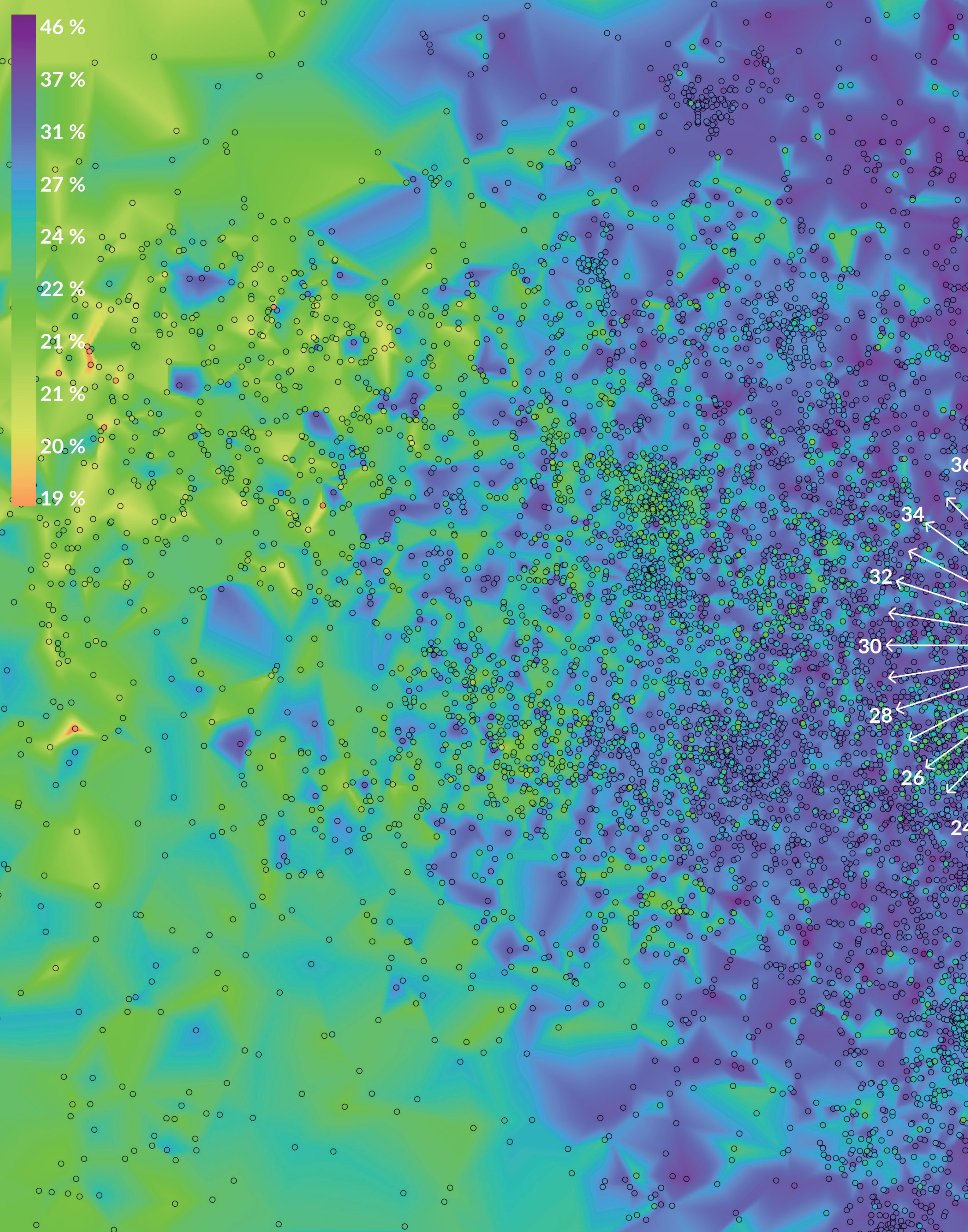
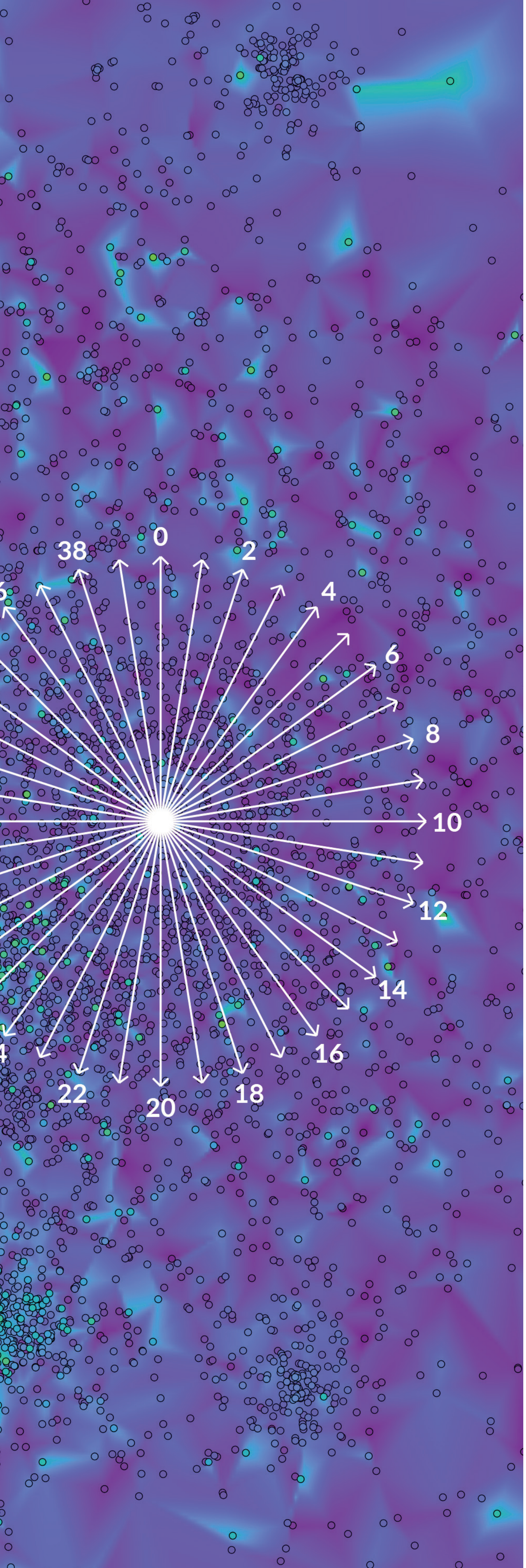
Model-based Optimization for Architectural Design: Optimizing Daylight and Glare in Grasshopper

