ICRS-Filter: A Randomized Direct Search Algorithm for Constrained Nonconvex Optimization Problems

Biyu Li[1], *Viet H. Nguyen*[1], *Chieh. L. Ng*[1], *E. A. del Rio-Chanona*[1]

Vassilios S. Vassiliadis[1]*, *Harvey Arellano-Garcia*[2]

[1]Department of Chemical Engineering and Biotechnology, University of Cambridge, Pembroke Street, Cambridge CB2 3RA, UK

[2]Department of Chemical and Process Engineering, University of Surrey, Guildford GU2 7XH, UK

Abstract

This work presents a novel algorithm and its implementation for the stochastic optimization of generally constrained Nonlinear Programming Problems (NLP). The basic algorithm adopted is the Iterated Control Random Search (ICRS) method of Casares and Banga (1987) with modifications such that random points are generated strictly within a bounding box defined by bounds on all variables. The ICRS algorithm serves as an initial point determination method for launching gradient-based methods that converge to the nearest local minimum. The issue of constraint handling is addressed in our work via the use of a filter based methodology, thus obviating the need for use of the penalty functions as in the basic ICRS method presented in Banga and Seider (1996),which handles only bound constrained problems. The proposed algorithm, termed ICRS-Filter, is shown to be very robust and reliable in producing very good or global solutions for most of the several case studies examined in this contribution.

*Corresponding Author: vsv20@cam.ac.uk

Keywords:

nonconvex programming problem; randomized search; nonlinear programming; stochastic search algorithms;

1 Introduction

Optimization of nonconvex programming problems has an important role in Applied Mathematics, Computer Science as well as scientific and engineering practices. The significance of the global solution in some cases is 'non-negotiable', as it could signify "profit or loss" for chemical manufacturers, or "make-or-break" functional properties of proteins in drugs research by predicting their conformational structure.

There are two main approaches to addressing global optimization problems: deterministic and stochastic methods. Reviews of the deterministic global optimization methods are given in Floudas (1999) and Floudas and Misener (2009). For a given problem, deterministic methods are able to provide a certificate of global optimality of the final solution. Deterministic methods generally tend to be computationally expensive with computational times growing very quickly with problem sizes.

The other approach, which is based on stochastic algorithms, improves an initial point using stochastic perturbations. In the stochastic approach, the objective function is evaluated at randomly generated points and the process terminates when there is no further improvement in the objective function value as well as satisfaction of convergence criteria. Stochastic methods can only guarantee solutions which are local optima, without being able to certify global optimality. However, the methods' ability in efficiently and reliably locating local optima has been proven in various practical applications, especially for very large problems when "good enough" solutions are acceptable. Stochastic methods frequently employ multiple starting points to increase the chance of finding the global optimum (Hickernell and Yuan (1997),Torn (1978), Fouskakis and Draper (2002)).

Our work falls into the latter category of optimization methods and a new method, termed ICRS-Filter Method, will be presented which is the combination between the

Integrated Controlled Random Search (ICRS) algorithm originally developed by Casares and Banga (1987) and the Filter approach (Fletcher and Leyffer (2000)) to deal with generally constrained NLP problems.

2 The generic ICRS method

The ICRS method was first developed by Casares and Banga (1987). Banga proposed the ICRS method as a stochastic search method for global optimization of problems with bounds on variables. The method operates by generating random points obeying a normal distribution within the bounds. As the iterations progress, and as acceptances of improving points become fewer, the standard deviation of the normal distribution is suitably reduced thus inducing a more localized search around a current point desired to be improved.

The original ICRS Algorithm applies to an unconstrained problem, which is assumed to have the following formulation (***P1***):

Problem P1

$$\min_x \ f(x) \tag{2.1a}$$

subject to

$$x^L \leq x \leq x^U \tag{2.1b}$$

where $x \in \mathbb{R}^n$

The Algorithm is presented as Algorithm 1. The ICRS Algorithm is a search method, which instead of employing a set of search directions, it uses randomly generated points. As the Algorithm generates points closer to a local minimum, the standard deviation σ is reduced, hence the "contracting spheres" picture as shown in Figure 2.1.

