Derivative-free Methods for Structural Optimization

Guilherme Ilunga[1], António Leitão[2]
[1,2]INESC-ID/Instituto Superior Técnico, Universidade de Lisboa
[1,2]{guilherme.ilunga|antonio.menezes.leitao}@tecnico.ulisboa.pt

The focus on efficiency has grown over recent years, and nowadays it is critical that buildings have a good performance regarding different criteria. This need prompts the usage of algorithmic approaches, analysis tools, and optimization algorithms, to find the best performing variation of a design. There are many optimization algorithms and not all of them are adequate for a specific problem. However, Genetic Algorithms are frequently the first and only option, despite being considered last resort algorithms in the mathematical field. This paper discusses methods for structural optimization and applies them on a structural problem. Our tests show that Genetic Algorithms perform poorly, while other algorithms achieve better results. However, they also show that no algorithm is consistently better than the others, which suggests that for structural optimization, several algorithms should be used, instead of simply using Genetic Algorithms.

Keywords: *Derivative-free Optimization, Black-box Optimization, Structural Optimization, Algorithmic Design*

INTRODUCTION

In the common workflow of an architectural project, the architect is responsible for creating a design, which is then given to the engineering team for analysis. The engineers perform their calculations to ensure, for instance, the structural stability of the project. If there is an issue with the design, they inform the architect and make suggestions to resolve it. The architect is then responsible for adapting the design. This process is repeated until there are no more issues to solve.

Usually, architects use Computer-Aided Design (CAD) and Building Information Modelling (BIM) tools to develop the building design, but these tools are unable to properly handle repetitive complex changes, such as those that may be suggested by the engineers. For example, in a design of a space frame which follows a sinusoidal shape, changing the amplitude of the sinusoidal shape should be an easy task. Unfortunately, both types of tools recognize the geometrical aspects of the design, but lack information about the overall concept of a sinusoidal shape. Because of this, if the architect wishes to alter the sinusoidal function, he must change each individual position and size of the bars in the space frame, which is extremely time-consuming. To support these types of changes, nowadays architects may follow the Parametric or Algorithmic Design paradigms.

In the Parametric Design (PD) paradigm, the architect can introduce variability in the design through the usage of different parameters that define it. In the Algorithmic Design (AD) paradigm, the architect constructs an algorithm with the logic of the design, i.e., an algorithm capable of generating

the design, enabling the construction of parametric designs (Terzidis 2006). AD also introduces the ability to generate several variations of the same building with ease, giving the architect creative freedom to explore the space of possible designs, and choose the one he prefers.

Due to the complexity and costs usually associated with buildings, the role of the engineer is not only to ensure that a building follows regulations, but to also assess how well it does that. It is not a good solution to have a building that is structurally sound but that costs much more than what is needed. As a result, one important aspect of the engineer's job is the evaluation of the building's performance regarding different criteria, e.g., structural, lighting, energy consumption, and cost.

To evaluate a building's performance, numerical analysis and simulation-based tools can be used. These types of tools can simulate a building's structural behavior given a specific load using finite element analysis. Similarly, they can compute the useful daylight illumination using raytracing techniques. However, in both cases, these tools need an analytical model with all the required information to perform the analysis, which can be quite different from a geometrical 3D model.

Analytical models can be generated in BIM tools that have that capability, or they may be modeled directly in the analysis tool, or, when using an AD approach, by accessing the Application Programming Interface (API) of the analysis tool. Using the latter method together with a generative process for creating the models makes it possible to guide the generation of the design based on the results of the analysis, following the Performance-based Design (PBD) paradigm (Oxman 2006). Recent approaches give the AD tool the ability to generate different analytical and 3D models from the algorithm provided by the architect, enabling the automatic analyses of different variations of the design (Aguiar et al. 2017), making it possible to find a cheaper design, a design with better lighting or heating conditions, or a design with less structural risks, early in the project's development. Also, this approach enables the usage of mathematical optimization in architecture, i.e., an optimization process can guide the generation of different designs, returning the best design in regard to a set of metrics.