Thomas Wortmann

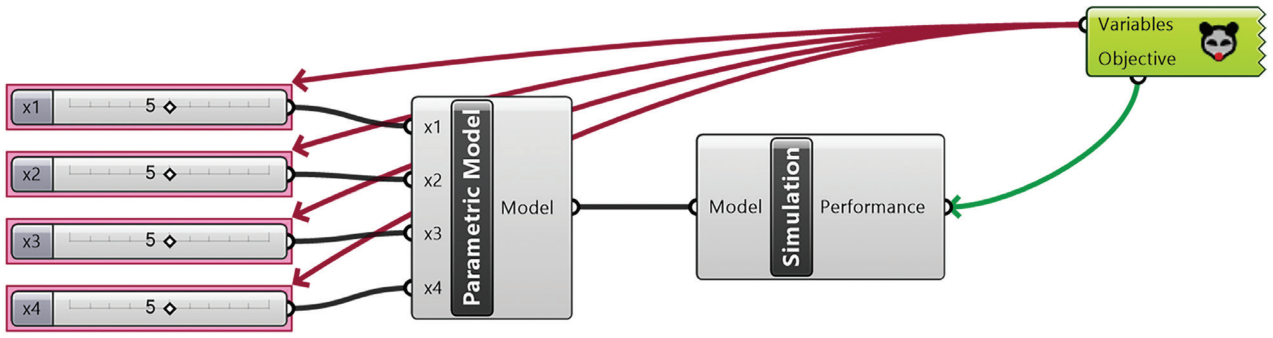

46 %
37 %
31 %
27 %
24 %
22 %
21 %
21 %
20 %
19 %
36
34
32
30
28
26
24



Thomas Wortmann
Singapore University of Technology and Design

Model-based Optimization for Architectural Design: Optimizing Daylight and Glare in Grasshopper

Model-based optimization is an innovative optimization strategy and particularly appropriate for time-intensive performance simulations. To demonstrate this appropriateness, this paper reviews simulation-based optimization algorithms and benchmarks several (single- and multi-objective) optimization tools on two problems involving annual daylight and glare simulations. The benchmarks demonstrate that model-based optimization outperforms other (single- and multi-objective) approaches on time-intensive, simulation-based optimization problems and thus puts new applications within reach. In this way, model-based optimization aids architectural designers and consultants to develop more resource- and energy-efficient buildings.



TAD 1 : 2

◁ Figure 1 (Previous spread). Radial mapping of (most of) the simulated solutions for Problem 2. The position of every circle represents the solution's parameters, and the color its performance in terms of a weighted sum of daylight and glare, on a logarithmic scale. The background colors are interpolated with barycentric coordinates.

The better solutions on the left (<22%) indicate that the screen should be more porous on the top and less porous on the bottom (Figure 9). There are several very good (<20%) solutions that do not form a clear pattern, which indicates several distinct screen designs that are close to optimal.

△ Figure 2. Opossum in Grasshopper. The curves on the left link to the variables and the one on the right to the objective.

▷ Figure 3. Three types of simulation-based, black-box optimization: (a) Optimize the output of the exact simulation directly, (b) optimize the approximating output of the surrogate model constructed from prior simulation results, and (c) optimize and update the surrogate model during the optimization process.

Introduction and Background

Structural, building energy, and daylighting simulations play a key role in architectural design processes. They allow the quantitative evaluation of design variants, while parametric modeling allows the fast and automated generation of design variants from numerical parameters.[1] When designers combine parametric models with performance simulations, optimization algorithms find well-performing design variants. Consequently, simulation-based optimization is increasingly applied in leading architectural and engineering practices such as SOM[2] and ARUP[3] and was, for example, used in the design of the Louvre Abu Dhabi.[4]

Model-based optimization is an innovative, machine learning-related optimization strategy and particularly appropriate for time-intensive performance simulations. To demonstrate this appropriateness, this paper benchmarks several optimization tools in Grasshopper—a popular software for parametric design and performance simulations—on two problems involving annual daylight and, for one problem, glare simulations. These benchmarks include the first freely available, model-based optimization tool aimed at architectural design optimization (ADO), Opossum (OPtimizatiOn Solver with SUrrogate Models), of which the author is the lead developer. (Opossum is available as a free download at www.food4rhino.com/app/opossum-optimization-solver-surrogate-models.)

Model (or surrogate)-based optimization methods find good results with small numbers of simulations.[5] This high speed of convergence is important for sustainable design problems such as daylighting and building energy, where a single simulation takes several minutes or hours to complete. Under such conditions, it is impractical to perform the thousands of simulations required by population-based metaheuristics such as genetic algorithms (GAs).

Global Black-Box Optimization

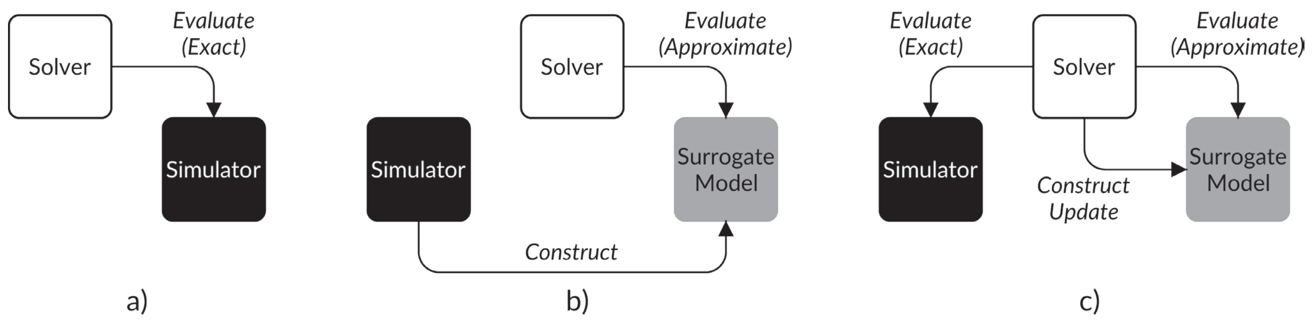
Unlike the majority of optimization methods, global black-box (or derivative-free) methods do not need mathematical formulations of optimization problems and—unlike local optimization methods—do not get trapped in local optima but instead consider the whole design space.

