Journal of Global Optimization
https://doi.org/10.1007/s10898-021-01085-0

Review and comparison of algorithms and software for mixed-integer derivative-free optimization

Nikolaos Ploskas[1] · **Nikolaos V. Sahinidis**[2,3]

Received: 1 July 2020 / Accepted: 20 August 2021
© The Author(s) 2021

Abstract
This paper reviews the literature on algorithms for solving bound-constrained mixed-integer derivative-free optimization problems and presents a systematic comparison of available implementations of these algorithms on a large collection of test problems. Thirteen derivative-free optimization solvers are compared using a test set of 267 problems. The testbed includes: (i) pure-integer and mixed-integer problems, and (ii) small, medium, and large problems covering a wide range of characteristics found in applications. We evaluate the solvers according to their ability to find a near-optimal solution, find the best solution among currently available solvers, and improve a given starting point. Computational results show that the ability of all these solvers to obtain good solutions diminishes with increasing problem size, but the solvers evaluated collectively found optimal solutions for 93% of the problems in our test set. The open-source solvers MISO and NOMAD were the best performers among all solvers tested. MISO outperformed all other solvers on large and binary problems, while NOMAD was the best performer on mixed-integer, non-binary discrete, small, and medium-sized problems.

Keywords Derivative-free optimization algorithms · Mixed-integer optimization · Direct search methods · Surrogate models · Stochastic methods

✉ Nikolaos V. Sahinidis
nikos@gatech.edu

[1] Department of Electrical and Computer Engineering, University of Western Macedonia, Kozani, Greece

[2] H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA

[3] School of Chemical & Biomolecular Engineering, Georgia Institute of Technology, Atlanta, GA, USA

Published online: 07 September 2021

1 Introduction

We consider the following bound-constrained mixed-integer problem:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & l_j \le x_j \le u_j \quad \forall j \in \{1, \ldots, n\} \\ & x_j \in \mathbb{R} \quad \forall j \in I_c \\ & x_j \in \mathbb{Z} \quad \forall j \in I_z \end{aligned} \tag{1}$$

where n is the number of variables, $I_z \subseteq \{1, \ldots, n\}$ is the set of the discrete variables, $I_c := \{1, 2, \ldots, n\} \setminus I_z$ is the set of the continuous variables, $l_j, u_j \in \mathbb{R}$ for all $j \in I_c$, and $l_j, u_j \in \mathbb{Z}$ for all $j \in I_z$. We assume that algebraic expressions of f and its derivatives are unavailable, so we treat these problems as black-box optimization problems.

Algorithms for optimization problems in the absence of derivatives have a long history and are known as derivative-free optimization (DFO) algorithms [10,16]. The majority of DFO algorithms address the problem in which $I_z = \varnothing$, i.e., all variables are continuous. We refer to Rios and Sahinidis [62] for a recent review of DFO algorithms for continuous problems. As previous comparative studies of DFO algorithms were limited to a small number of algorithms and problems, the work of [62] filled a gap in the DFO literature by providing a systematic comparison of 22 DFO implementations on a test set of 502 unconstrained or bound-constrained problems. In the current paper, we seek to extend the results of [62] from the continuous to the mixed-integer case.

The bound-constrained mixed-integer derivative-free optimization (MIDFO) problem is of interest because there are many applications that require it, including software tuning [41,63], optimizing the circuitry configuration of heat exchangers [58], optimal design of integrated circuits [15], access point communication problems [23], and groundwater supply and hydraulic capture community problems [22].

Several papers consider bound-constrained MIDFO problems. The first such algorithm was proposed by Audet and Dennis [7]. Abramson et al. [1] extended the Mesh Adaptive Direct Search (MADS) method of [9] to mixed-variable optimization problems. Recently, Audet et al. [11] also extended the MADS method to handle integer variables. Liuzzi et al. [42–44] used line searches and proved global convergence of their MIDFO algorithm. Newby and Ali [57] proposed an extension of the BOBYQA algorithm [60] to mixed-variable programming. Other approaches based on surrogate models were proposed in [17,19,25,29,36,38,53,54,61]. Additionally, various heuristics have been proposed for solving bound-constrained MIDFO problems [13,37,40,64].

Many of the above-cited works compare several DFO solvers in the context of specific applications or small collections of test problems. However, a systematic comparison of solvers on a large collection of problems is missing. The aim of this paper is to fill this void in the literature. We first discuss recent algorithmic developments in this field, followed by a software comparison. In this comparison, we investigate which solver is more likely to obtain global or near-global solutions on mixed-integer and pure-integer problems. We also investigate how the quality of the solutions obtained changes as the problem size is increased.

We adopt the classification of Rios and Sahinidis [62] and classify DFO algorithms as model-based and direct-search, depending on whether they build surrogate models or not. We also classify them as global and local, depending on whether they seek global solutions or not. Finally, we classify them as stochastic and deterministic, depending on whether they use random search strategies or not.

The remainder of this paper is organized as follows. Section 2 reviews algorithms for MIDFO. Section 3 gives a brief overview of software for bound-constrained MIDFO. In Sect. 4, we illustrate the search strategies employed by various solvers with two simple examples. Extensive computational experience with thirteen solvers is presented in Sect. 5. A total of 267 test problems were used, including 176 pure-integer problems and 91 mixed-integer problems. These problems were used to test the solvers using the same starting points and bounding boxes. Conclusions from our study are drawn in Sect. 6. The Online Supplement provides a complete listing of the test problems and model statistics, the average- and best-case performance for each derivative-free optimization solver, the best solutions found by each solver, and the problems for which starting points were improved by each solver.

2 MIDFO methods

Due to the mixed-integer nature of Problem (1), multiple definitions of a local minimum can be given. Let us introduce

$$\mathcal{X} := \left\{x \in \mathbb{R}^n : l \leq x \leq u\right\}$$
$$\mathcal{Z} := \left\{x \in \mathbb{R}^n : x_j \in \mathbb{Z},\, j \in I_z\right\}$$

where $\mathcal{X}$ is a compact set, hence l_j and u_j cannot be infinite, for $i = 1, 2, \ldots, n$.

Local optimality is defined in terms of local neighborhoods. This is well-defined for continuous variables. For integer variables, a local neighborhood must be defined by the user. A local minimum depends on the notion of the local neighborhood we use. Hence, we summarize different definitions of local neighborhoods with respect to the continuous and integer variables [3,42,47]. Given a point $\overline{x} \in \mathbb{R}^n$ and $\rho > 0$, let us define

$$\mathcal{B}_c\left(\overline{x}, \rho\right) = \left\{x \in \mathbb{R}^n : x_z = \overline{x}_z, \|x_c - \overline{x}_c\|_2 \leq \rho\right\}$$

and

$$\mathcal{N}_z\left(\overline{x}\right) = \left\{x \in \mathbb{R}^n : x_c = \overline{x}_c, \|x_z - \overline{x}_z\|_2 = 1\right\}$$

where $x_c = [x]_{i \in I_c}$ and $x_z = [x]_{i \in I_z}$.

Now we can define a local minimum point for Problem (1).

Definition 1 *(Local minimum point)* A point $x^* \in \mathcal{X}$ is a local minimum of Problem (1) if there exists an $\epsilon > 0$ such that

$$\begin{aligned} f(x^*) &\leq f(x), \quad \forall x \in \mathcal{B}_c(x^*; \epsilon) \cap \mathcal{X} \\ f(x^*) &\leq f(x), \quad \forall x \in \mathcal{N}_z(x^*) \cap \mathcal{X} \end{aligned} \tag{2}$$

Every point $\overline{x} \in \mathcal{N}_z\left(x^*\right) \cap \mathcal{X}$, with $\overline{x} \neq x^*$, such that $f\left(\overline{x}\right) = f\left(x^*\right)$, satisfies (2).

Next, we introduce the definitions of stationary and strong stationary points. We assume that $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function with respect to x_j, $j \in I_c$.

Definition 2 *(Stationary point)* A point $x^* \in \mathcal{X} \cap \mathcal{Z}$ is a stationary point of Problem (1) when it satisfies the following

$$\nabla_c f\left(x^*\right)^\top \left(x - x^*\right)_c \geq 0, \quad \forall x \in \mathcal{X} \tag{3}$$
$$f\left(x^*\right) \leq f(x), \quad \forall x \in \mathcal{N}_z\left(x^*\right) \cap \mathcal{X} \tag{4}$$

where $\nabla_c f(x)$ denotes the gradient of the function with respect to the continuous variables.

Definition 3 *(Strong stationary point)* A point $x^* \in \mathcal{X} \cap \mathcal{Z}$ is a strong stationary point of Problem (1) when it is a stationary point, and $\forall \overline{x} \in \mathcal{N}_z(x^*) \cap \mathcal{X}$ such that $f(\overline{x}) = f(x^*)$, it holds that

$$\nabla_c f(\overline{x})^\top (x - \overline{x})_c \geq 0, \quad \forall x \in \mathcal{X} \tag{5}$$

$$f(\overline{x}) \leq f(x), \quad \forall x \in \mathcal{N}_z(\overline{x}) \cap \mathcal{X} \tag{6}$$

Finally, Newby and Ali [57] present a definition of an improved local minimum, a "combined local optimum".

Definition 4 *(Combined local minimum point)* A point $x^* \in \mathcal{X}$ is a local minimum of Problem (1) if there exists an $\epsilon > 0$ such that

$$f(x^*) \leq f(x), \quad \forall x \in \mathcal{B}_c(x^*; \epsilon) \cap \mathcal{X} \tag{7}$$

$$f(x^*) \leq f(x), \quad \forall x \in \mathcal{N}_{comb}(x^*) \cup \mathcal{N}_z(x^*) \tag{8}$$

where $\mathcal{N}_{comb}(x^*)$ is the set of the smallest local minima on each feasible continuous manifold on which $\mathcal{N}_z(x^*)$ has a point.

For problems that are convex when integrality is relaxed, any point that is a combined local minimum will also be a local minimum. The converse is not true; local minima are not always combined local minima [57].

Equipped with the above definitions we can now proceed to discuss algorithms and their optimality properties.

2.1 Local search algorithms

In Hooke and Jeeves [30], direct search is defined as the sequential examination of trial solutions generated by a certain strategy. Based on whether or not these methods operate on simplices, Conn et al. [16] classify direct search algorithms into simplicial methods such as the Nelder–Mead algorithm [55], and directional methods such as the generalized pattern search algorithm [72]. The main advantages of these methods are their simplicity, flexibility, and reliability. On the other hand, direct local search methods are highly dependent on the initial point and can be trapped in local minima.

The first algorithm for MIDFO was proposed by Audet and Dennis [7]. These authors developed a direct search algorithm for problems with bounds on continuous and categorical variables. The algorithm explores the search space of the continuous variables using a generalized pattern search (GPS) method. In the absence of discrete variables, the algorithm of Audet and Dennis reduces to the method proposed by Lewis and Torczon [39] for continuous problems. For the discrete variables, the algorithm of Audet and Dennis evaluates all points in a user-defined discrete neighborhood. The polling step of the algorithm includes three stages. The first stage is identical to the typical polling in pattern search algorithms for continuous variables only (the discrete variables are held fixed). The second stage is the natural generalization to the discrete variables using a set of neighbors. The last stage (extended polling) performs a continuous poll around promising points found during the discrete polling in order to improve the quality of the limit points. In this way, extended polling will be carried out at more iterations, which may cost more function evaluations, but should give a better local minimizer.