To use the knowledge and ideas from the field of mathematical optimization for the benefit of architectural projects, it is necessary to translate an architectural optimization problem into a mathematical optimization problem, where the metric to optimize is treated as a function to minimize. The parameters of the optimization can be the same that were defined in an AD approach, and the constraints can be defined by imposing bounds on the parameters. As for the function to minimize, unfortunately, in most cases, it does not have a known mathematical expression. To overcome this problem, analysis tools need to resort to simulation techniques, such as raytracing or finite element analysis, to approximate results. Additionally, to minimize such an unknown function, one needs to treat it as a black-box and, for this case, there is a particular optimization technique, called black-box optimization.

With this approach, existing optimization algorithms can be used directly in architectural design, to obtain the best performing design according to specific criteria. Unfortunately, replacing unknown functions with simulation processes entails a performance penalty. This means that black-box optimization processes need to use a reduced budget of function evaluations.

OPTIMIZATION ALGORITHMS

In the field of mathematical optimization, several strategies for optimization have been considered over the years, which has resulted in the creation of several optimization algorithms with different properties, advantages, and disadvantages. Despite the variety, users often choose the simplest approaches because they are the ones have that are more easily available, or because they are easier to understand and implement (Conn et al. 2009). However, for better results, other algorithms should be used.

Optimization algorithms may be classified according to several different properties. One possible way of classifying algorithms is according to their determinism. Some algorithms may be given a starting point, or initial guess, from which they will begin their path towards a solution and, in this case, deterministic algorithms are sequential algorithms that always return the same value, given the same starting point. On the contrary, stochastic algorithms include some form of randomness, i.e., the sequence of steps may be random, therefore they may return different values for the same starting point.

Besides their deterministic or stochastic properties, another way of classifying algorithms is according to their mobility, i.e., if they are global or local. Global optimization algorithms try to find the best solution across the entire solution space, while local ones find the best solution only within a region of that space.

Another classification considers the information used, i.e., if the algorithm takes advantage of information regarding the derivates of the function. Derivative-based algorithms use the partial derivatives of a function, the gradient, to discover the direction of the greatest increase of the function. In the case of black-box functions, the derivatives of the function are unavailable, therefore, for black-box optimization, derivative-free algorithms are the only feasible option.

In their derivative-free optimization textbook, Conn et al. (2009) consider two types of deterministic derivative-free algorithms: direct-search and model-based algorithms. Metaheuristics are only briefly mentioned, despite being derivative-free algorithms and one of the most used and cited type of algorithm (Koziel and Yang 2011; Hare et al. 2013; Rios and Sahinidis 2013; Wortmann et al. 2017). The authors label them "methods of last resort (...) applicable to problems where the search space is necessarily large, complex, or poorly understood (...)". In a recent review of derivative-free algorithms, Rios and Sahinidis (2013) also consider the existence of direct-search and model-based algorithms, although they consider that these methods may be deterministic or stochastic. This paper follows a third approach, previously used in architectural design optimization, which considers three classes of derivative-free optimization algorithms: direct-search, metaheuristics, and model-based algorithms (Wortmann et al. 2017). The following sections describe these classes and provide examples of algorithms. The information about each algorithm, its class, and properties is summarized in Table 1.