Accordingly, they are appropriate for simulation-based optimization problems in ADO. Such problems define the relationship between variables and performance objectives not with a formula but by evaluating a parametric model with numerical simulations (Figure 2), and often exhibit local optima and complex interdependencies between variables.

Global black-box methods must compromise between exploring the design space as a whole with locally exploiting promising regions. They fall into three broad categories: Direct search, model-based methods, and metaheuristics. The following subsections discuss the three categories, before introducing Pareto-based optimization.

Direct Search

Direct search methods evaluate a deterministic sequence of solutions without trying to approximate the design space (Figure 3a). The well-known Hooke-Jeeves and Nelder-Mead Simplex algorithms are local direct search methods, while DIRECT[6] is a more recent, global method. DIRECT subdivides the design space into hyper-boxes and



excludes hyper-boxes from further subdivision by approximating the maximum change of the performance objective for each box (Figure 4). This paper presents results for DIRECT as implemented in the free, open-source NLopt library.[7]

Model-based Methods
Model-based methods approximate the design space as a whole by constructing surrogate models of the implicit mathematical formulations of simulation-based problems. Surrogate models are much faster to calculate than simulations and can accelerate optimization processes. However, as approximations, surrogate models are less accurate than the underlying simulations.

Accuracy is a concern especially for approaches that completely replace time-intensive simulations with surrogate models and then apply optimization (Figure 3b). Improving the models' accuracy requires a larger sample size, which can negate the initial speed advantage.

In contrast, model-based methods iteratively build and refine models during the optimization process (Figure 3c). At every iteration, a model-based method searches its model for a promising solution (deterministically, randomly, or with a metaheuristic). It simulates the found solution and updates the model with the exact simulation results (Figure 5). A model-based method thus improves its model's accuracy during the optimization process. For the second optimization problem discussed in the following section, the final models constructed by the model-based algorithm deviated at most 53% from the exact objective value, but only 11% on average.

Trust region methods employ local models, while more recent, global model-based methods approximate design spaces globally. Global methods construct surrogate models with a variety of statistical (e.g., Polynomial Regression and Kriging) and machine learning-related (e.g., Radial Basis Functions, Neural Networks and Support Vector Machines) techniques. Due to their ability to model complex design spaces, Kriging and radial basis functions are particularly suitable for simulation-based problems from engineering design[8] and, by extension, ADO. Opossum, the optimization tool presented here, interpolates design spaces with radial basis functions.[9]

Global model-based methods are particularly effective for optimizing problems with time-intensive simulations and complex relationships between variables and objective.[10] They converge—that is, find good solutions—quickly by alternating between evaluating the surrogate model, which is fast but approximate, and the exact simulation, which typically is much slower. The surrogate model not only improves convergence, but offers opportunities for visualization and interaction that are relevant for ADO.[11]

Compared to other types of optimization methods, model-based methods require additional calculations to construct and search the surrogate model at every optimization step. However, when a single function evaluation takes more than a few seconds, the time required for these calculations is negligible.[12]

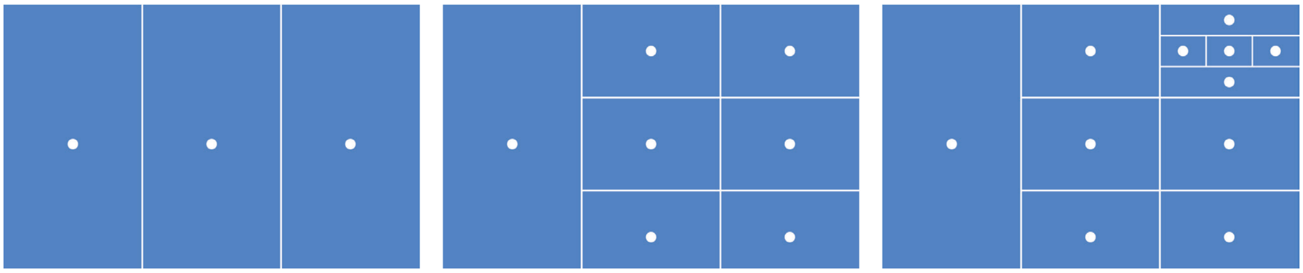
This study considers the performance of RBFOpt,[13] a free and open-source, state-of-the-art library for model-based optimization powering Opossum. Of the twenty-eight solvers competing in the 2015 GECCO Black-Box Competition, which consisted of 1.000 mathematical benchmark problems with two to sixty-four variables, RBFOpt ranked seventh overall and first among the open-source solvers.[14]

Based on previous benchmark results from building energy problems,[15] RBFOpt's continuously updated radial basis function model quickly identifies promising areas of the design space, but, in some cases, hinders the algorithm from focusing on promising areas. In other words, RBFOpt excels in global search but has weaknesses in local search. (RBFOpt v3.0.1, which dates from after the completion of the below experiments, addresses these weaknesses by incorporating a trust region method.)

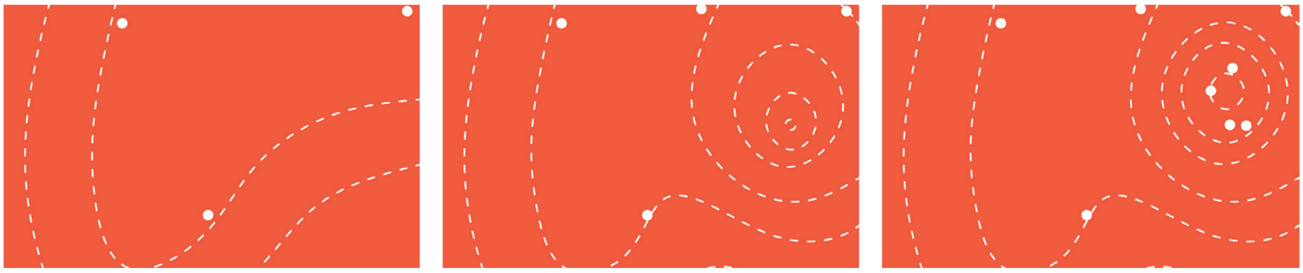
RBFOpt is controlled by more than forty parameters. Depending on individual problem characteristics, some of these parameters drastically affect RBFOpt's performance.[16] The most importance choices are between different optimization (Gutmann[17] and MSRSM[18]) and interpolation methods. (RBFOpt v3.0.1 introduces automatic model selection as a default. With this setting, the algorithm tries to choose the most accurate interpolation method—linear, multi-quadratic, cubic and thin plate spline—for each problem.) Opossum's graphical user interface (GUI) reduces RBFOpt's complexity into three tabs that afford increasing levels of control (Figure 7) and offers presets that are based on intensive testing with mathematical test functions.