Several authors have extended the Audet and Dennis algorithm to handle more general problems. Using a filter approach, Abramson et al. [3] extended the algorithm to handle

general constraints for continuous variables. Sriver et al. [69] extended the algorithm to linearly-constrained mixed-integer problems with a stochastic objective function. Lucidi et al. [47] generalized the approach to solving linearly-constrained mixed-integer problems. Instead of applying pattern search to the continuous variables, Lucidi et al. [47] update the continuous variables using a continuous local search.

Abramson et al. [1] extended the MADS method [9] to mixed-variable optimization problems with general constraints for continuous variables. This was achieved in a relatively straightforward manner, similar to the work of Audet and Dennis [7], which extended the GPS methods to bound-constrained mixed-integer problems. Abramson et al. also use continuous, discrete, and extended poll steps. Recently, Audet et al. [11] also extended the MADS method [9] to handle integer variables by harmonizing the minimal granularity of variables with the finest mesh containing all trial points.

Liuzzi et al. [42] presented three variants of a direct search algorithm for bound-constrained mixed-integer problems. All variants perform a continuous and discrete search. The main difference between these variants is the strategy used in the discrete search procedure. The first algorithm explores the search space of the discrete variables using a method similar to the one proposed in [48]. The main difference is that Liuzzi et al. include a sufficient decrease condition governed by a control parameter, which is reduced during the optimization process. The second algorithm removes the sufficient decrease condition on the discrete variables and updates the iterate by choosing the coordinate that yields the largest objective function reduction. The third algorithm performs a more extensive investigation of the discrete neighborhoods by means of a local search procedure. Essentially, it uses a discrete search procedure that is similar to the one used in the first algorithm, but also performs an extended polling step if a point with a sufficient decrease is not found. The last algorithm is convergent to strong stationary points. Additionally, Liuzzi et al. [43] extended the approach presented in [42] to handle general constraints using the sequential quadratic penalty approach presented in [45]. Liuzzi et al. [44] also proposed an algorithmic framework to handle integer variables based on primitive directions and nonmonotone line searches. These three variants, along with other algorithms, are available in the Derivative-Free Library (DFL) [46]. This library includes several algorithms for continuous and mixed-integer, local and global derivative-free optimization.

Vicente [73] considers a slightly different but related problem. Vicente worked with implicit and densely discrete black-box optimization problems, i.e., problems that are characterized by the existence of an implicit and unknown discrete set where optimization points are first projected (the projection operator 'projects' the values to nearby values where it is possible or desirable to evaluate the real function) before the objective function is evaluated. The algorithm proposed by Vicente is a modified direct search.

Newby and Ali [57] proposed an extension of the BOBYQA algorithm [60] for mixed-variable programming. Their algorithm is a model-based local search. It uses quadratic approximations and integer local search, with guaranteed convergence to a "combined local optimum", which they defined as an improved local optimum.

Finally, Porcelli and Toint [59,71] proposed a directional direct search algorithm for mixed-variable optimization problems, including those with ordinal categorical variables. The proposed algorithm implements a mesh-based direct search that aligns the poll points to respect the integer constraints. A recursive call of the algorithm reduces the number of integer variables by fixing a subset of these variables.

2.2 Global search algorithms

Holmström et al. [29] extended Response Surface Methods (RSMs) based on kriging [49] and Radial Basis Functions (RBFs) to solve mixed-integer constrained black-box problems that have an expensive objective function and inexpensive general constraints. Hemker et al. [25] replaced the black-box portions of the objective function by a stochastic surrogate model. Davis and Ierapetritou [19] proposed a surrogate model-based algorithm for mixed-integer problems with binary variables that contain black-box functions. They combine a branch-and-bound algorithm with a kriging surface. Global information is obtained using kriging models that are used to identify promising neighborhoods for local search. Local RSMs are optimized to refine the lower and upper bounds. Kleijnen et al. [36] proposed a method for solving pure-integer constrained nonlinear problems using kriging metamodeling. Kriging is used to approximate global input and output functions per output type implied by the simulation model, while pure-integer nonlinear programming techniques are utilized to estimate the optimal solution based on kriging models.

Müller et al. [53] developed a surrogate model-based algorithm, called SO-MI, for expensive mixed-integer black-box optimization problems with general constraints. They use an RBF surrogate model to select candidate points for both continuous and integer decision variables. The objective function and the constraints are then evaluated at these points. In each iteration, the method selects four new sampling sites. Four different strategies are used to select the candidate points of each sampling site. Müller et al. [54] extended their method for solving pure-integer black-box optimization problems with general constraints. The proposed algorithm, called SO-I, consists of two phases. The first phase is used to find a feasible solution, while the second phase continues evaluating candidate points until a stopping criterion is met. Müller et al. [51] extended their work in [53] by implementing a new algorithm, called MISO, which allows various sampling strategies and adds a local search in order to find high accuracy solutions.

Recently, Costa and Nannicini [17] proposed to generate and iteratively refine an RBF surrogate model of the objective function by exploiting a noisy but less expensive oracle to accelerate convergence to the optimum of the exact oracle. Larson et al. [38] proposed a model-based approach for the global optimization of black-box convex integer problems, where an underestimator that does not require access to gradients of the objective was used. The proposed underestimator uses secant linear functions that interpolate the objective functions at previously evaluated points. The underestimator is used to generate new candidate points until global optimality has been certified.

Finally, various heuristics have been proposed for solving bound-constrained MIDFO problems. Cao et al. [13] proposed an evolutionary programming technique for handling integer variables. Schlüter et al. [64] extended to the mixed-integer case the ant colony optimization metaheuristic for continuous search domains proposed by Socha and Dorigo [67]. Liao et al. [40] also extended an ant colony optimization algorithm for continuous optimization to tackle mixed-integer problems. Laguna et al. [37] proposed a black-box metaheuristic known as scatter search for pure-integer optimization problems with general constraints.

2.3 Milestones in the field

Table 1 presents a timeline of major accomplishments in the continuous and discrete DFO from 1961 to the present. In the continuous DFO literature, the Hooke–Jeeves [30] and

Nelder–Mead [55] algorithms were the dominant approaches in the 1960s and 1970s. Genetic algorithms were proposed in 1975 [26], and many algorithms in this category have been applied to black-box problems since then. The first textbook dedicated to DFO appeared only in 2009. Researchers have been developing new DFO methods for continuous problems for over five decades now.

On the other hand, DFO algorithms that handle integer variables were not introduced until 2000. Some genetic algorithms were applied to black-box problems with integer variables (e.g. [14,24]) but the first algorithm dedicated to MIDFO was proposed in 2000 [7]. In the same year, an evolutionary programming technique for handling integer variables was proposed [13]. The first use of surrogate models in mixed-integer DFO was introduced in 2008 [25,29], and since then, the emphasis in MIDFO has shifted towards the development of methods based on surrogate models. As seen in Table 1, these developments have led to a strong recent interest in MIDFO. Table 2 lists the top-cited works in the MIDFO literature. These numbers are strong but low in comparison to the number of citations received for papers in the continuous DFO area, suggesting that MIDFO is nowhere close to being a mature area. Yet, as the next section details, a large number of software implementations have emerged that can be used in applications.

3 Mixed-integer derivative-free optimization solvers

This section discusses software implementations of bound-constrained mixed-integer derivative-free optimization algorithms reviewed in Sect. 2. All these implementations can handle both continuous and integer variables, and some of them can also handle constraints. Our aim here is to provide pointers for practitioners interested in these codes and describe the main features of each implementation. We selected solvers that are available either as open-source or commercial tools and can handle discrete variables explicitly. We used all solvers with reported computational experience in the literature for this class of problems.

3.1 BFO

Brute Force Optimizer (BFO) [71] is an open-source MATLAB implementation for nonlinear bound-constrained derivative-free optimization and equilibrium computations with continuous and discrete variables. BFO implements a direct search method. It starts by generating a sequence of feasible iterates whose objective function values are decreasing. In an attempt to find a new point with a lower objective function value, the objective function is evaluated at a finite number of points on a mesh in the neighborhood of the current iterate. The best of the improving points becomes the next iterate [59]. Discrete variables can be handled as integers or according to user-specified discrete lattices. BFO also provides checkpointing and restart facilities. Finally, BFO can be automatically trained on a set of instances in order to identify its own optimal algorithmic parameters.

3.2 DAKOTA solvers

Design Analysis Kit for Optimization and Terascale Applications (DAKOTA) [5] is a project at the Sandia National Laboratories. DAKOTA is written in C++ and it is open-source. The initial scope of DAKOTA was to create a toolkit of black-box optimization methods. However, it was later expanded to include additional optimization methods and other engineering

Table 1 Timeline of innovation in continuous and discrete DFO

Continuous DFO	Discrete DFO
(1952) Coordinate search algorithm [21]	
(1961) Hooke and Jeeves algorithm [30]	
(1962) Simplex-based algorithm [68]	
(1965) Nelder–Mead simplex algorithm [55]	
(1969) First use of trust-regions [75]	
(1975) Genetic algorithms [26]	
(1979) Hit-and-run algorithms [12]	
(1983) First use of simulated annealing [35]	
(1991) Implicit filtering [76]	
(1995) Particle swarm algorithm [20,34]	
(1997) Generalized pattern search [72]	
(1999) Multilevel coordinate search [31]	
	(2000) First MIDFO algorithm [7]
	(2000) First use of evolutionary algorithms [13]
(2003) Nonsmooth convergence analysis [8]	
(2004) Incorporation of filters [3] and simplex derivatives [18] in pattern search	
(2006) The MADS algorithm [9]	
	(2008) First use of radial basis functions [29]
	(2008) First use of stochastic surrogate models [25]
(2009) First textbook dedicated to derivative-free optimization [16]	(2009) Extension of the MADS algorithm [1]
	(2009) First use of ant colony optimization [64]
	(2012) Linesearch-type algorithms [42]
	(2014) Scatter search algorithm [37]
	(2015) Extension of a linesearch algorithm to general constraints [43]
	(2015) Quadratic surrogate models [57]
(2017) Second textbook dedicated to derivative-free optimization [10]	(2017) A directional direct search algorithm [59]
	(2019) Convexity exploitation [38]

Table 2 Citations of top-cited works in mixed-integer DFO

Publication	Year appeared	Citations[a]
Audet and Dennis [7]	2000	210
Abramson, Audet, Chrissis, and Walston [1]	2009	138
Schlüter, Egea, and Banga [64]	2009	201
Liao, Socha, de Oca, Stützle, and Dorigo [40]	2013	182
Müller, Shoemaker, and Piché [53]	2013	151

[a]From Google Scholar on 15 March 2021

applications, including the design of experiments and nonlinear least squares. DAKOTA includes several algorithms for continuous and mixed-integer, local and global derivative-free optimization. Two solvers are available for solving mixed-integer derivative-free optimization problems:[1]

1. DAKOTA/MADS: an implementation of the mesh adaptive direct-search algorithm.
2. DAKOTA/SOGA: a global optimization method that implements a single-objective genetic algorithm.

3.3 DFL solvers

Derivative-Free Library (DFL) [46] is an open-source software library for derivative-free optimization. It includes several algorithms for continuous and mixed-integer, local and global derivative-free optimization. Two solvers in this collection are available for solving mixed-integer derivative-free optimization problems:

1. DFLBOX: a derivative-free linesearch algorithm for bound-constrained mixed-integer nonlinear programming [42]. As discussed in Sect. 2.1, DFLBOX [42] consists of a continuous and a discrete search procedure that calculates the step sizes.
2. DFLGEN: a derivative-free linesearch algorithm for general constrained mixed-integer nonlinear programming [43]. DFLGEN extends DFLBOX using a sequential quadratic penalty approach [45].