Direct-search algorithms

Direct-search algorithms are deterministic derivative-free algorithms, which choose their next action using a set of points that are sampled directly from the function at each iteration. Examples include the Nelder-Mead Simplex, DIRECT, DIRECT-L, PRAXIS, and Subplex algorithms. The Nelder-Mead Simplex (NMS) algorithm (Nelder and Mead 1965) uses a simplex - a generalized polyhedron - over the search space, located around an initial guess, and uses expansion, contraction, and shrinking operations to search for the best result, while sampling the function at each time step. The Dividing Rectangles (DIRECT) algorithm (Jones et al. 1993) is a global optimization algorithm which splits the search space into hyper-rectangles, and samples points in each hyper-rectangle to guide the search. The DIRECT-L algorithm (Gablonsky and Kelly 2001) is similar to DIRECT but it is more biased towards local search. The Principal Axis (PRAXIS) algorithm (Brent 1973) applies line search to each dimension and uses its results to determine a better solution. The Subplex (SBPLX) algorithm (Rowan 1990) selects sub-spaces to explore using the NMS algorithm, in order to find a better solution.

Metaheuristics

Metaheuristics are stochastic algorithms that also do not require derivatives. These algorithms are inspired by aspects of Nature, such as natural selection, evolution, and swarm intelligence. Metaheuristics are popular because they can be applied to almost any problem and are the focus of active research, despite

the fact that they do not offer any convergence guarantees (Conn et al. 2009). Some examples of metaheuristics are Simulated Annealing (Kirkpatrick et al. 1983), Genetic Algorithms (Goldberg 1989), Particle Swarm Optimization (Kennedy and Eberhart 1995), Controlled Random Search (Price 1983), Improved Stochastic Ranking Evolutionary Strategy (Runarsson and Yao 2000), and the ESCH algorithm (Santos 2010). The Simulated Annealing (SA) algorithm skips between neighboring states of a function, according to a temperature parameter. With high temperature, the algorithm may move towards worse states, but as temperature decreases, it becomes increasingly greedy in his search for a local optimum. Genetic Algorithms (GAs) apply the idea of survival of the fittest to a population of candidate solutions. Over time, weaker solutions will be replaced by better solutions, as stochastic genetic operators, such as crossover and mutation, are used to create new offspring. Particle Swarm Optimization (PSO) also contains a population, or swarm, of candidate solutions, and their positions on the search space are updated according to their current position and velocity. Each particle's velocity is updated according to its best-known value and also the swarm's best-known value. The Controlled Random Search (CRS2) algorithm combines random search with simplex approaches and heuristics, such as mutation. The Improved Stochastic Ranking Evolution Strategy (ISRES) algorithm evolves candidate solutions by stochastically ranking and selecting the best solutions. ESCH also uses an evolutionary strategy, but applies different genetic operators using non-uniform probability distributions.

Model-based algorithms

Model-based methods approximate the unknown black-box function and create a high-fidelity surrogate model (Rios and Sahinidis 2013). Over the years, several different strategies for building these models have been identified. Trust-region methods are a local model-based approach, where the model is believed to be accurate within a neighborhood. Examples of trust-region methods are COBYLA (Powell 1994), which builds linear approximations, and BOBYQA (Powell 2009), which creates a quadratic approximation. For global model-based optimization, several approaches have been developed, such as Radial Basis Function (RBF) interpolation (Regis and Shoemaker 2007), RBF-Linear for linear interpolation and RBF-Cubic for cubic interpolation, and Gaussian Processes (GPs), which define a probability distribution over functions (Murphy 2012).

Class	Algorithm	Deterministic	Global
Direct-search	DIRECT	Yes	Yes
	DIRECT-L	Yes	Yes
	NMS	Yes	No
	PRAXIS	Yes	No
	SBPLX	Yes	No
Metaheuristics	CRS2	No	Yes
	ESCH	No	Yes
	GA	No	Yes
	ISRES	No	Yes
	PSO	No	Yes
	SA	No	Yes
Model-based	BOBYQA	Yes	No
	COBYLA	Yes	No
	GP	No	Yes
	RBF	No	Yes

Table 1
Summary of the discussed derivative-free optimization algorithms and their properties.