It is important to note that the ICRS Algorithm is a *randomized* direct search method and this is to be constrasted with other well-known methods in which the search directions

Algorithm 1 ICRS Algorithm

Initial Guess $\leftarrow x_0$
Initial Deviation Factor $\leftarrow k_1$
Reduction Deviation Factor $\leftarrow k_2$
Expansion Deviation Factor $\leftarrow k_3$
Maximum Number of Samples $\leftarrow N_{\text{Sample}}$
Maximum Number of Failures $\leftarrow N_{\text{Failure}}$
Variable Convergence Tolerance $\leftarrow \varepsilon$
Evaluate Best Objective Function Value $f_{\text{Best}} \leftarrow f(x_0)$
Compute Initial Deviation Factor $\sigma \leftarrow k_1 \cdot (x^U - x^L)$
Set Current Solution Vector $x_{\text{Best}} \leftarrow x_0$
Set $ifailure \leftarrow 0$
for $i \leftarrow 1$ *to* N_{Sample} **do**
    Generate a new point x_{New} which is Normally distributed between x^U and x^L, given the Mean x_{Best} and Standard Deviation σ
    $f_{New} \leftarrow f(x_{New})$
    if $f_{\text{New}} < f_{\text{Best}}$ **then**
        Variable Tolerance $\leftarrow \phi(x_{\text{New}}, x_{\text{Best}})$
        Update Objective Value $f_{\text{Best}} \leftarrow f_{\text{New}}$
        Update Current Solution $x_{\text{Best}} \leftarrow x_{\text{New}}$
        Expand Deviation Factor $\sigma \leftarrow k_3 \cdot \sigma$
        if Variable Tolerance $< \varepsilon$ **then**
            Exit Sampling Loop
        end if
    else
        if $f_{\text{New}} \geq f_{\text{Best}}$ **then**
            $ifailure \leftarrow ifailure + 1$
            if $ifailure > N_{\text{Failure}}$ **then**
                Reduce Deviation Factor $\sigma \leftarrow k_2 \cdot \sigma$
                Reset Counter $ifailure \leftarrow 0$
            end if
        end if
    end if
end for
return Best Solution x_{Best} and Best Objective Value f_{Best}

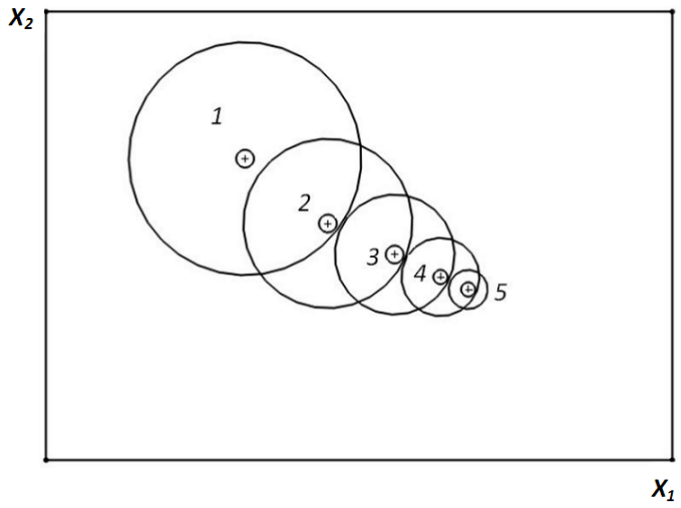

Fig. 2.1: Illustration of the ICRS Algorithm

are generated deterministically, such as the Nelder-Mead Simplex Algorithm (Correia et al. (2010) and Nelder and Mead (1965)). Their algorithm is evidently unable to handle any other constraints on the variables' domain, which can be easily induced by adding equalities or inequalities to the original ***(P1)*** problem. Consequently, the ICRS approach is only effective at solving unconstrained optimization problems.

The most important step in the ICRS algorithm is the generation of normally distributed points within given bounds. The following methods have been attempted in this work:

1. Projection to bounds method

The principle behind the method is very simple: given x_0 and σ, generate a random point x which is normally distributed with mean x_0 and with a standard deviation σ. The method used to generate the points x is adapted from Box and Muller (1958). Furthermore, if *any* elements in x are falling below the lower bound or exceeding the upper bound, they will be replaced by the correspoding lower or upper bound values, i.e. "clipped to the bounds".

The "*Projection to bounds method*" often causes the sampling points to "stick" onto the bounds too often and leads to an uneven distribution in the interior of the sampling region. Furthermore, in problems containing functions, which are undefined at the bounds, the method could lead to numerical instabilities. Therefore, the method is not

strongly recommended, but it is still included in the discussion as a legacy of the original implementation.

2. Rejection method

The approach is the same as before. However, this method would reject a point if *any* element in x lies outside its bounds. Consequently, generated points are verified whether they are within the bounds. If they are not, the corresponding variable values are rejected until a point is found to satisfy the bounds. The method possibly requires more iterations in the normal random number generator so the CPU time would be slightly more intensive. However, the method tends to sample more evenly in the variable domains, and the overall behaviour is 'smoother' in comparison to that of the original "*Projection to bounds method*".