DFLBOX and DFLGEN source codes are in Fortran.

3.4 MIDACO

Mixed Integer Distributed Ant Colony Optimization (MIDACO) [66] is a global evolutionary algorithm based on the ant colony optimization metaheuristic for continuous search domains proposed by Socha and Dorigo [67]. Its extension to mixed-integer domains is due to Schlüter et al. [64]. MIDACO can solve general constrained mixed-integer nonlinear programming problems by applying the oracle penalty method [65]. MIDACO's source code is in C and Fortran and provides interfaces to other programming languages as well. MIDACO is available under TOMLAB [28], a commercial optimization framework.

[1] DAKOTA/EA cannot be used to solve pure-integer derivative-free optimization problems, so it is not considered here.

3.5 MISO

MISO [52] is an open-source MATLAB implementation of the mixed-integer surrogate optimization framework presented in [51]. MISO can solve expensive black-box optimization problems with mixed-integer variables. It uses cheap surrogate models to approximate the expensive objective function and to decide at which points in the variable domain the expensive objective function should be evaluated. The framework combines different sampling strategies and local search to obtain highly-accurate solutions.

3.6 NOMAD

NOMAD [2] is an open-source C++ implementation of the LTMADS [9] and ORTHOMADS [4] methods. NOMAD is designed to solve nonlinear, nonsmooth, noisy optimization problems. It can handle categorical variables using a special step, the extended poll. NOMAD also includes a variable neighborhood search metaheuristic [6], a strategy to escape from local optima.

3.7 SNOBFIT

SNOBFIT [56] is an open-source MATLAB implementation of the branch-and-fit algorithm proposed by Huyer and Neumaier [32]. It combines global and local search by branching and local fits. Even though SNOBFIT was designed to handle only continuous variables, it offers an option that can be set to fix the resolution in the search space. When this option is set equal to 1, SNOBFIT will then search for integers only.

3.8 TOMLAB solvers

TOMLAB [28] is a commercial optimization platform and modeling language for solving optimization problems in MATLAB. It provides access to several derivative-free optimization solvers, the following of which were tested:[2,3]

- TOMLAB/GLCDIRECT [28, pp. 112–117]: an implementation of the DIRECT algorithm [33]. TOMLAB/GLCDIRECT is a Fortran implementation of TOMLAB/GLCSOLVE.
- TOMLAB/GLCFAST [28]: a fast and efficient implementation of the DIRECT algorithm [33].
- TOMLAB/GLCSOLVE [28, pp. 118–122]: a MATLAB implementation of the DIRECT algorithm [33].
- TOMLAB/MSNLP [27, pp. 9–10]: a multistart heuristic algorithm designed to find global optima of smooth constrained nonlinear programs.

[2] TOMLAB/EGO, TOMLAB/ARBFMIP, and TOMLAB/GLCCLUSTER cannot be used to solve pure-integer derivative-free optimization problems, so they are not considered here.

[3] TOMLAB/RBFSOLVE is using TOMLAB/GLCCLUSTER, so it is also not considered.

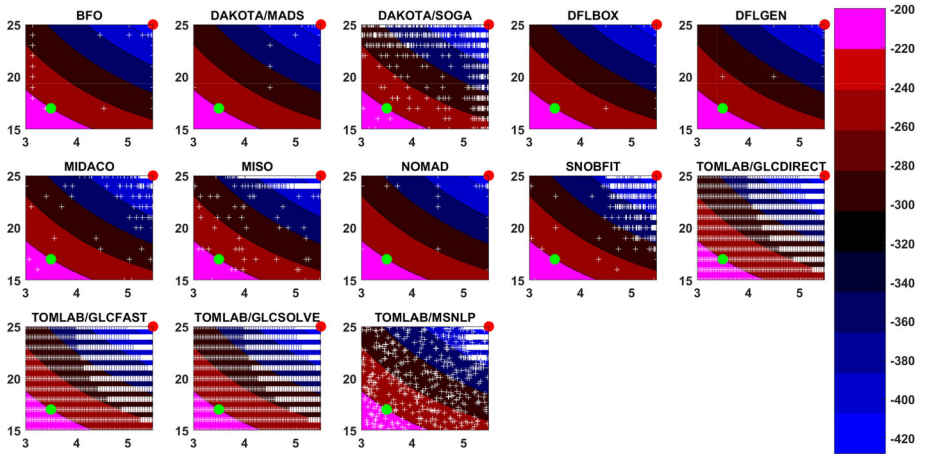

Fig. 1 Solver search strategy on st_e36. Red and blue hues represent high and low objective function values, respectively. The global minimum is located at [5.5, 25] and is marked with a magenta circle. Solvers that require a starting point were given the same starting point, which is located at [3.5, 17] and is marked with a green circle. The points evaluated by each solver are marked with white crosses, and the final solution is marked with a red circle

4 Illustrative examples

In this section, we present two illustrative examples in order to provide insights into the search strategies that are employed by the different mixed-integer DFO solvers. Each solver was limited to 2500 function evaluations. The first example was derived from problem `st_e36` in MINLPLib version 2 [74]. The original problem has two variables, one continuous and one integer, and two constraints. Since we are interested in solving bound-constrained mixed-integer problems, we omitted the constraints. The bound-constrained version of the problem is as follows:

$$\begin{aligned} \min \ & 2x_1^2 + 0.008x_2^3 - 3.2x_1x_2 - 2x_2 \\ \text{s.t.} \ & 3 \leq x_1 \leq 5.5 \\ & 15 \leq x_2 \leq 25 \\ & x_1 \in \mathbb{R}, x_2 \in \mathbb{Z} \end{aligned}$$

Figure 1 illustrates the search strategies of the thirteen derivative-free optimization solvers for the modified `st_e36`. All solvers were able to find the global optimum or a solution very close to the global optimum (within 0.0001%). The ability of most solvers to find the global optimum on small mixed-integer problems is also demonstrated in the computational study in Sect. 5. BFO, DAKOTA/MADS, DFLBOX, DFLGEN, and NOMAD performed less than 123 function evaluations. DAKOTA/SOGA performed 988 function evaluations, while MIDACO, MISO, SNOBFIT, and TOMLAB solvers reached the limit (2500 function evaluations). It is evident that global solvers need a relatively large amount of function evaluations that cover the entire search space, while local solvers terminate quickly. Also evident is the fact that deterministic solvers, such as TOMLAB/DLCDIRECT search over a pattern whereas stochastic solvers, such as TOMLAB/MSNLP, make random moves.

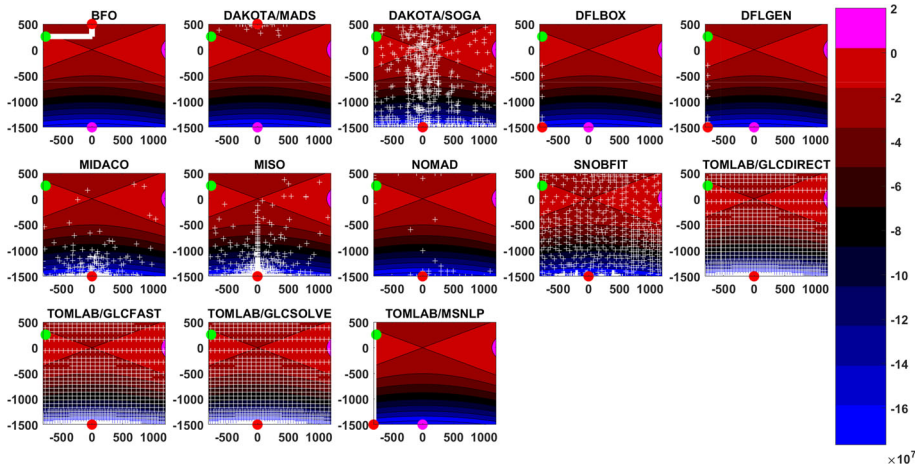

Fig. 2 Solver search strategy on synthetic example with large search domain. Red and blue hues represent high and low objective function values, respectively. The global minimum is marked with a magenta circle. Solvers that require a starting point were given the same starting point, which is located at $[-755, 255]$ and is marked with a green circle. The points evaluated by each solver are marked with white crosses and the final solution is marked with a red circle

We also constructed for experimentation the following example with two integer variables:

$$\begin{aligned} \min\ & 15.71x_1^2 - \log(1500.37 + x_1) - 78.54x_2^2 + \log(2500.37 + x_2) \\ \text{s.t.}\ & -800 \leq x_1 \leq 1200 \\ & -1500 \leq x_2 \leq 500 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

The global optimum for this problem occurs at $x_1 = 0$ and $x_2 = -1500$. Many solvers are unable to find this point due to the large size of the search domain. Such problems are very common in practice and in the test library that we used in the computational experiments of this paper. Figure 2 illustrates the search strategies of the thirteen derivative-free optimization solvers for this problem. DAKOTA/SOGA, MIDACO, MISO, NOMAD, SNOBFIT, TOMLAB/GLCDIRECT, TOMLAB/GLCFAST, and TOMLAB/GLCSOLVE found the global optimum or a solution very close to it (within less than 5% difference). BFO, DAKOTA/MADS, DFLBOX, DFLGEN, and TOMLAB/MSNLP improved the solution of the starting point locally. DAKOTA/MADS, DFLBOX, DFLGEN, and NOMAD performed less than 79 function evaluations. DAKOTA/SOGA performed 921 function evaluations, BFO performed 1492 function evaluations, while MIDACO, MISO, SNOBFIT, and TOMLAB solvers reached the limit (2500 function evaluations). It is again clear that global solvers need a relatively large number of function evaluations and cover the entire search space, while local solvers, except for BFO, terminate quickly. Most local solvers usually terminate quickly after improving the solution of the starting point locally. Although NOMAD is regarded as a local solver, it includes techniques to escape local minima. That enables it to find solutions near the global optimum.

5 Computational comparisons

This section presents a systematic comparison of thirteen available implementations of derivative-free optimization algorithms on bound-constrained mixed-integer problems. The testbed includes: (i) pure-integer and mixed-integer problems, and (ii) small, medium, and large problems covering a wide range of characteristics found in applications. We evaluate the solvers according to their ability to find a near-optimal solution, find the best solution among currently available solvers, and improve a given starting point.

5.1 Experimental setup

Since most derivative-free optimization solvers are designed for low-dimensional unconstrained problems, we consider problems with a maximum of 500 variables with bounds only. The thirteen derivative-free optimization solvers presented in Sect. 3 were tested on 267 problems from the MINLPLib 2 library. Most of the original problems from the MINLPLib 2 library have constraints. In this paper, we are interested in solving bound-constrained mixed-integer problems, so we omitted the constraints. We also eliminated the variables that were redundant after eliminating the constraints. Moreover, we also used 79 continuous problems from the MINLPLib 2 library and imposed integrality constraints on the variables in order to have a representative sample of non-binary discrete problems. Table S1 in the Online Supplement provides a complete listing of the test problems and model statistics. We used the general-purpose global optimization solver BARON [70] to obtain the global solution of each problem.

