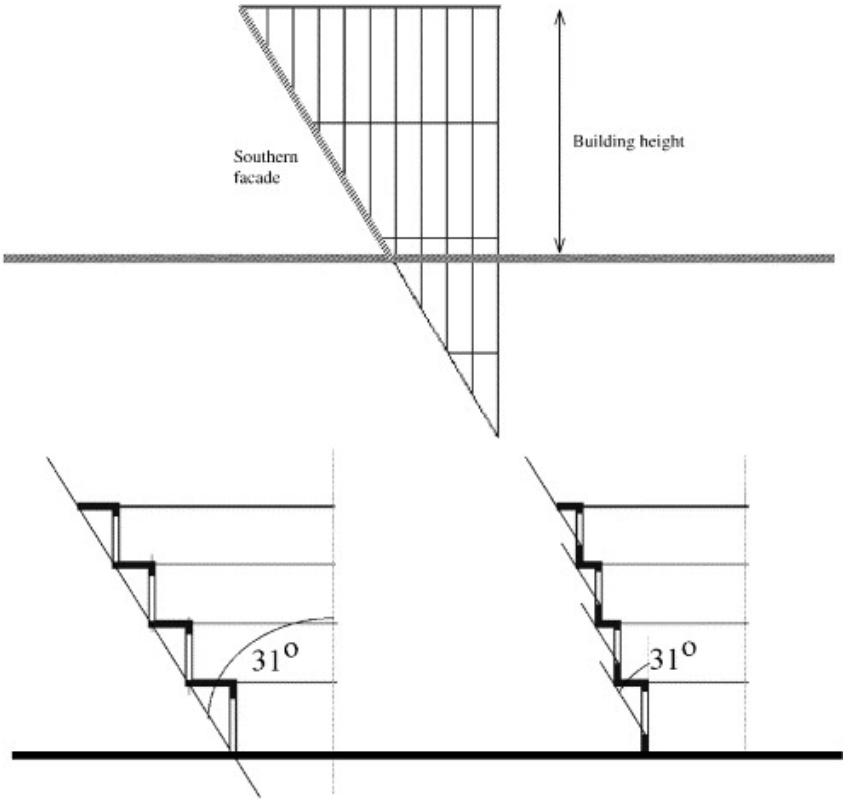


Figure 1: Types of self-shaded envelopes (The inclined facade and the extended facade Capeluto [1])

Figure 2: Self-shading buildings (left: London City Hall and right: Diamond Building in Malaysia)

graphics communities Ochoa et al. [7]. The simulation process takes place after specifying building geometry, objects materials, site parameters (i.e., Latitude and longitude), time of simulation, sky parameters (e.g., overcast vs. clear sky), viewpoints and rendering parameters. It is also added as plugins to many simulation software such as Honeybee and Ladybug tools for Grasshopper Roudsari et al. [8], IES-VE Crawley et al. [9], Reinhart et al. [10], DesignBuilderBaharvand et al. [11] and EcotectMarsh [12].

1.2. Differential Evolution

Differential evolution is a stochastic population-based method that is useful for global non-linear optimization problems by selecting the best cases at each iteration from each offspring and adding random mutation for each generation of solutions and mixing with other candidate solutions to create a trial candidate Chakraborty [13], Storn and Price [14], Storn [15].

To apply this algorithm in the current study, The Python scientific library *SciPy* is used as it has many high efficient scientific optimization models Jones et al. [16]. One of these models is differential evolutio Sci [17] as shown the following listing.

```
scipy.optimize.differential_evolution(func, bounds, args=()\
, strategy='best1bin', maxiter=None, popsize=15, tol=0.01,\
mutation=(0.5, 1), recombination=0.7, seed=None, callback=None,\
disp=False, polish=True, init='latinhypercube')\
```

Where $func$ is the objective function, *bounds* are the limits of random variables for example (from -2.0 to 2.0 meters), *popsize*: is the population size, *maxiter*: is the maximum iterations. In that case, the total number of instances is equal to number of variables * population size * maximum number of iterations.

1.3. Support Vector Machines (Machine Learning Regression Model)

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. In this project, a well-known Python library called scikit-learn developed by Pedregosa et al. [18] is used for training the dataset. The regression model i.e. Support Vector Regression is used for predicting the unknowns based on the trained known dataset, that is why it is referred to as supervised learning. The model takes

a set of data input X and their related output y. Then this data set is fit together to be ready for predicting unknowns with the same type and shape.

