Journal of Computational Design and Engineering 6 (2019) 414–428

Contents lists available at ScienceDirect

Journal of Computational Design and Engineering

journal homepage: www.elsevier.com/locate/jcde

Genetic evolution vs. function approximation: Benchmarking algorithms for architectural design optimization

Thomas Wortmann

Singapore University of Technology and Design, Singapore

ARTICLE INFO

Article history:
Received 5 April 2018
Received in revised form 19 July 2018
Accepted 11 September 2018
Available online 20 September 2018

Keywords:
Architectural design optimization
Black-box optimization
Benchmarking
Genetic algorithms
Model-based methods

ABSTRACT

This article presents benchmark results from seven simulation-based problems from structural, building energy, and daylight optimization. Growing applications of parametric design and performance simulations in architecture, engineering, and construction allow the harnessing of simulation-based, or black-box, optimization in the search for less resource- and/or energy consuming designs. In architectural design optimization (ADO) practice and research, the most commonly applied black-box algorithms are genetic algorithms or other metaheuristics, to the neglect of more current, global direct search or model-based, methods. Model-based methods construct a surrogate model (i.e., an approximation of a fitness landscape) that they refine during the optimization process. This benchmark compares metaheuristic, direct search, and model-based methods, and concludes that, for the given evaluation budget and problems, the model-based method (RBFOpt) is the most efficient and robust, while the tested genetic algorithms perform poorly. As such, this article challenges the popularity of genetic algorithms in ADO, as well as the practice of using them for one-to-one comparisons to justify algorithmic innovations.

1. Introduction

Growing applications of parametric design and performance simulations in architecture, engineering, and construction increase the relevance of simulation-based, architectural design optimization (ADO) in the search for less resource- and/or energy consuming designs (Wortmann & Nannicini, 2017). Performance simulations play an increasingly larger role in architectural design processes and were for example employed by the designers of the Louvre Abu Dhabi (Imbert et al., 2013).

This article presents benchmark results from seven simulation-based ADO problems—including structural, building energy and daylight optimization—that (1) challenge the popularity of genetic algorithms (GAs) in ADO, (2) demonstrate the efficiency of model-based methods for simulation-based ADO problems, and (3) allow provisional recommendations for ADO practitioners.

Model (or surrogate)-based optimization methods find good results with small numbers of simulations (Costa & Nannicini, 2014; Holmström, 2008). This efficiency is especially important when a single simulation takes several minutes to complete, for example for sustainable design problems such as daylighting and building energy. In such cases, it often is impractical to perform the thousands of simulations required by population-based metaheuristics such as genetic algorithms (GAs). The need for rapid design iterations during concept design stages compounds this impracticality. Arguably, the inefficiency of GAs has been a major challenge to the wider application of ADO (Wortmann, 2018).

2. Benchmarking Black-Box algorithms

This section briefly discusses the optimization algorithms, evaluation criteria, and methodology for this benchmark. The benchmark uses Grasshopper, a popular parametric modeling and simulation platform that can be extended with various optimization plug-ins (Touloupaki & Theodosiou, 2017), and tests all publicly available, global Black-Box algorithms.

2.1. Algorithms

Simulation-based optimization problems define the relationship between variables and performance objectives not with an explicit mathematical function but by evaluating a parametric model with numerical simulations. This relationship often exhibits local optima and complex, non-linear dependencies between

Peer review under responsibility of Society for Computational Design and Engineering.

E-mail address: thomas_wortmann@alumni.sutd.edu.sg

https://doi.org/10.1016/j.jcde.2018.09.001
2288-4300/

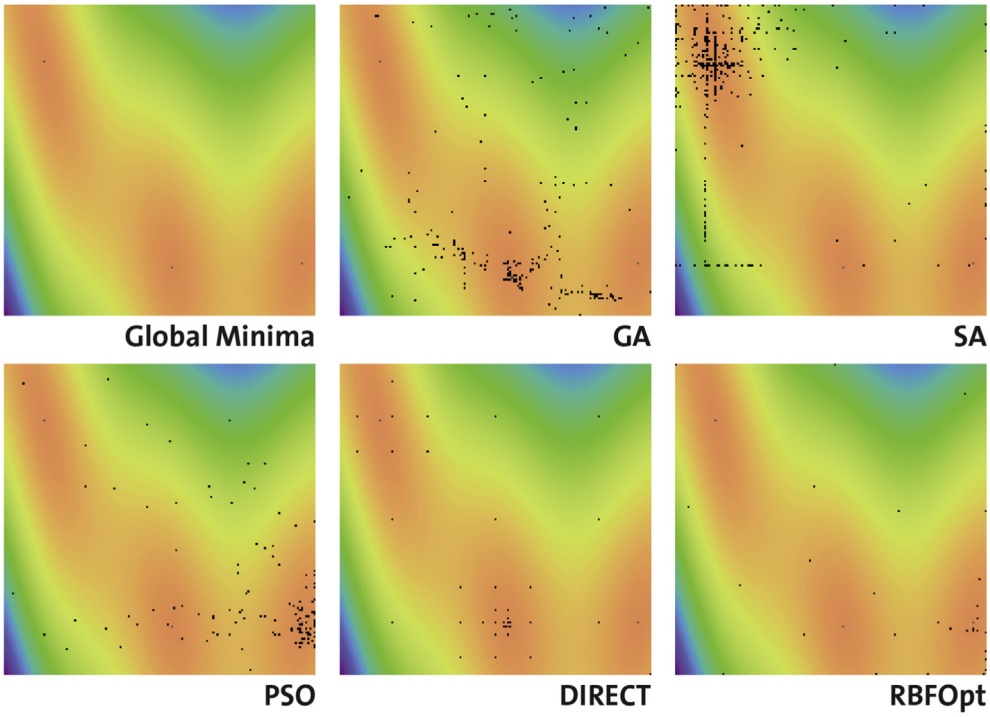

Fig. 1. Behavior of black-box algorithms on the Branin function. This function has two variables (with ranges of [−5.0, 10.0] and [0.0, 15.0) and three global minima (indicated as gray dots). The design candidates evaluated during the algorithms' searches are indicated as black dots.

variables (Attia, Hamdy, O'Brien, & Carlucci, 2013; Evins, 2013). Since global black-box (or derivative-free) optimization methods do not get "trapped" by local optima and do not require explicit mathematical formulations, they are particularly appropriate for simulation-based ADO problems. Following (Wortmann & Nannicini, 2017), this article distinguishes three categories of black-box methods: (1) Direct search, (2) model-based methods, and (3) metaheuristics.

Fig. 1 visualizes the varying behavior of different algorithms from these categories on a mathematical test function. The algorithms optimized this function until they found a solution within 1% of a global minimum. Note that the metaheuristics (GA, SA, and PSO) required more function evaluations than the global direct search (DIRECT) and model-based (RBFOpt) methods.

2.1.1. Direct search

Direct search methods evaluate design candidates in a typically deterministic sequence. The Hooke-Jeeves and Nelder-Mead Simplex algorithms (Nocedal & Wright, 2006) are classic examples of local direct search methods. Subplex (Rowan, 1990) adapts the Nelder-Mead Simplex algorithm for global search by applying it to a sequence of subspaces, i.e., hyper-boxes. DIRECT (Jones, Perttunen, & Stuckman, 1993) is a global direct search algorithm that recursively subdivides the design space into hyper-boxes.

This article benchmarks implementations of DIRECT and Subplex (SBPLEX) from the free, open-source NLopt library (Johnson, 2010), which are available in Grasshopper via the Goat plug-in.

2.1.2. Model-based methods

Model-based methods employ surrogate models (explicit approximations of the implicit mathematical functions, i.e. the unknown fitness landscapes, of black-box problems) to guide the search for good solutions. Trust region methods (Nocedal & Wright, 2006) employ local models, while more recent, global model-based methods model design spaces completely. Global methods construct models with a variety of statistical (e.g., Polynomial Regression and Kriging) and machine learning-related (e.g., Neural Networks and Support Vector Machines) techniques (Koziel, Ciaurri, & Leifsson, 2011). RBFOpt, the model-based algorithm tested in this article, approximates a design space's performance with radial basis functions (Costa & Nannicini, 2014). Radial basis functions and Kriging are particularly suitable for approximating simulation-based problems (Forrester, Sóbester, & Keane, 2008), such as in ADO.

Surrogate models accelerate optimization processes since they are much faster to calculate than the underlying simulations. Approaches that completely replace time-intensive simulations with surrogate models and then apply optimization methods (e.g., Yang, Sun, di Stefano, Turrin, & Sariyildiz, 2016) are limited by the models' initial precision. Increasing the models' precision requires a larger sample size, which can negate the initial speed advantage. Contrastingly, model-based methods iteratively build and refine models during the optimization process (Fig. 2). Model-based methods thus are particularly effective for optimizing problems with time-intensive simulations and complex relationships between variables and objective (Holmström, 2008). This effectiveness, combined with opportunities for visualization and interaction afforded by the surrogate model, makes model-based optimization attractive for ADO (Wortmann, Costa, Nannicini, & Schroepfer, 2015).

In Grasshopper, RBFOpt is available via Opossum, a free plug-in whose development was led by the author (Wortmann, 2017). Opossum is the first established model-based optimization tool aimed at ADO practitioners.[1]

[1] The building optimization literature contains examples of GAs that employ surrogate models that are refined during the optimization process (e.g., Brownlee & Wright, 2015). But these are not "established" model-based algorithms since the selection of design candidates for evaluation does not explicitly consider the predicted improvement of the models' accuracy.

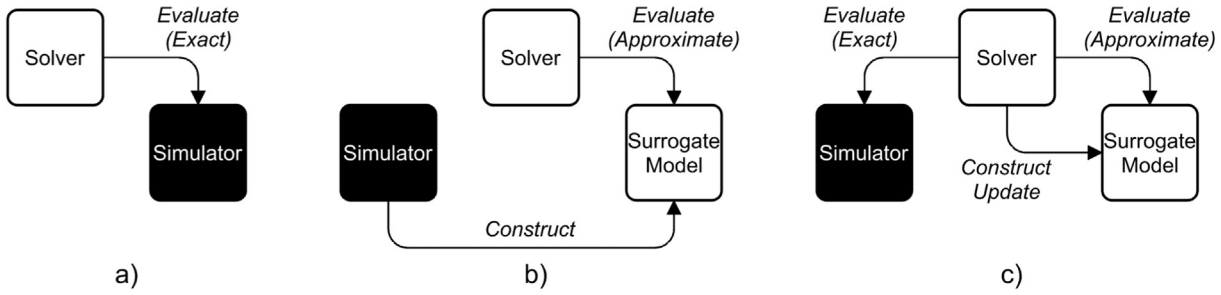

Fig. 2. Three types of simulation-based, black-box optimization: Optimize the output of the exact simulation directly (a), optimize the approximating output of the surrogate model constructed from prior simulation results (b) and optimize and update the surrogate model during the optimization process (c).

