DIRECT using local search on surrogates

Technical Report

March 20, 2011

Thomas Hemker*, Christian Werner

Simulation, Systems Optimization and Robotics, Department of Computer Science, Technische Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, Germany

* Corresponding author: hemker@sim.tu-darmstadt.de

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Abstract

The solution of noisy nonlinear optimization problems with nonlinear constraints and derivative information is becoming increasingly important, as many practical applications can be described by this type of problem in e.g., engineering applications. Existing local optimization methods show good convergence properties. However, they often depend on sufficiently good starting points and/or the approximation of gradients. In turn, global derivative free methods, which need no starting values to be initialized, require many evaluations of the objective function, particularly in the vicinity of the solution.

A derivative free optimization algorithm is developed that combines advantages of both local and global methods. The DIRECT algorithm, which is often used for problems where no prior knowledge is available as kind of a brute force start, is extended by an inner loop using a surrogate based optimization method. The local search on the surrogate function determines better candidates for sampling than the hypercube center points chosen by DIRECT, especially if constraints are arising. This inner loop needs no additional evaluation of the original problem. Standard test problems and a computational more expensive test problem are chosen to show the performance of the new algorithm.

Contents

1 Introduction

Derivative free optimization (DFO) methods are widely used in many disciplines due to their easy use and the need for only a little input information, typically just the objective function value and some initial data. The development in the field of computer-aided engineering is to use computer simulations more frequently for optimization processes [23]. The complexity of the modeled systems implicates high computational costs for simulation runs. The evaluation of a set of input values can take a lot of time (minutes, hours, days) until a response is obtained. So, optimization methods are needed that achieve a fast reduction of the objective function within a low number of evaluations.

The field of derivative free optimization (DFO) deals with such problems [4] and algorithms exist that are capable of finding reasonably good values within just a few function evaluations. In the area of simulation based optimization, objective functions and constraints are based on some sort of black box evaluations and have characteristically high frequency and low amplitude noise included which impedes

the direct or approximate use of gradient information. Additionally high computational costs are often connected to calls of the objective function that forbid high numbers of evaluations. Therefore, DFO methods are the only choice for optimization without changing the underlying function value evaluator.

*DI*viding *RECT*angles [18] is a DFO solver belonging to the category of global optimization methods and is used for different black box based optimization problems [3, 6, 15]. It is an advancement of Shubert's Algorithm for Lipschitz continuous objective functions [29]. A combination of function values and Lipschitz constants is used as an indicator for promising regions of the variable space for a minimum. DIRECT works on a variable space restricted by box constraints and iteratively divides it into subhyperrectangles. A ranking of these hyperrectangles is done by function value at the midpoint and the size of the hyperrectangle.

DIRECT is intended for problems with box-constraints that act as an initial rectangle. Furthermore, apart from the problem bounds, no other parameters, in particular no starting values, need to be set. This makes DIRECT easy to use, especially on optimization problems for which no deeper knowledge about the underlying systems exists. But as Jones mentions himself in [17],

> *[. . .] DIRECT quickly gets close to the optimum but takes longer to achieve a high degree of accuracy. This suggests that the best performance would be obtained by combining DIRECT with a good local optimizer.*

Motivated by Jones' suggestion, a hybrid optimization method based on DIRECT is proposed and analyzed, which adds an inner loop using a local solver. The local search is done by a combination of an interior point method and statistical emulation using Gaussian processes to build sequentially updated surrogate functions. DIRECT is an optimization algorithm that has almost no parameters that have to be set, and we are trying to keep this benefit by using the default parameter sets for all elements of the new algorithm.

Hybrid optimization approaches can be separated into at least three groups of methods. First, the group of methods that run sequentially, using the notation from [25]: Collaborative combinations with sequential execution. An example is the hybrid of DIRECT and IFFCO [21], which has been implemented in [3] and [26]. The above mentioned weakness of DIRECT to converge slowly once it is close to a solution is compensated by starting IFFCO from the best value found by DIRECT after a predetermined number of evaluations. In turn DIRECT supplies IFFCO with the needed starting points.

The second group stands for approaches where optimization methods run in parallel, collaborative methods with parallel or intertwined execution [25]. These methods share evaluation data and intermediate results, but the optimization algorithms are not changed like presented e.g., in [13]. In this hybrid DIRECT is coupled into an optimization framework as one solver, competing with others, here generating set search (GSS) [12].

The last group stands for hybrid optimization approaches that are coupled on the iteration level, integrative combinations [25] where one method depends on the coupled one to finish the iteration. The approach presented in this paper belongs to this group of methods. DIRECT provides the outer loop for each iteration and a second optimization method runs as an inner loop.

The rest of the paper is organized as follows. Section 2 describes the DIRECT algorithm, the subsequent Section introduces the methods later used for the inner loop, local search on surrogates. A description of the new algorithm, called DIRECToS, and the framework used for implementation is presented in Section 4. The experimental setup is outlined in Section 5 and Section 6 describes the numerical experiments, before we discuss the results in the final Section.

2 DIRECT

The DIRECT algorithm of Jones et al. [18] is a deterministic sampling method designed for problems of the type

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & l \leq x \leq u \\ & x, l, u \in \mathbb{R}^n,\ f : \mathbb{R}^n \to \mathbb{R} \end{aligned} \tag{1}$$

where f is continuous.

Instead of evaluating the function f at the corners like in Shubert's algorithm [29], f is evaluated at the center points of the subdivided hypercubes. An extension on how to deal with noisy objective functions is given in [5]. In an n-dimensional search space, this means that the algorithm can be initialized by sampling just one point, as opposed to all 2^n vertices of the space.

Algorithm 1 Procedure for dividing hyperrectangles

1: Identify the set I of dimensions with the maximum side length. Let δ equal one-third of this maximum side length.
2: Sample the function at the points $c \pm \delta e_i$ for all $i \in I$, where c is the center of the hyperbox and e_i is the ith unit vector.
3: Divide the hyperbox containing c into thirds along the dimensions in I, starting with the dimension with the lowest value of $w_i = \min\{f(c + \delta e_i), f(c - \delta e_i)\}$, and continuing to the dimension with the highest w_i.

In the course of the algorithm, the space bounded by l and u is partitioned into hyperrectangles. Since there are now n possibilities to divide the hypercube into thirds, a procedure is needed to decide in which order the dimensions will be partitioned. Algorithm 1 describes the strategy for dividing a hyperrectangle. It follows that hyperrectangles are always divided along their longest sides, ensuring their shrinkage in every dimension. This is essential for the proof of convergence, as shown in more detail in [7].

Those hyperboxes are chosen for further sampling and subdivision, that are potentially optimal as defined in [9, 18]. The default settings of the used DIRECT implementations are adopted to define potentially optimal hyperboxes in all numerical experiments. No parameters have to be adjusted by the user to balance between searching locally and globally. The chosen subspaces are divided into thirds, followed by a new ranking including the two new hyperrectangles. By repeating this, we obtain the skeleton Algorithm 2.

Algorithm 2 DIRECT

1: Normalize the search space to be the unit hypercube.
2: Evaluate f in the center point c_1 of the hypercube.
Set $f_{\min} = f(c_1)$ and $t = 0$ (iteration counter).
3: Identify the set J of "potentially optimal" hyperrectangles.
4: Select any hyperrectangle $j \in J$.
5: Using Algorithm 1, determine where to sample within hyperrectangle j and how to divide it into subhyperrectangles.
6: Update f_{min} and set $l = l + \delta l$, where δl is the number of new points sampled.
7: Set $J = J - j$. If $S \neq \emptyset$, go to 4.
8: Handle infeasible points.
9: Set $t = t + 1$. If $t = t_{max}$, stop (iteration limit reached). Otherwise go to 3.

Many optimization problems at hand require efficient treatment of linear and nonlinear constraints in addition to simple bounds. Although, DIRECT can handle hidden constraints [10] it makes no use of quantitative information about the constraint violation. DIRECT does not differentiate between hidden and explicitly given constraints and uses a heuristic giving infeasible points a weight depending on the value of feasible evaluations in the neighborhood [9].

DIRECT converges faster than Shubert's algorithm [18] but it is still considered to be too slow. As can be seen in [17] DIRECT is capable of quickly identifying areas with potential global optima. But the grid based iteration scheme leads to slow convergence once DIRECT is close to an optimal value, especially in the presence of nonlinear or hidden constraints close to the optimum [15]. Thus, Jones suggests to combine DIRECT with a local optimizer to refine a solution.

3 Surrogate based local search

Surrogate optimization represents solving problems generated with surrogate functions by approximating components of an original objective function. Surrogate based methods are of special interest in cases where the evaluation of the objective function f is connected with high computational costs. The idea is to avoid iterating directly on f. Instead, calls to a surrogate function $\hat{y}$ are performed to determine a new promising candidate x_{cand} for the evaluation by f. The response of each evaluation of the objective function $f(x_{\text{cand}})$ is stored and the surrogates are generated using the information from this storage. Barthelemy and Hafka [1] proposed a classification into three groups: local, med-range and global approximations. Local approximations are valid in a vicinity of the point around which they are build. Global approximations can be used on the whole variable space of the optimization problem, even outside of the feasible regions. Mid-range approximation are in between, originally defined as an approximation using objective function evaluations at several points as shown in [4], or a combination of a local and a global approximation.