Support Vector Machines have many advantages such as:

1. Effective in high dimensional spaces.
2. Still effective in cases where number of dimensions is greater than the number of samples.
3. Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
4. Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

1.4. Limitations of the current study

There are some limitations set in this case-study: 1) the location of the building is in Singapore, this means that the altitude of the sun is very high during the daytime, this also means that the sun direction will be in the north during the period from December to March and it will be in the south during the period from June to September. 2) The current simulation was carried out in the 21st of December at 12:30, that is, the lowest altitude of the sun in the year at the northern façade and 3) Only the northern façade was taken into consideration for design purposes.

2. Methodology

2.1. The optimization work flow

The initial model is represented as a building with curved façade which contains a row of 17 spaces, each space is 2.5 m width and 3.5 m height, and this row is duplicated 6 times to form the case-study as shown in Figure 3. This façade is facing the north direction, this means that it will be exposed to solar radiation during the time from December to March at which the scope of the simulation took place. This model was designed using IES-VE simulation software as it was intended to use the Python API script that exists in this software, however, it took much time for the generation of each instance which caused slow simulation process. Thus, the model was exported to Radiance file (.rad) and then a new code from scratch was developed that can read the base-case, and generate random cases then runs the simulation, then collect the simulation results and finally optimize the model based on each iteration as demonstrated in the following section.

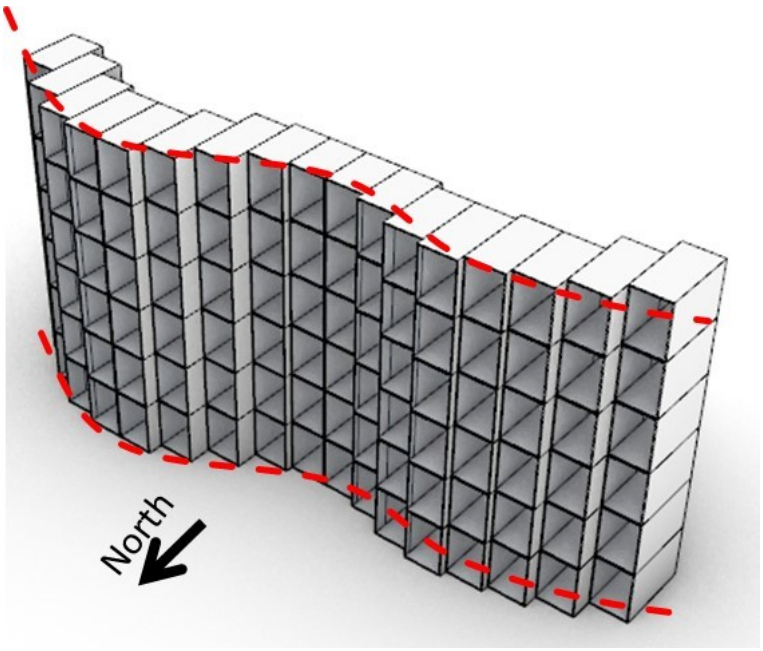


Figure 3: Case-study basic form

2.2. Simulation design

For optimizing the self-shaded façade, each instance of the optimization process was generated randomly by allowing each spaces exterior wall to move forward or backward in a range of [-2.0 m to 2.0 m] Figure 4, this process is repeated for all the 102 spaces that exist on the building's façade. This process is automated by using Python code as shown in Figure 5.

Figure 5 shows the workflow that is used for the optimization process. This workflow is applied to two different processes that will be mentioned in the next section. In this workflow, firstly: the base case is read using python code. The Python code is available on GitHub AbdelRahman 2018[19] This code firstly read the base-case rad file using the function `def readRadFile(path_)`, and break it down into parts e.g. (the exterior walls, the interior walls, glazing, glazing frames ...etc) then it also parametrizes this file in such a way that it allows only the northern faade to be moved but at the same time all the other surfaces still in place using the function `def editRadFile(dataFile, x)`. where x here is the random variables that should be automatically generated by the differential evolution algorithm as shown in Figure 6. Then another function:

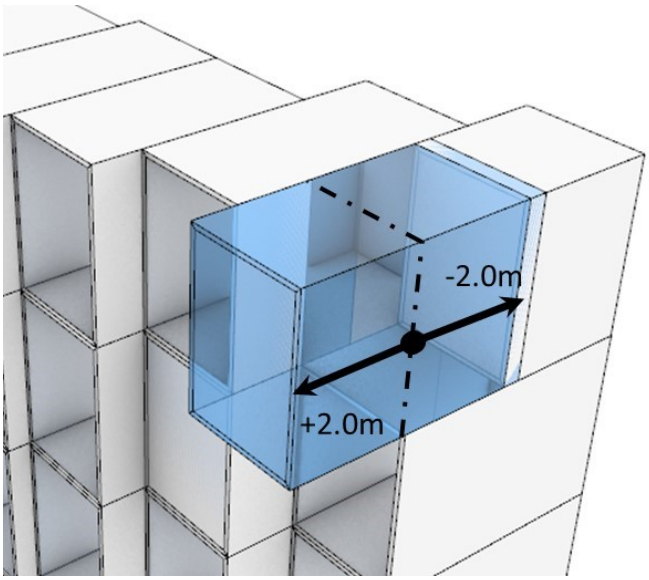


Figure 4: Each space exterior wall could be moved 2.0 forward or backward

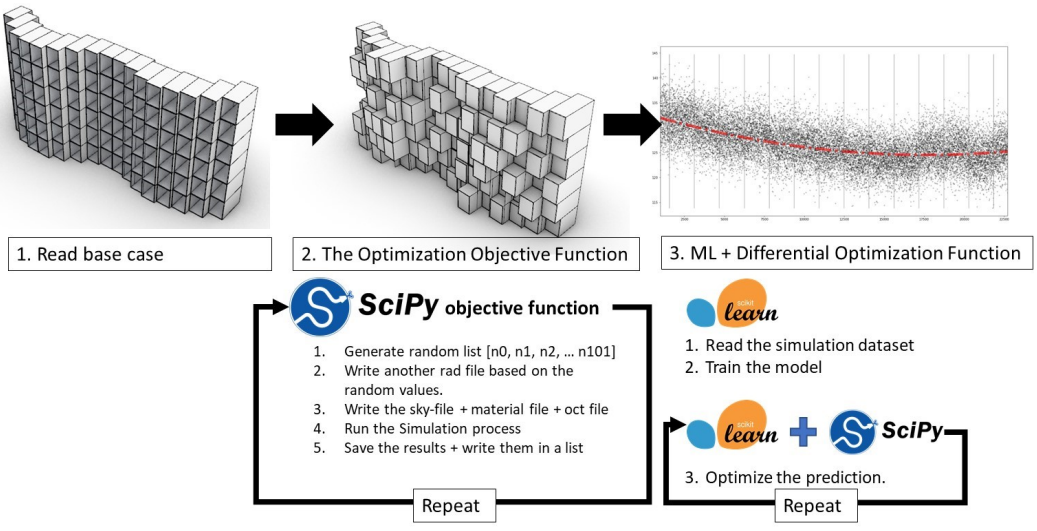


Figure 5: The optimization work-flow

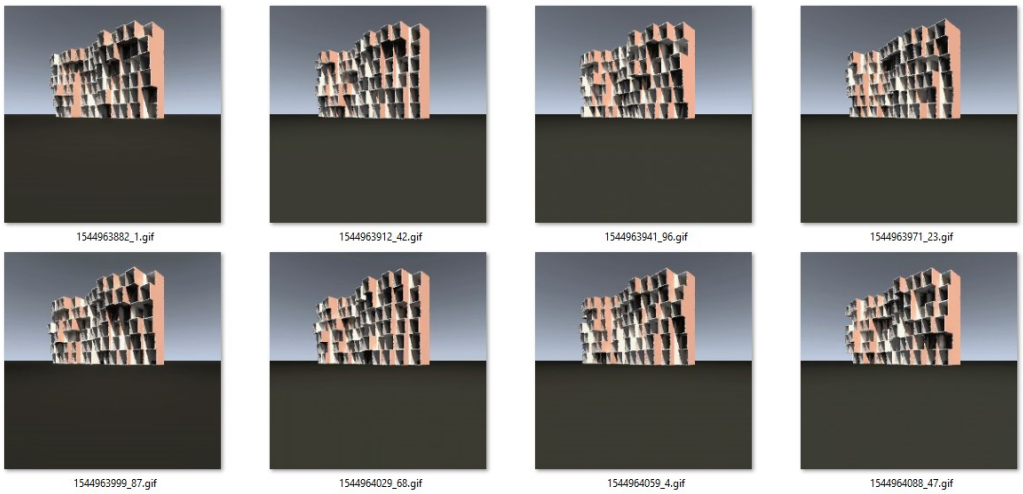


Figure 6: Generated images saved with unique names

`def writeModifiedRadFile(dataFile, x)` creates the new radiance file based on the parametrized random variables, and write it in a specific location. Finally, comes the function `def RunRadiance(filename)` that runs the new radiance file and wait until the simulation process ends, then it writes all the output data in a separate file and render the output images in a separate folder with unique names as shown in Figure 6.