The computational experiments were performed on an Intel Xeon CPU W-2123 with 32 GB of main memory and a clock of 3.6 GHz, running under Centos 7 64-bit. All solvers were tested using a limit of 2500 function evaluations for each run. All solvers require variable bounds except for NOMAD. For problems with missing bounds in the problem formulation, we restricted all variables to the interval $[-10,000, 10,000]$. Whenever starting points were required, they were drawn from a uniform distribution from the box-bounded region. We generated five random starting points for each problem. Solvers that use the provided starting point (BFO, DAKOTA/MADS, DFLBOX, DFLGEN, MIDACO, NOMAD, SNOBFIT, TOMLAB/MSNLP) ran once from each of the five different starting points. The same randomly generated starting points were used for all solvers. MISO supplements the provided starting point with a set of points sampled via its default sampling strategy, i.e., symmetric Latin hypercube sampling. DAKOTA/SOGA, does not use the provided starting points but uses randomly chosen starting points, thus it was also ran five times. All other solvers that do not use the provided starting point and are deterministic solvers (TOMLAB/GLCDIRECT, TOMLAB/GLCFAST, TOMLAB/GLCSOLVER) were ran once.

In order to assess the quality of the solutions obtained by different solvers, we compared the solution obtained by the derivative-free optimization solvers against the globally optimal solution for each problem. A solver was considered to have successfully solved a problem if it returned a solution with an objective function value within 1% or 0.01 of the global optimum, whichever was larger. Since we performed five runs for each solver that utilizes the provided starting point, starting each time from a different starting point, we compared the average- and best-case behavior of each solver. Finally, we used the default algorithmic parameters for each solver, i.e., we did not tune solvers in any way to the problems at hand. Table 3 lists the specific versions of solvers used in this computational study.

Table 3 Derivative-free optimization solvers used in this computational study

Solver	Version
BFO	2
DAKOTA/MADS	6.13
DAKOTA/SOGA	6.13
DFLBOX	–
DFLGEN	–
MIDACO	8.7
MISO	–
NOMAD	3.9.1
SNOBFIT	2.1
TOMLAB/GLCDIRECT	8.7
TOMLAB/GLCFAST	8.7
TOMLAB/GLCSOLVE	8.7
TOMLAB/MSNLP	8.7

5.2 Computational results

Tables S2–S14 in the Online Supplement provide for each solver the median over the five optimization runs. Tables S15–S26 present the best-case performance out of all five runs. For each solver, we report the execution time, the number of iterations (function evaluations), the solution, and the optimality gap (% difference between solution returned by the solver and the global solution). A dash ("–") is used when the optimality gap is larger than or equal to 100%. In order to compare the quality of solutions returned, we compared the average- and best-case behavior of each solver. For the average-case behavior, we compared solvers using the median objective function value over the five different runs. For the best-case comparison, we compared the best solution found by each solver after all five runs. Best-case behavior is presented in the figures and analyzed below unless explicitly stated otherwise. The figures in this subsection are performance profiles [50] and present the fraction of problems solved by each solver within an optimality tolerance of 1%. The figures in Section B of the Online Supplement present the fraction of problems for which each solver achieved a solution as good as the best solution among all solvers, without regard to the global solution of the problems. When multiple solvers achieved the same solution, they were all credited as having the best solution among the solvers.

Figure 3 presents the fraction of problems solved by each solver. The horizontal axis shows the progress of a solver as the number of function evaluations gradually reaches 2500. If we only consider the solutions obtained by the solvers at the 2500 function evaluation limit, the best solvers, NOMAD and MISO, solved 77% and 76% of the problems, respectively. SNOBFIT solved 69% of the problems, while DAKOTA/MADS solved 56% of the problems. Most of the remaining solvers were able to solve more than 32% of the problems. TOMLAB/MSNLP had the worst performance, solving only 12% of the problems. Only four solvers could find an optimal solution for more than half of the problems. It is evident that most solvers can solve only a small number of problems. Next, we investigate the performance of solvers on different subsets of problems, dividing the problems based on the number and type of variables involved (mixed-integer or pure-integer).

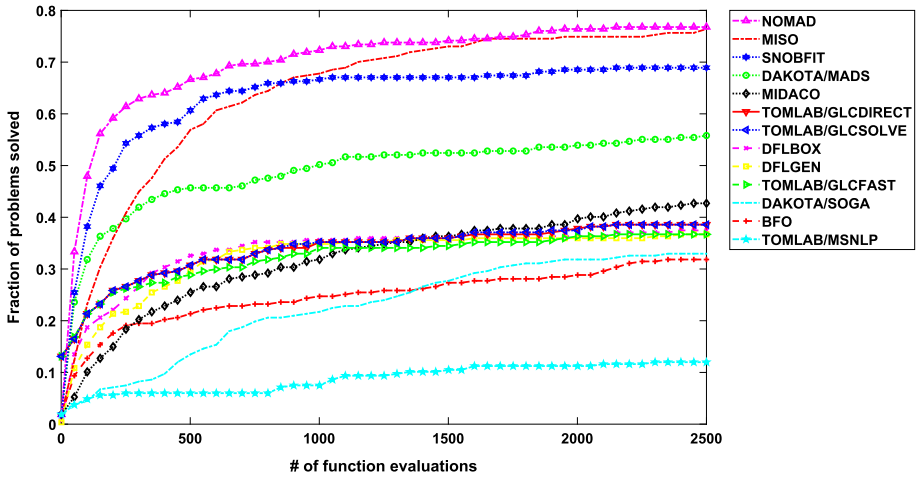

Fig. 3 Fraction of problems solved as a function of allowable number of function evaluations

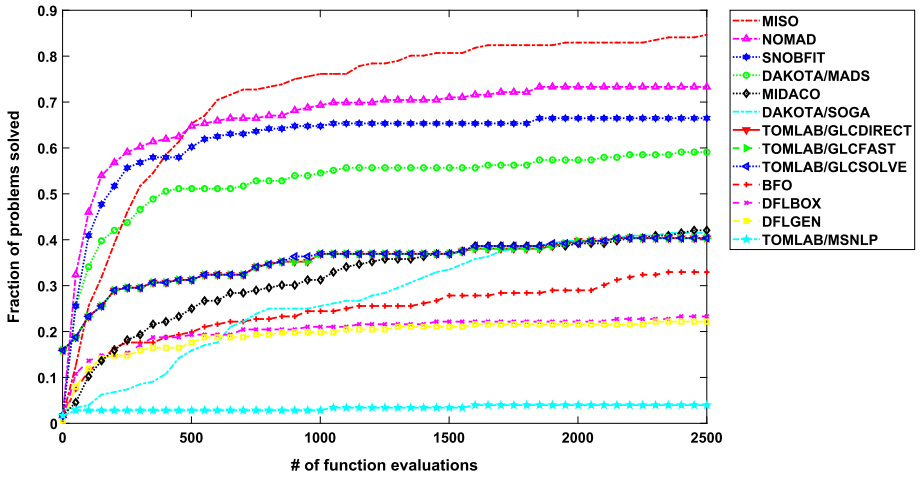

Fig. 4 Fraction of pure-integer problems solved as a function of allowable number of function evaluations

The test set includes 176 pure-integer problems and 91 mixed-integer problems. Figure 4 presents the fraction of pure-integer problems solved by each solver within the optimality tolerance. MISO found the optimal solution on 85% of the problems, while NOMAD found the optimal solution 73% of the problems. SNOBFIT and DAKOTA/MADS found the optimal solution on 66% and 59% of the problems, respectively. DAKOTA/SOGA, MIDACO, and TOMLAB solvers found the optimal solution on 40−42% of the problems. BFO found the optimal solution on 33% of the problems, while DFLBOX, DFLGEN, and TOMLAB/MSNLP had the worst performance, solving only less than 24% of the pure-integer problems. MISO is clearly the best solver for solving these pure-integer problems. DFLBOX, DFLGEN, and TOMLAB/MSNLP are not good options for solving this collection of pure-integer problems.

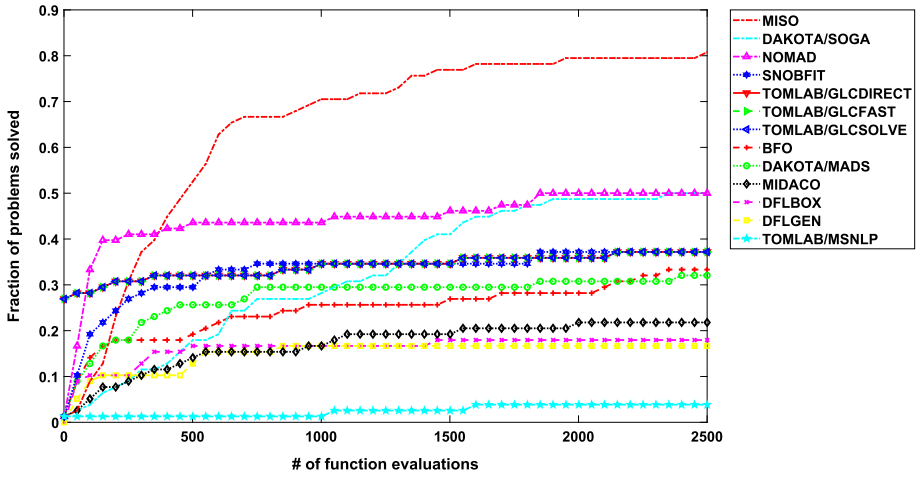

Fig. 5 Fraction of binary problems solved as a function of allowable number of function evaluations

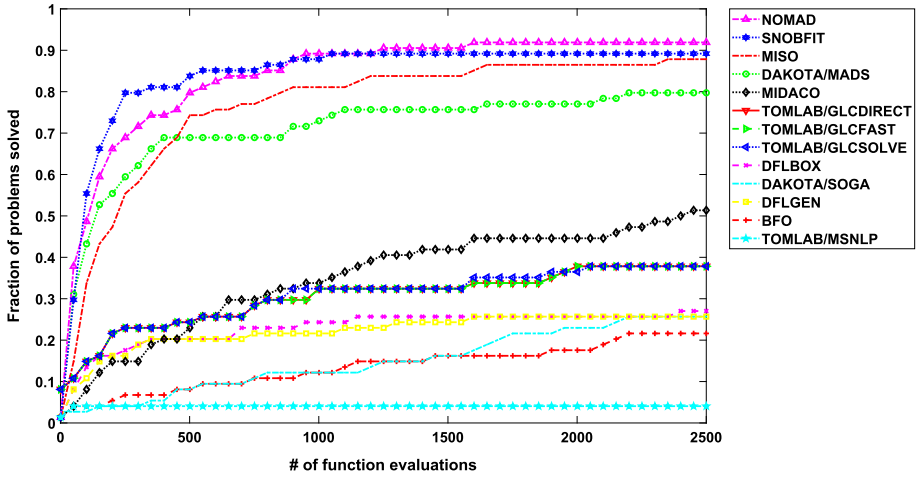

Fig. 6 Fraction of non-binary discrete problems solved as a function of allowable number of function evaluations

We also study the performance of algorithms on binary and non-binary discrete problems. The test set includes 78 binary and 74 non-binary discrete problems. Figures 5 and 6 present the fraction of binary and non-binary discrete problems, respectively, solved by each solver within the optimality tolerance. MISO outperforms all other solvers on binary problems. More specifically, MISO can solve 81% of the binary problems, almost twice as many as the second best performers, DAKOTA/SOGA and NOMAD, can solve. All other solvers can solve less than 39% of binary problems. DAKOTA/MADS, MISO, NOMAD, and SNOBFIT are the best performers on non-binary discrete problems, solving 80−92% of the problems. MIDACO can solve 51% of these problems, while all other solvers can solve less than 38% of the problems.