The general procedure is the same for most approaches: The first step is fitting a surrogate function to a set of evaluated points. If there are no evaluations yet, an initial design must be determined. Then, using the surrogate function and based on a defined criterion, a new candidate for evaluation is chosen and extends the set of evaluated points. These two steps are repeated until a stopping condition is satisfied.

The choice of the initial design points is done by expert's guess or with a space-filling design. Both options are not desired in the context considered here. On the one hand the choice should follow a general procedure to be independent from the given problem. On the other hand the need of a low number of evaluations prohibits an extensive evaluation of the problem in the initialization phase as needed by Latin hypercube sampling ($10n$ evaluations are common) [19].

The functions used for approximation are mostly compositions of algebraic functions. Thus, they have the benefit of being a smooth approximation of the original problem filtering noise and making the usage of derivative information for gradient-based solvers possible. Surrogate optimization methods differ in three key features: (1) The technique for approximation, (2) the choice of the initial set of data points, on which the first surrogate is fit, and (3) the choice of the next candidate.

Efficient implementations are the EGO algorithm that was developed by Jones, Schonlau and Welch in 1998 [19], which is extended by using DIRECT for the EGO specific criterion [30]. CORS-RBF by Regis and Shoemaker [26] implements the Constrained Optimization using Response Surfaces method. In [34] a multistart procedure working on local radial basis function models is introduced. BOBYQA [24] makes use of quadratic approximations defined on a trust region. Booker et al. [2] introduced the surrogate management framework (SMF) as a generalized pattern search method operating on function approximations, which was recently extended [27]. APPS-TGP by Taddy, Lee, Gray, and Griffin [31] is a loose coupling of a pattern search implementation APPS and TGP. SUROPT [15] aims to solve nonlinear mixed-integer problems with nonlinear constraints. An approach where DIRECT is used for optimization in combination with different combined surrogates is presented in [35]. Apart from the mentioned approaches, a lot more algorithms are developed and available through different solvers.

For those methods that build a surrogate on the complete domain and not only on a region of it, the stochastic approximation methods showed a good performance in practical examples [15, 31]. Thus, we will use Treed Gaussian Processes (TGP) [11] as a surrogate function builder for our hybrid approach. TGP provides an approximation of the objective function for the basis points, but no exact interpolation.

When optimization is carried out on surrogate functions, beside DFO methods, nonlinear programming methods can be applied. The nature of the originally considered optimization problem does not allow the use of any derivative information and thus does not allow direct application of such solvers. But the surrogate problem is a standard nonlinear programming (NLP) problem. The functions involved are mathematically given and are computational cheap to evaluate.

Efficient gradient-based solvers have been developed to solve NLP problems, interior point (IP) methods are one kind and have their origin in barrier methods which were developed already in the 1960s and in Karmarkar's algorithm [20]. IPOPT is one efficient implementation of an IP method, but it differs in some details from the basic interior point method algorithm. For further information the reader is referred to [32].

Beside the efficiency of NLP methods to solve the surrogate problem, IPOPT and others are made to solve nonlinear optimization problems including nonlinear constraints given by h, which is an extension of Equation 1, or equal to it,

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & l \leq x \leq u, \quad h(x) \leq 0. \\ & x, l, u \in \mathbb{R}^n,\ f : \mathbb{R}^n \to \mathbb{R},\ h : \mathbb{R}^n \to \mathbb{R}^m. \end{aligned} \tag{2}$$

The bounds and constraints are used explicitly within optimization with an NLP method and this ability is kept for the hybrid methods in combination with DIRECT as we will see in the following section.

4 DIRECT using local search

Due to the arguments presented in the previous sections, a combination of the exploratory and space-filling property of DIRECT, the cost-reducing and smoothing characteristics of surrogate methods and the efficiency of gradient-based solvers are promising.

4.1 Algorithm

DIRECT extended by local search on surrogates (DIRECToS) is a DFO algorithm using DIRECT, TGP and IPOPT. In contrast to sequential or parallel couplings mentioned, the coupling within the DIRECToS algorithm is performed at the iteration level. Each time the DIRECT algorithm requests an evaluation of a new center point, not this point is evaluated, but the candidate returned from the inner loop. This candidate is found by using the center point as the starting point for a local search on a surrogate. Thus, DIRECT internally ranks the points by their starting value quality for the local search, not by their function value (cf. Fig. 7).

To satisfy the need for keeping the number of function evaluations low, the local search is performed on a surrogate function created by TGP. The smooth nature of the surrogate permits the use of IPOPT to find a solution on the surrogate that satisfies the nonlinear constraints. During the first iterations of the algorithm, the surrogate $\hat{y}$ cannot be expected to capture enough characteristics of the objective function. So, the real function value of the surrogate's solution $f(\hat{x}_{\text{opt}})$ does not need to be smaller than the function value $f(c)$ at the hyperrectangle's center point c. However, it has the effect of building a set of design points that are not fixed on the grid given by DIRECT. This way a design is generated that tends to be non-collapsing and leads to a better dispersed sampling in the feasible region. Thus, the quality of surrogates for successive local searches is expected to be better in terms of catching global trends.

For the new algorithm we decompose it into an outer and an inner loop. The outer one is Algorithm 2 and the inner one is started whenever DIRECT requests an evaluation of the objective function. The

algorithm starts with the initial phase of DIRECT, which performs $2n+1$ evaluations to be able to decide how the initial unit hypercube is to be divided. These evaluations are performed directly on the original objective function. Each subsequent evaluation request starts a local search on the surrogate. This leads to an earlier use of surrogates, compared to other approaches based on different ways to define the initial set of points.

Now, the iteration phase of the outer loop begins. A surrogate is fit to all evaluated points $\{x_1, \ldots, x_p\}$. The evaluation request of the center point c_i of the hyperrectangle is redirected to IPOPT and serves as the starting point ($x_{\text{start}} := c_i$) for the local search on the surrogate. In the inner loop IPOPT tries to find the minimum on the surrogate within a given maximal number ν of evaluations on the surrogate, if no other default stopping criterion is reached. The best value found is interpreted as $\hat{x}_{\text{opt}}$ and is evaluated on the original function f. This completes the inner loop. The data $(\hat{x}_{\text{opt}}, f(\hat{x}_{\text{opt}}))$ is stored for surrogate fitting in later iterations and the value $f(\hat{x}_{\text{opt}})$ is returned to DIRECT, but not the requested value of the center point. This finalizes an iteration of the outer loop.

Preliminary experiments showed that this pure strawman approach for choosing the next evaluation candidate led to an early clustering of the design points. This can be explained by the fact that in early stages the surrogate only captures a global trend. Thus, it tends to have one single optimal value to which the NLP-solver converges from any given starting point. This strongly diminishes the desired effect of the explorative behavior that DIRECT contributes to the algorithmic framework. Thus, similar to [26], we will introduce bounds which aim to prevent the local search from producing clusters of design points. This is done by constraining the local search on the hyperrectangle j of which c_j is the current starting value of the local search. This makes the surrogate to a kind of mid-range approximation, built using points from the entire domain, but used only on the hyperrectangle around the relevant center point.

The presence of nonlinear constraints requires some modifications to the algorithm. In the inner loop, it is straight forward for IPOPT to include constraints, since their derivative information is given explicitly. Special handling is necessary if the local search is unable to find a feasible solution or if some evaluations in DIRECT's initial phase are infeasible. If such a case occurs, a message is returned to DIRECT that the requested evaluation is infeasible. Nevertheless, the evaluation response is stored for surrogate fitting. Note that this is reasonable since the problem is relaxable.

An algorithmic description of DIRECToS is easily obtained by taking the description of DIRECT (Algorithm 2) and substituting just the sampling in the algorithm in Step 5 with the routine shown in Algorithm 3. The general algorithmic structure of DIRECT and especially the way how the domain is separated is not changed, but just the order how the domain is explored. The theoretical convergence properties of DIRECT [18] hold unchanged for any problem, where the preconditions and especially Lipschitz continuity is given.

Algorithm 3 SAMPLEoS(x)

1: **if** number of evaluations of objective function $< 2 \cdot$ dimension of problem $+1$ **then**
2: Call objective function and return result.
3: **else**
4: Create surrogate using all previous evaluations.
5: Start local search with IPOPT on surrogate with $x_{\text{start}} = x$.
6: Evaluate objective function in found local optimum x_{optLoc}.
7: Return $f(x_{optLoc})$.
8: **end if**

4.2 Implementation details

A combination of IPOPT and TGP is introduced in this work: IPOPT is an interior-point method and available from the pages of the Computational Infrastructure for Operations Research (COIN-OR) (`https://projects.coin-or.org/Ipopt`). TGP is available as a package for the free statistical soft-

ware R. It provides several methods for the creation of Bayesian regression models. It is distributed under the Lesser GNU Public License (LPGL) and is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/`.

All components are mostly unmodified in their default settings, though some changes are necessary. The Hessian of the Lagrangian, needed by IPOPT, is approximated since it is not explicitly given [33]. Furthermore, the TGP package does not offer any method to compute gradients of the surrogate. Since the random responses of the predict-method for surrogate function evaluation prevent the application of finite differences to approximate the gradients, this randomization is deactivated within the code of TGP. Hence it is possible to use central differences for approximation of the surrogate's gradient.