2.3. Two optimization approaches

The aim of this study is to maximize the shadow area of the faade, to achieve this, two main approaches were adopted: 1) Irradiance based approach and 2) Image processing based approach, each of them utilizes the work-flow that is mentioned in Figure 5. However, for the irradiance approach, the irradiance values at different points in each space at height 0.5 from the floor as shown in figure 7-Left have been calculated and then averaged and optimized over time. However, this approach proved inefficiency because it requires more time in simulating the indirect light bounces and requires more accuracy as well. On the other hand, for the image-processing base approach, after the simulation process, an isometric elevation projection of the façade is captured and then converted into white/black colours using Pythons open computer vision module OpenCV. Then the ratio between the black and white colour is measured, this should indicate the shaded/exposed ratio, then the optimization function is working on maximizing this ratio.

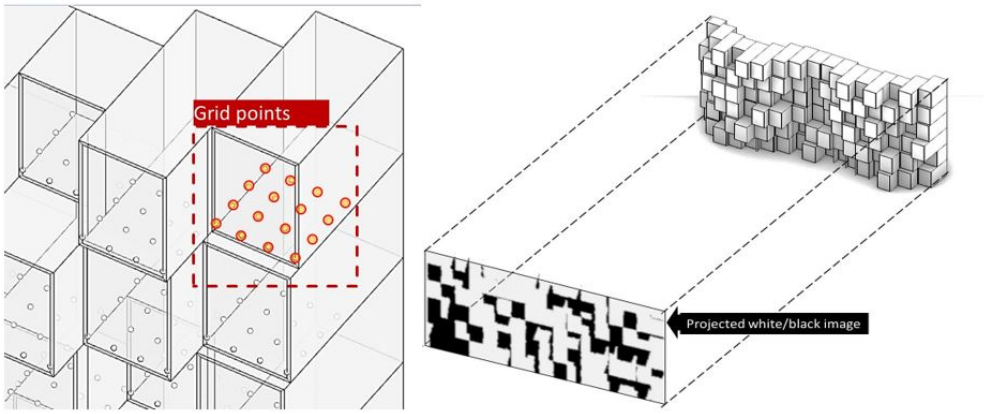


Figure 7: Left: irradiance-based simulation — Right: image processing-based simulation

This approach proved much more efficiency because the indirect bounces are neglected in this case. The two approaches are shown in Figure 7.

2.4. Optimization process

2.4.1. Irradiance-based optimization

In this process, about 22719 random samples were generated and optimized over time, but due to the noisy nature of this type of problems, the convergence criteria never met as shown in Figure 8, In addition, after the 17500 iteration, the researcher suddenly changed some of the radiance parameters such as number of `ambient bounces (ab)` and the `ambient accuracy (aa)` which lead to increase in slight divergence as shown in the highlighted area of figure 8. thus, the machine learning algorithm was used to benefit from those 22719 samples as training dataset, and then accordingly use a hybrid system of differential optimization algorithm alongside with machine learning prediction to come up with an optimum global minimum value after 14000 iterations as shown in Figure 9. The IPyton notebook of this simulation process is available at:

https://github.com/MahmoudAbdelRahman/Scientific-computing/tree/master/self-shading-optimization

2.4.2. Image-Processing-based optimization

In this process, more than 100000 iterations were generated, however, because it is more efficient, it didnt take as much time as the previous simulation process. However, the results also have been very noisy as shown