OPTIMIZATION IN ARCHITECTURE

With the recent advancements in AD and analysis tools, performing optimization on an architectural design is an easier task from the architect's viewpoint. This has led to optimization being an increasingly desired step in the typical design process, especially structural and daylight optimization (Cichocka et al. 2017a). Due to the popularity of AD approaches and the Grasshopper3D tool, several optimization plugins have been developed for it. A large portion of these tools are not new implementations of known optimization algorithms, instead they connect Grasshopper with existing, well-known, and tested optimization software packages.

Some examples of optimization plugins for

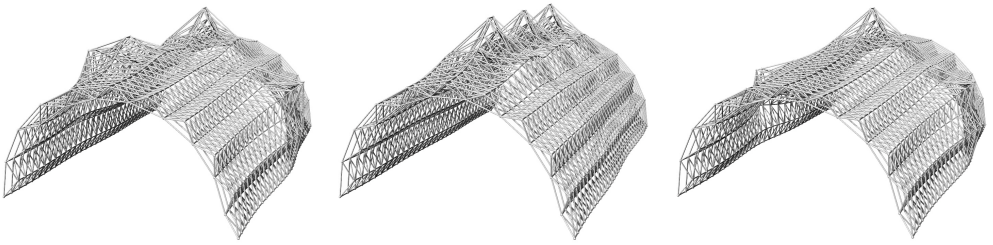

Figure 1
Variations of a space frame with three attractor points.

Grasshopper are Goat ([1] - Flöry no date), Galapagos ([2] - Rutten 2010), Silvereye (Cichocka et al. 2017b), and Opossum (Wortmann 2017). Goat offers several types of algorithms, namely DIRECT, SBPLX, CRS2, COBYLA, and BOBYQA. Silvereye is an implementation of the PSO algorithm. Galapagos is currently included with Grasshopper and offers its users two implementations of metaheuristics - a GA and the SA algorithm. Finally, Opossum connects the RBFOpt library (Costa and Nannicini 2014) to Grasshopper, enabling the usage of RBFs for optimization.

Optimization has been applied more frequently in architecture within recent years. Wortmann (Wortmann et al. 2015; Wortmann and Nannicini 2016; Wortmann et al. 2017; Wortmann and Nannicini 2017) has conducted several studies on black-box optimization methods for architectural design, including daylighting optimization, and building energy optimization. Other studies have also been made for structural optimization: Hare et al. (2013) study derivative-free algorithms and their usage in structural optimization, and Zavala et al. (2014) study the effectiveness of multi-objective metaheuristics.

Regarding the different classes of algorithms, it is usually preferable to use methods with proven convergence properties, which is not the case of metaheuristics. In architectural optimization, GAs are the most used algorithms, but when tested against other algorithms, e.g., in building energy optimization problems, they tend to perform poorly (Wortmann et al. 2017).

CASE STUDY EVALUATION

Our case study is an arc-shaped space frame deformed by three attractor points intended to give the structure a non-uniform shape. Our workflow uses the Rosetta AD tool (Lopes and Leitão 2011) to generate 3D models of truss variations for visualization, as well as the corresponding analytical models for analysis. The analysis is performed using the Autodesk Robot Structural Analysis tool to evaluate the maximum vertical displacement, which the optimization algorithms will attempt to minimize by changing the locations of the attractor points. Figure 1 illustrates three variations of the structure.

For optimization, our workflow enables the usage of existing software packages, namely DEAP, NLopt, and PySOT. DEAP (Fortin et al. 2012) is an evolutionary computation framework, which allows the usage of several algorithms, such as GAs and Evolution Strategies. NLopt ([3] - Johnson 2009) is an open-source package for nonlinear optimization containing several optimization algorithms, both gradient-based as well as derivative-free ones. PySOT ([4] - Eriksson et al. 2015) is an optimization framework for global model-based black-box optimization, providing several different types of approximation models.

For the purposes of this paper, we selected five direct-search methods (DIRECT, DIRECT-L, PRAXIS, NMS, and SBPLX), four metaheuristics (CRS2, ISRES, GA, and ESCH), and five model-based methods (RBF-Linear, RBF-Cubic, GP, COBYLA, and BOBYQA). In total, fourteen different algorithms were tested using their default parameters, simulating a situation where the

architect has little knowledge about the algorithms and their parameters.