A further difficulty arising with TGP is that predictions can only be performed for points lying in the hyperrectangle R, where R is the smallest hyperrectangle that contains all design points and its sides are parallel or orthogonal to the canonical unit vectors. Figure 2 illustrates the two-dimensional case. The arithmetic mean of the function values of the design points is returned for predictions outside of this box. In order to obtain a global valid surrogate it is necessary to evaluate the two points $l_- := (l_1 - \delta, \ldots, l_n - \delta)$ and $u_+ := (u_1 + \delta, \ldots, u_n + \delta)$, with a small δ, at the beginning of the initial phase of DIRECToS.

Two options of the TGP package are varied to analyze the impact of the performance of DIRECToS. The first is the choice of the mean function in the stochastic process. Surrogates with constant mean functions have been used in [15, 28]. Regis and Shoemaker choose a linear polynomial to describe the global trend of the surrogate [26]. Taddy et al. left the default setting for TGP, which selects a linear mean [31]. Later, results obtained with the linear mean are marked by an L, those obtained with a constant mean are marked with a C.

The second option to decide on, is the nugget. In [31] Taddy et al. give arguments for a modification of the nugget parameter. Instead of using the default distribution $\sim \Gamma^{-1}(1,1)$ with an expected value of 1, they choose a small nugget with a distribution $\sim \Gamma^{-1}(1,100)$ and an expected value of 0.01. This leads to closer approximating surrogates, but which are more sensitive to noise. The choice is justified by the absence of random noise in the objective functions, but limits the approach not to these kind of problems. These two variations are marked in the numerical results by 1, respectively 100. It always depends on the optimization problem under consideration how to choose these parameters for a certain approximation accuracy and this has to be focus of further research.

4.3 Framework

The framework connecting the codes of DIRECT, IPOPT and TGP has been implemented in Python (www.python.org). The diagram in Figure 3 gives an idea of the implementation. An alternative framework written in Python is *Pyomo* [14] but it does not offer the solvers needed in the context of this work. Although the combination of DIRECT, IPOPT and TGP is considered in this work, any optimization solver or surrogate code could be included within this framework easily. The programming effort depends on the language the connecting code is written in. Languages for which interfaces have already been implemented include C/C++, R, Fortran and Matlab. The infrastructure is based on remote procedure calls and offers the possibility to run the different modules on different computers, as long as there is a network connection between them.

5 Experimental Setup

For two reasons a limit of objective function evaluations has been passed to DIRECToS. First, the focus lies on how well the solver performs with a low number of evaluations, so an exhaustive search is not needed. The second reason is that with the increasing number of design points, the time needed for the model fitting grows disproportionally since the number of covariances to be computed grows quadratically. The left plot in Figure 4 illustrates this. Additionally, the right plot shows the time needed for fitting a model with 37 design points depending on the number of variables n.

The actual limit was motivated by Moré and Wild [22]. They used a stopping criteria for the solvers of at least 100 simplex gradients. Since their benchmark problems have at most 12 variables, the limit

was set to 1300. In our case no simplex gradients are computed. Thus, we wanted DIRECToS to use at least 100 evaluations of the objective function. This number is augmented by $2n+1$ for each individual problem. Note that this is the number of evaluations performed in the initial phase of the algorithm, until the first surrogate is generated. This does mean that high dimensional problems are easier to solve than lower dimensional ones, but it is owed to the fact that building surrogates in higher dimensional domains becomes computational expensive.

Data profiles [22] are used to analyze the performance on the test set. They are defined in terms of a performance measure $t_{p,s}$ obtained for each $p \in \mathscr{P}$ and $s \in \mathscr{S}$, where $\mathscr{P}$ and $\mathscr{S}$ denote the set of problems and the set of solvers, respectively. Here, this measure is the number of function evaluations required to satisfy the convergence test:

$$f(x_0) - f(x) \geq (1-\theta)(f(x_0) - f_L), \tag{3}$$

where $\theta > 0$ is a tolerance. x_0 is the starting point for the problem. f_L is computed for each problem as the smallest value of f obtained by any solver within a given number μ_f of function evaluations. Users with expensive optimization problems are interested in the performance of solvers in terms of the number of function evaluations. *Data profiles* capture this information and provide information about the short-time behavior of a solver s. [22] provides the following definition for the data profile of a solver s:

$$d_s(\kappa) = \frac{1}{|\mathscr{P}|} \left| \left\{ p \in \mathscr{P} : \frac{t_{p,s}}{n_p + 1} \leq \kappa \right\} \right| \tag{4}$$

where n_p is the number of variables in problem p. If $n_p + 1$ in the denominator is omitted, the data profile would be a function giving the fraction of problems that can be solved with κ function evaluations for some solver s. This definition of data profiles would be independent of the dimension of the problem. This is unrealistic because the number of function evaluations needed to satisfy the convergence test (3) is likely to grow as the number of variables increases. Thus this scaling is introduced. By defining $\frac{t_{p,s}}{n_p+1}$ as *normalized number of evaluations*, the data profile $d_s(\kappa)$ is the fraction of the problems that can be solved with the equivalent of κ normalized evaluations.

6 Numerical Results

DIRECToS has been tested on two different groups of test problems, one containing only bound constraints and a second group of general NLP problems restricted by nonlinear constraints, too. Since we are particularly interested in the short time behavior of DIRECToS, we will concentrate on the results given by data profiles. The parameter μ_f is set to a value that at least the intended $2n+1+100$ evaluations are proceeded with each problem. This leads to the drawback that some problem stop contributing to the results depending on the problem dimension. But, as we will see, the most interesting part of the profiles is mostly reached before. If not given otherwise, θ will be set to 0.1 to emphasize the short-time behavior of the solvers.

6.1 Bound constraints problem

In this work problems from two sources are used to define the first set of problems. The *convex quadratics* defined in [21], where each of these problems has two realizations. A smooth one and one including "deterministic noise" [21, Equation(6.12)], resulting from an added low-amplitude high frequency perturbation. These will be called *Kelley problems*. The problems used in [22], which are taken from the CUTEr data base (`http://hsl.rl.ac.uk/cuter-www/index.html`) complete the set of box constrained problems. These problems are used without any variation of the starting point, as we make no use of such information. These problems have a smooth and a noisy version as well. They will be called *CUTEr*

problems. In total we get 30 Kelley problems and 40 CUTEr problems problems, each in a smooth and a noisy version.

The results for the smooth Kelley problems are summarized in Figure 5, the results for the noisy ones in Figure 6. Almost all problems from this test set are solved by DIRECToS, not more than 60% by DIRECT without using local search. But we also see that DIRECT is performing better on the noisy problems than on the smooth ones. The profiles in Figure 7 (smooth problems) and 8 (noisy problems) show the disadvantage of DIRECToS' need for the two additional evaluations to guarantee the existence of the surrogate (see 4.2). This results in low success rate values of DIRECToS right at the beginning of the data profiles. But after a few evaluations DIRECToS solves more problems successfully at almost all normalized iterations except those iterations around 10. The mean number of function evaluations used to find for the best point in all numerical experiments with the 80 CUTEr problems is 124.4, where the maximum number of evaluations is 330. The best points were found with DIRECToS for only 11 problems in less than 100 evaluations, for DIRECT only for 3 problems. We also recognized that both approaches perform better on the noisy problems and that all four DIRECToS versions have a comparable performance.

6.2 Noisy nonlinear constrained problems

The set of NLP problems is taken from [16] and comprises theoretical as well as practical problems of varying dimensions, types of objective functions and types of constraints. A categorization of the NLP problems is given in Table 1. Note that a problem can belong to multiple categories. Furthermore, to meet the specifications of the assumed optimization problem 2, deterministic noise is added to the objective function of each problem as proposed for box constrained problems in [21].

The experimental setup is the same as for the boxed problems. Additionally, since the constraints of the NLP problems are mathematically available, we give IPOPT access to them. In Figure 9 the data profile for all NLP problems is shown. It gives an idea of the overall good performance of DIRECToS compared to DIRECT. The profile shows that all four tested versions of DIRECToS perform similarly well as we have seen earlier. The influence of θ as the criteria to define success for an optimization run is shown in Figure 10. To examine the results in more detail, data profiles for classes of the NLP problems regarding to Table 1 are shown in Figures 11 to 16.

DIRECT was not able to solve any of the equality constrained problems without changes like adding tolerances. This is not surprising, since the feasible domain of these problems is a hyperplane and thus it is very unlikely that DIRECT samples a feasible point. Due to the integration of IPOPT, DIRECToS is able to sample feasible points as shown Figure 11. Thus, numerical results from the equality constrained problems are not included if not mentioned explicitly.

A different situation is given in Figure 12 which shows the data profile for problems with inequality constraints, and DIRECT solves successfully half of the number of problems that are solved by its extension DIRECToS. Figures 13 and 14 show the data profiles for the problems with quadratic and polynomial objective functions respectively. In these cases DIRECT is dominated again by all versions of DIRECToS. However, DIRECToS C1 performs best after 11 normalized evaluations.

The last comparison is performed between linear and nonlinear constrained problems. Here again only problems with inequality constraints are considered. Figures 15 and 16 show the corresponding data profiles. In both cases DIRECToS performs better than DIRECT, although the difference is bigger for the nonlinearly constrained problems, especially during the first iterations.