Figure 7 presents the fraction of mixed-integer problems solved by each solver. NOMAD leads over the entire range of function evaluations, finding an optimal solution on 84%

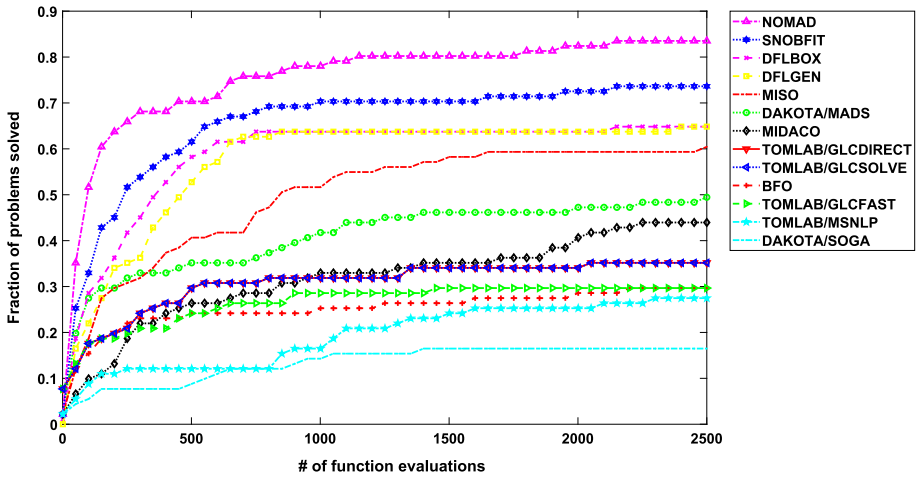

Fig. 7 Fraction of mixed-integer problems solved as a function of allowable number of function evaluations

of the problems. DFLBOX, DFLGEN, MISO, and SNOBFIT are also performing well in this category, solving 60−74% of the problems. Contrary to their poor performance on pure-integer problems, DFLBOX and DFLGEN are able to solve a significant number of mixed-integer problems. On the other hand, DAKOTA/SOGA performed much better on pure-integer problems (solved 31% of the problems) than on mixed-integer problems (solved 13% of the problems). DAKOTA/SOGA had the worst performance, solving only 16% of the problems. NOMAD is the best solver for solving this collection of mixed-integer problems, followed by DFLBOX, DFLGEN, MISO, and SNOBFIT.

The computational results show that only two solvers can solve more than half of the problems. One factor that may significantly impact solver performance is the problem size. To investigate the effect of size on problem performance, we divided the problem set into three categories: (i) small problems with one to ten variables, (ii) medium problems with 11 to 50 variables, and (iii) large problems with 51 to 500 variables. The problem set includes 52 small problems, 102 medium problems, and 113 large problems.

Figure 8 presents the fraction of small problems solved by each solver within the optimality tolerance. Eight solvers were able to solve more than 84% of the problems with one to 10 variables. More specifically, DAKOTA/MADS and NOMAD found an optimal solution on all of the problems. Additionally, DAKOTA/MADS solved all small problems in less than 178 function evaluations on average. SNOBFIT found an optimal solution on 97% of the problems, while MIDACO and MISO found an optimal solution on 96% of the problems. TOMLAB/GLCDIRECT, TOMLAB/GLCFAST, and TOMLAB/GLCSOLVE found an optimal solution on 84% of the problems. TOMLA/MSNLP had the worst performance, only solving 40% of the problems.

Figure 9 presents the fraction of medium problems solved by each solver within the optimality tolerance. Similar to small problems, NOMAD was the best solver, solving 93% of the problems with 11 to 50 variables. DAKOTA/MADS, MISO, and SNOBFIT are also performing well, solving 82% and 85% of the problems, respectively. On the other hand, TOMLAB/MSNLP had the worst performance on this collection by solving only 11% of the problems.

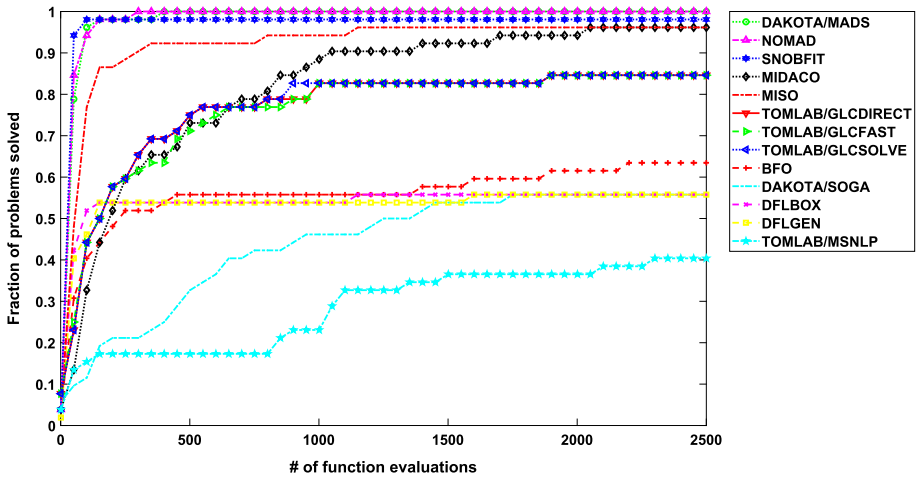

Fig. 8 Fraction of small problems (one to ten variables) solved as a function of allowable number of function evaluations

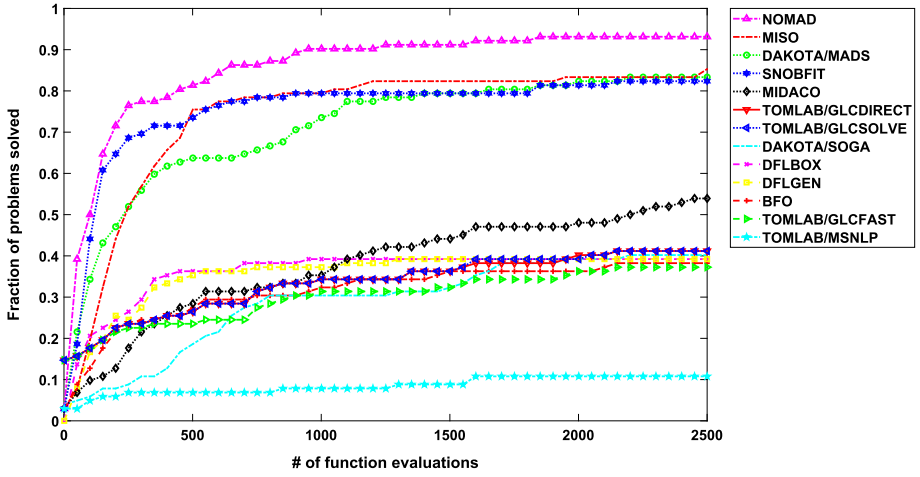

Fig. 9 Fraction of medium problems (11 to 50 variables) solved as a function of allowable number of function evaluations

Figure 10 presents the fraction of large problems solved by each solver. All solvers had lower success rates for these problems in comparison to their performance for smaller problems. MISO was able to solve 59% of the problems, followed by NOMAD and SNOBFIT that solved 51% and 43% of the problems, respectively. The remaining solvers solved fewer than 28% of the problems. TOMLAB/MSNLP did not solve any of these problems.

5.3 Improvement from starting point

Moré and Wild [50] proposed a benchmarking procedure for derivative-free optimization solvers that measures each solver's ability to improve a starting point. For a given $0 \leq \tau \leq 1$ and starting point x_0, a solver is considered to have successfully improved the starting point

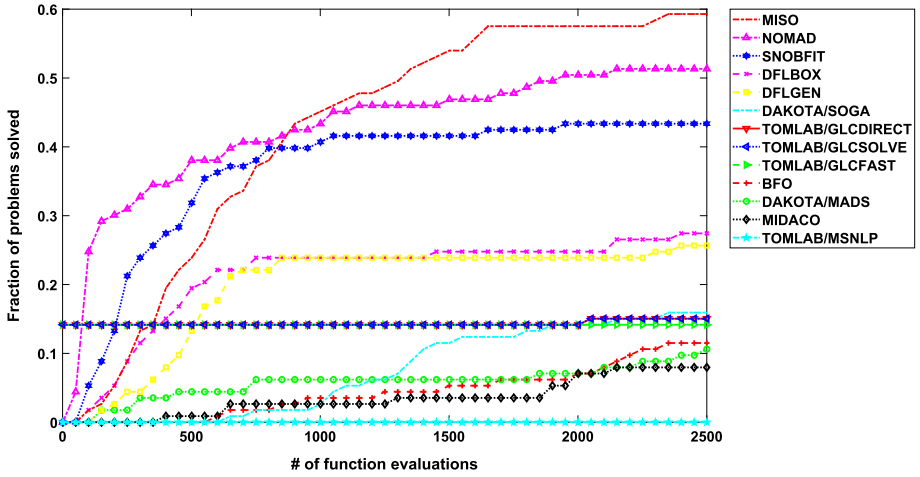

Fig. 10 Fraction of large problems (51 to 500 variables) solved as a function of allowable number of function evaluations

if

$$f_{x_0} - f_{solver} \geq (1 - \tau)(f(x_0) - f_L)$$

where $f(x_0)$ is the objective value at the starting point, f_{solver} is the solution reported by the solver, and f_L is the global solution. We used this measure to evaluate the best-case performance of each solver. In other words, a problem was considered solved by a solver if the best solution from the five runs improved the associated starting point by at least a fraction of $1 - \tau$ of the largest possible reduction. The starting points were drawn from a uniform distribution from the box-bounded region.

Figure 11 presents the fraction of problems for which the starting point was improved. NOMAD improved the starting points for 85% of the problems for $\tau = 1\text{e}{-1}$, and its ability to improve the starting points is slightly reduced for smaller values of τ. MISO improved the starting points for 96% of the problems for $\tau = 1\text{e}{-1}$, but its ability to improve the starting points dropped considerably for smaller values of τ. SNOBFIT improved the starting points for 80% of the problems for $\tau = 1\text{e}{-1}$, and its ability to improve the starting points is slightly reduced for smaller values of τ. DAKOTA/SOGA and MIDACO are also performing well for $\tau = 1\text{e}{-1}$ but they are not very efficient for larger values of τ.

In Section C of the Online Supplement, we present the fraction of problems for which starting points were improved for each type of problem, i.e., (i) pure-integer and mixed-integer problems, and (ii) small, medium and large problems. The results are very similar to those in the figures of this section and demonstrate higher success rates for smaller problems. More specifically, NOMAD leads over most values of τ in all categories. DAKOTA/SOGA improved the starting points for larger values of τ, but its performance dropped considerably for smaller values of τ.