2.1.3. Metaheuristics

Unlike direct search and model-based methods, metaheuristics do not rely on mathematical proofs of convergence but draw their inspirations from natural processes, such as genetic evolution or "swarm intelligence." Due to this lack of rigor and often poor performance on benchmarks (Costa & Nannicini, 2014; Rios & Sahinidis, 2013), the mathematical optimization community regards metaheuristics as "methods of last resort" (Conn, Scheinberg, & Vicente, 2009). Nevertheless, metaheuristics are by far the most popular category in ADO, and GAs the most popular algorithm (Evins, 2013). This popularity is due to a relative ease of implementation, a wide availability, and a perception that metaheuristics are especially appropriate for complex black-box problem with multiple optima (e.g., Attia et al., 2013; Evins, 2013).

This article tests implementations of GAs, particle swarm optimization (PSO) and simulated annealing (SA). It also tests CRS2 (Kaelo & Ali, 2006), a metaheuristic that performs randomized global search, but employs direct search for local search, and Covariance Matrix Adaptation Evolution Strategy (CMAES) (Hansen & Ostermeier, 2001), an evolutionary algorithm that samples new design candidates according to an "evolving" normal distribution.

Grasshopper's Galapagos provides a single-objective GA and SA (Rutten, 2013). A PSO implementation is available via the Silvereye plug-in (Cichocka, Migalska, Browne, & Rodriguez, 2017), and CRS2 is included in NLopt, which is available via the Goat plug-in.

This article also tests implementations of CMAES (Hansen, 2018) and the simple GA (SGA) (Waibel, 2018). Wetter and Wright's (2004) recommendation that the SGA is a "good choice" when "a user is willing to accept a slight decrease in accuracy at the benefit of fewer simulations" is widely cited in the building optimization literature as a motivation for using GAs.[2]

2.2. Evaluation criteria

Selection criteria for single-objective algorithms include speed of convergence, algorithmic overhead, stability, and robustness.

2.2.1. Speed of convergence

Most often, speed of convergence (i.e., effectiveness) is the most critical criterion. It measures how fast an optimization algorithm improves the objective value in terms of the number of function evaluations. To benchmark the speed of convergence, one graphs the values of the objective function, or an aggregate thereof, over the number of function evaluations, i.e., simulations (Moré & Wild, 2009).

2.2.2. Algorithmic overhead

Measuring speed of convergence in terms of function evaluations makes it independent of (1) an optimization algorithm's complexity, (2) the speed of particular implementations, and (3) computers. Algorithmic overhead (i.e., computational cost) thus is an additional consideration that is not considered in this benchmark. Many metaheuristics employ relatively simple calculations that require little time per iteration. Model-based methods, by contrast, typically require many more calculations to construct and search surrogate models.

When objective functions require more than a few seconds to evaluate—which is the case for the building energy and daylight simulations in this benchmark—time per iteration quickly becomes negligible, which is why in many practical cases the number of function evaluations is the most relevant performance criterion. This benchmark measures speed of convergence only in terms of the number of function evaluations.

2.2.3. Stability

Direct search methods that are fully deterministic always result in identical outcomes. But many optimization algorithms, including most model-based methods that employ RBFs, combine random and deterministic elements. Optimization algorithms that exploit randomness achieve different results when applied repeatedly to the same problem. In such cases, more stable methods result in a smaller range of outcomes. Stability is an important criterion since an unstable method can sometimes yield unsatisfactory results even if it displays satisfactory performance on average. Due to their heavy reliance on randomness, stability is crucial especially for metaheuristics. This benchmark measures stability in terms of the range of objective values attained by repeated runs of the same algorithm and problem, for a fixed evaluation budget.

2.2.4. Robustness

The criterion of robustness refers to the range of outcomes that optimization algorithms achieve across different problems and/or for different algorithmic parameters. The ADO literature typically considers GAs to be particularly robust (e.g., Attia et al., 2013; Wright & Alajmi, 2005). This benchmark measures robustness in terms of (ranges of) objective values achieved over different problems.

2.3. Methodology

The function evaluation budget and algorithmic parameters are important methodological choices for measuring the criteria of speed of convergence and stability.

2.3.1. Function evaluation budget

This benchmark considers the objective values from twenty runs on each problem.[3] To allow comparisons across all problems,

[2] Cited 241 times, according to Google Scholar on 26.3.2018.

[3] Since DIRECT is deterministic, a single run suffices for it.

the evaluation budget is identical for all problems (500 function evaluations, i.e., simulations). Wetter and Wright (2004) employ the same number in their widely-cited building energy benchmark.

This budget represents a compromise between practical situations where only a small number of function evaluations is possible and ADO benchmarks, which—perhaps reflecting the relatively slow speed of convergence of the employed metaheuristic and local direct search methods—sometimes employ (tens of) thousands of functions evaluations (e.g., Hasançebi, Çarbaş, Doğan, Erdal, & Saka, 2009).

2.3.2. Algorithmic parameters

Choice of parameters significantly affects the performance of optimization algorithms, especially for metaheuristics (Talbi, 2009). Ideally, algorithmic parameters should be "tuned" to individual problem characteristics. Nevertheless, we assume that the algorithm's authors chose sensible default parameters. Sensible defaults are important when, as in architectural practice, time pressure does not allow for extensive parameter tuning but calls for algorithms that are immediately efficient and usable by non-experts. Accordingly, the benchmark employs default parameters, but reduces the population sizes of GA and HypE to 25 to achieve a larger number of generations, i.e., a larger number of optimization steps.[4] The SGA employs the parameters recommended by (Wetter & Wright, 2004).

3. Benchmark problems

This section presents the benchmark problems employed in this article. They include structural, building energy, or daylighting simulations and have four to forty variables, see Table 1. Structural problems 1 and 2 are adapted from (Wortmann & Nannicini, 2016), building energy problems 3, 4, and 5 from (Wortmann et al., 2017), daylight problem 6 from (Wortmann et al., 2015), and daylight problem 7 from (Wortmann, 2017). This article combines new and extended benchmark results for these problems and provides a more generalizable overview of the algorithms' performance.

3.1. Problem 1: Structural, 8 discrete variables

The first benchmark problem, which is a standard problem in the structural optimization literature (Hasançebi et al., 2009), concerns a small—5.08 m tall—transmission tower with 25 structural members belonging to one of eight beam sets (Fig. 3). Each beam set can be assigned one of 30 pre-selected, circular cross sections.

The optimization problem consists in finding an assignment of cross sections to the eight beam sets that minimizes the weight of the tower while meeting stress and displacement constraints under horizontal and vertical point loads. In other words, the design problem is to find the lightest selection of cross sections that is structurally feasible for a fixed geometry and load.

As is typical for black-box optimization problems, the objective function includes "soft constraints." Such a problem does not formulate constraints explicitly but includes constraint violations as penalties in the objective function. For the objective function $f(x)$ indicated below, $p(x)$ are the penalty parameters for constraints of the form $c(x) \leq cmax$. The penalty weight w_p is 200 kg.

$$p(x) = \begin{cases} 0 & c(x) \leq c_{max} \\ \frac{c(x)}{c_{max}} - 1 & c(x) > c_{max} \end{cases}$$

$$f(x) = w(x) + w_p * \sqrt{p_1(x) + p_2(x) + p_3(x)}$$

$p_1(x)$ corresponds to maximum stress, $p_2(x)$ to maximum displacement in the x-axis, and $p_3(x)$ to maximum displacement in the y-axis. For a given set of cross sections, the Grasshopper plug-in Karamba did a first-order analysis of the stresses and displacements for problems 1 and 2.

3.2. Problem 2: Structural, 22 discrete variables

The second structural design problem, also drawn from (Hasançebi et al., 2009), involves the assignment of cross sections to the 22 beam sets of a braced dome truss (Fig. 4). The dome has a diameter of 40 m and a height of 8.28 m. This problem is a more complex variant of the previous problem and considers displacements and stresses for three different load cases (Dead load and snow (1) without wind, (2) with negative wind pressure, and (3) with positive wind pressure).

The objective is to minimize the weight of the dome, subject to the two "soft" constraints of displacement (along all axes) and stress utilization of the structural members. Accordingly, the objective function is the same as for Problem 1, but with two instead of three constraints, and w_p as 20,000 kg.

3.3. Problem 3: Building energy, 4 continuous variables

Building energy problems 3, 4, and 5 originate from the seminal benchmark by Wetter and Wright (2004). Problem 3 concerns the annual energy consumption of an office building in Seattle with four variables: building orientation α in degrees, window width for the West and East façades w_W and w_E, and the shading device transmittance τ. For simplicity, the simulation only calculates two zones, one east-facing and one west-facing (Fig. 5a).

The objective function to be minimized is annual energy consumption in kWh/m^2 a, calculated as the sum of annual heating and cooling loads ($Q_h(x)$ and $Q_c(x)$, divided by the typical plant efficiencies $\eta_h = 0.44$ and $\eta_h = 0.77$, respectively) and the energy for artificial lighting (E_l, converted into primary fuel consumption with a multiplication by three).

$$f(x) = \frac{Q_h(x)}{\eta_h} + \frac{Q_c(x)}{\eta_c} + 3E_l(x)$$

EnergyPlus 8.5.0 performed the energy simulations for problems 3, 4, and 5.

3.4. Problem 4: Building energy, 13 continuous variables

Problem 4 concerns a more detailed office building in Houston (Fig. 5b). The thirteen variables control the window width and heights for the North, West, East and South façade (w_N, w_W, w_E, w_S), the depth of the window overhangs in West, East and South (o_W, w_E, w_S), the setpoint of the shading devices in W/m^2 at the West, East and South façade (s_W, s_E, s_S), the setpoint for the zone air temperature for night cooling during summer and winter (T_u, T_i), and the cooling design supply air temperature used for the HVAC system sizing (T_d). The objective function is identical to problem 3.