Metaheuristics
Stochastic, population-based metaheuristics[19] often are inspired by natural processes, such as genetic evolution or "swarm intelligence," and require extensive tuning of optimization parameters. Due to a lack of mathematical proofs of convergence and inferior performance on benchmarks,[20] the mathematical optimization community regards them as "methods of last resort."[21]

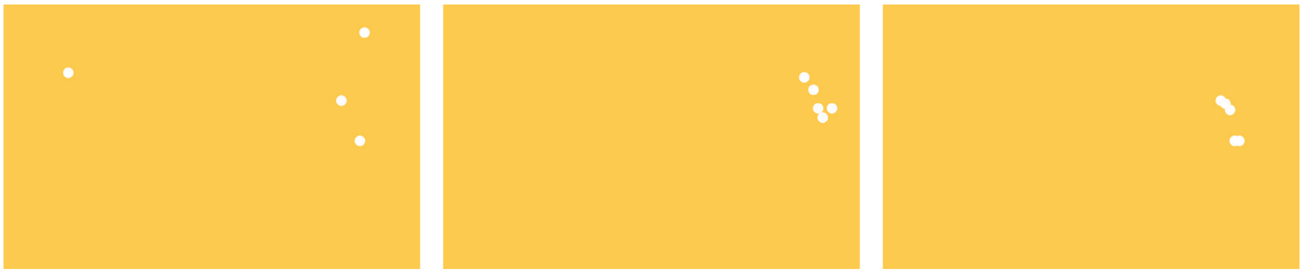
Nevertheless, metaheuristics are by far the most popular category of optimization methods in ADO, with GAs as the most



△ Figure 4. Diagram of three iterations of the DIRECT algorithm. Note how DIRECT subdivides promising regions of the design space while ignoring others.



△ Figure 5. Diagram of three iterations of a global, model-based algorithm. The dashed lines indicated the approximated design space.



△ Figure 6. Diagram of three iterations of a genetic algorithm.

prominent exponent.[22] Reasons for this popularity include a relative ease of implementation, wide availability, applicability to a wide range of problems, and a perception that metaheuristics are especially appropriate for complex, simulation-based problems.[23] This paper presents results for single- and multi-objective GAs, particle swarm optimization (PSO) and simulated annealing (SA).

GAs evolve a population of solutions through genetics-inspired operations (Figure 6), such as crossover, mutation, and selection. Schooling behaviors exhibited by, for example, birds and fish inspire PSO, where a "swarm" of solutions converges gradually in a good region of the space. SA considers only a single solution. Mimicking the movement of an atom in a cooling metal, the solution initially changes more randomly and stabilizes as the "temperature" drops.

Pareto-based Optimization

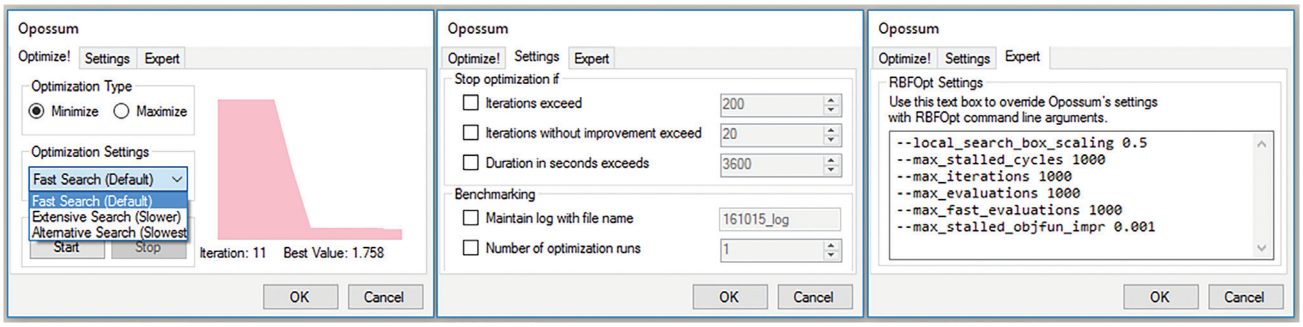
Pareto-based, multi-objective algorithms, which often are GAs, search for solutions that perform well in terms of more than one performance objective. This approach is popular in ADO, but less so in the mathematical optimization community, leading to a comparatively small number of optimization methods and benchmark results.

This paper considers a benchmark problem with the two objectives of daylight and glare, but combines the two into a single objective by subtracting them from each other. Alternatively, one can express multiple objectives as "soft constraints," for example, by penalizing the weight of a structure when it exceeds stress or deflection constraints.[24]

Pareto-based algorithms do not define such weighted or penalized sums, but instead optimize all objectives in parallel. Pareto-based optimization is useful only when there is a tradeoff between objectives: for example, allowing more daylight into a room can lead to more glare. Pareto-based optimization explores such tradeoffs by searching for nondominated solutions, that is, solutions where improving one of the objectives is only possible by worsening others.

However, since Pareto-based optimization aims to not only find good solutions, but a set of nondominated solutions that represent different tradeoffs, it is less efficient than its single-objective counterpart. For example, in a benchmark of a building energy problem with two objectives, it took 1400–1800 function evaluations for the Pareto fronts to stabilize.[25]

The ADO literature does not address the efficiency difference between single- and multi-objective algorithms. For example, one paper characterizes Pareto-based optimization as "getting more for less" and mentions computation time only as a general limitation.[26] Similarly, a foundational paper perceives an affinity between



Optimization Type
(● Minimize) ○ Maximize

Optimization Settings
Fast Search (Default)
Fast Search (Default)
Extensive Search (Slower)
Alternative Search (Slowest

Iteration: 11 Best Value: 1.758

Stop optimization if
☐ Iterations exceed 200
☐ Iterations without improvement exceed 20
☐ Duration in seconds exceeds 3600

Benchmarking
☐ Maintain log with file name 161015_log
☐ Number of optimization runs 1

RBFOpt Settings
Use this text box to override Opossum's settings with RBFOpt command line arguments.

```
--local_search_box_scaling 0.5
--max_stalled_cycles 1000
--max_iterations 1000
--max_evaluations 1000
--max_fast_evaluations 1000
--max_stalled_objfun_impr 0.001
```

△ Figure 7. Opossum's GUI. The first tab lets users choose one of three pre-sets of parameters, and start and stop the optimization. The second tab provides options for benchmarking. The third tab offers full control by accepting command line parameters for RBFOpt.

the many tradeoffs addressed by architectural design and the comparatively smaller number of multiple objectives addressed by Pareto-based optimization, but acknowledges the performance tradeoffs of this approach only in a general sense.[27]

This paper speculates that architects often prefer Pareto-based optimization, because it offers a range of alternatives rather than a single result and not because it accurately represents trade-offs. It evaluates apples as oranges, and vice versa, by evaluating the Pareto-based GA HypE[28] as a single-objective algorithm and comparing the Pareto fronts found by HypE with the fronts found implicitly by the five single-objective algorithms.