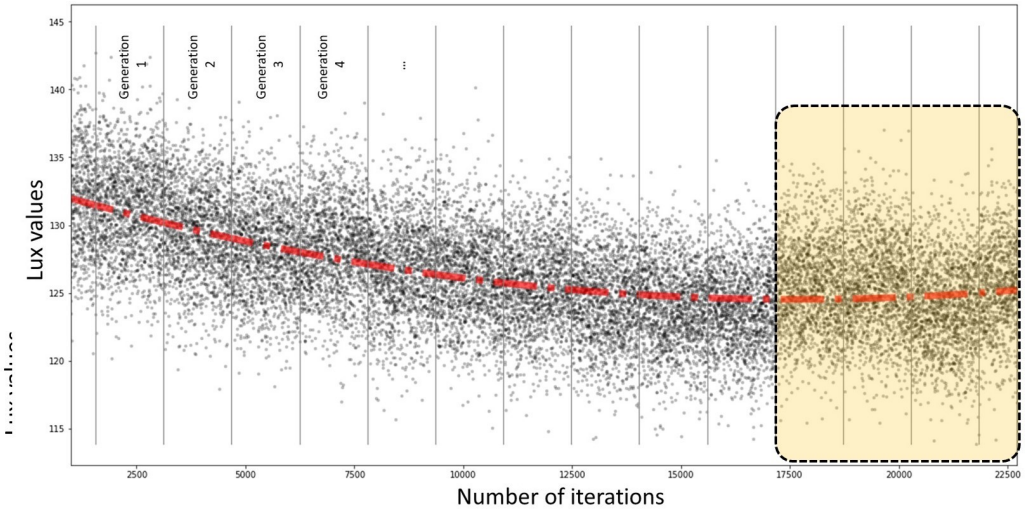


Figure 8: Irradiance based optimization - The highlighted area shows increase in values, this is due to the sudden change in simulation parameters

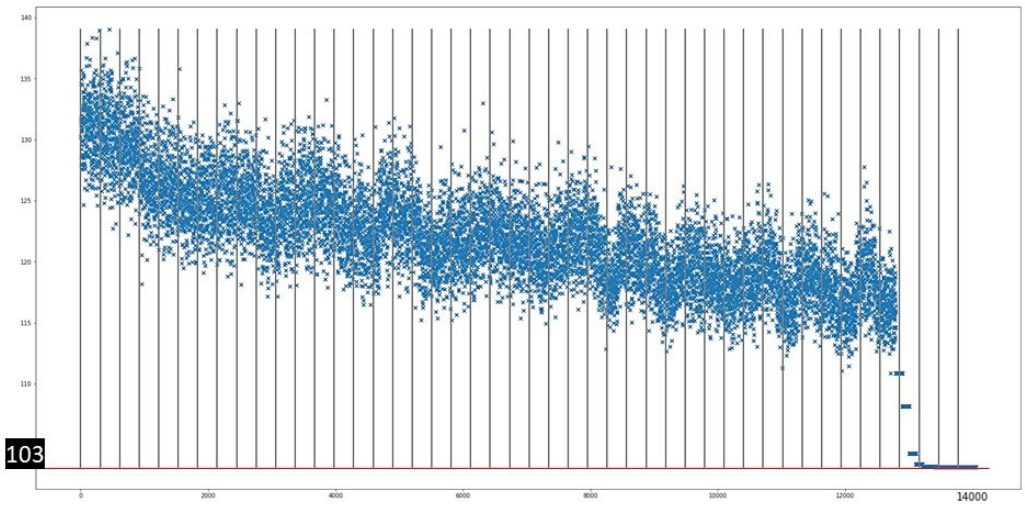


Figure 9: Irradiance machine learning optimization outcome

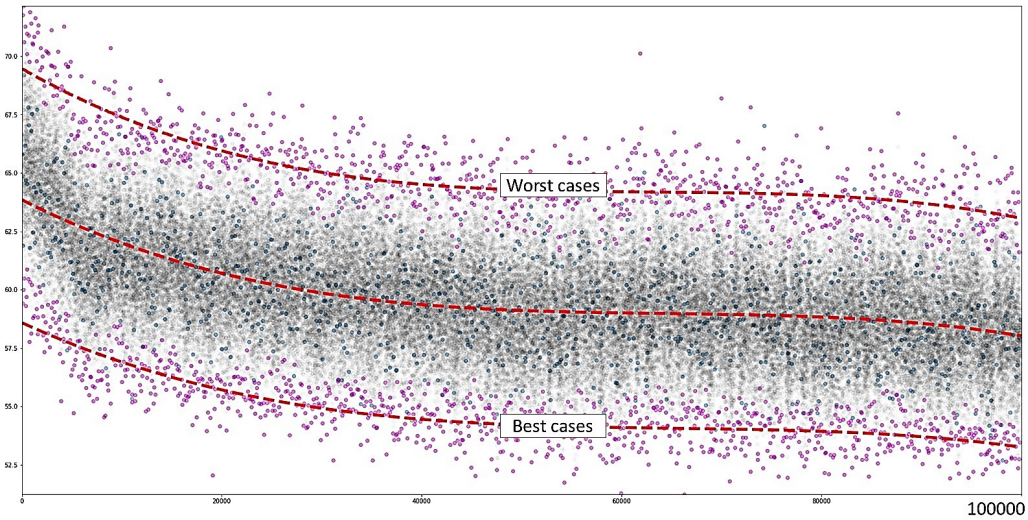


Figure 10: Image based optimization process output.

in Figure 10. For this simulation, the best cases and the worst cases are shown in Figure 11 and Figure 12 with the best and worst cases respectively highlighted and their values are : 48.7% and 27.8% respectively. After this process, a machine learning algorithm coupled with the differential optimization algorithm is also applied to this dataset, it converged after 23832 iterations Figure 13, and the optimal solution was much better than the simulation outcome as shown in Figure 14 and the comparative results are summarized in the next section in Table 1. Results summary.