3.5. Problem 5: Building energy, 13 discrete variables

Problem 5 is identical to 4 but uses discrete variables and Chicago Weather.

3.6. Problem 6: Daylight, 15 discrete variables

Problem 6 considers a single room (11.25 by 7.5 m, with a floor to ceiling height of 4.3 m), located on the Southwest corner of the third floor of a building in Singapore (Fig. 6). The room has two facades, one with nine façade components, and one with six.

[4] Silvereye's PSO automatically adjusts its "Max. Velocity" parameter based on the ranges of a problem's variables.

Table 1
Overview of benchmark problems in this article. n_C indicates the number of continuous variables, n_D the number of discrete variables, and t the required time for a single function evaluation in milliseconds on an Intel i7 6700 K CPU with 4.0 Ghz and eight threads. Center indicates the objective value for a solution with values in the center of each variable range and Best the objective value for the best solution found in the benchmarks. Due to their formulation, the improvement from Center to Best for problems 5 and 6 is comparatively small.

	Type	n_C	n_D	t [ms]	Center	Best	Units
1	Structural	0	8	~25	319	220	kg
2	Structural	0	22	~300	38,500	4581	kg
3	Building Energy	4	0	~4000	153.2	132.9	kWh/m^2 a
4	Building Energy	13	0	~8000	214.6	175.5	kWh/m^2 a
5	Building Energy	0	13	~8000	179.0	166.3	kWh/m^2 a
6	Daylight	0	15	~45,000	13% (87% UDI)	11% (89% UDI)	
7	Daylight and Glare	40	0	~120,000	45% (43% UDI/32% DGP)	20% (85% UDI/25% DGP)	

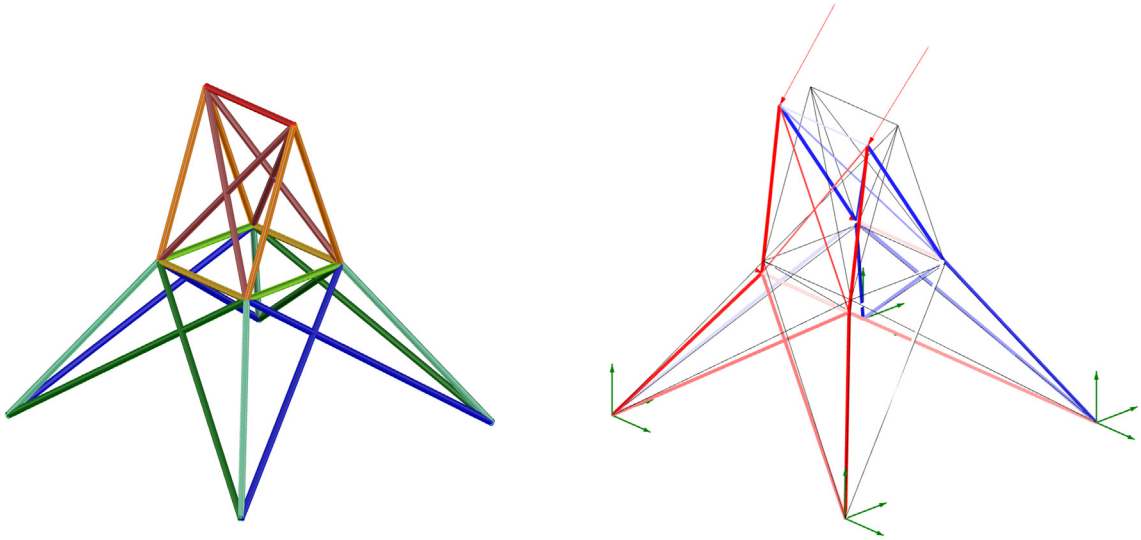

Fig. 3. Problem 1's small transmission tower. The left diagram represents each of the 8 beam sets with a color. The right diagram shows the tower deforming under load (1000% exaggerated).

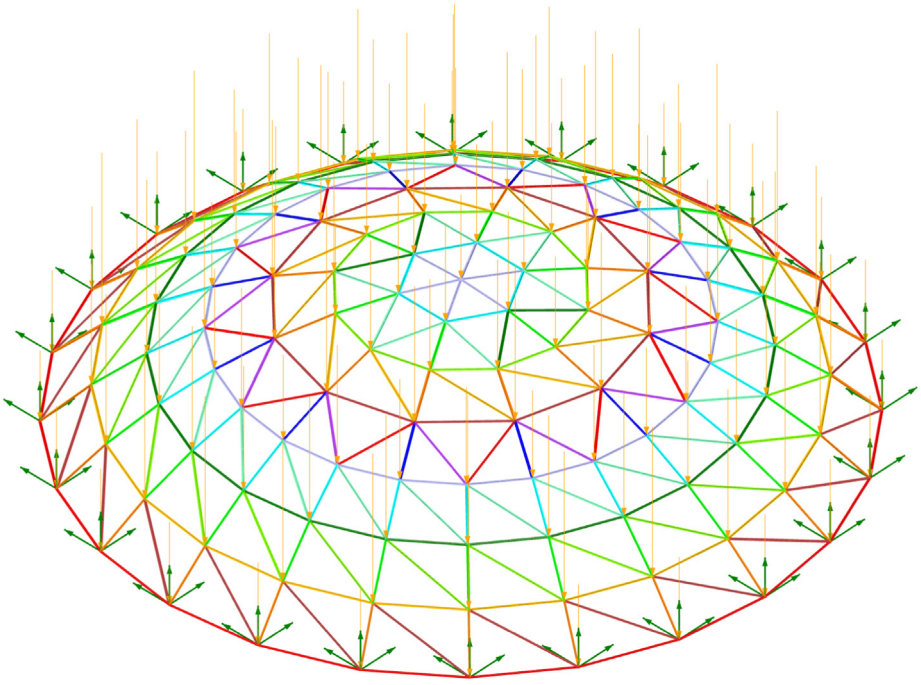

Fig. 4. Problem 2's dome under dead load. Each color represents one of the 22 beam sets.

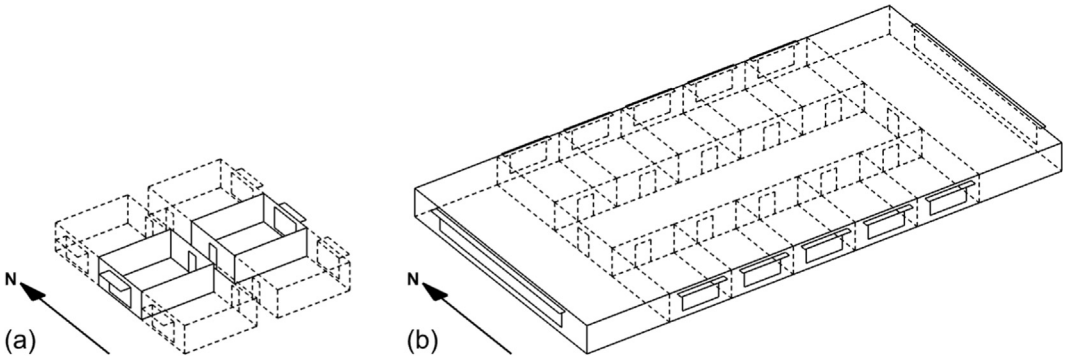

Fig. 5. Energy models of office buildings for problem 3 (a) and problems 4 and 5 (b). The small model on the left simulates only two zones (i.e., rooms), while the larger model on the right simulates a complete floor.

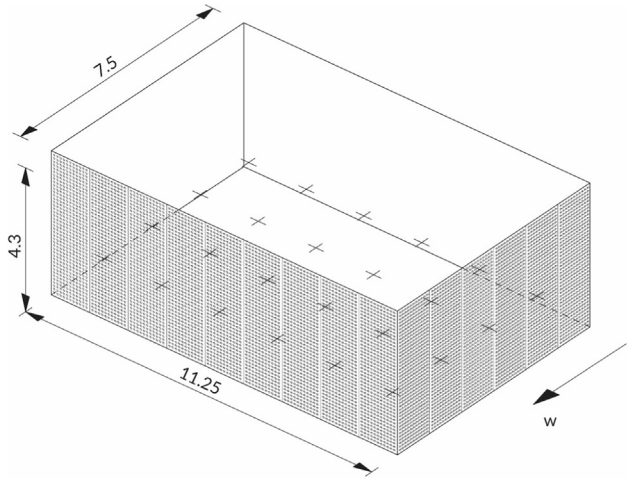

Fig. 6. Diagram of the room being optimized in terms of daylight. The crosses indicate the sensor grid for simulating UDI.

We are searching for a façade configuration that maximizes daylight while avoiding heat gains and glare and maintaining a unified appearance of the facade. Every façade component is 1.25 m wide and perforated with a grid of 426 circular holes, with a diameter of 50 mm each. The holes have "flaps," or micro-louvers, that, for each façade component, can take angles between 0° and 180° (in increments of 5°). In this way, we standardize the façade components into 37 distinct, daylight-modulating types.

The optimization problem aims to find a configuration for the angles of the fifteen louvered 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 (100–2000 lx) (Mardaljevic, Andersen, Roy, & Christoffersen, 2012).

In addition, we penalize design candidates for angle differences larger than 10° between neighboring façade components, to ensure a coherent appearance of the façade from the outside, and a subtle modulation of daylight on the interior.[5] The penalty function below computes a penalty value p_i for an individual façade component with angle d_i. If the angle difference with the previous neighbor is smaller than 10°, the penalty pi is zero, otherwise it is a squared error term. The objective value to be maximized is the difference of the average UDI $u(x)$ and the sum of the penalties of the fifteen façade components $p(x)$.

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

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

RADIANCE performed the daylighting and glare simulations for problems 6 and 7, via the DIVA 4.0.2 Grasshopper plug-in.

3.7. Problem 7: Daylight and Glare, 40 continuous variables

Problem 7 considers a single room in Singapore (Fig. 7). The rectangular room has a South-facing, 10.8 m long and 3.6 m high façade and is 7.2 m deep. The room's floor is raised 20 m above ground level. The façade has a porous screen with a triangular grid of 1.692 circular openings.

To avoid controlling every opening with an individual variable and to create a graduated, cloudy appearance, a grid of forty "attractor points" controls the openings, with weights in the range [0.0, 1.0]. To create a soft falloff, the parametric model calculates the radius of every opening as the average of the values of all attractor points, weighted by the inverse squares of their distances to the opening and multiplied by the maximum radius of 65 mm.