Optimizing Daylight and Glare

To evaluate RBFOpt and Opossum, we compare its performance with the remaining global solvers available in Grasshopper: the GA and SA included in Galapagos,[29] the PSO of Silvereye,[30] the DIRECT algorithm included in Goat and HypE included in Octopus.[31] We test the solvers on two problems involving time-intensive daylighting simulations: optimizing a screened façade with discrete louver angles for daylight and optimizing a perforated façade with continuous opening sizes for daylight and glare. We simulate daylight and glare with DIVA-for-Rhino 4.0.[32]

Problem 1 (15 Discrete Variables)

The first optimization problem, defined by Wortmann et al.[33] and presented here with new and expanded benchmark results, considers a single room that is located on the Southwest corner of a building in Singapore (Figure 8).

The room has two facades, one with nine façade components and one with six. Every façade component has micro-louvers, which, for each façade component, can take angles between 0° and 180° (in increments of five). In other words, there are 37 daylight-modulating types of façade components.

The optimization problem aims to find a configuration of façade components that maximizes Useful Daylight Illuminance (UDI). UDI measures the annual percentage of time during which a sensor point receives an amount of daylight that is sufficient for office work, while avoiding glare and excessive heat gains (300–3000 lux). One simulates UDI via a grid of sensor points (Figures 8 and 9).[34]

In addition, we penalize configurations with angle differences larger than 10 degrees between neighboring façade components to ensure a coherent appearance of the façade and a subtle modulation of daylight. The penalty function in the following equation computes a penalty value pi for an individual façade component with angle di. If the angle difference with the previous neighbor is smaller than ten degrees, the penalty pi is zero; otherwise, it is a squared error term. The minimization objective is 100% minus average UDI u(x) plus the sum of the penalties of the fifteen façade components p(x):

$$p_i(x) = \begin{cases} 0 & d_i - d_{i-1} \leq 10^\circ \\ \left(\dfrac{|d_i - d_{i-1}| - 10^\circ}{170^\circ} \right)^2 & d_i - d_{i-1} > 10^\circ \end{cases}$$

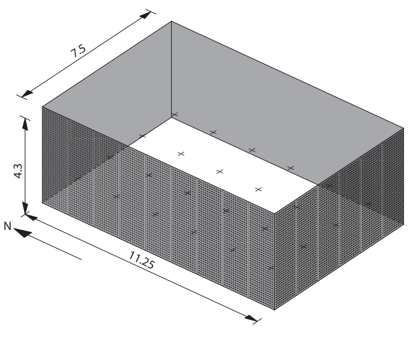
$$\min f(x) = 1.0 - \bar{u}(x) + \sum_{i=0}^{n} p_i(x)$$

On an Intel Xeon E5-1620 CPU with sixteen threads and 3.6 GHz, one evaluation of this objective, that is, generating the parametric geometry and performing the daylighting simulation, takes about 75 seconds.

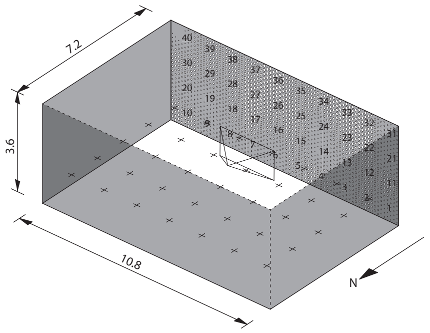
Problem 2 (40 Continuous Variables)

The second problem, defined by Wortmann[35] and presented here with a more rigorous comparison of the Pareto fronts, also considers a room in Singapore (Figure 9). The room has a south-facing façade, perforated by 1.692 openings of varying sizes. The continuous weights of a grid of forty attractor points, which are the problem's variables, controls these sizes and thus the façade's appearance and daylight performance.

This optimization problem aims to find a façade design that maximizes UDI and minimizes Daylight Glare Probability (DGP). DGP measures glare as a percentage for a specific camera view and point-in-time and classifies this percentage as imperceptible, perceptible, disturbing, or intolerable.[36] A recent evaluation of five glare indices concludes that all indices exhibit inaccuracies and inconsistencies. However, the authors state that "DGP shows



△ Figure 8. Diagram of the room optimized in terms of daylight. The crosses indicate the sensor grid for simulating UDI.



△ Figure 9. Diagram of the room optimized in terms of daylight and glare. The crosses indicate the sensor grid for simulating UDI and the cone the camera position and view for simulating DGP.

the best evaluation performance among the five indices" and that DGP "would be still very effective when we would like to find out whether or not discomfort glare exists," which is the case for the problem at hand.[37]

Although a more realistic glare assessment requires several camera views representing the users' true field of view, we calculate this value only for a single camera to hasten the many runs required for benchmarking. To further reduce calculation time, we approximate annual glare as an average of 59 representative daylight hours instead of a full annual glare simulation.[38] Although less accurate than a full annual simulation, this approach yields a good qualitative assessment of the presence or absence of discomfort glare.

Assuming that quality of daylight and avoidance of glare are equally important, subtracting average annual DGP g from average annual UDI u yields a single minimization objective (both UDI and DGP are in the range [0,1]):

$$\min f(x) = (1.0 - u(x) + g(x))/2$$

On an Intel Xeon E5-1620 CPU with sixteen threads and 3.6 GHz, one simulation of UDI and approximated annual glare takes about 90 seconds.

Methodology

We optimize both problems for 200 function evaluations: ten times for each nondeterministic solver (RBFOpt, GA, SA, PSO, and HypE) and one time for the deterministic DIRECT.