3. Conclusion

The study in this research focused on optimizing self-shaded façade by randomly distributing blocks of buildings stacked over each other. The optimization algorithm that was used in the study is genetic algorithm namely, differential evolution coupled with Machine Learning algorithm. Two main approaches were adopted, the first one is a grid-point based simulation approach, at which irradiance values are measured at each grid point for each space and then are averaged. Whereas the second approach is an image processing-based optimization wherein a black/white image of the façade is captured at each iteration where the black pixels represent the shadow, and the white pixels represent the exposed areas ,and using OpenCV to count

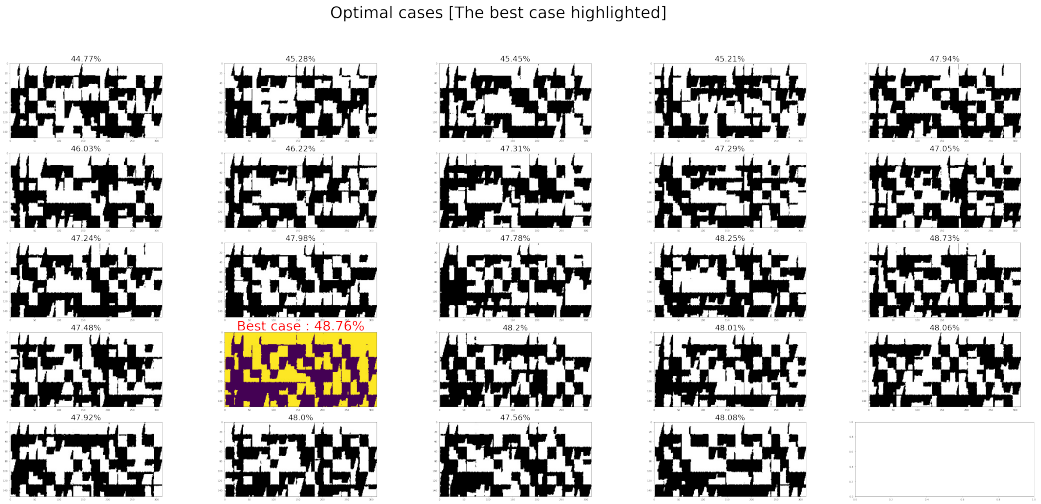


Figure 11: Best cases of the image-based optimization, each shadow/exposed ratio is shown on top of the case with the best case highlighted.

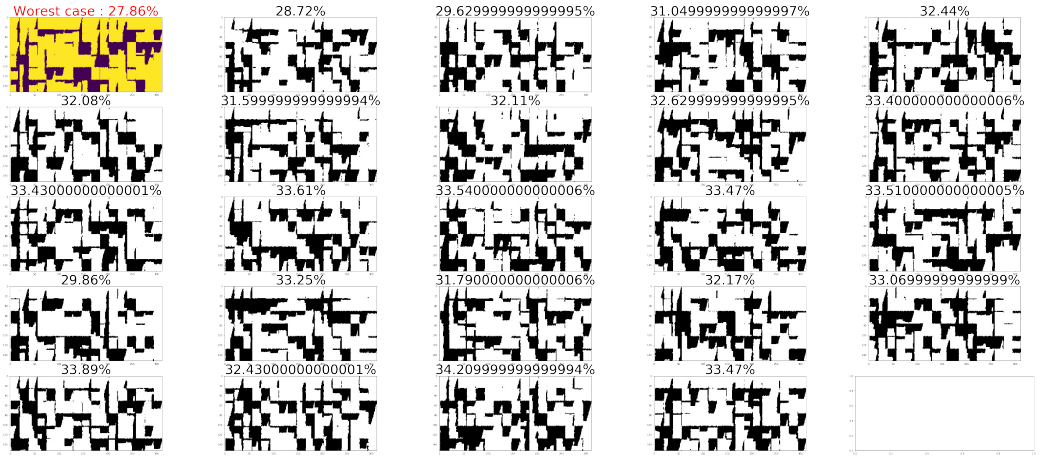


Figure 12: Worst cases from the image-based optimization process, each shadow/exposed ratio is shown on top of the case with the worst case highlighted.