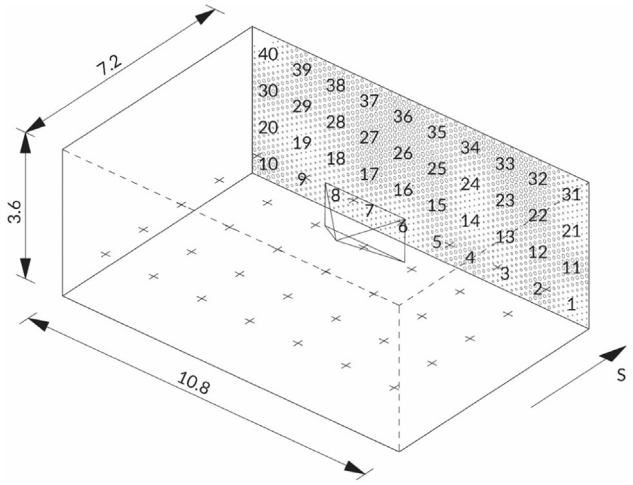

Fig. 7. Diagram of the room Problem 7 optimizes in terms of daylight and glare. The crosses indicate the sensor grid for simulating UDI, the cone the camera position and view for simulating DGP, and the numbers the attractor points regulating the façade's porosity.

[5] This problem formulation assumes that, for daylight quality, a room's orientation matters more than height, and thus does not consider vertical angle differences between floors.

Openings with a radius below 10 mm are closed completely. This formulation results in a problem with forty continuous variables.

The objectives of the optimization are to (1) maximize UDI while (2) minimizing Daylight Glare Probability (DGP). This problem calculates UDI as the average from a seven by five grid of sensor points. DGP measures glare as a percentage for a specific camera view and for a specific point-in-time (Wienold, 2010). To reduce calculation time, this value is simulated only for a single camera. The south-facing camera points directly at the screen and is in the center of the room at a height of 1.6 m from the floor.[6]

The problem approximates annual glare as the average of 59 DGP values corresponding to 59 daylight hours on which more extensive annual glare simulations rely. Although less accurate than a full annual simulation—which can take hours even at low quality settings—this approach yields a good qualitative assessment of the presence or absence of discomfort glare.

If quality of daylight and avoidance of glare are equally important, subtracting average annual UDI u from 100% and adding (approximated) average annual DGP g yields a single minimization objective. (Both UDI and DGP are in the range [0, 1].):

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

Wortmann (2017) considers a multi-objective version of this problem and concludes 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."

4. Benchmark results

This section presents benchmark results individually for each problem and in combination.

4.1. Problem 1: Structural, 8 discrete variables

For problem 1, RBFOpt converges most rapidly and the Galapagos GA and SGA most slowly. CRS2 converges slowly but steadily and ends up with a very good median solution. The remaining algorithms exhibit similar speeds of convergence (Fig. 8). Note that RBFOpt converges rapidly until about 100 evaluations, then improves more slowly until about 300 evaluations, and then again improves comparatively rapidly. This behavior probably is due to a combination of restarts (which trigger after 100 evaluations without improvements) and an increase in refinement (i.e., local searches). This "unlimited refinement" starts after the algorithm has expended 60% of the function evaluation budget, i.e., after 300 evaluations. In terms of stability, the deterministic DIRECT is the most stable, followed by SA and CMAES. The remaining algorithms display comparable variance (Fig. 9). For RBFOpt, PSO, and CRS2, this variance tends towards solutions that are better than those found by DIRECT, SA and CMAES. In summary, DIRECT is a good choice when stability is critical, but RBFOpt and CRS2 find better solutions 95% of the time.

4.2. Problem 2: Structural, 22 discrete variables

For problem 2, RBFOpt again is the fastest converging algorithm, with SBPLEX and SA "catching up" at around 200 function evaluations. DIRECT converges far too slowly, probably due to this problems' considerable number of variables (22 discrete variables) (Fig. 10). Interestingly, DIRECT's recursive subdivision is discernable in the "staircase" pattern of its convergence graph.

[6] *A more realistic glare assessment would evaluate several camera views to cover the users' field of view.*

Apart from DIRECT, SGA and RBFOpt are the most stable algorithms, closely followed by SA (Fig. 11). Accordingly, RBFOpt is the best choice for this problem and evaluation budget.

4.3. Problem 3: Building energy, 4 continuous variables

For problem 3, SA, PSO, DIRECT, RBFOpt, and CMAES find median solutions of very similar quality (within 0.05 kWh/m^2 a) (Fig. 12). SA finds the best median solution, but DIRECT, and especially RBFOpt, converge faster for the first hundred function evaluations. After DIRECT, RBFOpt is the most stable algorithm (Fig. 13). Accordingly, DIRECT is the best choice for this problem and evaluation budget. GA and SGA perform comparatively poorly.

4.4. Problem 4: Building energy, 13 continuous variables

For problem 4, SBPLEX, SA, DIRECT, and RBFOpt find the best median solutions (within 0.5 kWh/m^2 a of each other) (Fig. 14). RBFOpt is the fastest converging algorithm until about 200 function evaluations. After DIRECT, RBFOpt is the most stable algorithm (Fig. 15). SBPLEX displays one significant outlier. Accordingly, RBFOpt is the best choice for smaller function evaluation budgets and DIRECT the best choice when stability is critical. GA, SGA, and CR2 perform similarly poorly.

4.5. Problem 5: Building energy, 13 discrete variables

For problem 5, SA, SBPLEX, and RBFOpt find the best median solutions (within 0.2 kWh/m^2 a of each other) (Fig. 16). RBFOpt is fastest converging algorithm until about 200 function evaluations. After DIRECT, RBFOpt is the most stable algorithm (Fig. 17). SA, PSO, and SBPLEX exhibit two outliers. Accordingly, RBFOpt provides the best balance between speed of convergence and stability for this problem and evaluation budget. Again, GA, SGA and CRS2 perform similarly poorly.

4.6. Problem 6: Daylight, 15 discrete variables

For problem 6, RBFOpt and DIRECT find the best median solutions (within 0.1% of each other) (Fig. 18). DIRECT evaluates the "Center" solution—which for this problem is very good (Table 1)—first. RBFOpt catches up at around 100 function evaluations. PSO and SA converge slower, while the GA, CMAES, CRS2, and SGA do not find useful solutions. After DIRECT, RBFOpt is the most stable algorithm (Fig. 19). Accordingly, DIRECT provides the best balance between speed of convergence and stability for this problem and evaluation budget.

4.7. Problem 7: Daylight and Glare, 40 continuous variables

On problem 7, DIRECT is the worst-performing algorithm because its recursive subdivision proceeds too slowly in the forty dimensions corresponding to the variables (Fig. 20). SBPLEX performs similarly poorly. The metaheuristic SGA, SA, GA, CRS2, and PSO perform similarly and comparatively poorly. RBFOpt is the best-performing algorithm, followed by CMAES. Except for the deterministic DIRECT, CMAES is the most stable algorithm, followed by RBFOpt (Fig. 21).

4.8. Combined results

Figs. 22 and 23 combine the results from all problems. They indicate the (median) objective value achieved by each algorithm as a percentage of the best-know solution. (Since all problems are minimization problems, these values are larger than 100%.) Apart from the first forty function evaluations, during which

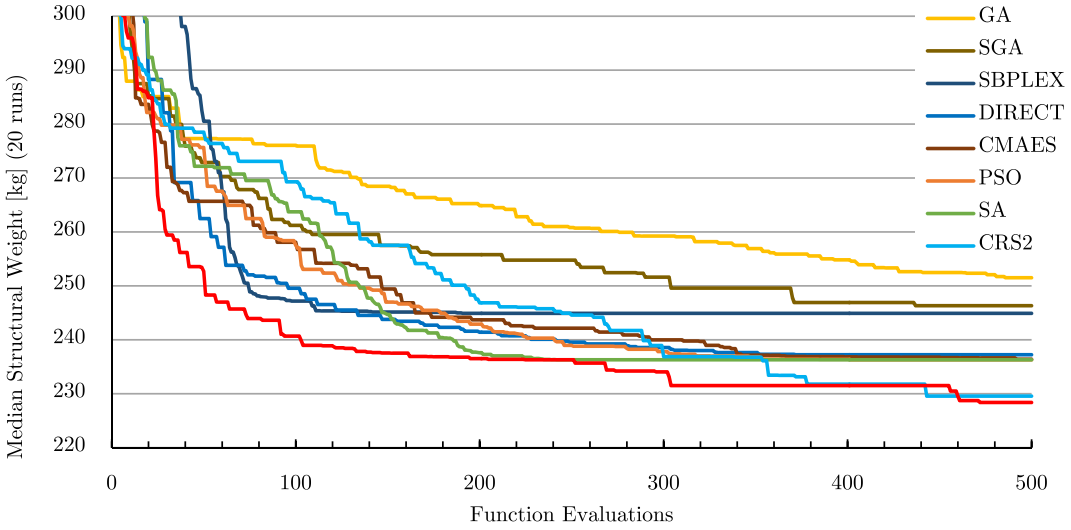

Fig. 8. Problem 1: Convergence. The legend displays the algorithms according their final solutions' values, from bottom to top.

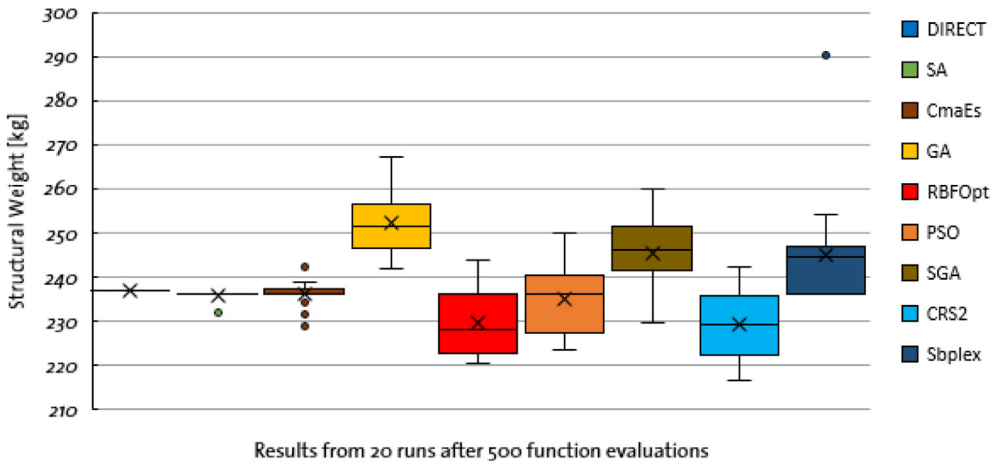

Fig. 9. Problem 1: Stability. This figure orders the algorithms according their results' standard deviations, increasing from left to right. In this "box-and-whisker plot," the top lines of the colored boxes represent the upper quartiles—i.e., the 25% worst solutions are above this line—and the bottom lines the lower quartiles. The boxes' center lines represent medians, and the crosses averages. The parallel lines at top and bottom represent the values' ranges, except for outliers, which exceed the respective quartile by more than 150%.