6.3 Heart rate regulation problem

In [8] Fowler et al. introduce a 17-parameter identification problem for a model that can predict heart rate regulation during the act of standing up from a sitting posture. It is a problem where only box constraints arise. The evaluation of the objective function implies the solution of a sequence of differential equations in Matlab making up the heart rate model. Computational noise and complexity arising from

this give the objective function typical black box characteristics. The problem is originally solved with the Nelder-Mead simplex algorithm, implicit filtering using a user chosen starting value, and a genetic algorithm with a large budget of function evaluations.

The number of evaluations performed is plotted in Figure 17 against the best found value. The relevant part starts with evaluation 36, because the first 35 calls to the objective function take place in the initial phases of the solvers. DIRECToS again needs 2 more evaluations to be able to create a search space covering surrogate.

Table 2 gives the results of [8] in contrast to the ones obtained within this work. The abbreviations used are NM for the Nelder-Mead simplex method, IF for implicit filtering and GA for a genetic algorithm. IFNM and IF2 denote restarts of implicit filtering with previous best points from Nelder-Mead and implicit filtering respectively. Thus, they can be considered as sequential hybrid methods. Even without using any starting value DIRECT realized an objective function value decrease of 73% , DIRECToS of more than 84%, but both failed even after 1000 evaluations to find the originally identified best value that was obtained using prior knowledge or after an exhaustive search of the GA [8]. However, DIRECToS achieved a comparable improvement of the objective function without a starting point or other prior problem specific knowledge.

7 Conclusion

This work intends to improve DIRECT by coupling it with a surrogate optimization method. The latter is a combination of the freely available NLP-solver, IPOPT, and of the, also freely available, TGP package. With only minor changes to the individual program codes, the derivative free optimization method, DIRECToS, is implemented in a new framework for hybrid optimization methods. The framework was implemented in Python and it offers the possibility to connect any solver or surrogate generator to it. DIRECToS has no process parameters that have to be set, or which have to be tuned for the benchmark problems used in this work.

Numerical experiments show that the choice of found optimal points of surrogates as candidates to replace center points for evaluations can improve the performance of DIRECT. Furthermore, a larger set of NLP problems is used to conduct a comparison of DIRECToS with DIRECT regarding constraints handling, where the objective function is extended by a deterministic term simulating numerical noise. It is shown that by including a local search all types of constraints can be handled naturally without adding problem specific penalty or barrier functions. Additionally, as a practical application, a heart rate modeling problem has been used for comparison where DIRECToS achieved better results than DIRECT. The results confirm the superior performance of the coupled algorithm DIRECToS over DIRECT.

DIRECToS and the framework used for implementation offer various possibilities for further work, an important one is the extension of using additional surrogates to replace nonlinear constraints in case these are based on black box or simulation evaluations. As DIRECToS is designed as an initial optimization method, it looks promising to be tested in a larger hybrid framework of collaborative parallel methods, where also measures are developed to decide whether the use of surrogate function information is helpful or not.

References

[1] J.-F. M. Barthelemy and R. T. Haftka, *Approximation concepts for optimum structural design - a review*, Structural Optimization **5** (1993), 129–144.

[2] A. J. Booker, J. E. Dennis jr., P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset, *A rigorous framework for optimization of expensive functions by surrogates*, Structural Optimization **17** (1999), 1–13.

[3] R. G. Carter, J. M. Gablonsky, A. Patrick, C. T. Kelley, and O. J. Eslinger, *Algorithms for noisy problems in gas transmission pipeline optimization*, Optimization and Engineering **2** (2001), 139–157.

[4] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*, MPS-SIAM, 2009.

[5] G. Deng and M. C. Ferris, *Extension of the direct optimization algorithm for noisy functions*, WSC '07: Proceedings of the 39th conference on Winter simulation (Piscataway, NJ, USA), IEEE Press, 2007, pp. 497–504.

[6] D. E. Finkel, *Global optimization with the direct algorithm*, Ph.D. thesis, North Carolina State University, 2005.

[7] D. E. Finkel and C. T. Kelley, *Convergence analysis of the DIRECT algorithm*, Tech. report, North Carolina State University, 2004.

[8] K. R. Fowler, G. Gray, and M. Olufsen, *Modeling heart rate regulation - Part II: Parameter identification and analysis.*, Cardiovasc Eng **8** (2008), 109–119.

[9] J. M. Gablonsky, *DIRECT version 2.0 user guide*, Tech. report, North Carolina State University, Department of Mathematics, 2001.

[10] ———, *Modification of the DIRECT algorithm*, Ph.D. thesis, North Carolina State University, 2001.

[11] R. B. Gramacy, *`tgp`: An R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models*, Journal of Statistical Software **19** (2007), no. 9.

[12] J. D. Griffin and T. G. Kolda, *Nonlinearly-constrained optimization using heuristic penalty methods and asynchronous parallel generating set search*, Applied Mathematics Research eXpress (2010), in press.

[13] Joshua D. Griffin and Tamara G. Kolda, *Asynchronous parallel hybrid optimization combining direct and gss*, Optimization Methods and Software (2009).

[14] W. E. Hart, *Python optimization modeling objects (Pyomo)*, Tech. report, Sandia National Laboratories, Discrete Math and Complex Systems Department, 2008.

[15] T. Hemker, *Derivative free surrogate optimization for mixed-integer nonlinear black box problems in engineering*, Ph.D. thesis, Department of Computer Science, Technical University Darmstadt, 2008.

[16] W. Hock and K. Schittkowski, *Test examples for nonlinear programming codes*, Springer, 1981.

[17] D. R. Jones, *DIRECT - Global optimization algorithm*, Encyclopedia of optimization **XXVII** (2001), 431–440.

[18] D. R. Jones, C. D. Pertunnen, and B. E. Stuckman, *Lipschitzian optimization without the lipschitz constant*, Journal of optimization theory and application: Vol. 79, No. 1, OCTOBER 1993 **Vol. 79, No. 1** (1993), 25.

[19] D. R. Jones, M. Schonlau, and W. J. Welch, *Efficient global optimization of expensive black-box functions*, Journal of Global Optimization **13** (1998), 455–492.

[20] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorics **4** (1984), 373–395.

[21] C. T. Kelley, *Iterative methods for optmization*, SIAM, 1999.

[22] J. J. Moré and S. M. Wild, *Benchmarking derivative-free optimization algorithms*, SIAM J. Optimization **20** (2009), no. 1, 172–191.

[23] J. T. Oden, T. Belytschko, J. Fish, T. J. R. Hughes, C. Johnson, D. Keyes, A. Laub, L. Petzold, D. Srolovitz, and S. Yip, *Simulation-based engineering science - revolutionizing engineering science through simulation*, Tech. report, Report of the National Science Foundation Blue Ribbon Panel on Simulation-Based Engineering Science, 2006.

[24] M. J. D. Powell, *The bobyqa algorithm for bound constrained optimization without derivatives*, Tech. report, Department of Applied Mathematics and Theoretical Physics, Centre for Mathematical Sciences, Wilberforce Road, Cambridge CB3 0WA, England., 2009.

[25] J. Puchinger and G. R. Raidl, *Artificial intelligence and knowledge engineering applications: A bioinspired approach*, Lecture Notes in Computer Science, ch. Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification, pp. 41–53, Springer Berlin / Heidelberg, 2005.

[26] R. G. Regis and C. A. Shoemaker, *Constrained global optimization of expensive black box functions using radial basis functions*, Journal of Global Optimization **31** (2005), 153–171.

[27] S. Sankaran, C. Audet, and A. L. Marsden, *A method for stochastic constrained optimization using derivative-free surrogate pattern search and collocation*, J. Comput. Phys. **229** (2010), no. 12, 4664–4682.

[28] M. Schonlau, W.J. Welch, and D. R. Jones, *Global versus local search in constrained optimization of computer models*, New Developments and Applications in Experimental Design IMS Lecture Notes - Monograph Series **34** (1998), 11–25.

[29] B. O. Shubert, *A sequential method seeking the global maximum of a function*, SIAM Journal on Numerical Analysis, Vol. 9, No. 3 (Sep., 1972), **9** (1972), no. 3, 379–388.

[30] E. S. Siah, M. Sasena, J. L. Volakis, P. Y. Papalambros, and R. W. Wiese, *Fast parameter optimization of large-scale electromagnetic objects using direct with kriging metamodeling*, Microwave Theory and Techniques, IEEE Transactions on **52** (2004), no. 1, 276 – 285.

[31] M. Taddy, H.-K. H. Lee, G. A. Gray, and J. D. Griffin, *Bayesian guided pattern search for robust local optimization*, 2009.

[32] A. Wächter, *An interior point algorithm for large-scale nonlinear optimization with application in process engineering*, Ph.D. thesis, Carnegie Mellon University, 2002.

[33] _____, *Introduction to ipopt: A tutorial for downloading, installing, and using ipopt.*, 2008.

[34] Stefan M. Wild and Christine A. Shoemaker, *Gorbit: Global optimization by multistart radial basis function algorithms*, Tech. Report ORIE-1473, Cornell University School of Operations Research and Information Engineering, July 2009.

[35] L. E. Zerpa, N. V. Queipo, and S. Pintos, *An optimization methodology of alkaline-surfactant-polymer flooding processes using field scale numerical simulation and multiple surrogates*, SPE/DOE Symposium on Improved Oil Recovery, 17-21 April 2004, Tulsa, Oklahoma, 2004.