For comparing the algorithms, we are interested on answering the following questions:

- How do the results found by the algorithms evolve as the number of function evaluations increases?
- Is there any algorithm or class of algorithms that is consistently better or worse than the others?

To answer these questions, the algorithms were executed for a fixed number of evaluations, and the best result at each iteration was stored. Most algorithms rely on an initial guess to perform their optimization, and the process of choosing this guess was randomized. This is consistent with the typical black-box optimization problem - since the function is unknown, it is hard to propose a good first guess. Due to this randomization, and because some algorithms are inherently stochastic, the optimization results depend on the random seed that is used. Therefore, to properly compare them, the algorithms were executed three times, to analyze the mean best value at each evaluation. By performing these tests, it is possible to properly evaluate and compare the algorithms' convergence capabilities and compare the different categories.

Figure 2 illustrates the mean best result of each algorithm, as a function of the number of evaluations. Of the direct-search methods, DIRECT and DIRECT-L almost achieved the best overall result and converged quickly - DIRECT required thirty evaluations and DIRECT-L thirty-five. The other direct-search methods, PRAXIS, NMS, and SBPLX, performed much worse, and are within the four worst algorithms. For metaheuristics, the common GA was the second worst algorithm, and barely improved after the 10th iteration. ISRES was the best metaheuristic, constantly improving its result until the 57th iteration, and achieving the 6th best result. CRS2 and ESCH only achieved the 9th and 10th best result.

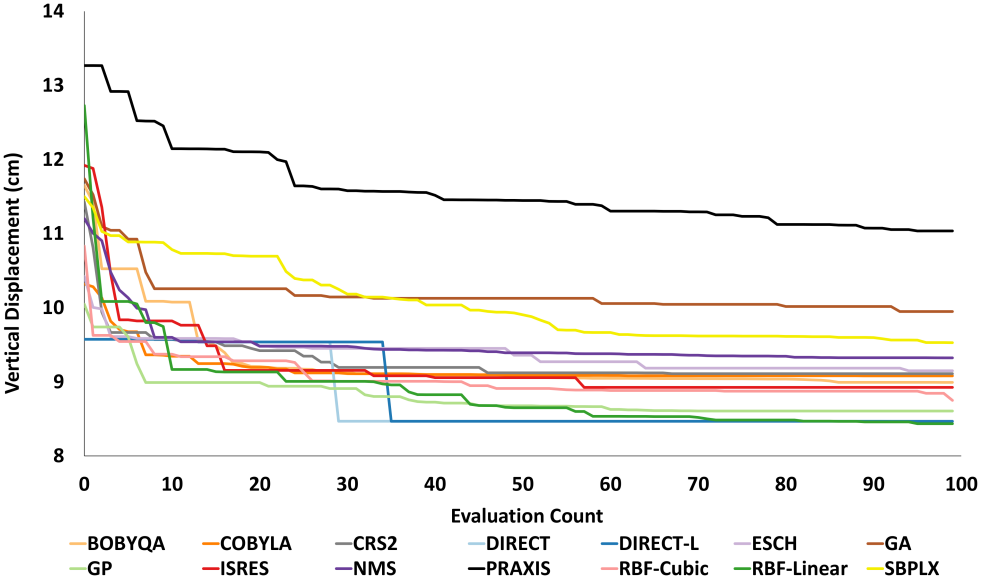

Figure 2
The mean best result of each algorithm, as a function of the number of evaluations, over the course of 100 evaluations. Each algorithm was executed three times, to calculate the mean value.

Finally, the best performing algorithms were model-based. COBYLA and BOBYQA, the local methods, achieved the 7th and 8th best result. The global algorithms, RBF-Linear, RBF-Cubic, and GP, were some of the best algorithms, only matched by DIRECT and DIRECT-L. RBF-Linear achieved the best overall result. Both RBF variants appear to have not stagnated, i.e., if given a bigger evaluation budget, perhaps they could have found even better results.