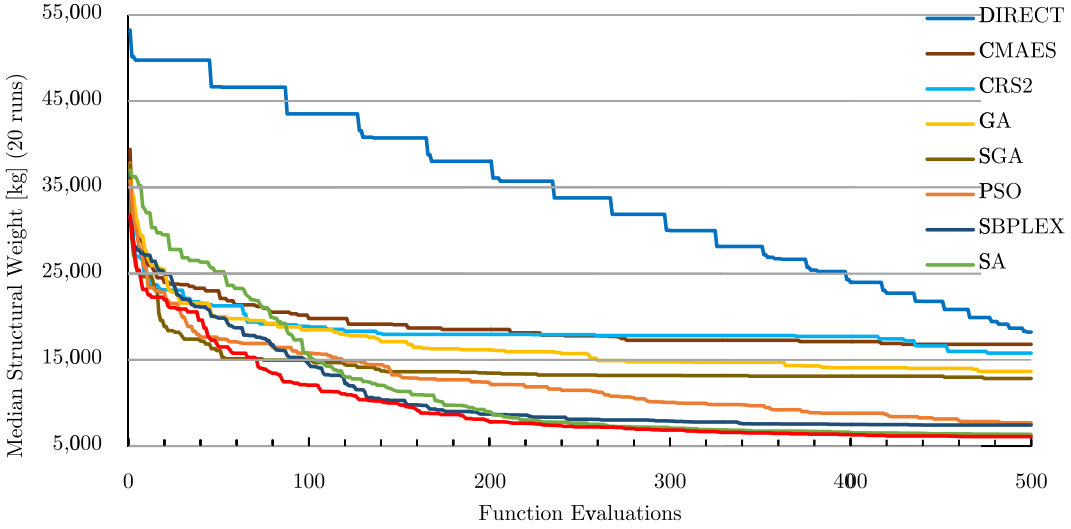

Fig. 10. Problem 2: Convergence.

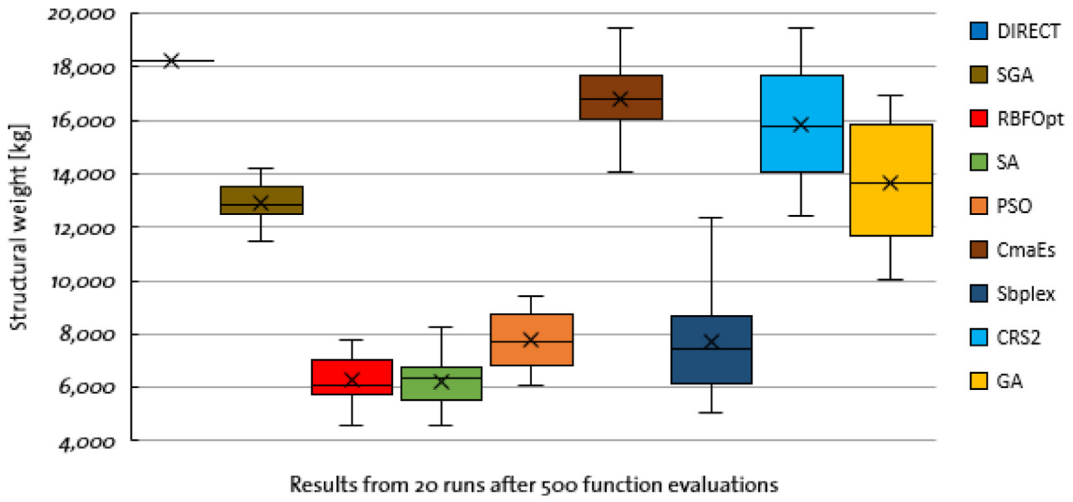

Fig. 11. Problem 2: Stability.

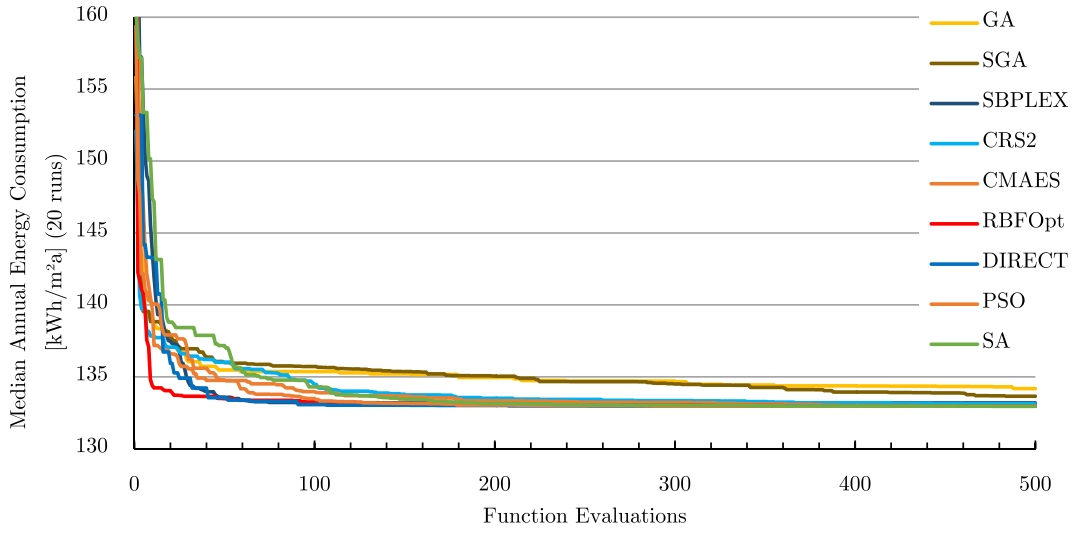

Fig. 12. Problem 3: Convergence.

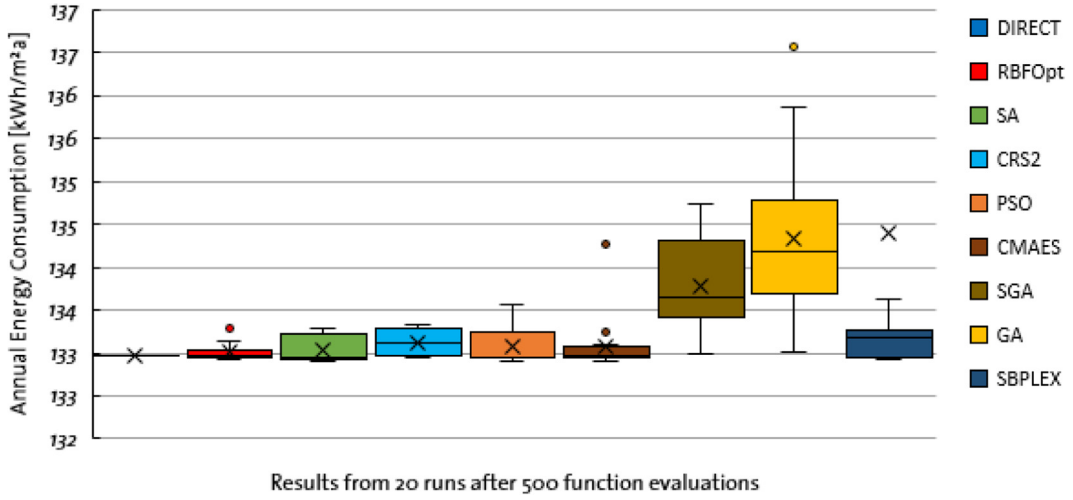

Fig. 13. Problem 3: Stability.

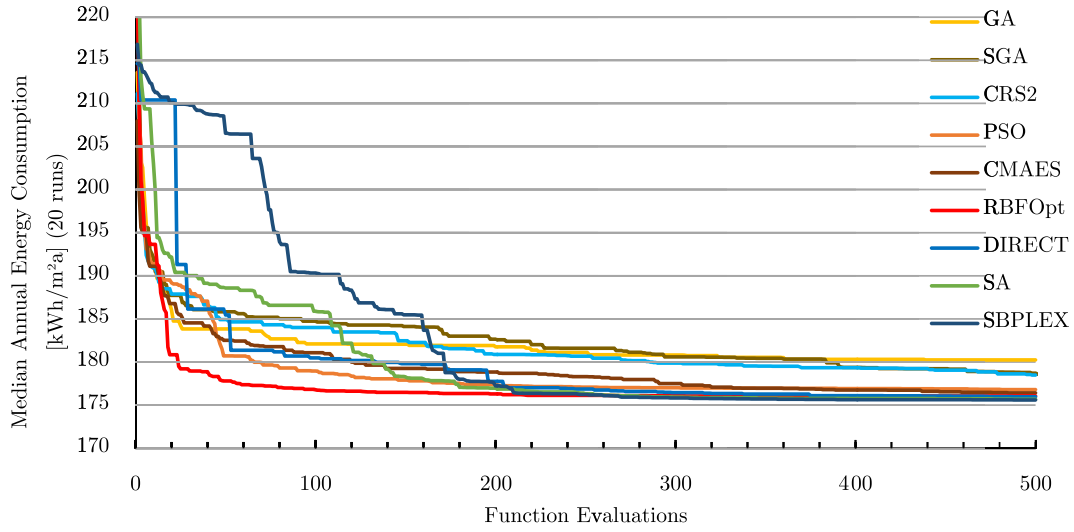

Fig. 14. Problem 4: Convergence.