5.4 Minimal sufficient set of solvers

The computational results revealed that MISO and NOMAD are clearly superior to other MIDFO solvers for the problems considered in this study. However, neither solver was able

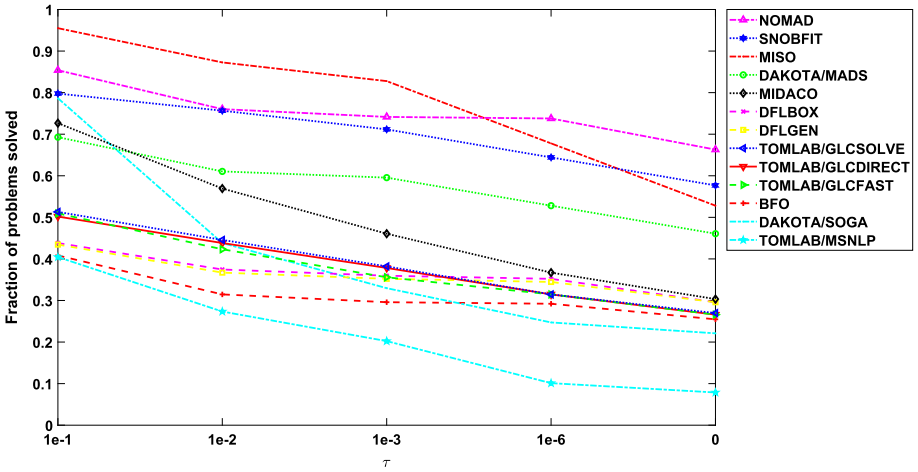

Fig. 11 Fraction of problems for which starting points were improved within 2500 function evaluations vs. τ values

to find an optimal solution for all our test problems. For instance, NOMAD solved only 51% of the large problems, while MISO solved only 60% of the mixed-integer problems. Therefore, it is worthwhile to find a minimal cardinality subset of the solvers capable of collectively solving as many problems in our test set as possible. Toward this end, we considered the best solution derived by each solver in the five runs. For each problem size, we first identify all problems that can be solved. Then, we determine the smallest number of solvers that collectively solve all these problems. A solver is not included in a minimally sufficient set of solvers if the problems solved by this solver form a strict subset of the problems solved by another solver. Figure 12 presents the minimum number of solvers required to solve the problems in our collection broken down by problem type. BFO, DAKOTA/SOGA, DFLBOX, MISO, NOMAD, and SNOBFIT collectively solve 93% of all problems and 92% of the pure-integer problems. Finally, DFLBOX, NOMAD, and SNOBFIT collectively solve 95% of the mixed-integer problems. Interestingly, even though MISO is a very good solver for all problem classes, it is not in the minimal set of solvers for the mixed-integer problems because it is completely dominated by NOMAD for these problems. On the other hand, MISO contributes the most on the solution of the pure-integer problems.

Figure 13 presents the minimum sufficient number of solvers as a function of problem size. NOMAD solved all small problems and 93% of the medium problems, while MISO solved 54% of the large problems. Finally, BFO, DFLBOX, MISO, NOMAD, and SNOBFIT collectively solved 85% of the large problems.

5.5 Variance of the results

The previous figures were presented in terms of the best results among five problem instances for each solver with a limit of 2500 function evaluations. In this subsection, we discuss the variance of the results, as many solvers have varying performance as a function of the starting point given as input and random seeds used in the computations. Although DAKOTA/SOGA does not utilize the provided starting points, it was executed five times since it is a stochastic solver. TOMLAB/GLCDIRECT, TOMLAB/GLCFAST, and TOMLAB/GLCSOLVER were

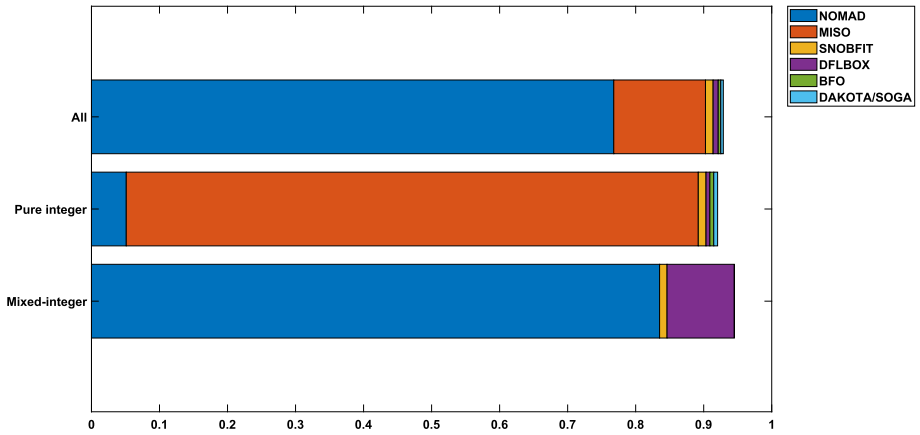

Fig. 12 Minimum sufficient number of solvers as a function of problem type

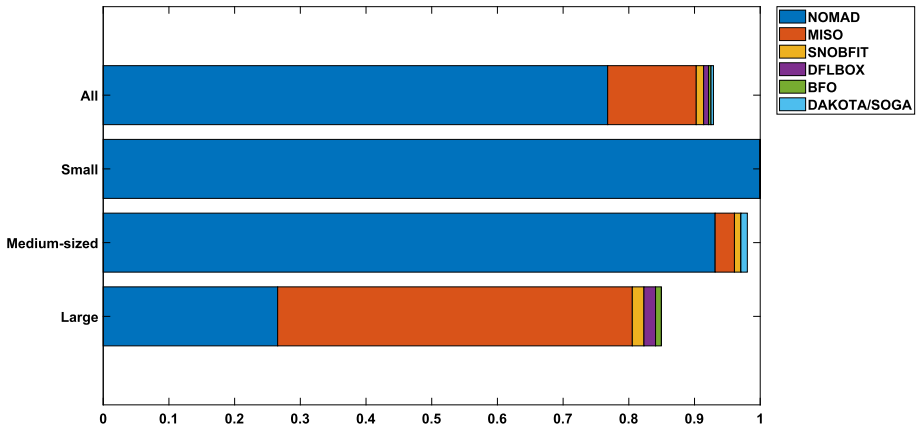

Fig. 13 Minimum sufficient number of solvers as a function of problem size

only run once since they do not utilize the provided starting points and they are deterministic solvers.

The difference in scales of the global solutions and the range of values of the objective function of the test problems make a direct comparison difficult. Therefore, we scale the objective function values as follows:

$$f_{scaled} = 1 - \frac{|f_L - f_{solver}|}{(1\text{e}{-10} + |f_L|)}$$

where f_{solver} is a solution obtained by the solver and f_L is the global solution. If $f_{scaled} < 0$ (i.e., the optimality gap is larger than 100%), we set the scaled objective function value equal to 0. Hence, the resulting scaled objective function value is in the interval [0, 1]. A value of 1 corresponds to the global solution, while a value of 0 corresponds to a solution with an optimality gap of larger than or equal to 100%.

Figure 14 presents the average scaled best, mean, median, and worst results among the five optimization instances for all test problems. BFO, DAKOTA/MADS, DFLBOX, DFLGEN, MIDACO, MISO, NOMAD, SNOBFIT, and TOMLAB/MSNLP use the starting point provided, while the remaining solvers do not use it. As expected, most local solvers and global

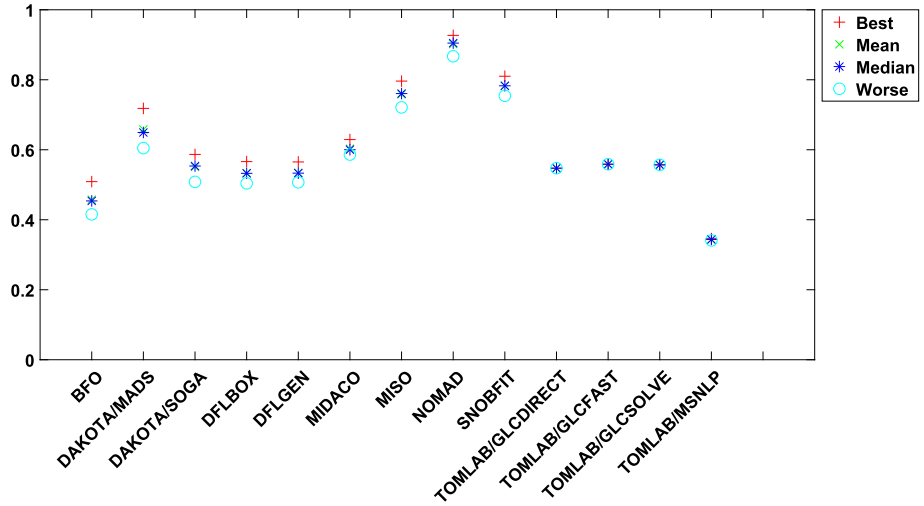

Fig. 14 Scaled results for the best, mean, median, and worst result among the five optimization instances for each solver (larger values are better)

stochastic and hybrid solvers produce varying results in the different runs because of the starting point given as input and the random seeds used in the computations. However, solution variability is small for all solvers, indicating that starting points and random seeds do not significantly affect them. The solver with the largest variability is DAKOTA/MADS.

5.6 Computational effort

In this section, two metrics are considered in comparing the computational effort of different solvers: (i) the number of functions evaluations required by each solver, and (ii) each solver's execution time (CPU time). By combining former analysis on solution quality and computational effort, further analysis is then proposed to show solver efficiency in solving the problems of the testbed. Depending on whether or not an evaluation of the objective function is time-consuming, different metrics are more important for obtaining a solution faster. Since the test problems are algebraically and computationally simple and small, the total time required for function evaluations for all runs was negligible. Most of the solvers' execution time was spent on processing function values and determining the sequence of iterates. In cases where the evaluation of the objective function is not time-consuming, the execution time of the solvers is more important. However, in applications where an evaluation of the objective function requires a significant amount of time, the number of function evaluations that the solvers perform is the factor that determines computational efficiency. Global optimization methods will perform more iterations and will likely require more execution time than local methods.

Tables 4 and 5 present the computational efficiency of the solvers in terms of function evaluations and execution time. Table 4 shows that function evaluations of various solvers differ greatly. MIDACO, MISO, SNOBFIT, and all TOMLAB solvers require more than 2000 function evaluations. In some cases, this is due to the solver performing a large number of samples at early function evaluations. In other cases, solvers employ restart strategies to minimize the likelihood of getting trapped in local solutions. As mentioned before, these

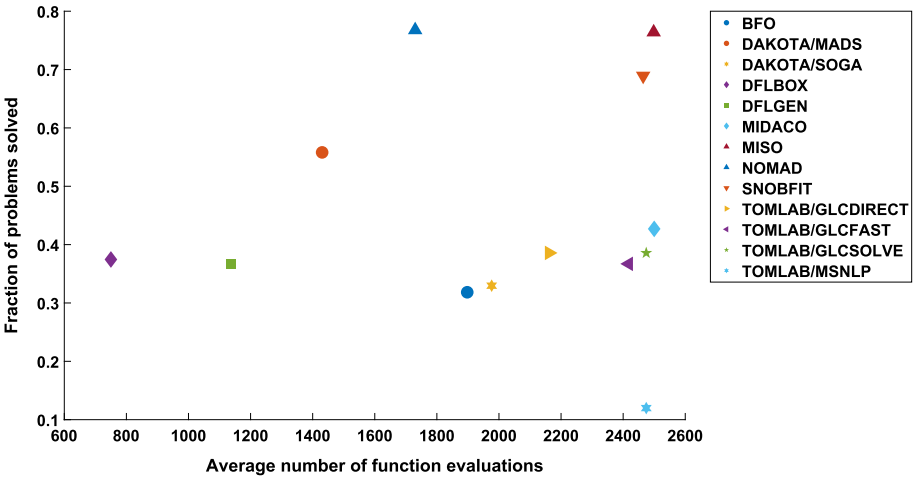

Fig. 15 Efficiency of solvers as a function of the number of function evaluations

methods are global optimization methods and thus require more function evaluations. BFO, DAKOTA/MADS, DAKOTA/SOGA, and NOMAD require 1400 to 2000 function evaluations, while DFLBOX, and DFLGEN require fewer than 1138 function evaluations. On the other hand, most solvers can find the best solution on relative few function evaluations. DFLBOX, DFLGEN, and NOMAD find their best solution in less than 650 function evaluations on average. In terms of execution time, all solvers, except for the DAKOTA solvers, MISO, NOMAD, and SNOBFIT, need less than 43 seconds on average to solve the instances, regardless of problem size. The DAKOTA solvers, MISO, NOMAD, and SNOBFIT require considerably more time to solve the problems. DAKOTA, MISO, and SNOBFIT demand CPU times that increase with problem size. Interestingly, NOMAD requires fewer iterations for large than medium problems and solves large problems faster. Even though MIDACO and TOMLAB solvers implement global optimization methods, they do not require much time compared to other global methods like MISO, NOMAD, and SNOBFIT.