3. *Truncated normal distribution method*

This method guarantees to generate points precisely within the bounds. To illustrate the key idea, the problem is first restated below

> *Given the mean* x_0 *and standard deviation* σ *and bounds* x^L *and* x^U*, generate a random point* x *such that* $x \in N(x_0, \sigma)$ *and* $x^L \leq x \leq x^U$

Given a random variable x, the cumulative distribution function $I(x)$ is defined as the area under the Normal Distribution curve between $-\infty$ and x:

$$I(x) = CDF(x_0, \sigma, x) \tag{2.2}$$

where CDF is the Cumulative Distribution Function.

Conversely, given $I(x)$, the inverse of the cumulative distribution function is defined as the random variable x which satisfies the above relation:

$$x = CDF^{-1}(x_0, \sigma, I(x)) \tag{2.3}$$

Since $I(x)$ is a monotonic increasing function of x , there is a one-to-one correspondence between $I(x)$ and x. Based on this idea, the *"Truncated normal distribution method"* is presented in Algorithm 2:

Algorithm 2 Truncated Normal Distribution Method

1: Calculate $I^L = I(x^L)$ and $I^U = I(x^U)$
2: Generate a random number u which is uniformly distributed between 0 and 1
3: Compute the cumulative distribution function $I \leftarrow I^L + (I^U - I^L) \cdot u$
4: Invert I to obtain the corresponding random variable x using equation (2.2)

Since $I^L \leq I \leq I^U$, and the function is monotically increasing, this leads to $x^L \leq x \leq x^U$. Furthermore, to generate a randon number which is uniformly distributed between 0 and 1, the work by Park and Miller (1988) has been adapted in our implementation.

The *"Truncated normal distribution method"* method is considered to be the most preferred among the three approaches mentioned above because it neither tends to favor sampling on the bounds nor rejects any points as the other two methods would do, respectively. Therefore, it has been used for all the case studies in this work.

In addition to the modifications made to generate points within bounds, another improvement of the original ICRS method corresponds to the application of the filter concept to handle additional constraints. To be specific, it is a modification of the ICRS method acceptance criterion from strict improvement of an unconstrained function to that of a filter method for constraint handling, and thus, taking into account directly the values of the objective and the constraints in deciding whether a candidate point is acceptable as an improving point for generally constrained optimization problems. The next section will show how the filter method is adapted to deal with constraints.

3 Filter methods for constraint handling

For a generally constrained optimization problem, the following formulation ***(P2)*** is assumed:

Problem P2

$$\min_x \ f(x) \tag{3.1a}$$

subject to

$$h(x) \ = 0 \tag{3.1b}$$

$$g(x) \leq 0 \tag{3.1c}$$

and

$$x^L \leq x \leq x^U \tag{3.1d}$$

where $x \in \mathbb{R}^n$, $h \in \mathbb{R}^{n_e}$and $g \in \mathbb{R}^{n_i}$. A well-known approach is to convert the constrained problem into an unconstrained one. Popular methods are the penalty functions, barrier methods, and the augmented Lagrangian method (Edgar et al., 2001; Powell, 1969). All of these methods in essence absorb the equalities, inequalities and objective function into a single function. Such an approach is not always successful since the resulting function is often highly nonlinear and nonconvex. Thus, the determination of local and global optima can be very challenging.

The original ICRS algorithm is effective at solving problem *(**P1**)*, but is expected to perform poorly for the *(**P2**)* type problem. To handle the extra issue of having constraints the concept of a filter is applied, which leads to the ICRS-Filter method, the key original contribution in this work.

A brief review of the filter concept is given by Fletcher et al. (2006). Similar discussions can also be found in Correia et al. (2010) and Karas et al. (2006). The following definitions are central to the construction of a filter:

Definition 1. Let $F(x)$ and $G(x)$ be two real-valued scalar functions. A point x is said to dominate point y if and only if $F(x) < F(y)$ and $G(x) < G(y)$. Or equivalently, the entry $(F(x), G(x), x)$ is said to dominate the entry $(F(y), G(y), y)$.

Definition 2. A filter $\mathcal{F}$ is a list of entries $(F(x), G(x), x)$ such that no entry dominates the others.

The ICRS-Filter method separates the objective function from the equalities and inequalities. The method essentially attempts to solve a bi-objective problem: minimizing the objective value of $f(x)$ while keeping the constraints satisfied by reducing their violation norm.