List of Figures

List of Tables

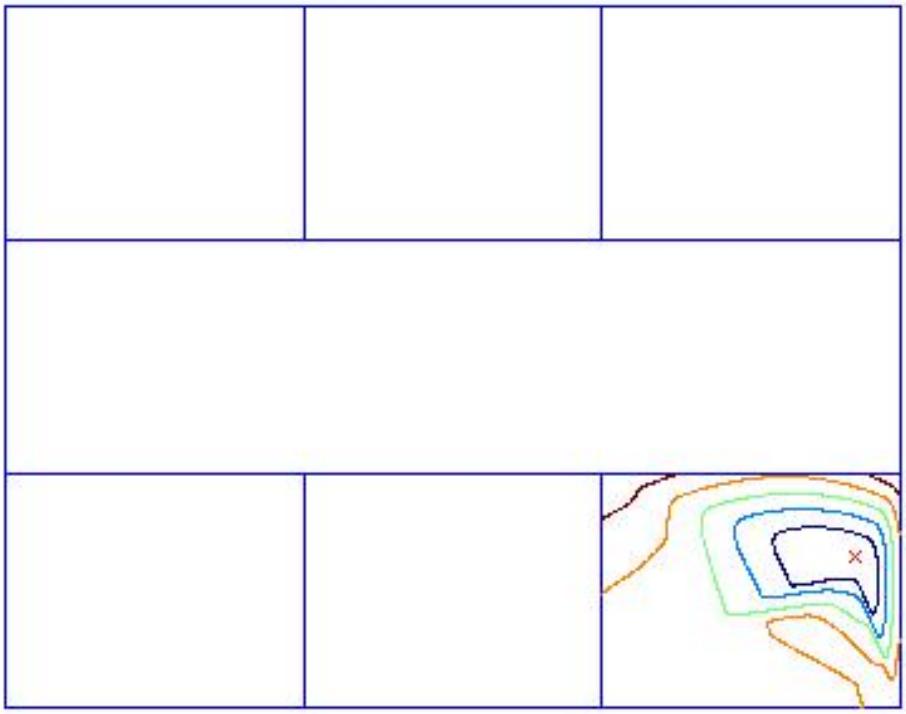

Figure 1: Illustration of the hybrid DIRECToS on a 2-D example. The lower right corner is covered by a surrogate to find the next candidate on it, instead of evaluating the center point.

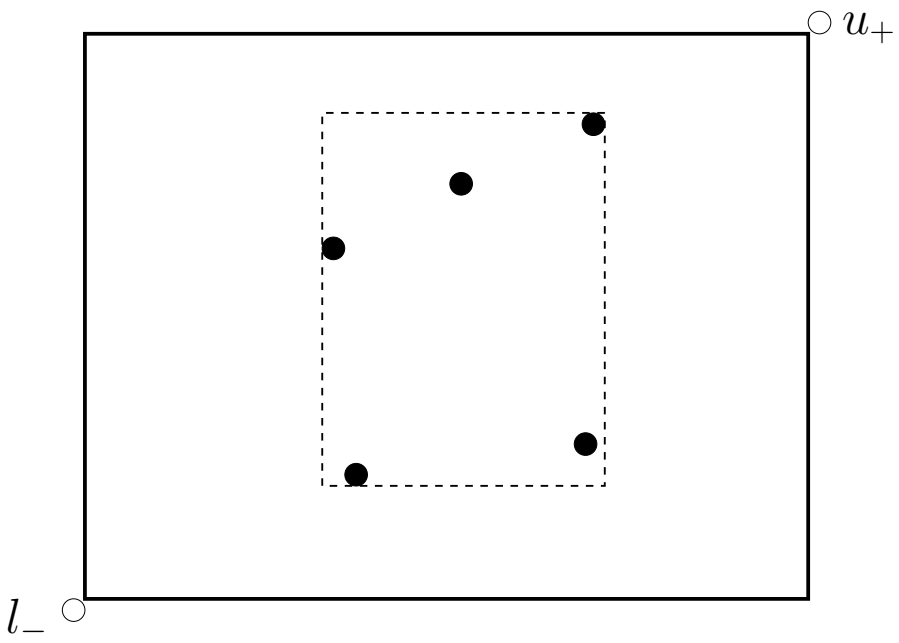

Figure 2: Valid box for surrogate evaluations (dashed) and the problem bounds (solid).

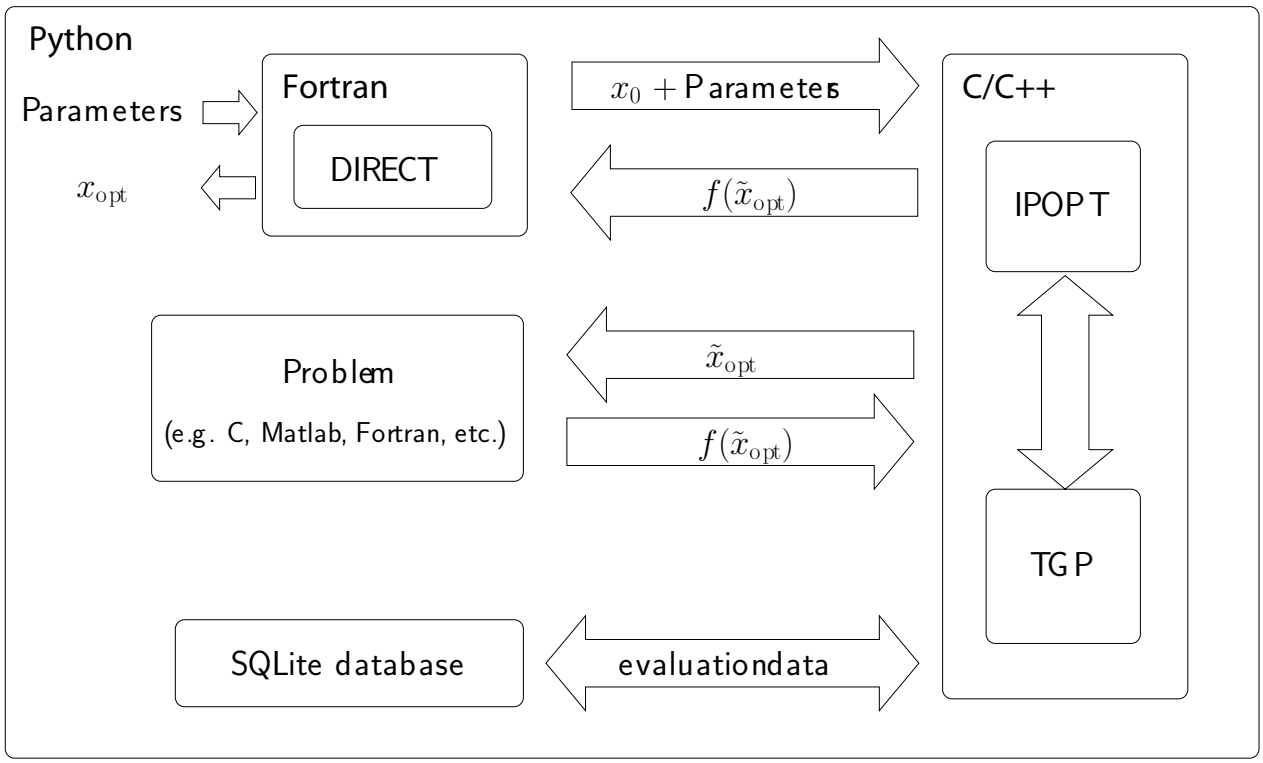

Figure 3: The python framework of DIRECToS.

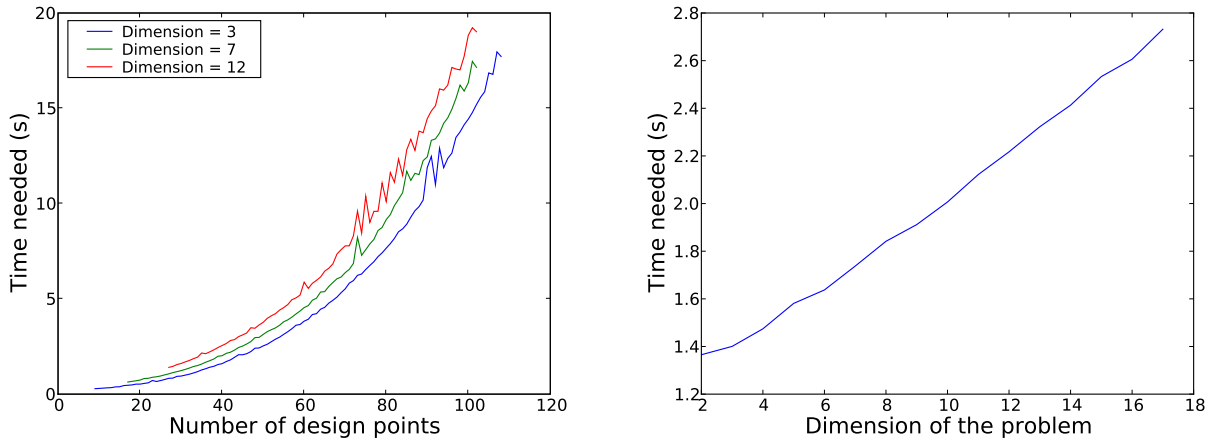

Figure 4: Fitting Performance using TGP.