Figure 15 presents the efficiency of each solver in terms of the number of function evaluations performed. The average number of function evaluations, represented by the horizontal axis, presents the computational effort of each solver, while the vertical axis indicates the quality of solutions in terms of the percentage of problems solved. Solvers that are located on the upper left corner of the figure indicate good efficiency. Approaches found on the lower right area indicate poor efficiency. BFO, DAKOTA/MADS, DAKOTA/SOGA, DFLGEN, MIDACO, MISO, NOMAD, SNOBFIT, TOMLAB/GLCDIRECT, TOMLAB/GLCFAST, and TOMLAB/GLCSOLVE required more than 1135 function evaluations and solved more than 31% of the problems. The least efficient solver is TOMLAB/MSNLP, which required 2474 function evaluations on average and solved 12% of the problems. DFLBOX required only 750 function evaluations on average and solved 38% of the problems. Figure 15 can also be interpreted as a Pareto front, in which case DAKOTA/MADS, DFLBOX, and NOMAD dominate all others.

Figure 16 presents the efficiency of each solver in terms of execution time. Even though MISO solved 76% of the problems, it required 4385 seconds on average. Similarly, NOMAD solved 77% of the problems, but it required 2229 seconds on average. DAKOTA/MADS solved 56% of the problems requiring 1266 seconds on average, while SNOBFIT solved

Table 4 Computational effort of solvers in terms of function evaluations (tot: average total number of function evaluations, opt: average function evaluations by which best solution was found)

Solver	All problems		Small problems		Medium problems		Large problems	
	tot	opt	tot	opt	tot	opt	tot	opt
BFO	1898	1788	1430	1247	1682	1580	2309	2225
DAKOTA/MADS	1430	1247	152	77	944	787	2459	2201
DAKOTA/SOGA	1977	1797	1302	1041	1970	1748	2295	2190
DFLBOX	750	361	131	92	387	168	1365	659
DFLGEN	1136	482	237	113	841	241	1818	869
MIDACO	2500	1642	2500	720	2500	1547	2500	2110
MISO	2498	1084	2492	677	2500	809	2500	1518
NOMAD	1730	649	586	63	2251	412	1787	1132
SNOBFIT	2464	618	2373	52	2471	463	2500	1018
TOMLAB/GLCDIRECT	2162	1065	2311	862	2291	1245	1979	995
TOMLAB/GLCFAST	2418	1440	2325	831	2406	1357	2473	1794
TOMLAB/GLCSOLVE	2474	1235	2377	887	2496	1374	2500	1269
TOMLAB/MSNLP	2474	1347	2377	873	2496	1393	2500	1524

Table 5 Computational effort of solvers in terms of average execution time (s)

Solver	All problems	Small problems	Medium problems	Large problems
BFO	16	12	13	21
DAKOTA/MADS	1266	11	228	2781
DAKOTA/SOGA	693	68	247	1382
DFLBOX	26	3	14	48
DFLGEN	42	6	33	68
MIDACO	21	19	20	23
MISO	4385	1120	1723	8290
NOMAD	2229	1042	3613	1526
SNOBFIT	208	57	81	391
TOMLAB/GLCDIRECT	19	18	18	20
TOMLAB/GLCFAST	21	19	19	24
TOMLAB/GLCSOLVE	22	20	21	25
TOMLAB/MSNLP	21	18	19	23

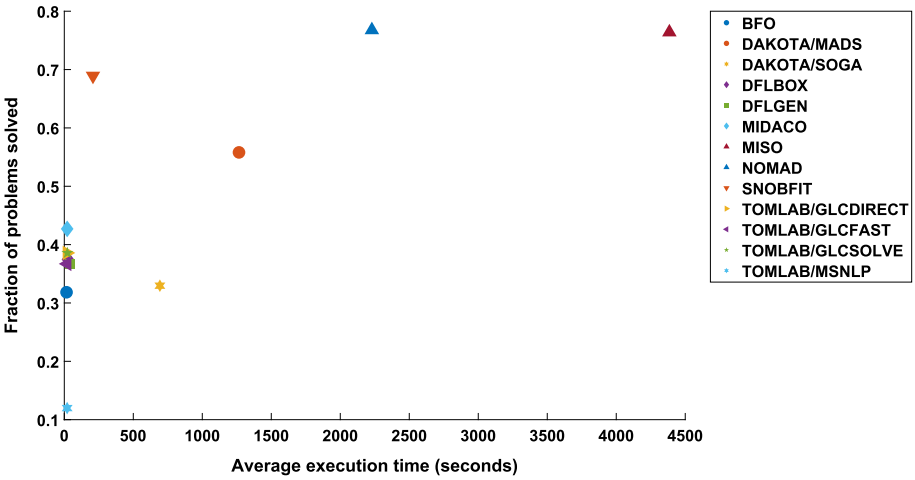

Fig. 16 Efficiency of solvers in terms of execution time

69% of the problems requiring 208 seconds. The best solvers in terms of time efficiency were MIDACO, TOMLAB/GLCDIRECT, TOMLAB/GLCFAST, and TOMLAB/GLCSOLVE, which solved 37−43% of the problems, requiring less than 23 seconds. BFO, DFLBOX, and DFLGEN were less efficient than the aforementioned solvers as they solved more than 31% of the problems requiring less than 43 seconds. The remaining solvers were not very efficient because of either large execution times (DAKOTA/SOGA) or the small fraction of problems that they solved (TOMLAB/MSNLP).

6 Conclusions

Although significant progress has been made on the algorithmic and theoretical aspects of derivative-free optimization over the past three decades, algorithms addressing problems with discrete variables have not yet attracted much attention. In order to assess the current state-of-the-art in this area, in this paper, we presented a systematic comparison of thirteen mixed-integer derivative-free optimization implementations on a large collection of test problems. Our computational results show that the existing solvers cannot solve large problems and pure-integer problems efficiently.

Figure 17 presents the fraction of problems solved to optimality by all solvers collectively. Solvers were able to solve 93% of the problems. Mixed-integer problems are easier to be solved to optimality than pure-integer problems. Solvers found the optimal solution for 95% of the mixed-integer problems and 92% of the pure-integer problems. Moreover, non-binary discrete problems are easier to be solved to optimality than binary problems. Solvers were able to solve 85% and 99% of the binary and non-binary discrete problems, respectively. Additionally, solvers are very efficient when solving problems with up to 50 variables. More specifically, solvers found the optimal solution on all small problems (one to ten variables), 98% of the medium problems (11 to 50 problems), and 85% of the large problems (51 to 500 variables). Moreover, solvers found the optimal solution on 79% of the large pure-integer problems.

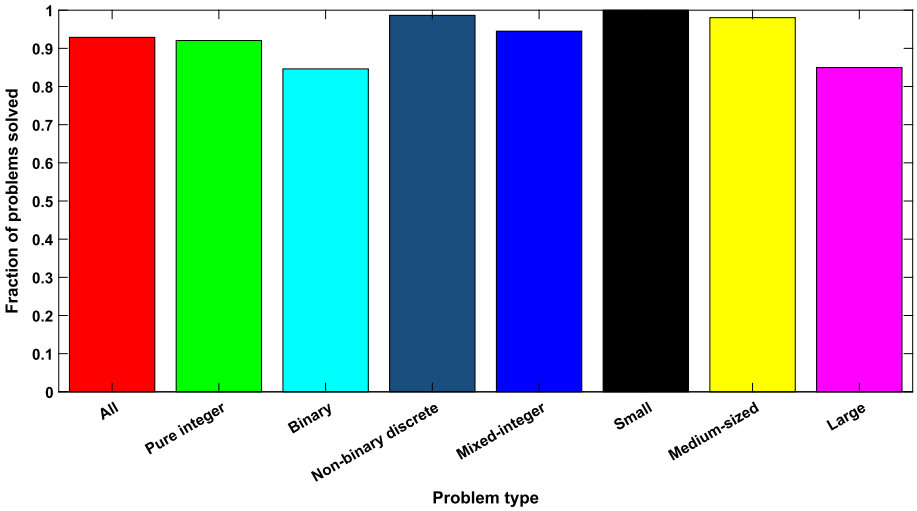

Fig. 17 Problems solved by all solvers collectively as a function of problem type and size

The open-source solvers MISO and NOMAD provided the best solutions among all the solvers tested. MISO is the best performer on large and binary problems, while NOMAD outperforms all solvers on mixed-integer, non-binary discrete, small, and medium-sized problems. DAKOTA/MADS and SNOBFIT also performed well on most types of problems. BFO, DFLBOX, MISO, NOMAD, and SNOBFIT collectively solve 93% of the problems used in this study.

Overall, existing algorithms in this field are complementary in their performance. They should be used collectively rather than individually. The collection of current solvers is capable of solving most small and even medium-sized problems. Clearly, there is a need for the development of algorithms and implementations for large-scale problems. Future work should also investigate the computational performance of derivative-free optimization algorithms on constrained problems.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10898-021-01085-0.

Acknowledgements We thank the anonymous referees, whose constructive comments helped improve our manuscript.