The second objective (*i.e.* constraint satisfaction) is formulated by aggregating the equalities and inequalities into a single function. This single function effectively measures the overall deviation of the constraints from zero. Some possible functions are:

$$\text{Norm-1} \qquad \Phi_1(x) = \sum_{i=1}^{n_e} |h_i(x)| + \sum_{j=1}^{n_i} |\max\{0, g_j(x)\}| \tag{3.2a}$$

$$\text{Norm-2} \qquad \Phi_2(x) = \sum_{i=1}^{n_e} (h_i(x))^2 + \sum_{j=1}^{n_i} (\max\{0, g_j(x)\})^2 \tag{3.2b}$$

$$\text{Norm-}\infty \qquad \Phi_\infty(x) = \max_{i\in\{1,\ldots,n_e\}} \{|h_i(x)|\} + \max_{j\in\{1,\ldots,n_i\}} \{\max\{0, g_j(x)\}\} \tag{3.2c}$$

which are respectively the sums of the norm-1, norm-2, and the infinity norm of $h(x)$ and the violations of $g(x)$.

In this work, the norm-1 in equation (3.2a) is chosen for the implementation of the ICRS-Filter method due to its equal weighting of the constraint values.

The final formulation of problem *(**P2**)* becomes problem *(**P2'**)*:

Problem P2'

$$\min_x \left(f(x), \Phi(x)\right) \tag{3.3a}$$

subject to

$$x^L \leq x \leq x^U \tag{3.3b}$$

It is noted that in equations (3.3a-3.3b) and in the rest of this work, the subscript 1 in $\Phi_1(x)$ is dropped to simplify notation and allow for generality. To apply the filter concept to the above problem, from **Definition 1**, let $f(x)$ be the objective function and the aggregated constraint norm be $\Phi(x)$. Thus, each point in the filter is represented by the pair $(f(x), \Phi(x))$ or entry $(f(x), \Phi(x), x)$. Additionally, during the construction of a filter, two steps are required. The first step is to decide whether or not to accept a new point to the current filter list:

Acceptance Criterion 1:

A new point x is accepted to the filter if and only if it is not dominated by any present entry in the list. Therefore, for a newly generated point, the above criterion is equivalent to checking Definition 1 for the point against all other points in the current filter. This is perhaps the simplest and most straightforward criterion. Another simple criterion regards the magnitude of the constraint norm $\Phi(x)$.

Acceptance Criterion 2:

A new point x is accepted to the filter if and only if $\Phi(x) \leq \Phi_{\max}$, where $\Phi_{\max}$ is a user-defined upper bound on the norm. Hence, if $\Phi(x)$ is too large then the point is rejected, which is useful when the filter already has many entries.

The second step in the filter construction is to update the filter. It essentially consists of two basic steps:

Filter Updating Step 1:

Given that a new point x is accepted to the filter, check and eliminate all the current entries, which are dominated by x.

Filter Updating Step 2:

Reorganize the filter points according to the ascending order of the constraint norms $\Phi(x)$ (for convenience in our implementation).

Thus, after the filter updating steps, the points are organized based on their constraint satisfaction (*i.e.* feasibility) with the left-most entries being the most feasible and the right-most entries being the least feasible. The construction of the filter in this work follows the above description. Fletcher and Leyffer (2000) and Fletcher et al. (2006) add the extra following steps to their algorithm:

1. Removal of the blocking entries from the filter.
2. Addition of an 'envelope' to the current filter.
3. For a new point with a reduction in constraint norm, check that there is also a "sufficient reduction" in the objective value.

Point 1 and 2 prevent their algorithm from converging to infeasible local minima and hence are not considered in this work as the ICRS method is stochastic and not locating local minima precisely. Furthermore, since it is possible that the filter points form a monotonic decreasing sequence of constraint norms $\Phi(x)$, but the objective value $f(x)$ may form an increasing sequence at the same time, Point 3 guards against such an issue by accepting only points which show some degree of reduction in the value of $f(x)$. The work in this contribution does not use these criteria, but it could be easily adapted in a future implementation.

By repetitively applying the acceptance and updating steps, a filter with decreasing constraint norm values (*i.e.* more feasible points) is constructed. It is also worth mentioning that due to the updating steps, the filter is a dynamic object which changes size frequently.