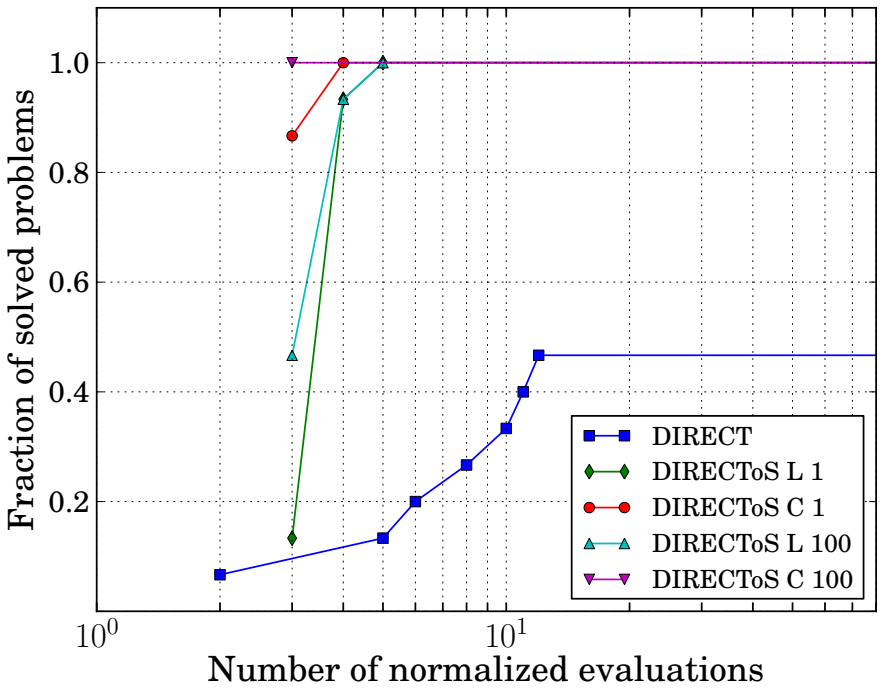

Figure 5: Data profile for the smooth "Kelley quadratics".

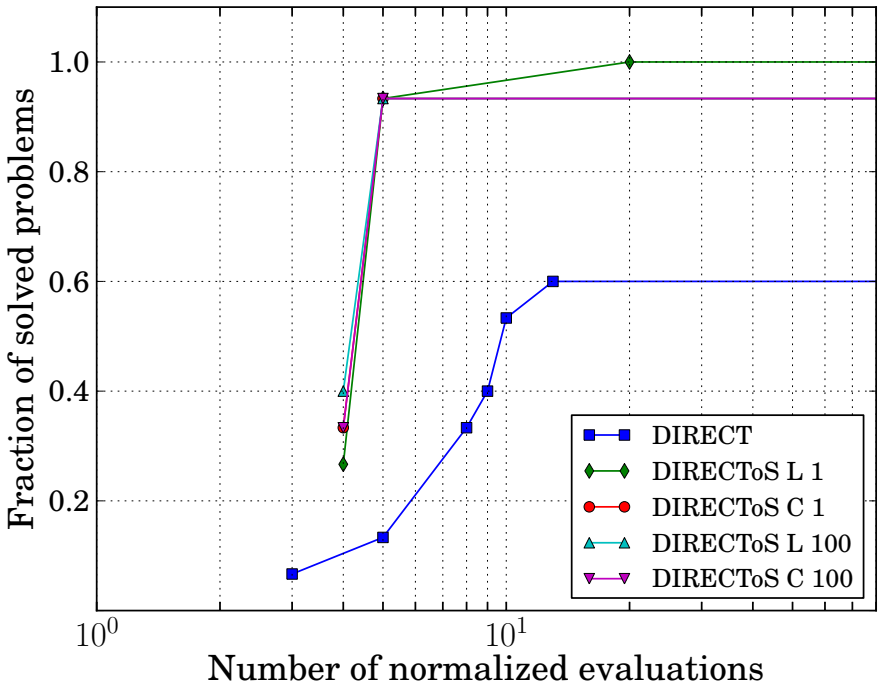

Figure 6: Data profile for the noisy "Kelley quadratics".

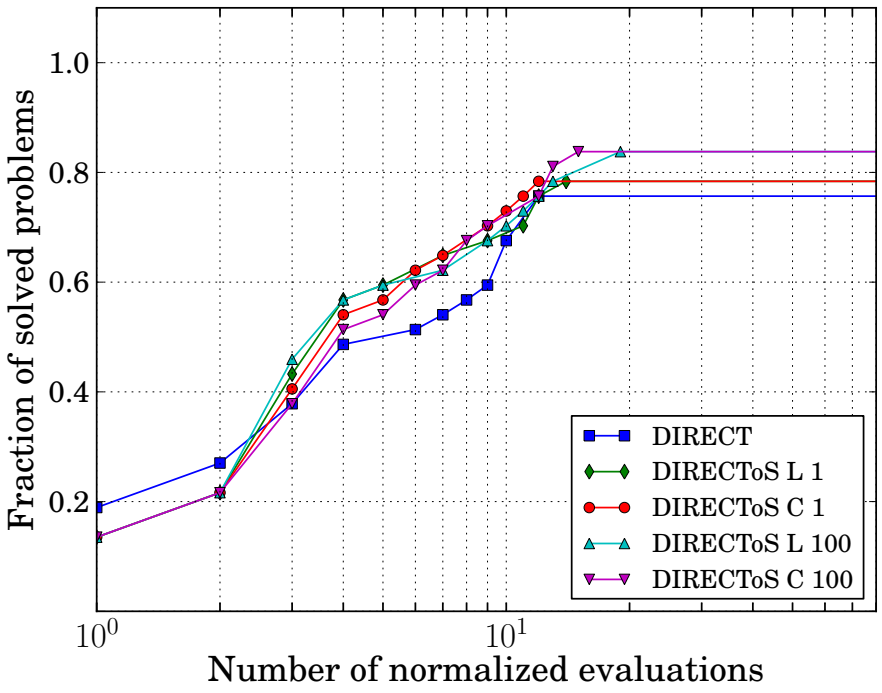

Figure 7: Data profile for the smooth CUTEr problems.

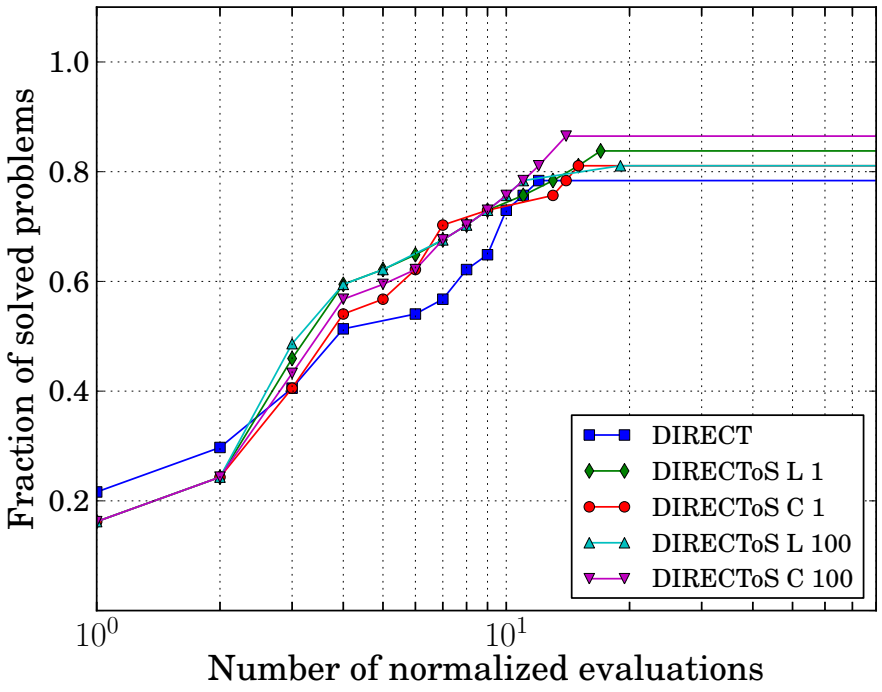

Figure 8: Data profile for the noisy CUTEr problems.

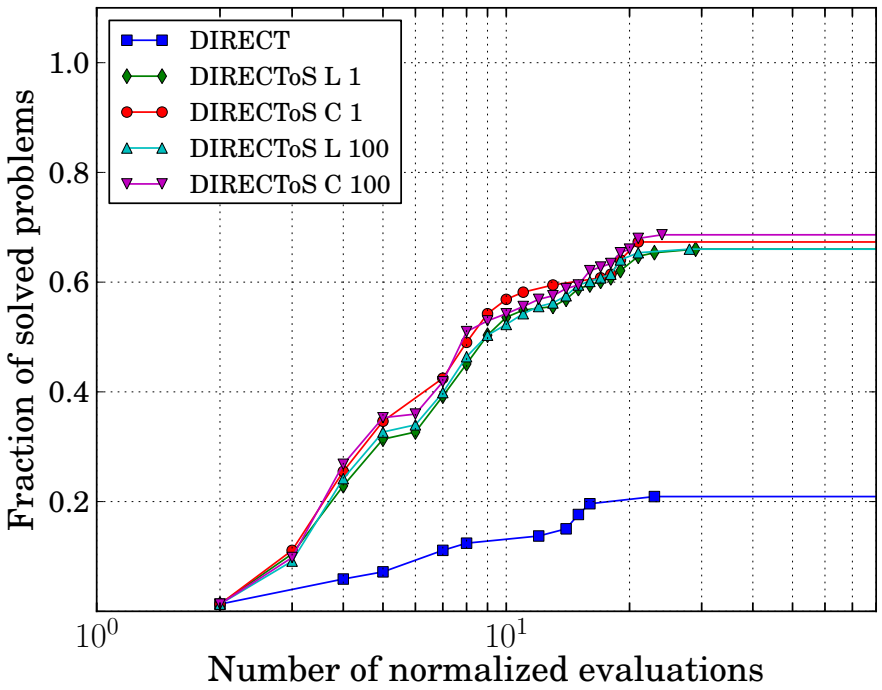

Figure 9: Data profile for the noisy nonlinear problems.