Choice of parameters significantly affects the performance of optimization algorithms, especially for metaheuristics, and is problem-dependent.[39] Nevertheless, we assume sensible default parameter choices on part of the solvers' authors. Sensible defaults are important when, as in architectural practice, time pressure does not allow for extensive parameter tuning but calls for solvers that are efficient and immediately usable by nonexperts. Accordingly, we employ default parameters, except for the GA and HypE, where we halve the population sizes to 25 to achieve a larger number of generations.

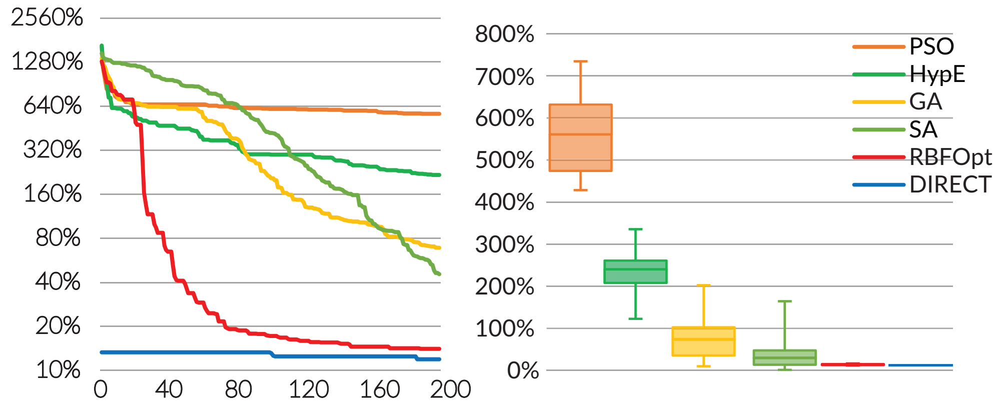
Following standard benchmarking methodology, we evaluate the solvers relative to the number of function evaluations, that is, simulations, and not relative to running time. This methodology ensures the results' independence from computing speed and implementation details. In practice, compared to the time required for simulations, running time differences between solvers often are insignificant.

We employ two evaluation criteria: speed of convergence and stability. Speed of convergence, which is the more critical, measures how fast an algorithm approaches the optimum in terms of function evaluations. But repeated runs of stochastic algorithms for identical problems and settings can have dramatically different results. Stability, which is a concern for metaheuristics and other stochastic algorithms, is a statistical measure for an algorithm's reliability. Stability is important, because, in practice, one should not expect users to run an optimization algorithm more than once.

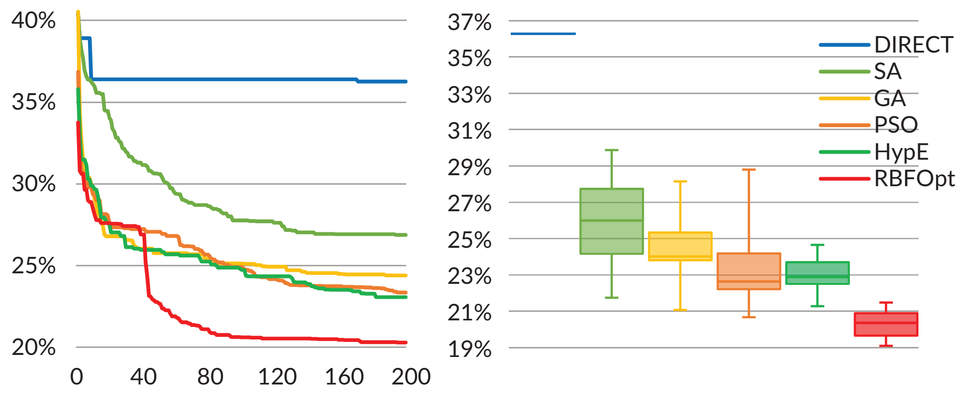
For problem 1, Octopus supplements the single objective with a second objective that aims to diversify the solution set found by HypE. In problem 2, we compare the Pareto-based solver with the single objective ones by calculating the weighted sum objective for the solutions found by HypE. We compare the single-objective solvers with HypE by recording individual UDI and DGP values and calculating the resulting hypervolume. Hypervolume is a quality measure for Pareto fronts calculated as a percentage of the "covered" objective space based on the currently nondominated solutions.[40]

Results

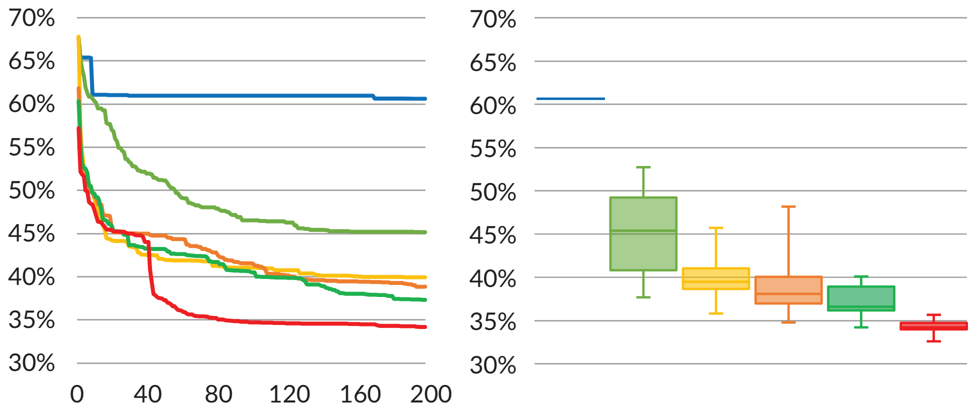
The convergence graph on the left in Figure 10 depicts the average, current best value found by the solvers relative to the number of evaluations on problem 1. DIRECT is the best performing solver, and RBFOpt the second best. DIRECT finds



◁ Figure 10. Benchmark results for problem 1. The convergence graph on the left displays the number of function evaluations on the x, and the average objective value on the y axis. The x axis is in logarithmic scale (base 2) to accommodate big differences between objective values caused by the penalty function. The box plot on the right indicates the range of objective values found by the six solvers in ten runs.



◁ Figure 11. Benchmark results for problem 2 in terms of the weighted objective function.