An illustration of the working mechanisms of a dynamic filter is given in Figure 3.1. With reference to this figure, the following steps illustrate the operation of the filter:

1. Suppose that the filter currently has four points 1, 2, 3 and 4

2. If a point such as A is generated, it will be rejected because it is dominated by other points in the current filter based on *Acceptance Criterion 1*

3. If a point such as B is generated, it will be accepted because it is not dominated by any other point in the current filter based on *Acceptance Criterion 1*

4. If a point such as C is generated:

   (a) It will be accepted because it is not dominated by any other point in the current filter

   (b) The Filter Updating Steps will remove Points 2, 3 and B because they are dominated by C

   (c) The current Filter now has Points 1, C and 4

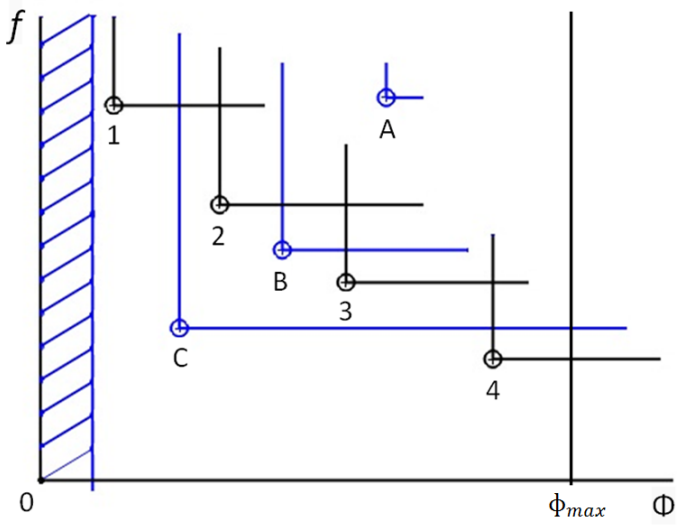

Fig. 3.1: Filter method illustration

Having introduced the ICRS algorithm and shown the complete construction of a filter, the next section will discuss the implementation of the ICRS-Filter method and show how it is used to solve constrained optimization problems.

4 The ICRS-Filter method Algorithm and Implementation

4.1 ICRS-Filter Method

Assuming the formulation ***(P2')*** above, the ICRS-Filter Method is presented in Algorithm 3. In this algorithm $FilterSize$ is the number of points in the filter.

Algorithm 3 ICRS-Filter Method

Initial Guess $\leftarrow x_0$
Initial Deviation Factor $\leftarrow k_1$
Deviation Reduction Factor $\leftarrow k_2$
Deviation Expansion Factor $\leftarrow k_3$
Maximum Number of Samples $\leftarrow N_{\text{Sample}}$
Maximum Number of Failures $\leftarrow N_{\text{Failure}}$
Maximum Number of Elements in Filter $\leftarrow N_{\text{max filter}}$
Maximum Constraint Norm $\leftarrow \Phi_{\max}$
Variable Convergence Tolerance $\leftarrow \varepsilon_{\text{Variable}}$
Objective Convergence Tolerance $\leftarrow \varepsilon_{\text{Objective}}$
Constraint Norm Convergence Tolerance $\leftarrow \varepsilon_{\text{Norm}}$
Evaluate the Objective Function Value $f_0 \leftarrow f(x_0)$
Evaluate the Constraint Norm Value $\Phi_0 \leftarrow \Phi(x_0)$
Compute Initial Deviation $\sigma \leftarrow k_1 \cdot (x^U - x^L)$
Initialize the Filter with the first entry f_0, Φ_0, x_0
Set the Current Centre $x \leftarrow x_0$
for $i \leftarrow 1$ *to* N **do**
  Apply Algorithm 2 to generate a new point x_{New}
  $f_{New} \leftarrow f(x_{\text{New}})$
  $\Phi_{New} \leftarrow \Phi(x_{\text{New}})$
  Decide whether the entry $(f_{New}, \Phi_{New}, x_{\text{New}})$ is accepted to the current filter using the acceptance criteria
  if the point is accepted **then**
    Updade the filter using the updating steps
    if $FilterSize > N_{\text{Max Filter}}$ **then**
      Remove right-most filter entries to restore $N_{\text{Max Filter}}$
    end if
    Update Centre $x \leftarrow x_1$ where x_1 is the left-most entry in the current filter
    Compute Variable Tolerance $\leftarrow \phi_1(x_{\text{New}}, x_1)$
    Compute Objective Tolerance $\leftarrow \phi_2(f_{New}, f(x_1))$
    Compute Constraint Norm Tolerance $\leftarrow \phi_3(\Phi_{New}, \Phi x_1)$
    if $|\phi_1| < \varepsilon_{\text{Variable}}$ and $|\phi_2| < \varepsilon_{\text{Objective}}$ and $|\phi_3| < \varepsilon_{\text{Norm}}$ **then**
      Exit Sampling Loop
    end if
    Reset Counter $ifailure \leftarrow 0$
    Expand Deviation $\sigma \leftarrow k_3 \cdot \sigma$
  else
    if the point is not accepted **then**
      $ifailure \leftarrow ifailure + 1$
      if $ifailure > N_{\text{Failure}}$ **then**
        Reduce Deviation $\sigma \leftarrow k_2 \cdot \sigma$
        Reset Counter $ifailure \leftarrow 0$
      end if
    end if
  end if
end for
Filter points are used to initialize a local optimization solver
The smallest objective value obtained is the final solution