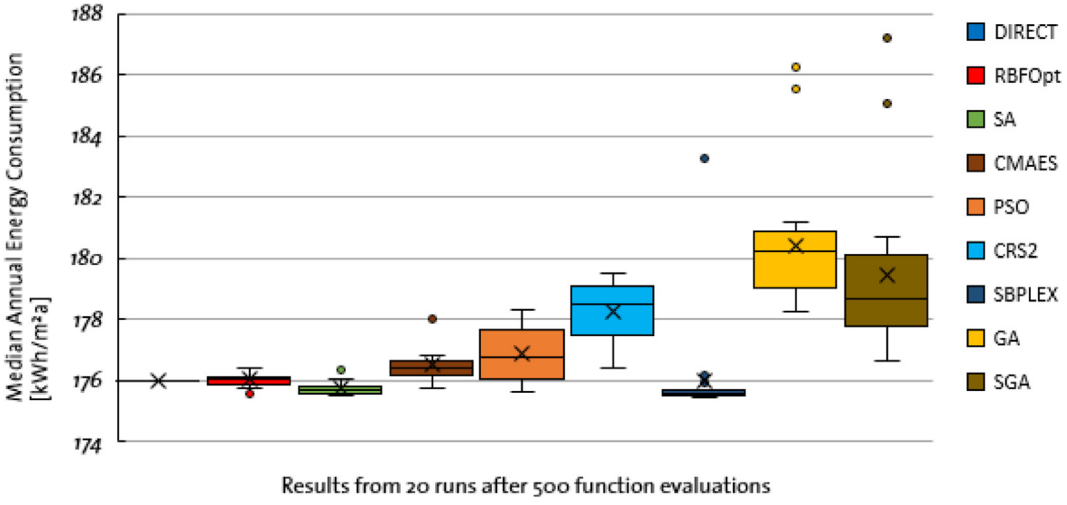

Fig. 15. Problem 4: Stability.

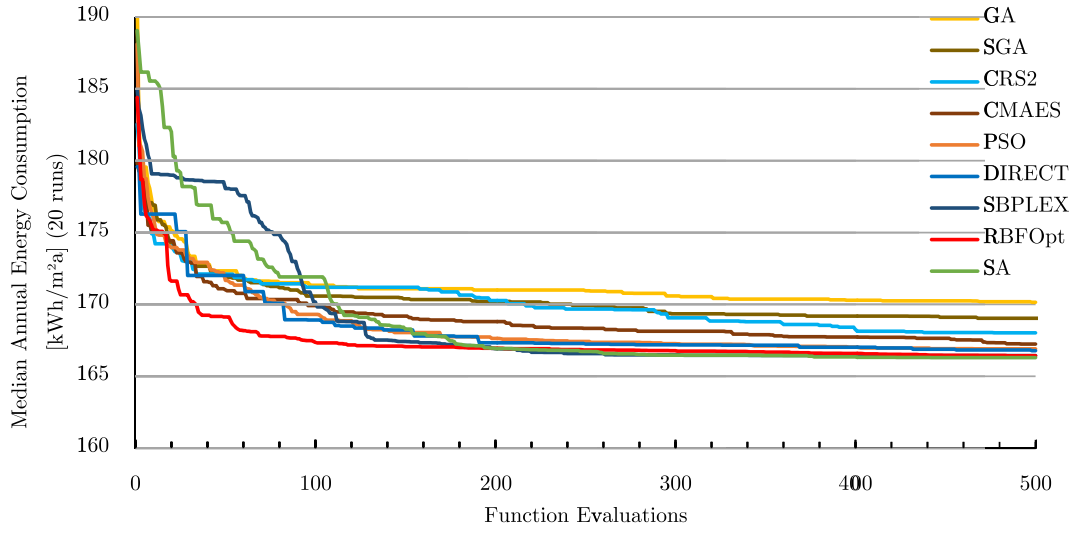

Fig. 16. Problem 5: Convergence.

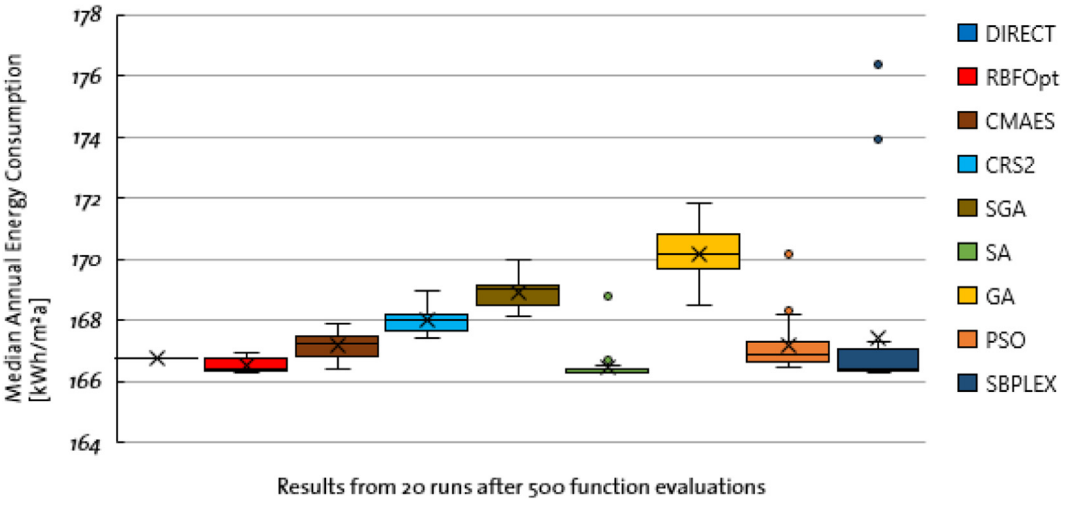

Fig. 17. Problem 5: Stability.

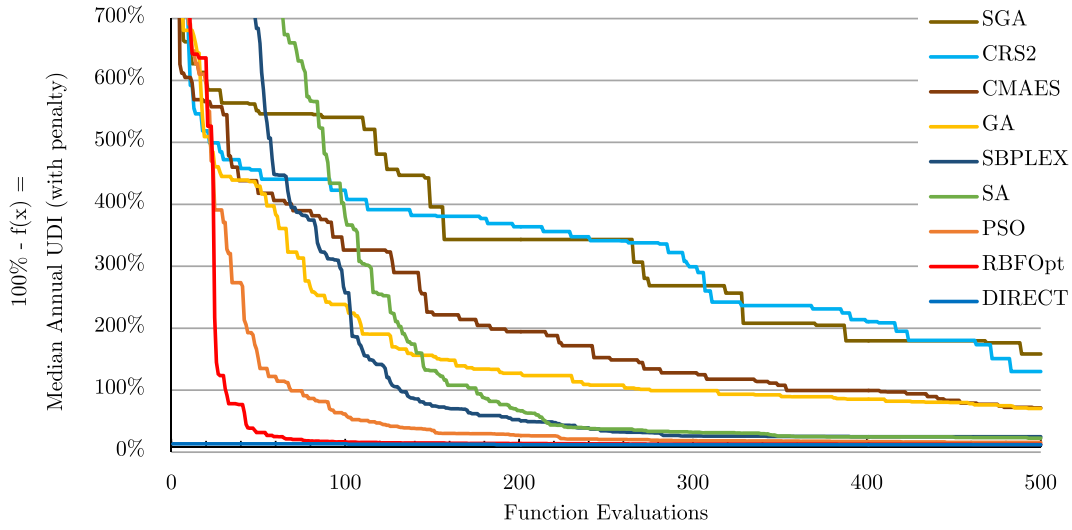

Fig. 18. Problem 6: Convergence.

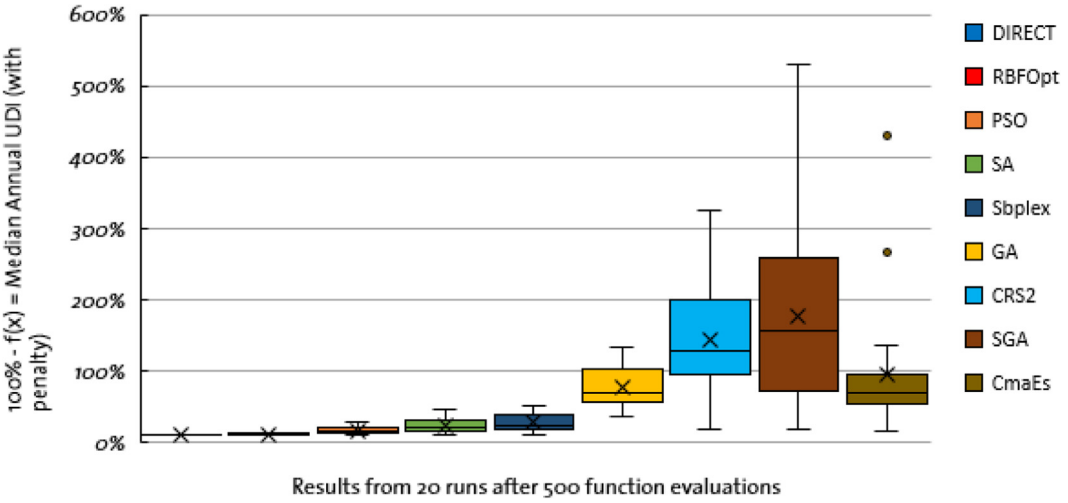

Fig. 19. Problem 6: Stability. Note the vast range of outcomes for the SGA.

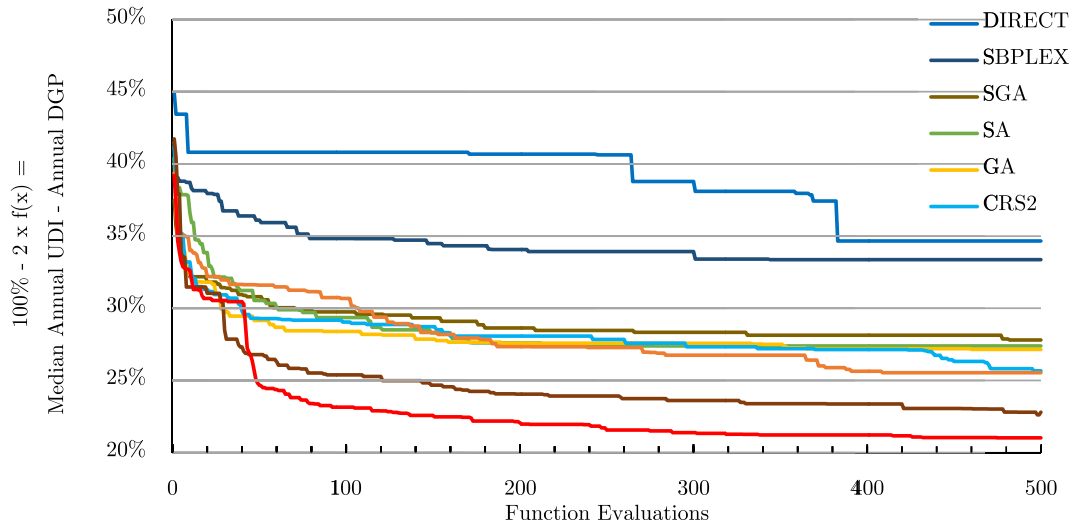

Fig. 20. Problem 7: Convergence. Note the rapid improvement of RBFOpt after 40 iterations. Until this point, RBFOpt is quasi-randomly simulating design candidates. After 40 iterations (i.e., the number of variables of this problem), RBFOpt constructs the first surrogate model, which almost immediately results in a substantial improvement.