From the observation of the results at each evaluation, it seems that no algorithm is consistently better than the others. The DIRECT and DIRECT-L algorithms and the global model-based algorithms are better than others after the 40th iteration. Several factors could change this ranking, such as a different set of parameters for the algorithms, i.e., a different population size, a random step taken by a metaheuristic, or a lucky random start for a local algorithm. Therefore, the conclusion is that it is preferable to try different algorithms during the optimization process. Also, running just one algorithm for a long time appears to be wasteful. Instead, allowing two or three algorithms to run for a small amount of time seems to have a higher chance of achieving a good result.

This research also shows that our hypothesis was confirmed: compared to other methods, metaheuristics do not appear to be the best choice for structural optimization, and they are certainly not adequate as the only choice. This was independently confirmed by other studies, e.g., for building energy optimization (Wortmann et al. 2017).

CONCLUSIONS

The increase in processing power of computers has enabled the usage of optimization processes in architecture. AD, along with analysis tools, allows the direct integration of analysis and optimization in the design process, leading to the PBD paradigm. According to recent studies, GAs are the most used algorithms for optimization and, in the case of architectural optimization, they are most of the times the only algorithm used. This happens because they are simple to use, easy to understand, and already available in popular modelling tools. However, similarly to studies in building energy optimization, our research shows that GAs are not the best algorithm for structural optimization. Our results also show that, despite the advantages of other optimization algorithms compared to GAs, no algorithm consistently outperforms the others. Therefore, we suggest that, for structural optimization, several different optimization algorithms should be used, particularly global model-based optimization algorithms.

In the future, we plan to include, in an AD tool, a framework for optimization containing the most relevant optimization algorithms, making it as easy to use a good optimization algorithm as it is, nowadays, to use the GA implementations available in the most used AD tools.

ACKNOWLEDGEMENTS

This work was supported by national funds through Fundação para a Ciência e Tecnologia (FCT) with reference UID/CEC/50021/2013.

REFERENCES

Aguiar, R, Cardoso, C and Leitão, A 2017 'Algorithmic Design and Analysis Fusing Disciplines', *Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*

Brent, RP 1973, *Algorithms for Minimization Without Derivatives*, Dover Publications

Cichocka, J, Browne, W and Rodriguez, E 2017a 'Optimization in the Architectural Practice', *Proceedings of the 22nd International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA)*

Cichocka, J, Migalska, A, Browne, WN and Rodriguez, E 2017b 'SILVEREYE - The Implementation of Particle Swarm Optimization Algorithm in a Design Optimization Tool', *Proceedings of the 17th Computer-Aided Architectural Design Futures Conference*

Conn, AR, Scheinberg, K and Vicente, LN 2009, *Introduction to Derivative-Free Optimization*, Society for Industrial and Applied Mathematics

Costa, A and Nannicini, G 2014 'RBFOpt : an open-source library for black-box optimization with costly function evaluations', *Optimization online 4538*

Fortin, FA, De Rainvile, FM, Gardner, MA, Parizeau, M and Gagné, C 2012, 'DEAP: Evolutionary Algorithms Made Easy', *Journal of Machine Learning Research*, 13, pp. 2171-2175

Gablonsky, JM and Kelley, CT 2001, 'A Locally-Biased form of the DIRECT Algorithm', *Journal of Global Optimization*, 21(1), pp. 27-37

Goldberg, DE 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman

Hare, W, Nutini, J and Tesfamariam, S 2013, 'A survey of non-gradient optimization methods in structural engineering', *Advances in Engineering Software*, 59, pp. 19-28

Jones, DR, Perttunen, CD and Stuckman, BE 1993, 'Lipschitzian Optimization without the Lipschitz Constant', *Journal of Optimization Theory and Applications*, 79(1), pp. 157-181