4.2 Implementation of the ICRS-Filter Method

The ICRS-Filter method was implemented in *Mathematica*™ (Version 8.0). The ICRS-Filter method has 9 parameters which are to be initialized before any computations. The parameter values used are listed in Table 1:

Tab. 1: ICRS-Filter Method Parameters' Values

Parameter	Value
Initial Deviation Factor k_1	$1/6$
Deviation Reduction Factor k_2	$1/2$
Deviation Expansion Factor k_3	1
Maximum Number of Samples N_{Samples}	10^6
Maximum Number of Failures N_{Failure}	$25 \cdot N_{\text{Variable}}$
Maximum Constraint Norm $\Phi_{\max}$	10^{30}
Variable Convergence Tolerance $\varepsilon_{\text{Variable}}$	10^{-3}
Objective Convergence Tolerance $\varepsilon_{\text{Objective}}$	10^{30}
Constraint Norm Convergence Tolerance $\varepsilon_{\text{Norm}}$	10^{30}
Maximum Filter Size $N_{\max \text{filter}}$	N_{Sample}

N_{Variable} is the number of variables in a specific problem. The values for k_2 and N_{Variable} are adapted from Banga and Seider (1996). It is noted that in their paper, $k_1 = 1/3$ and $\varepsilon_{\text{Variable}} = 10^{-4}$, *i.e.* their algorithm allows a wider search region as well as a more stringent convergence condition.

It is evident that setting tight values to $\varepsilon_{\text{Variable}}$, $\varepsilon_{\text{Objective}}$ and $\varepsilon_{\text{Norm}}$ (*e.g.* 10^{-3}) would demand a great deal of computation, so it has been decided to set a tight tolerance only on the variables and set the other tolerances to relaxed values. Consequently, since $\varepsilon_{\text{Objective}}$ and $\varepsilon_{\text{Norm}}$ are very large, the functional forms of ϕ_2 and ϕ_3 would not be factors in deciding the convergence of the method. By contrast, the functional form of the variable tolerance is the most important factor and the following has been used (Banga and Seider (1996)):

$$\phi_1 = \max\left(\frac{|x_{\text{New}} - x_1|}{|x^U - x^L|} \right) \tag{4.1}$$

Due to *Acceptance Criterion 2*, $\Phi_{\max}$ has been set to a high value to increase the chance of a point being accepted to the filter . In addition, the expression of $\Phi_{\max}$ (3.2a) implies that it is more difficult to satisfy equality than inequality constraints, hence in problems with many equalities and/or large variable bounds, $\Phi_{\max}$ could become and remain large

during the sampling loop. Therefore, a stringent acceptance condition (*i.e.* low $\Phi_{\max}$) would lead to a filter with very few elements.

It is likely that many samples would be required before the variable tolerance is satisfied, hence a large number of samples N_{Sample} was also chosen. A large number is also set for $N_{\max\,\text{filter}}$ because it is intended to keep as many points in the final filter as possible. Additionally, it is worth noting that the step of checking (and possibly trimming) the filter size is entirely optional and can be omitted without affecting the overall behaviour of the algorithm. (may require more storage if left large)

The local optimization solver used in this study is the *FindMinimum* provided in *Mathematica*$^{\text{TM}}$ (Version 8.0). The solver could handle both constrained and unconstrained problems but occasionally requires good starting points for successful convergence. Thus, in the last two steps in Algorithm 3, the symbolic problem model with objective function, constraints and variable bounds is passed to *FindMinimum* along with the initial value for variables taken from the filter points. Multiple local minima are to be expected and the smallest value is taken as the final solution.

Furthermore, the initial vector x_0 is chosen to be the bounds' midpoint. This is by no means the only option, but seems to be the simplest one:

$$x_0 = \frac{x^L + x^U}{2} \tag{4.2}$$

Like other stochastic algorithms, the ICRS-Filter Method cannot guarantee the global optimum. In addition, the most feasible points in the filter may not always lead to the global solution. As a consequence, a balance needs to be kept between pushing for more feasible points (*i.e.* reductions of constraint norm) and accepting points from wider regions to the filter. The second objective may lead to more infeasible points in the final filter, but also increases the chance of starting the local solver into the attraction region of the global optimum. This justifies the reason for setting a high $\Phi_{\max}$ and large $N_{\max\,\text{filter}}$. Obviously, such objective can be assisted by having $k_3 > 1$. In some sense, this could be viewed as being equivalent to taking extra sampling loops at $k_3 = 1$.