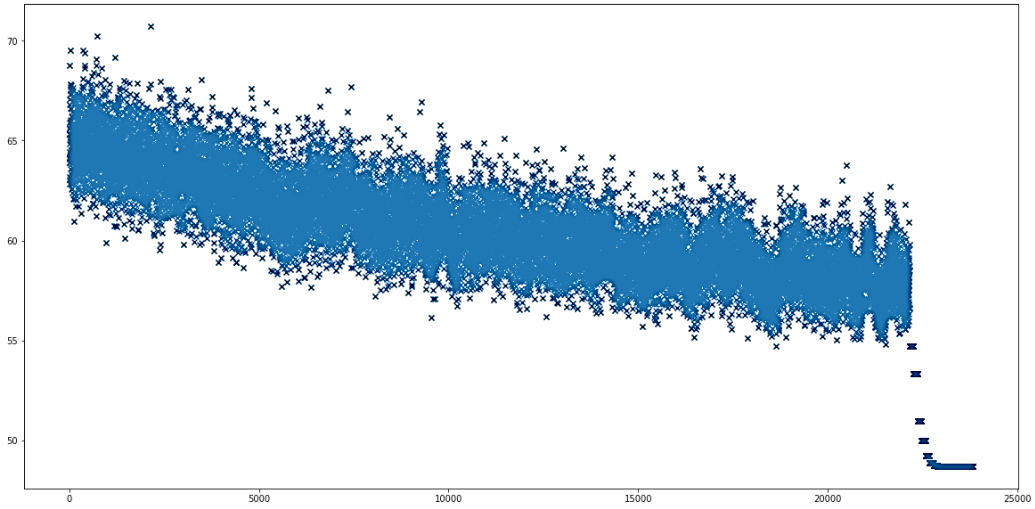


Figure 13: Image-based Machine-Learning optimization outcome.

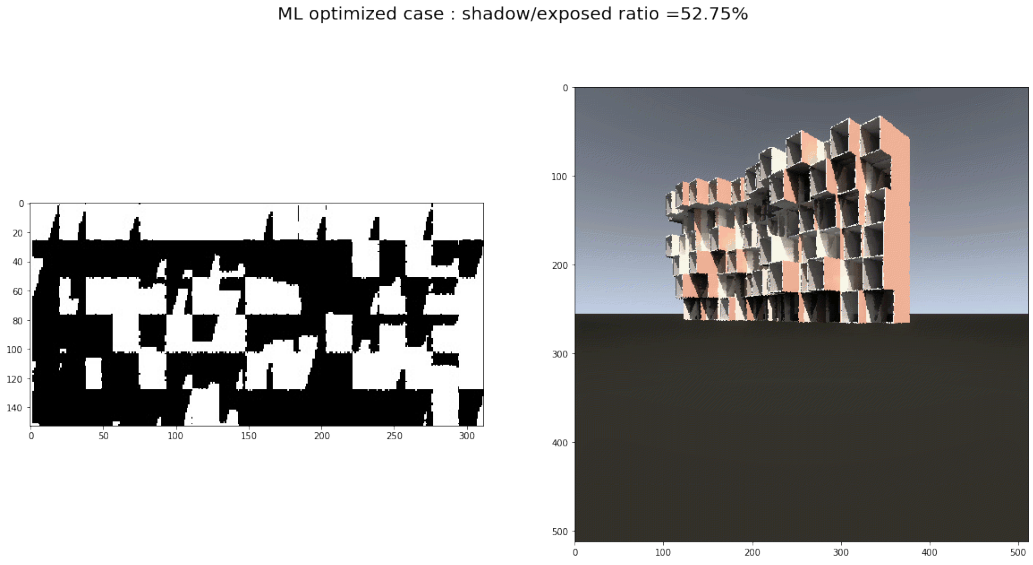
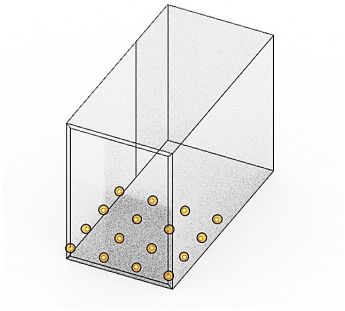
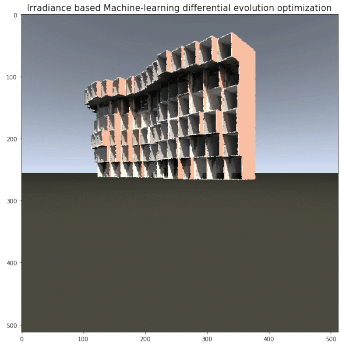
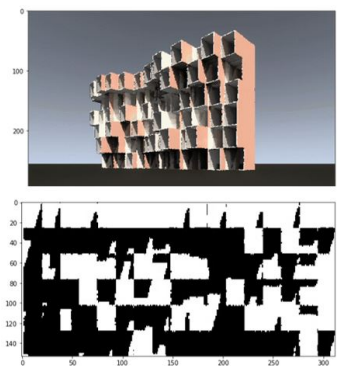