Kennedy, J and Eberhart, R 1995 'Particle Swarm Optimization', *Proceedings of the 1995 IEEE International Conference on Neural Networks*

Kirkpatrick, S, Gelatt, CD and Vecchi, MP 1983, 'Optimization by Simulated Annealing', *Science*, 220(4598), pp. 671-680

Koziel, S and Yang, XS 2011, *Computational Optimization, Methods and Algorithms*, Springer, Berlin, Heidelberg

Lopes, J and Leitão, A 2011 'Portable Generative Design for CAD Applications', *Proceedings of the 31st Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*

Murphy, KP 2012, *Machine Learning: a Probabilistic Perspective*, The MIT Press, Cambridge, MA

Nelder, JA and Mead, R 1965, 'A Simplex Method for Function Minimization', *The Computer Journal*, 7(4), pp. 308-313

Oxman, R 2006, 'Theory and design in the first digital age', *Design Studies*, 27(3), pp. 229-265

Powell, MJD 1994, 'A direct search optimization method that models the objective and constraint functions by linear interpolation', *Advances in Optimization and Numerical Analysis*, 275(1), pp. 51-67

Powell, MJD 2009 'The BOBYQA algorithm for bound constrained optimization without derivatives', *Department of Applied Mathematics and Theoretical Physics, University of Cambridge*

Price, WL 1983, 'Global optimization by controlled random search', *Journal of Optimization Theory and Applications*, 40(3), pp. 333-348

Regis, RG and Shoemaker, CA 2007, 'A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions', *INFORMS Journal on Computing*, 19, pp. 497-509

Rios, LM and Sahinidis, NV 2013, 'Derivative-free optimization: A review of algorithms and comparison of software implementations', *Journal of Global Optimization*, 56(3), pp. 1247-1293

Rowan, T 1990, *Functional stability analysis of numerical algorithms*, Ph.D. Thesis, University of Texas at Austin

Runarsson, T and Yao, X 2000, 'Stochastic ranking for constrained evolutionary optimization', *IEEE Transactions on Evolutionary Computation*, 4, pp. 284-294

Santos, CHS 2010, *Computação bio-inspirada e paralela para a análise de estruturas metamateriais em microondas e fotônica*, Ph.D. Thesis, University of Campinas

Terzidis, K 2006, *Algorithmic Architecture*, Routledge

Wortmann, T 2017 'Opossum: Introducing and Evaluating a Model-based Optimization Tool for Grasshopper', *Proceedings of the 22nd International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA)*

Wortmann, T, Costa, A, Nannicini, G and Schroepfer, T 2015, 'Advantages of surrogate models for architectural design optimization', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 29(04), pp. 471-481

Wortmann, T and Nannicini, G 2016 'Black-box optimization methods for architectural design', *Proceedings of the 21st International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA)*

Wortmann, T and Nannicini, G 2017, 'Introduction to Architectural Design Optimization', in Karakitsiou, A, Migdalas, A, Rassia, ST and Pardalos, PM (eds) 2017, *City Networks - Planning for Health and Sustainability*, Springer International Publishing, Cham, pp. 1-22

Wortmann, T, Waibel, C, Nannicini, G, Evins, R, Schroepfer, T and Carmeliet, J 2017 'Are Genetic Algorithms Really the Best Choice for Building Energy Optimization?', *Symposium on Simulation for Architecture and Urban Design (SimAUD)*

Zavala, GR, Nebro, AJ, Luna, F and Coello Coello, CA 2014, 'A survey of multi-objective metaheuristics applied to structural optimization', *Structural and Multidisciplinary Optimization*, 49(4), pp. 537-558

[1] https://www.rechenraum.com/en/goat.html

[2] http://www.grasshopper3d.com/profiles/blogs/evolutionary-principles

[3] http://ab-initio.mit.edu/nlopt

[4] https://github.com/dme65/pySOT