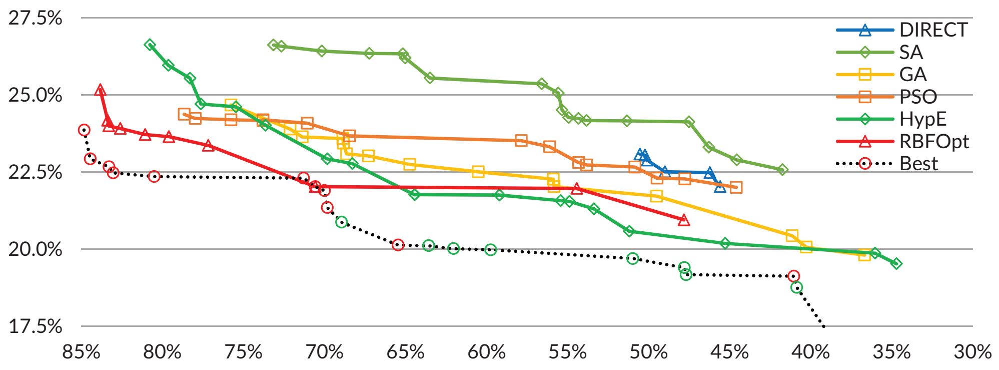


◁ Figure 12. Benchmark results for problem 2 in terms of the hypervolume.

an excellent solution immediately, that is, it "gets lucky" in terms of its initialization sequence. Note RBFOpt's rapid progress after 15 evaluations: Here the algorithm starts profiting from the surrogate model, while the earlier evaluations sample the objective function with a quasi-random Latin Hypercube Design. The number of quasi-random samples is equal to the number of problem dimensions, i.e., variables. The remaining solvers perform poorly. The box plot on the right in Figure 10 indicates the range of objective values found by the solvers in ten runs.

On problem 2, DIRECT is the worst-performing solver because its recursive subdivision proceeds too slowly in the forty dimensions corresponding to the variables (Figure 11). SA performs comparatively poorly, while the remaining metaheuristics, including the Pareto-based HypE, perform similarly and improve the objective by around 40%. Opossum's RBFOpt is the best-performing solver with an improvement of 50%. For both problems, RBFOpt is the most stable nondeterministic algorithm.

Figure 12 graphs not the algorithms' objectives, but their hypervolumes. The results indicate that, on this problem, the single-objective RBFOpt is more stable and finds more accurate Pareto fronts than the Pareto-based HypE. The curves in Figure 12 resembles the ones in Figure 11, since an improvement of the weighted



◁ Figure 13. Pareto fronts found during each solver's most representative (i.e., closest to the mean) run. "Best" is the combined front from all solvers and runs. The markers' color indicates the solver. UDI indicated on the x-, and average DGP indicated on the y-axis.

objective value implies an improvement of the Pareto front.

Figure 13 displays representative examples of Pareto fronts found by the solvers, which further illustrate the performance differences between them. Compared to HypE, RBFOpt has found a shallower front, that is, it has found better tradeoffs between maximizing daylight and minimizing glare, especially for high-daylight solutions. (Note that, although large improvements of daylight quality require only small increases in glare, low average DGP values can contain isolated instances of intolerable glare.)

Conclusion

The performance of optimization algorithms depends on the characteristics of individual optimization problems. Nevertheless, based on the benchmark results presented here and elsewhere[41] and results from the literature,[42] a rough, general recommendation can be made: Designers should apply metaheuristics primarily on problems where thousands, or tens of thousands, function evaluations are possible, direct search on problems with relatively small numbers of variables and model-based methods especially on problems with larger numbers of variables and limited function evaluation budgets. Accordingly, model-based methods are particularly attractive for ADO, where design problems often are complex and involve time-intensive simulations. For the complex and time-intensive daylighting problems presented here, Opossum's RBFOpt is the best choice overall.

The comparison with a Pareto-based algorithm indicates that designers should employ Pareto-based optimization judiciously and only when a large evaluation budget is available to avoid an inaccurate approximation of the Pareto front. Currently, model-based optimization methods are used only rarely in ADO,[43] but results like the ones presented here highlight their potential for optimizing building designs based on time-intensive, structural and environmental simulations.

Nevertheless, this study takes only a small step on the road to more in-depth and rigorous studies of the performance of optimization methods in ADO. Such studies will require a dramatically enlarged catalogue of simulation-based benchmark problems, (single- and multi-objective) optimization methods and results. Another research direction is the improved integration of ADO with architectural design processes, for example through more interactive optimization algorithms and more insightful visualizations of design and objective spaces.

Acknowledgement

The development of RBFOpt and Opossum was supported by the SUTD-MIT International Design Centre (IDG215001100, PIs: Thomas Schroepfer and Giacomo Nannicini). The author thanks the anonymous reviewers, editors and J. Alstan Jakubiec for their detailed and valued feedback.

Notes

1. Robert F. Woodbury, *Elements of Parametric Design* (London; New York: Routledge, 2010).
2. Keith Besserud, Neil Katz, and Alessandro Beghini, "Structural Emergence: Architectural and Structural Design Collaboration at SOM," *Architectural Design* 83, no. 2 (2013): 48–55.
3. Chris Luebkeman and Kristina Shea, "CDO: Computational Design + Optimization in Building Practice," *The Arup Journal*, 2005.
4. Frédéric Imbert et al., "Concurrent Geometric, Structural and Environmental Design: Louvre Abu Dhabi," in *Advances in Architectural Geometry 2012*, ed. Lars Hesselgren et al. (Springer Vienna, 2013), 77–90.
5. Kenneth Holmström, Nils-Hassan Quttineh, and Marcus M. Edvall, "An Adaptive Radial Basis Algorithm (ARBF) for Expensive Black-Box Mixed-Integer Constrained Global Optimization," *Optimization and Engineering* 9, no. 4 (2008): 311–39; Alberto Costa and Giacomo Nannicini, "RBFOpt: An Open-Source Library for Black-Box Optimization with Costly Function Evaluations," Optimization Online (Singapore University of Technology and Design, 2014).
6. Donald R. Jones, Cary D. Perttunen, and Bruce E. Stuckman, "Lipschitzian Optimization without the Lipschitz Constant," *Journal of Optimization Theory and Applications* 79, no. 1 (1993): 157–181.
7. Stephen G. Johnson, *The NLopt Nonlinear-Optimization Package*, 2010, http://ab-initio.mit.edu/nlopt.
8. Alexander I. J. Forrester, András Sóbester, and A. J. Keane, *Engineering Design via Surrogate Modelling: A Practical Guide* (Chichester, UK: J. Wiley, 2008).
9. Rommel G. Regis and Christine A. Shoemaker, "A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions," *INFORMS Journal on Computing* 19, no. 4 (2007): 497–509.