Having completed the presentation of the implementation, the next section will il-

lustrate the performance of the method in solving a number of standard constrained optimization problems.

5 Numerical Results

The ICRS-Filter method has been tested using 104 cases studies taken from Hock and Schittkowski (1981a), Floudas et al. (1999), Rumarsson and Yao (2000) and Al et al. (2012). All case studies were modelled in *Mathematica*^{TM} (Version 8.0) and ran on a standard desktop PC with AMD Athlon^{TM} II X2 250 Processor at 3.00 GHz. The method was initially tested with simple models from Hock and Schittkowski (1981a). Case studies from Floudas et al. (1999) have been selectively chosen to represent diverse classes of problems in Optimization. They are Quadratic Programming Problems, Quadratically Constrained Problems, Bilinear Problems, Biconvex and Difference of Convex Functions Problems, Generalized Geometric Programming Problems, Parameter Estimation and Equations of State Problems. Additional problems are supplied from literature (Rumarsson and Yao (2000) and Al et al. (2012)).

A summary of the problems' characteristics is given in Tables 2, 3, 4, 5, and 6. Each case study has been run 10 times. The information regarding the CPU Time, Number of Samples (*i.e.* Sample Size), and Filter Points is given in Tables 7 and 8 ("Sub" means the suboptional result and "Fail" indicates that the local solver fails to converge). Summary of average values of the objective and constraint norm for the most and least feasible points are shown in Tables 9 and 10. Furthermore, for each problem, the best and the worst solution are identified for each run and the number of times in which they are found are highlighted in the corresponding columns in Tables 7 and 8. Vector solutions for cases that show improved solutions are summarized in Table 11. In addition, the original "maximization problems" 2.1.9, 5.2.2 Case 1, 5.2.2 Case 2, 5.2.2 Case 3, 5.2.4 and 5.2.5 have been reformulated as the "minimization problems" by reversing the signs of the corresponding objective functions. Thus, to keep the consistency, the signs of the best known solutions in literature were also reversed (Table 7).

Out of 104 case studies, global or best known solutions for 96 of them have been

confirmed. Our solutions match those reported in literature by at least 3 significant figures in all cases. This represents a 92% successs rate for our proposed method. In addition, 7 out of 96 cases are found to yield slightly better solutions. For the cases with better solutions, tests have been carried out to ensure that the objective values obtained from our models are consistent with the given vector solutions in literature.

In Tables 7 and 8, it is seen that 60 out of 104 cases require less than 5 seconds of CPU time. For other case studies CPU times are higher. It is possible that by implementing the algorithm in C++ or another language, the computational times can be significantly reduced as *Mathematica*$^{\text{TM}}$ is slow. The average amount of time in which the best solution was found is 5.7 seconds and 50 cases have obtained the best solutions for every run.

FindMinimum has failed to converge in 4 case studies, namely HS_101, HS_103, 5.3.3 and 8.6.3 (Tables 7 and 8). In the first three case studies, it is noted that the issue happens frequently when the solver is initialized with the left-most (*i.e.* most feasible) points in the filter. By constrast, *FindMinimum* converges very well given the right-most (*i.e.* least feasible) points. However, the situation is reversed in case 8.6.3 in which *FindMinimum* was able to converge without difficulties when it was initialized with the left-most points.

In addition, from Table 3, case 5.3.3 was the largest problem that was attempted. *FindMinimum* was found to fail consistently given any point in the filter in any run. Some details of the running of the problem are given in Table 12. In runs 3, 4, 5, and 6, the most feasible points in the filter already approximate the known solution (*i.e.* 3.324 Floudas et al. (1999)). The fact that *FindMinimum* fails to converge implies either *i)* the starting points are not good enough or *ii)* the local solver's algorithm has difficulties in locating the local minimum. Regarding the first implication, it is possible that even though the objective values are similar, the starting variable vectors could be still very far away from the neighbourhood of the desired local optimum.

In case 7.2.5, it was found that by putting the vector solutions published in the literature into the objective function, the objective value computed did not match the given value in the literature. Hence, it is suspected that there might be a typing error in the literature. This explaines why the solutions yielded by our tests are different from the

known solutions.