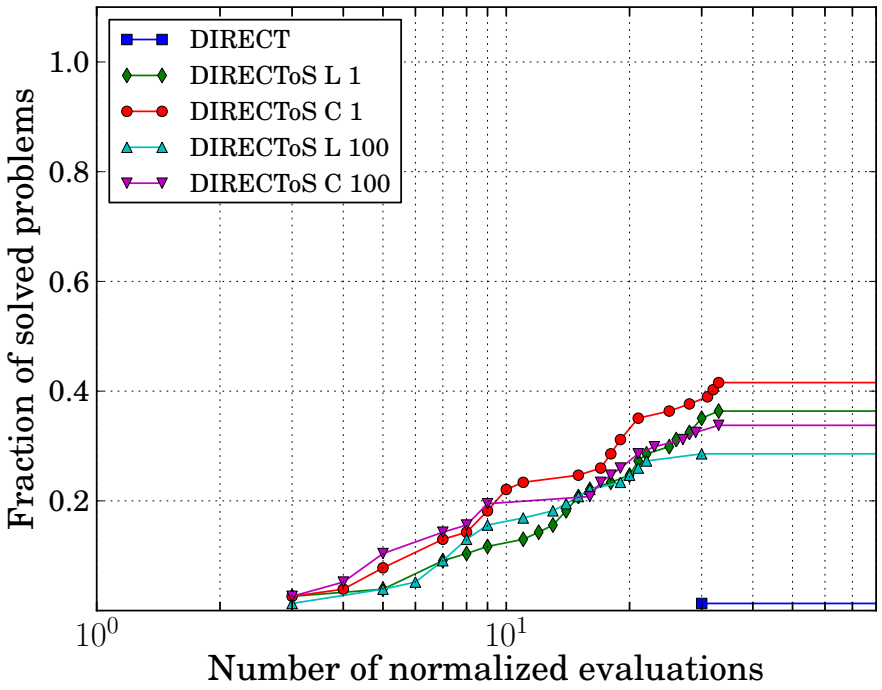

Figure 10: Data profile for the noisy nonlinear problems. (Tolerance $\theta = 10^{-5}$)

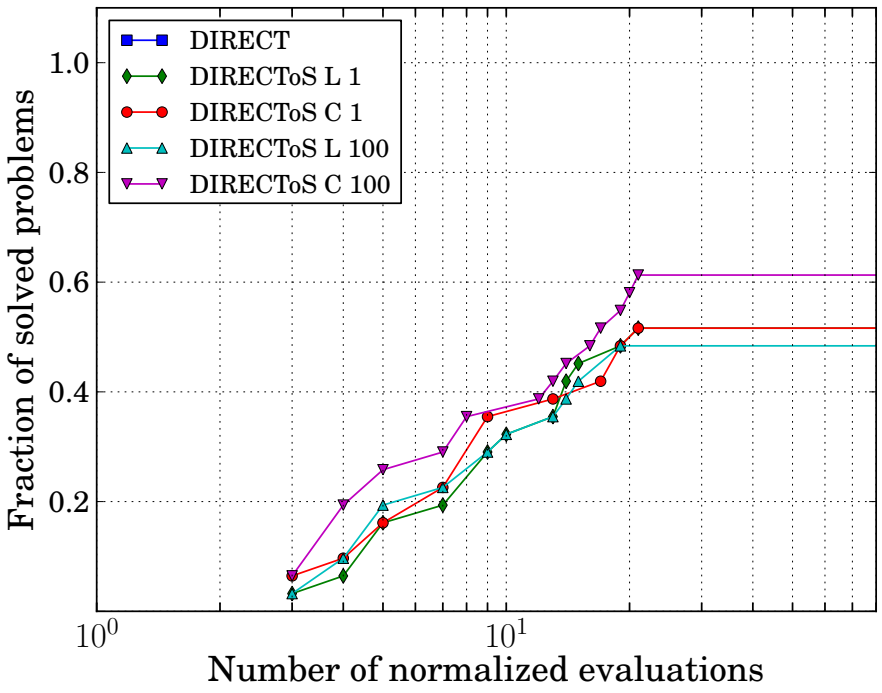

Figure 11: Data profile for the NLP problems with equality constraints.

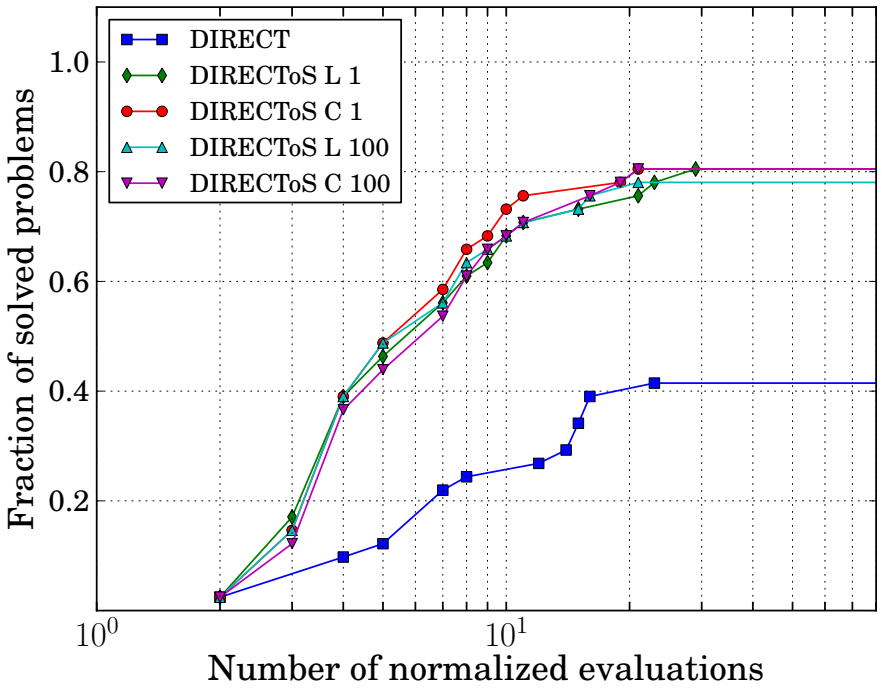

Figure 12: Data profile for the NLP problems with inequality constraints.

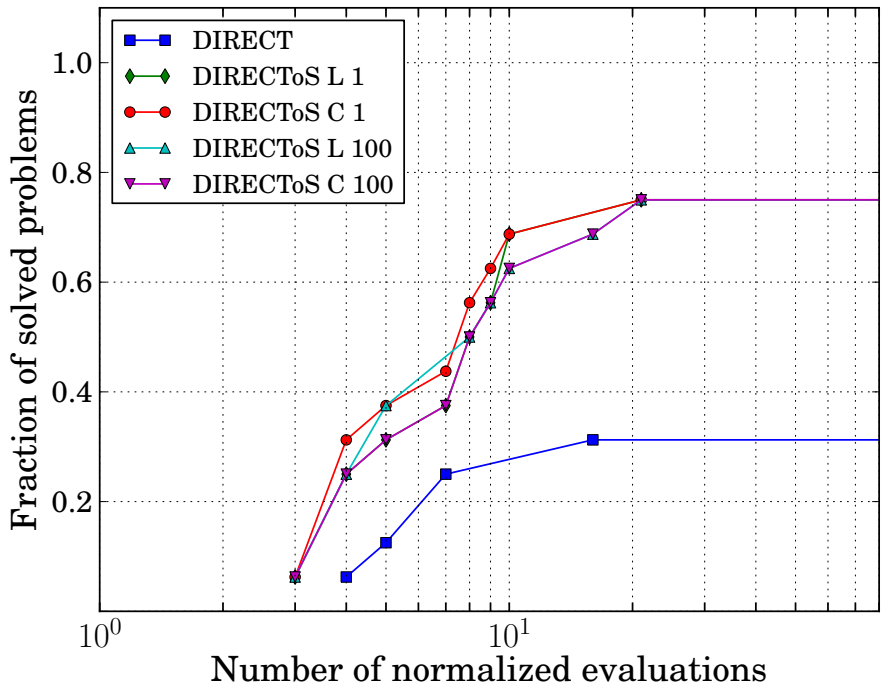

Figure 13: Data profile for the NLP problems with a quadratic objective function and inequality constraints.