10. Holmström, Quttineh, and Edvall, "An Adaptive Radial Basis Algorithm" (see note 5 above).
11. Thomas Wortmann et al., "Advantages of Surrogate Models for Architectural Design Optimization," *AIEDAM* 29, no. 4 (2015): 471–481.
12. Thomas Wortmann and Giacomo Nannicini, "Black-Box Optimization for Architectural Design: An Overview and Quantitative Comparison of Metaheuristic, Direct Search, and Model-Based Optimization Methods," in *Proceedings of the 21th CAADRIA Conference*, ed. Sheng-Fen Chien et al. (CAADRIA 21, Hong Kong: CAADRIA, 2016), 177–186.
13. Costa and Nannicini, "RBFOpt: An Open-Source Library" (see note 5 above).
14. Ilya Loshchilov and Tobias Glasmacher, "Black-Box Optimization Competition," 2017, bbcomp.ini.rub.de.
15. Thomas Wortmann et al., "Are Genetic Algorithms Really the Best Choice for Building Energy Optimization?," in *Proceedings of the Symposium on Simulation for Architecture & Urban Design* (SimAUD 2017, Toronto, CA, 2017), 51–58.
16. Ibid.
17. H.-M. Gutmann, "A Radial Basis Function Method for Global Optimization," *Journal of Global Optimization* 19, no. 3 (2001): 201–227.
18. Regis and Shoemaker, "A Stochastic Radial Basis" (see note 9 above).
19. El-Ghazali Talbi, *Metaheuristics: From Design to Implementation* (Hoboken, NJ: John Wiley & Sons, 2009).
20. Luis Miguel Rios and Nikolaos V. Sahinidis, "Derivative-Free Optimization: A Review of Algorithms and Comparison of Software Implementations," *Journal of Global Optimization* 56, no. 3 (2013): 1247–1293; Holmström, Quttineh, and Edvall, "An Adaptive Radial Basis Algorithm" (see note 5 above).
21. A. Conn, K. Scheinberg, and L. Vicente, *Introduction to Derivative-Free Optimization*, MOS-SIAM Series on Optimization (Philadelphia, PA: Society for Industrial and Applied Mathematics, 2009), 6.
22. Ralph Evins, "A Review of Computational Optimisation Methods Applied to Sustainable Building Design," *Renewable and Sustainable Energy Reviews* 22 (2013): 230–245.
23. Wortmann et al., "Are Genetic Algorithms Really the Best Choice" (see note 15 above).
24. Wortmann and Nannicini, "Black-Box Optimization" (see note 12 above).
25. Mohamed Hamdy, Anh-Tuan Nguyen, and Jan L. M. Hensen, "A Performance Comparison of Multi-Objective Optimization Algorithms for Solving Nearly-Zero-Energy-Building Design Problems," *Energy and Buildings* 121 (2016): 57–71.
26. Ralph Evins et al., "Multi-Objective Design Optimisation: Getting More for Less," *Proceedings of the ICE - Civil Engineering* 165, no. 5 (2012): 5–10.
27. Anthony D. Radford and John S. Gero, "On Optimization in Computer Aided Architectural Design," *Building and Environment* 15 (1980): 73–80.
28. Johannes Bader and Eckart Zitzler, "HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization," TIK-Report (Zurich, CH: Computer Engineering and Networks Laboratory (TIK) Department of Electrical Engineering Swiss Federal Institute of Technology (ETH), 2008).
29. David Rutten, "Evolutionary Principles Applied to Problem Solving," 2010, www.grasshopper3d.com/profiles/blogs/evolutionary-principles.
30. Judyta Cichocka, Will Browne, and Edgar Rodriguez, "Evolutionary Optimization Processes as Design Tools: Implementation of a Revolutionary Swarm Approach," in *Proceedings of 31th International PLEA Conference* (Bologna, IT, 2015).
31. Robert Vierlinger, "Multi Objective Design Interface" (Msc Thesis, Technische Universität Wien, 2013).
32. J. Alstan Jakubiec and Christoph F. Reinhart, "DIVA 2.0: Integrating Daylight and Thermal Simulations Using Rhinoceros 3D, Daysim and EnergyPlus," in *Proceedings of Building Simulation 2011* (Sydney, AUS: IBPSA, 2011), 2202–2209.
33. Wortmann et al., "Advantages of Surrogate Models" (see note 11).
34. John Mardaljevic et al., "Daylighting, Artificial Lighting and Non-Visual Effects Study for a Residential Building," Velux Technical Report (Loughborough, UK: School of Civil and Building Engineering, Loughborough University, 2012).
35. Thomas Wortmann, "Opossum—Introducing and Evaluating a Model-Based Optimization Tool for Grasshopper," in *Protocols, Flows and Glitches*, ed. Patrick Janssen et al. (CAADRIA 2017, Hong Kong, CN: CAADRIA, 2017), 283–92.
36. Jan Wienold, *Daylight Glare in Offices* (Stuttgart, DE: Fraunhofer IRB Verlag, 2010).
37. Jae Yong Suk, Marc Schiler, and Karen Kensek, "Investigation of Existing Discomfort Glare Indices Using Human Subject Study Data," *Building and Environment* 113 (2017): 121–130.
38. Wortmann, "Opossum—Introducing and Evaluating" (see note 35 above).
39. Talbi, *Metaheuristics* (see note 19 above).
40. Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele, "The Hypervolume Indicator Revisited: On the Design of Pareto-Compliant Indicators Via Weighted Integration," in *EMO 2007*, ed. S Obayashi et al., vol. 4403, Lecture Notes in Computer Science (Springer Berlin Heidelberg, 2007), 862–876.
41. Wortmann et al., "Are Genetic Algorithms Really the Best Choice" (see note 15 above).
42. Holmström, Quttineh, and Edvall, "An Adaptive Radial Basis Algorithm" (see note 5 above); Rios and Sahinidis, "Derivative-Free Optimization" (see note 20 above).
43. Evins, "A Review of Computational Optimisation Methods" (see note 22 above).

Thomas Wortmann is a PhD candidate in the Architecture and Design Pillar at Singapore University of Technology and Design. His research interests are computational design, benchmarking of optimization algorithms, and interactive, visual optimization tools based on surrogate models. Thomas is the lead developer of Opossum, a model-based optimization tool for Grasshopper. Opossum is available for free on www.food4rhino.com.