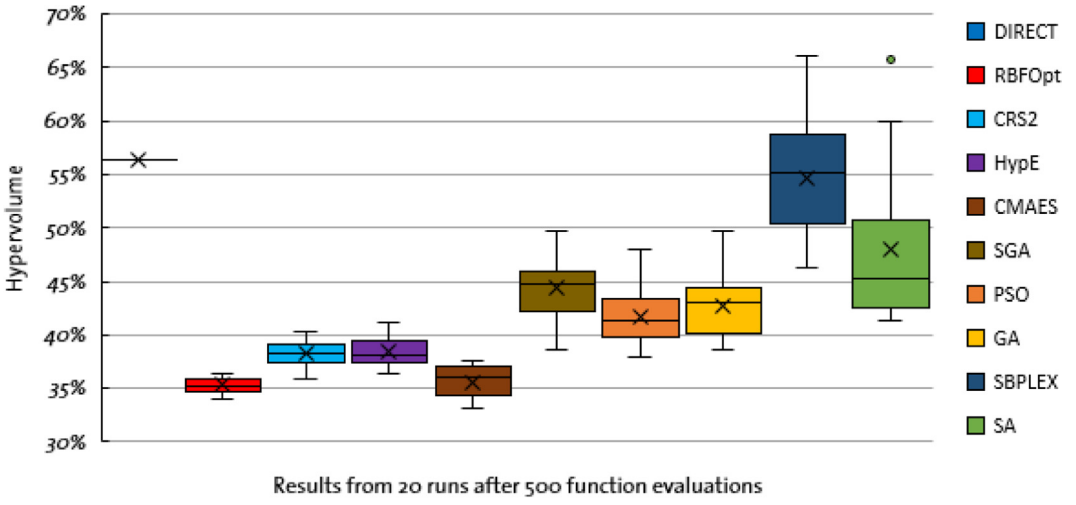

Fig. 21. Problem 7: Stability.

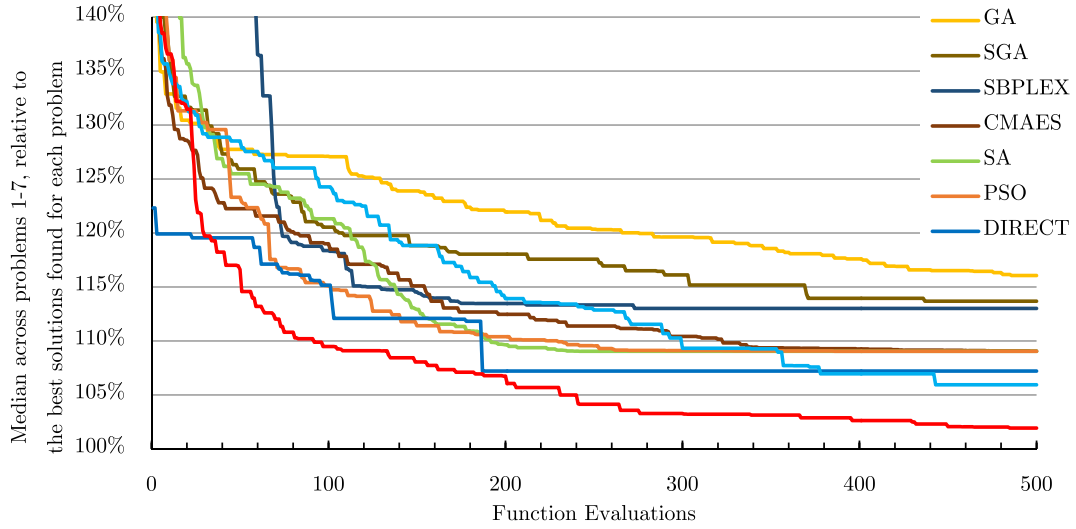

Fig. 22. Problems 1–7: Convergence.

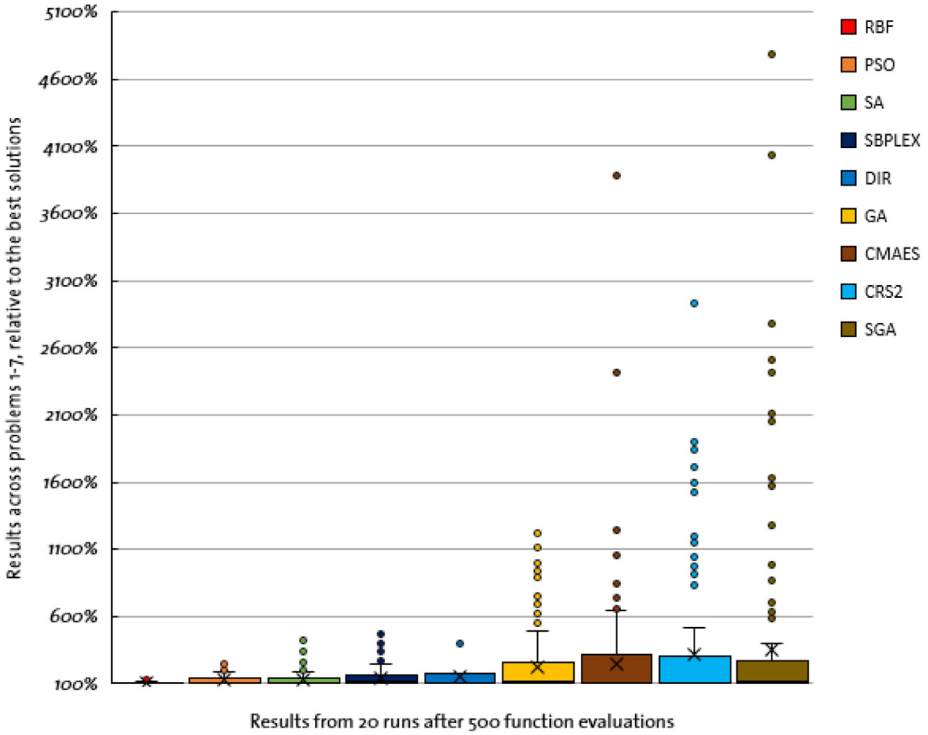

Fig. 23. Problems 1–7: Stability. Note that DIRECT exhibits identical performance on individual problems, but varying performance relative to different problems.

DIRECT often finds the best solutions due to its deterministic search sequence, RBFOpt is the fastest converging algorithm, followed by CRS2, DIRECT, PSO, SA, and CMAES (Fig. 22). Remarkably given its specialization on small evaluation budgets, RBFOpt exhibits strong improvement also after 200 function evaluations, which is where DIRECT, PSO, SA, and CMAES begin to stagnate. RBFOpt is the most robust algorithm, and the metaheuristic SGA, CRS2, CMAES, and GA the least robust (Fig. 23). As such, RBFOpt is the only algorithm that performs well in terms of convergence and robustness and that shows significant improvement after 200 function evaluations.

DIRECT performs well on problems with smaller numbers of variables. As such, it is worth trying on such problems, especially because its deterministic nature makes it more stable than CRS2, PSO and SA, which otherwise perform comparably.

GA and SGA perform significantly worse than the remaining algorithms. Given their popularity in architectural theory (e.g., De Landa, 2002) and ADO, the GAs' performance is remarkably disappointing, especially relative to other metaheuristics such as SA and PSO. CMAES is an example of a better performing evolutionary algorithm, but also displays low robustness.

5. Discussion

This section discusses limitations and critiques the wide-spread use of GAs in ADO.

5.1. Limitations

In addition to the choices of the function evaluation budget and algorithmic parameters, this section identifies two limitations: (1) the size of the problem set and (2) the number of tested algorithms.

5.1.1. Size of the problem set

The most critical limitation of this benchmark is the size of its problem set since, according to the Almost No Free Lunch theorem (Droste, Jansen, & Wegener, 2002), the performance of optimization algorithms is problem-dependent.

But one can somewhat generalize algorithmic performance for sets of problems with shared characteristics. For example, all problems in this benchmark employ simulations and exhibit nonlinear relationships between variables and objective values, as well as multiple local optima.

Compared to mathematical benchmarks such as (Rios & Sahinidis, 2013), this problem set is small. Nevertheless, in the author's opinion, the problem set combines distinct types of simulation-based ADO problems, numbers and types of variables, and evaluation times in a diverse and relevant manner. Note that, in contrast to mathematical benchmarks, this benchmark employs simulation-based problems instead of mathematical test functions.

The number of problems (seven) also is comparable to well-cited ADO papers such as (Wetter & Wright, 2004)—two building energy problems with three different weather files, i.e., six problems—and (Hasançebi et al., 2009), with five structural problems. Most ADO papers present only a single ADO problem and algorithm (Evins, 2013; Touloupaki & Theodosiou, 2017). Compared to other ADO papers, this problem set is more varied since it covers several ADO domains.

5.1.2. Number of tested algorithms

Similarly, the number of algorithms (seven) that were tested across all problems is small compared to, for example, mathematical optimization competitions. It is slightly smaller compared to (Wetter & Wright, 2004) and identical to (Hasançebi et al., 2009). But Wetter and Wright (2004) test only local direct search, metaheuristic, and hybrid methods, while Hasançebi et al. (2009) test only metaheuristics, while this benchmarks tests global direct search and model-based methods, as well as metaheuristics.

5.2. Critique of genetic algorithms in ADO

This benchmark indicates that—for practical, simulation-based, and time-intensive ADO problems with modest evaluation

budgets—a global model-based method such as RBFOpt is the most likely to yield the best results.