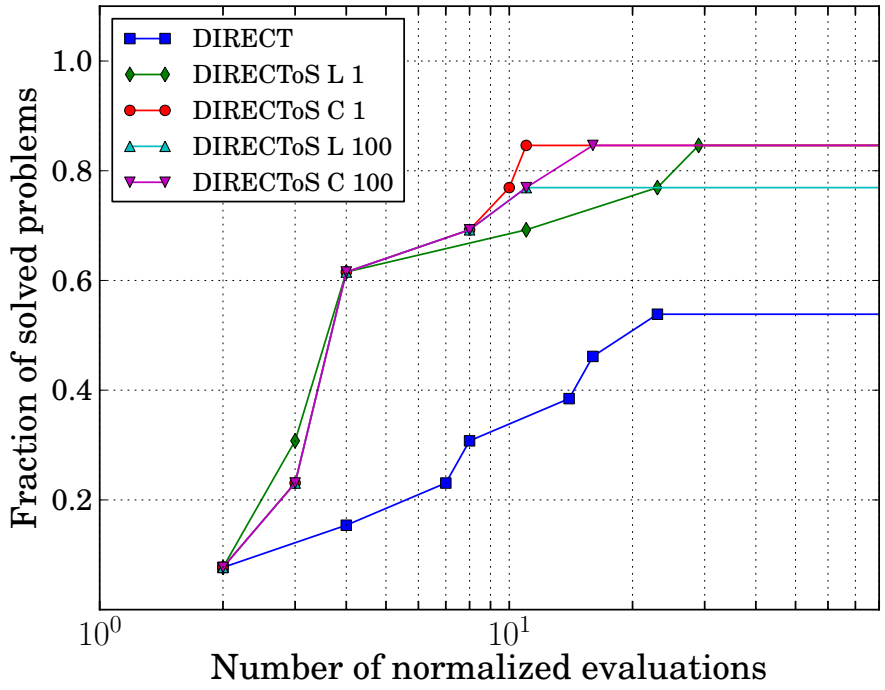

Figure 14: Data profile for the NLP problems with a polynomial objective function and inequality constraints.

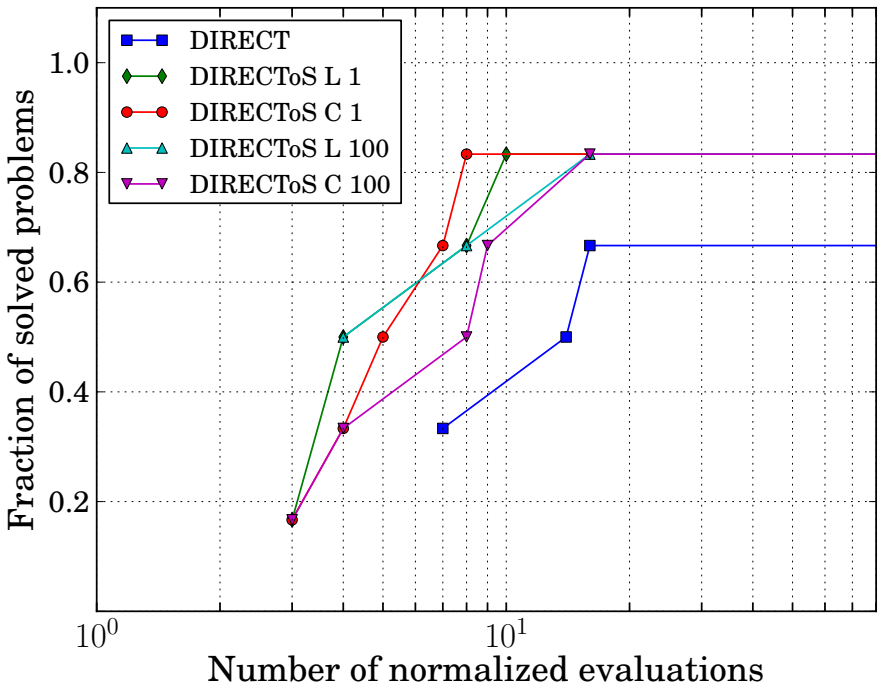

Figure 15: Data profile for the NLP problems with linear inequality constraints.

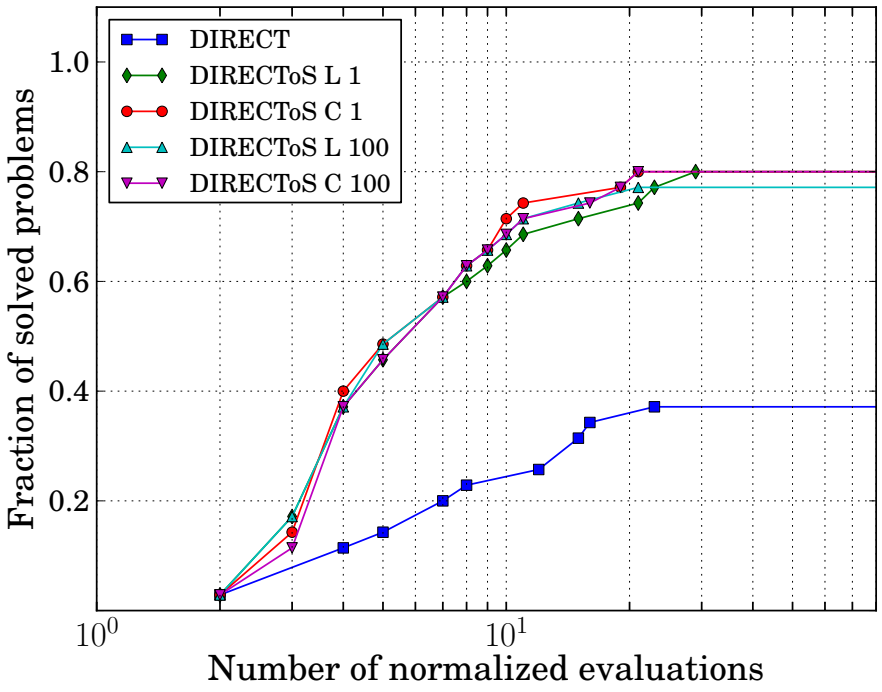

Figure 16: Data profile for the NLP problems with nonlinear inequality constraints.

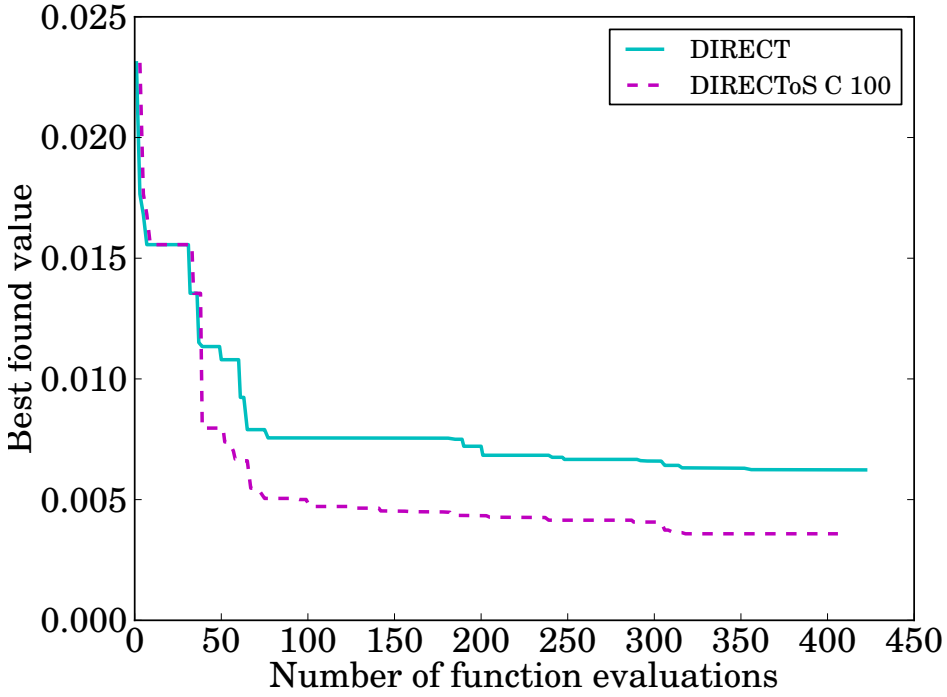

Figure 17: Comparison of DIRECT and DIRECToS on the heart rate regulation problem. Plotted is the number of objective function calls versus the best value found.

Typ of problem chracteristic	Number of problems
Inequality constraints only	53
Equality constraints only	37
Mixed constraints	8
Linear constraints	24
Nonlinear constraints	74
Linear objective function	12
Quadratic objective function	32
Polynomial objective function	43
General objective function	17

Table 1: Classifications of NLP problems with the number of associated problems. A problem can belong to multiple categories.

Method	Initial f	Minimized f	Percent decrease	Number of function evaluations
NM	0.01700	0.00246	83,3%	477
IF	0.01700	0.00246	83,4%	1981
IF2	0.00282	0.00246	$<$ 1% more	390 (+ IF)
IFNM	0.00283	0.00246	2% more	414 (+ NM)
GA	None	0.00246		5810
DIRECT	0.02309	0.00623	73,0%	424
DIRECToS	0.02309	0.00358	84,4%	407

Table 2: Optimized and initial function values for each method applied to the heart rate regulation problem compared to the results obtained in [8].