Figure 14: The optimal solution after ML optimization.

Table 1: Result summary

	Irradiance-based optimization	Image-based optimization
Output	Average lux values of all grid points	Facade elevation captured image showing shaded and exposed areas of the envelop
Sample		
Radiance main function	RTRACE	RPICT
Radiance input values	-ab 2 -aa 0.3 -ar 64	-ab 0 -aa 0.0 -ar 64
No. of iterations	22719	100000
Best simulation results	126.4 LUX	48.76 %
Best ML optimization result	103.0 LUX	52.75%
Render		

the black and white pixels and calculate the ratio between shaded/exposed area of the façade, both approaches proved good results, however the image-processing approach showed more efficiency in terms of time and resources wherein this ratio increased from 3% for the initial case up to 52% for the optimized case. Future studies include coupling these results with the cooling loads simulation. Table 1 summarizes the results from both approaches.

4. References

[1] I. G. Capeluto, Energy performance of the self-shading building envelope, Energy and Buildings 35 (2003) 327–336.

[2] A. L. Chan, T. T. Chow, Calculation of overall thermal transfer value (OTTV) for commercial buildings constructed with naturally ventilated double skin façade in subtropical Hong Kong, Energy and Buildings 69 (2014) 14–21.

[3] S. Chou, Y. L. Energy, u. 1988, A simplified overall thermal transfer value equation for building envelopes, Elsevier (????).

[4] K. J. Chua, S. K. Chou, An ETTV-based approach to improving the energy performance of commercial buildings, Energy and Buildings 42 (2010) 491–499.

[5] G. M. Singapore, BCA Green Mark for New Residential Buildings, BCA Green Mark (2010) 0–18.

[6] G. J. Ward, G. J., The RADIANCE lighting simulation and rendering system, in: Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94, ACM Press, New York, New York, USA, 1994, pp. 459–472.

[7] C. E. Ochoa, M. B. Aries, J. L. Hensen, State of the art in lighting simulation for building science: A literature review, Journal of Building Performance Simulation 5 (2012) 209–233.

[8] M. S. Roudsari, M. Pak, A. Smith, Gordon Gill Architecture, Ladybug: a Parametric Environmental Plugin for Grasshopper To Help Designers Create an Environmentally-Conscious Design, 13th Conference of International building Performance Simulation Association (2013) 3129 – 3135.

[9] D. Crawley, J. Hand, M. Kummert, B. G. B. environment, , u. 2008, Contrasting the capabilities of building energy performance simulation programs, Elsevier (????).

[10] C. Reinhart, D. Bourgeois, F. Dubrous, A. Laouadi, P. Lopez, O. Stelescu, DAYLIGHT1-2-3-A STATE-OF-THE-ART DAYLIGHTING/ENERGY ANALYSIS SOFTWARE FOR INITIAL DESIGN INVESTIGATIONS, in: Building Simulation, pp. 1669–1676.

[11] M. Baharvand, A. M. Hamdan, M. R. Abdul, DesignBuilder verification and validation for indoor natural ventilation, Journal of Basic and Applied Scientific Research (JBASR) 3 (2013) 8.

[12] A. Marsh, ECOTECT and EnergyPlus, Building Energy Simulation User News 24 (2003) 2–3.

[13] U. K. Chakraborty, Advances in differential evolution, volume 143, Springer, 2008.

[14] R. Storn, K. Price, Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization 11 (1997) 341–359.

[15] R. Storn, Differential evolution research - Trends and open questions, Studies in Computational Intelligence 143 (2008) 1–31.

[16] E. Jones, T. Oliphant, P. Peterson, ${$SciPy$}$: open source scientific tools for ${$Python$}$ (2014).

[17] scipy.optimize.differential_evolution SciPy v0.17.0 Reference Guide, ????

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research (2011).

[19] M. M. AbdelRahman, Python optimiztion code on GitHub, 2018.