In the seminal study by Wetter and Wright (2004), the authors conclude that the hybrid PSO/Hook-Jeeves algorithm finds the best solutions and the SGA offers faster convergence at a slight decrease in solution quality. In this benchmark, the Galapagos GA is the worst-performing, and SGA the second-worst-performing, algorithm on problems 1, 3, 4, and 5. The latter three problems replicate problems from (Wetter & Wright, 2004). The Galapagos GA and SGA also are the worst-performing algorithm overall. Without comparison with the SGA, one might suspect that the poor performance of the GA is due to an ineffective implementation by Rutten (2013). But the comparison shows that the SGA is only slightly worse than the GA.

In a follow-up study, Wright and Alajmi (2005) conclude that the SGA " was insensitive to the choice of GA control parameters". But the SGA's inferior performance on the structural and daylighting problems indicates that it is highly sensitive to individual problems. This sensitivity indicates that the SGA's good performance in (Wetter & Wright, 2004) might be due to problem-specific tuning of its parameters.

These results contradict not only (Wetter & Wright, 2004) and (Wright & Alajmi, 2005), but most of the ADO literature, which often exhibits a bias towards GAs. On the other hand, the results are unsurprising in light of mathematical benchmark results and proofs of convergence (Conn et al., 2009; Costa & Nannicini, 2014; Rios & Sahinidis, 2013). Generalizations such as "evolutionary algorithms are robust in exploring the search space for a wide range of building optimization problems" (Attia et al., 2013) demand critical scrutiny and more extensive benchmarking.

The poor performances of the GAs in this benchmark also problematize the common practice (Evins, 2013; Hare, Nutini, & Tesfamariam, 2013) of using GAs as a baseline in (already scarce) ADO benchmarks. At least compared to the Galapagos GA, almost any algorithm will be better most of the time, which can give rise to the spurious innovations of novel metaheuristics decried by Sörensen (2015). Instead, benchmarks should include state-of-the-art methods such as DIRECT and RBFOpt.

6. Conclusions

Whenever possible, ADO practitioners should test more than one algorithm on their problems. Based on this benchmark, Goat's DIRECT is a good choice when stability is critical and the number of variables is not too large. Opossum's RBFOpt is a good all-round choice, especially for small evaluation budgets employed for time-intensive simulations such as daylighting and/or during concept design phases. For larger evaluation budgets, the Galapagos SA and Silvereye's PSO can deliver good results as well.

Due to its low robustness, CRS2 cannot be recommended. Contrasting with most of the ADO literature, the uses of at least the Galapagos GA and SGA cannot be recommended as well. Instead, this benchmark confirms an insight from the mathematical optimization community: Mathematically well-grounded methods often achieve good and robust results also on practical problems where proofs of convergence might not necessarily hold, such as in ADO.

Future work includes more extensive benchmarking with more problems and algorithms, the development of a multi-objective, model-based optimization algorithm and tool, and research into ways to (visually and interactively) present optimization results in ways that are more meaningful and relevant for designers.

Conflict of interest declaration

There is no conflict of interest with this submission.

Acknowledgments

This article is based on a chapter in the author's PhD thesis (Wortmann, 2018). 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 Michael Wetter and Jonathan Wright for providing the original files for Problems 3, 4, and 5, Christoph Waibel for updating these files to EnergyPlus 8.5.0 and preparing them in Grasshopper, and the anonymous reviewers for their detailed comments. Alstan Jakubiec provided free, computer-independent educational licenses for DIVA.

References

Attia, S., Hamdy, M., O'Brien, W., & Carlucci, S. (2013). Assessing gaps and needs for integrating building performance optimization tools in net zero energy buildings design. *Energy and Buildings, 60*, 110–124.

Brownlee, A. E. I., & Wright, J. A. (2015). Constrained, mixed-integer and multi-objective optimisation of building designs by NSGA-II with fitness approximation. *Applied Soft Computing, 33*, 114–126.

Cichocka, J. M., Migalska, A., Browne, W. N., & Rodriguez, E. (2017). SILVEREYE – The implementation of particle swarm optimization algorithm in a design optimization tool. In G. Çagdas, M. Özkar, L. F. Gül, & E. Gürer (Eds.), *Computer-aided architectural design. Future trajectories* (pp. 151–169). Singapore: Springer Singapore. https://doi.org/10.1007/978-981-10-5197-5_9.

Conn, A., Scheinberg, K., & Vicente, L. (2009). *Introduction to derivative-free optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Costa, A., & Nannicini, G. (Eds.). (2014). *RBFOpt: An open-source library for black-box optimization with costly function evaluations (Optimization Online No. 4538)*. Singapore University of Technology and Design.

De Landa, M. (2002). Deleuze and the use of the genetic algorithm in architecture. In N. Leach (Ed.), *Designing for a digital world* (pp. 117–118). London, UK: John Wiley & Sons.

Droste, S., Jansen, T., & Wegener, I. (2002). Optimization with randomized search heuristics—The (A)NFL theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science, 287*(1), 131–144. https://doi.org/10.1016/S0304-3975(02)00094-4.

Evins, R. (2013). A review of computational optimisation methods applied to sustainable building design. *Renewable and Sustainable Energy Reviews, 22*, 230–245.

Forrester, A. I. J., Sóbester, A., & Keane, A. J. (2008). *Engineering design via surrogate modelling: A practical guide*. Chichester, UK: J. Wiley.

Hansen, N. (2018). *pycma: Python implementation of CMA-ES*. CMA-ES: Python https://github.com/CMA-ES/pycma.

Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation, 9*(2), 159–195.

Hare, W., Nutini, J., & Tesfamariam, S. (2013). A survey of non-gradient optimization methods in structural engineering. *Advances in Engineering Software, 59*, 19–28.

Hasançebi, O., Çarbaş, S., Doğan, E., Erdal, F., & Saka, M. P. (2009). Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. *Computers & Structures, 87*(5–6), 284–302.

Holmström, K. (2008). An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization. *Journal of Global Optimization, 41*(3), 447–464.

Imbert, F., Frost, K. S., Fisher, A., Witt, A., Tourre, V., & Koren, B. (2013). Concurrent geometric, structural and environmental design: Louvre Abu Dhabi. In L. Hesselgren, S. Sharma, J. Wallner, N. Baldassini, P. Bompas, & J. Raynaud (Eds.), *Advances in architectural geometry 2012* (pp. 77–90). Springer Vienna.

Johnson, S. G. (2010). *The NLopt nonlinear-optimization package*. Cambridge, MA: Massachusetts Institute of Technology http://ab-initio.mit.edu/nlopt.

Jones, D. R., Perttunen, C. D., & Stuckman, B. E. (1993). Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications, 79*(1), 157–181.

Kaelo, P., & Ali, M. M. (2006). Some variants of the controlled random search algorithm for global optimization. *Journal of Optimization Theory and Applications, 130*(2), 253–264.

Koziel, S., Ciaurri, D. E., & Leifsson, L. (2011). Surrogate-based methods. In S. Koziel & X.-. S. Yang (Eds.), *Computational optimization, methods and algorithms* (pp. 33–59). Heidelberg, DE: Springer.

Mardaljevic, J., Andersen, M., Roy, N., & Christoffersen, J. (2012). *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.

Moré, J. J., & Wild, S. M. (2009). Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization, 20*(1), 172–191.

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (2nd ed.). New York, NY: Springer.

Rios, L. M., & Sahinidis, N. V. (2013). Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization, 56*(3), 1247–1293.

Rowan, T. H. (1990). *Functional stability analysis of numerical algorithms (Ph.D. Dissertation)*. Austin, TX: The University of Texas.
Rutten, D. (2013). Galapagos: On the logic and limitations of generic solvers. *Architectural Design, 83*(2), 132–135.
Sörensen, K. (2015). Metaheuristics-the metaphor exposed. *International Transactions in Operational Research, 22*(1), 3–18.
Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*. Hoboken, N.J: John Wiley & Sons.
Touloupaki, E., & Theodosiou, T. (2017). Performance simulation integrated in parametric 3D modeling as a method for early stage design optimization—A review. *Energies, 10*(5), 637.
Waibel, C. (2018). *BB-O: A black-box optimization library*. CH: Zurich https://github.com/christophwaibel/BB-O.
Wetter, M., & Wright, J. (2004). A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization. *Building and Environment, 39*(8), 989–999.
Wienold, J. (2010). *Daylight glare in offices*. Stuttgart, DE: Fraunhofer IRB Verlag.
Wortmann, T. (2017). Model-based optimization for architectural design: Optimizing daylight and glare in grasshopper. *Technology | Architecture + Design, 1*(2), 176–185.
Wortmann, T. (2018). *Efficient, visual, and interactive architectural design optimization with model-based methods (Ph.D. Dissertation)*. Singapore: Singapore University of Technology and Design.
Wortmann, T., Costa, A., Nannicini, G., & Schroepfer, T. (2015). Advantages of surrogate models for architectural design optimization. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 29*(4), 471–481.
Wortmann, T., & Nannicini, G. (2016). Black-box optimization for architectural design: An overview and quantitative comparison of metaheuristic, direct search, and model-based optimization methods. In S.-F. Chien, S. Choo, M. A. Schnabel, W. Nakapan, M. J. Kim, & S. Roudavski (Eds.), *Proceedings of the 21th CAADRIA conference* (pp. 177–186). Hong Kong, CN: CAADRIA.
Wortmann, T., & Nannicini, G. (2017). Introduction to architectural design optimization. In A. Karakitsiou, A. Migdalas, P. M. Pardalos, & S. Rassia (Eds.). *City networks – planning for health and sustainability* (Vol. 128, pp. 259–278). Cham, CH: Springer International Publishing.
Wortmann, T., Waibel, C., Nannicini, G., Evins, R., Schroepfer, T., & Carmeliet, J. (2017). Are genetic algorithms really the best choice for building energy optimization? In *Proceedings of the symposium on simulation for architecture & urban design* (pp. 51–58). Toronto, CA: SCS.
Wright, J., & Alajmi, A. (2005). The robustness of genetic algorithms in solving unconstrained building optimization problems. *Proceedings of building simulation*. Montreal, CA: IBPSA.
Yang, D., Sun, Y., Stefano, D. di, Turrin, M., & Sariyildiz, S. (2016). Impacts of problem scale and sampling strategy on surrogate model accuracy: An application of surrogate-based optimization in building design. In 2016 IEEE congress on evolutionary computation (CEC) (pp. 4199–4207).