Data availability All data generated or analyzed during this study are available from https://sahinidis.coe.gatech.edu/bbo?q=midfo.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Abramson, M.A., Audet, C., Chrissis, J.W., Walston, J.G.: Mesh adaptive direct search algorithms for mixed variable optimization. Optim. Lett. **3**, 35–37 (2009)
2. Abramson, M.A., Audet, C., Couture, G., Dennis, Jr., J.E., Le Digabel, S.: The NOMAD project (Current as of 15 March, 2021). http://www.gerad.ca/nomad/
3. Abramson, M.A., Audet, C., Dennis, J.E., Jr.: Filter pattern search algorithms for mixed variable constrained optimization problems. Department of Computational and Applied Mathematics, Rice University, Tech. rep. (2004)
4. Abramson, M.A., Audet, C., Dennis, J.E., Jr., Le Digabel, S.: OrthoMADS: a deterministic MADS instance with orthogonal directions. SIAM J. Optim. **20**, 948–966 (2009)
5. Adams, B.M., Ebeida, M.S., Eldred, M.S., Geraci, G., Jakeman, J.D., Maupin, K.A., Monschke, J.A., Swiler, L.P., Stephens, J.A., Vigil, D.M., Wildey, T.M., Bohnhoff, W.J., Dalbey, K.R., Eddy, J.P., Hooper, R.W., Hu, K.T., Hough, P.D., Ridgway, E.M., Rushdi, A.: DAKOTA, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: version 6.5 user's manual. Sandia National Laboratories, Albuquerque, NM and Livermore, CA (2016). https://dakota.sandia.gov/
6. Audet, C., Béchard, V., Le Digabel, S.: Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. J. Glob. Optim. **41**, 299–318 (2008)
7. Audet, C., Dennis, J.E., Jr.: Pattern search algorithms for mixed variable programming. SIAM J. Optim. **11**, 573–594 (2000)
8. Audet, C., Dennis, J.E., Jr.: Analysis of generalized pattern searches. SIAM J. Optim. **13**, 889–903 (2003)
9. Audet, C., Dennis, J.E., Jr.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**, 188–217 (2006)
10. Audet, C., Hare, W.: Derivative-free and Blackbox Optimization. Springer, Cham (2017)
11. Audet, C., Le Digabel, S., Tribes, C.: The mesh adaptive direct search algorithm for granular and discrete variables. SIAM J. Optim. **29**, 1164–1189 (2019)
12. Boneh, A., Golan, A.: Constraints' redundancy and feasible region boundedness by random feasible point generator (RFPG). In: Third European Congress on Operations Research (EURO III). Amsterdam (1979)
13. Cao, Y., Jiang, L., Wu, Q.: An evolutionary programming approach to mixed-variable optimization problems. Appl. Math. Model. **24**, 931–942 (2000)
14. Chipperfield, A.J., Fleming, P.J., Fonseca, C.M.: Genetic algorithm tools for control systems engineering. In: Proceedings of Adaptive Computing in Engineering Design and Control, vol. 128, p. 133 (1994)
15. Ciccazzo, A., Latorre, V., Liuzzi, G., Lucidi, S., Rinaldi, F.: Derivative-free robust optimization for circuit design. J. Optim. Theory Appl. **164**, 842–861 (2015)
16. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-free Optimization. SIAM, Philadelphia (2009)
17. Costa, A., Nannicini, G.: RBFOpt: an open-source library for black-box optimization with costly function evaluations. Math. Program. Comput. **10**, 597–629 (2018)
18. Custódio, A.L., Vicente, L.N.: Using sampling and simplex derivatives in pattern search methods. SIAM J. Optim. **18**, 537–555 (2007)
19. Davis, E., Ierapetritou, M.: A kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions. J. Glob. Optim. **43**, 191–205 (2009)
20. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43. Nagoya, Japan (1995)
21. Fermi, E., Metropolis, N.: Numerical solution of minimum problem. Los Alamos Unclassified Report LA–1492, Los Alamos National Laboratory, Los Alamos (1952)
22. Fowler, K., Reese, J., Kees, C., Dennis, J., Jr., Kelley, C., Miller, C., Audet, C., Booker, A., Couture, G., Darwin, R., Farthing, M., Finkel, D., Gablonsky, J., Gray, G., Kolda, T.: Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. Adv. Water Resour. **31**, 743–757 (2008)
23. García-Palomares, U., Costa-Montenegro, E., Asorey-Cacheda, R., González-Castaño, F.: Adapting derivative free optimization methods to engineering models with discrete variables. Optim. Eng. **13**, 579–594 (2012)
24. Gross, B., Roosen, P.: Total process optimization in chemical engineering with evolutionary algorithms. Comput. Chem. Eng. **22**, S229–S236 (1998)
25. Hemker, T., Fowler, K.R., Farthing, M.W., von Stryk, O.: A mixed-integer simulation-based optimization approach with surrogate functions in water resources management. Optim. Eng. **9**, 341–360 (2008)
26. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press (1975)

27. Holmström, K., Göran, A.O., Edvall, M.M.: User's guide for TOMLAB/OQNLP. Tomlab Optimization (2007). http://tomopt.com
28. Holmström, K., Göran, A.O., Edvall, M.M.: User's guide for TOMLAB 7. Tomlab Optimization (Current as of 15 March, 2021). http://tomopt.com
29. Holmström, K., Quttineh, N.H., Edvall, M.M.: An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization. Optim. Eng. **9**, 311–339 (2008)
30. Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. J. Assoc. Comput. Mach. **8**, 212–219 (1961)
31. Huyer, W., Neumaier, A.: Global optimization by multilevel coordinate search. J. Glob. Optim. **14**, 331–355 (1999)
32. Huyer, W., Neumaier, A.: SNOBFIT—stable noisy optimization by branch and fit. ACM Trans. Math. Softw. **35**, 1–25 (2008)
33. Jones, D.R.: The DIRECT global optimization algorithm. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, vol. 1, pp. 431–440. Kluwer Academic Publishers, Boston (2001)
34. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948. Piscataway, NJ, USA (1995)
35. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
36. Kleijnen, J.P.C., Van Beers, W., Van Nieuwenhuyse, I.: Constrained optimization in expensive simulation: novel approach. Eur. J. Oper. Res. **202**, 164–174 (2010)
37. Laguna, M., Gortázar, F., Gallego, M., Duarte, A., Martí, R.: A black-box scatter search for optimization problems with integer variables. J. Glob. Optim. **58**, 497–516 (2014)
38. Larson, J., Leyffer, S., Palkar, P., Wild, S.: A method for convex black-box integer global optimization (2019). arXiv:1903.11366
39. Lewis, R.M., Torczon, V.J.: Pattern search algorithms for bound constrained minimization. SIAM J. Optim. **9**, 1082–1099 (1999)
40. Liao, T., Socha, K., de Oca, M., Stützle, T., Dorigo, M.: Ant colony optimization for mixed-variable optimization problems. IEEE Trans. Evol. Comput. **18**, 503–518 (2013)
41. Liu, J., Ploskas, N., Sahinidis, N.: Tuning baron using derivative-free optimization algorithms. J. Glob. Optim. **74**(4), 611–637 (2019)
42. Liuzzi, G., Lucidi, S., Rinaldi, F.: Derivative-free methods for bound constrained mixed-integer optimization. Comput. Optim. Appl. **53**, 505–526 (2012)
43. Liuzzi, G., Lucidi, S., Rinaldi, F.: Derivative-free methods for mixed-integer constrained optimization problems. J. Optim. Theory Appl. **164**, 933–965 (2015)
44. Liuzzi, G., Lucidi, S., Rinaldi, F.: An algorithmic framework based on primitive directions and nonmonotone line searches for black-box optimization problems with integer variables. Math. Program. Comput. **12**, 673–702 (2020)
45. Liuzzi, G., Lucidi, S., Sciandrone, M.: Sequential penalty derivative-free methods for nonlinear constrained optimization. SIAM J. Optim. **20**, 2614–2635 (2010)
46. Lucidi, S.: DFL—derivative-free library (current as of 15 March, 2021). http://www.dis.uniroma1.it/~lucidi/DFL/
47. Lucidi, S., Piccialli, V., Sciandrone, M.: An algorithm model for mixed variable programming. SIAM J. Optim. **15**, 1057–1084 (2005)
48. Lucidi, S., Sciandrone, M.: A derivative-free algorithm for bound constrained optimization. Comput. Optim. Appl. **21**, 119–142 (2002)
49. Matheron, G.: Principles of geostatistics. Econ. Geol. **58**, 1246–1266 (1967)
50. Moré, J., Wild, S.: Benchmarking derivative-free optimization algorithms. SIAM J. Optim. **20**, 172–191 (2009)
51. Müller, J.: Miso: mixed-integer surrogate optimization framework. Optim. Eng. **17**, 177–203 (2016)
52. Müller, J.: Miso: mixed-integer surrogate optimization framework (current as of 15 March, 2021). https://optimization.lbl.gov/downloads#h.p_BjSaeAORU9gm
53. Müller, J., Shoemaker, C.A., Piché, R.: SO-MI: a surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. Comput. Oper. Res. **40**, 1383–1400 (2013)
54. Müller, J., Shoemaker, C.A., Piché, R.: SO-I: a surrogate model algorithm for expensive nonlinear integer programming problems including global optimization applications. J. Glob. Optim. **59**, 865–889 (2014)
55. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)
56. Neumaier, A.: SNOBFIT—stable noisy optimization by branch and FIT (current as of 15 March, 2021). http://www.mat.univie.ac.at/~neum/software/snobfit/

57. Newby, E., Ali, M.M.: A trust-region-based derivative free algorithm for mixed integer programming. Comput. Optim. Appl. **60**, 199–229 (2015)
58. Ploskas, N., Laughman, C., Raghunathan, A., Sahinidis, N.: Optimization of circuitry arrangements for heat exchangers using derivative-free optimization. Chem. Eng. Res. Des. **131**, 16–28 (2018)
59. Porcelli, M., Toint, P.L.: BFO, a trainable derivative-free brute force optimizer for nonlinear bound-constrained optimization and equilibrium computations with continuous and discrete variables. ACM Trans. Math. Softw. **44**, 1–25 (2017)
60. Powell, M.J.D.: The BOBYQA algorithm for bound constrained optimization without derivatives. Tech. rep., Department of Applied Mathematics and Theoretical Physics, University of Cambridge (2009)
61. Rashid, K., Ambani, S., Cetinkaya, E.: An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization. Eng. Optim. **45**, 185–206 (2013)
62. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Glob. Optim. **56**, 1247–1293 (2013)
63. Sauk, B., Ploskas, N., Sahinidis, N.: GPU parameter tuning for tall and skinny dense linear least squares problems. Optim. Methods Softw. **35**, 638–660 (2018)
64. Schlüter, M., Egea, J.A., Banga, J.R.: Extended ant colony optimization for non-convex mixed integer nonlinear programming. Comput. Oper. Res. **36**, 2217–2229 (2009)
65. Schlüter, M., Gerdts, M.: The oracle penalty method. J. Glob. Optim. **47**, 293–325 (2010)
66. Schlüter, M., Munetomo, M.: MIDACO user guide. MIDACO-SOLVER (2016). http://www.midaco-solver.com/
67. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. Eur. J. Oper. Res. **185**, 1155–1173 (2008)
68. Spendley, W., Hext, G.R., Himsworth, F.R.: Sequential application for simplex designs in optimisation and evolutionary operation. Technometrics **4**, 441–461 (1962)
69. Sriver, T.A., Chrissis, J.W., Abramson, M.A.: Pattern search ranking and selection algorithms for mixed variable simulation-based optimization. Eur. J. Oper. Res. **198**, 878–890 (2009)
70. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**, 225–249 (2005)
71. Toint, P.L., Porcelli, M.: BFO—brute-force optimizer (current as of 15 March, 2021). https://sites.google.com/site/bfocode/home
72. Torczon, V.J.: On the convergence of pattern search algorithms. SIAM J. Optim. **7**, 1–25 (1997)
73. Vicente, L.N.: Implicitly and densely discrete black-box optimization problems. Optim. Lett. **3**, 475–482 (2009)
74. Vigerske, S.: Minlplib 2. In: Proceedings of the XII Global Optimization Workshop MAGO, pp. 137–140 (2014)
75. Winfield, D.: Function and functional optimization by interpolation in data tables. Ph.D. thesis, Harvard University, Cambridge, MA (1969)
76. Winslow, T.A., Trew, R.J., Gilmore, P., Kelley, C.T.: Simulated performance optimization of GaAs MESFET amplifiers. In: IEEE/Cornell Conference on Advanced Concepts in High Speed Semiconductor Devices and Circuits, pp. 393–402. Piscataway, NJ (1991)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.