Tab. 2: Problem description for cases from Hock and Schittkowski (1981b)

Problem Name	Number of Variables	Number of Equalities	Number of Inequalities	Solution Type	Remark
HS_018	2	0	2	Global	Matched
HS_019	2	0	2	Global	Matched
HS_021	2	0	1	Global	Matched
HS_023	2	0	5	Global	Matched
HS_030	3	0	1	Global	Matched
HS_038	4	0	2	Global	Matched
HS_041	4	1	0	Global	Matched
HS_059	2	0	3	Global	Matched
HS_062	3	1	0	Global	Matched
HS_071	4	1	1	Global	Matched
HS_080	5	3	0	Global	Matched
HS_083	5	0	6	Global	Matched
HS_085	5	0	38	Global	Matched
HS_093	6	0	2	Global	Matched
HS_095	6	0	4	Global	Matched
HS_098	6	0	4	Global	Matched
HS_101	7	0	6	Global	Local solver did not converge
HS_103	7	0	6	Global	Local solver did not converge
HS_104	8	0	6	Global	Matched
HS_109	9	6	4	Global	Improved solution
HS_114	10	3	8	Global	Matched
HS_118	15	0	17	Global	Improved solution
HS_119	16	8	0	Global	Matched

Tab. 3: Problem description for cases from Floudas et al. (1999)

Problem Name	Number of Variables	Number of Equalities	Number of Inequalities	Solution Type	Remark
2.1.1	5	0	1	Global	Matched
2.1.2	6	0	2	Global	Matched
2.1.3	13	0	9	Global	Matched
2.1.4	6	0	5	Global	Matched
2.1.5	10	0	11	Global	Matched
2.1.6	10	0	5	Global	Matched
2.1.7_Case 1	20	0	10	Global	Matched
2.1.7_Case 2	20	0	10	Global	Matched (Bounds provided)
2.1.7_Case 3	20	0	10	Global	Matched (Bounds provided)
2.1.7_Case 4	20	0	10	Global	Matched (Bounds provided)
2.1.7_Case 5	20	0	10	Global	Matched (Bounds provided)
2.1.8	24	10	0	Global	Matched (Bounds provided)
2.1.9	10	1	0	Global	Matched
2.1.10	20	0	10	Global	Matched
3.1.1	8	0	6	Global	Matched
3.1.2	5	0	6	Global	Matched
3.1.3	6	0	6	Global	Matched
3.1.4	3	0	3	Global	Matched (Bounds provided)
5.2.2_Case 1	9	4	2	Global	Matched (Bounds provided)
5.2.2_Case 2	9	4	2	Global	Matched
5.2.2_Case 3	9	4	2	Global	Matched
5.2.4	7	1	5	Global	Matched
5.2.5	32	3	16	Global	Matched
5.3.2	22	16	0	Global	Matched
5.3.3	62	53	0	Best Known	Local solver did not converge
5.4.2	8	0	6	Best Known	Matched
5.4.3	16	13	0	Global	Matched
5.4.4	27	19	0	Global	Matched

Tab. 4: Problem description for cases from Floudas et al. (1999)

Problem Name	Number of Variables	Number of Equalities	Number of Inequalities	Solution Type	Remark
6.3.1	8	6	0	Global	Matched
6.3.2	4	3	0	Global	Matched
6.3.3	12	9	0	Global	Matched
6.3.4	6	4	0	Global	Matched
6.4.2	9	3	0	Global	Matched
6.4.3	3	1	0	Global	Matched
6.4.4	9	3	0	Global	Matched
6.4.5	3	1	0	Global	Matched
6.4.7	4	2	0	Global	Matched
6.4.8	9	3	0	Global	Matched
6.4.9	3	1	0	Global	Matched
6.4.11	4	2	0	Global	Matched
6.4.12	6	3	0	Global	Inferior point
6.4.14	4	2	0	Global	Improved soluton
7.2.1	7	0	14	Global	Matched
7.2.2	6	4	1	Global	Matched
7.2.3	8	0	6	Global	Matched
7.2.4	8	0	4	Global	Improved solution
7.2.5	5	0	5	Global	Error in reported solution
7.2.6	3	0	1	Global	Matched
7.2.7	4	0	2	Global	Matched
7.2.8	8	0	4	Global	Matched
7.2.9	10	0	7	Best Known	Matched
7.2.10	11	0	9	Global	Matched
8.5.1	24	10	0	Global	Matched
8.5.2	24	10	0	Global	Matched
8.5.3	52	25	0	Global	Matched
8.5.4	17	12	0	Global	Matched
8.5.5	15	11	0	Global	Matched
8.5.6	14	8	0	Global	Inferior point
8.5.7	52	20	0	Global	Improved solution
8.5.8	22	10	0	Global	Matched
8.6.1	4	2	0	Global	Inferior point
8.6.2	4	2	0	Global	Inferior point
8.6.3	3	2	0	Global	Local solver did not converge
8.6.4	3	2	0	Global	Inferior point
8.6.5	3	2	0	Global	Inferior point
8.6.6	4	2	0	Global	Matched

Tab. 5: Problem description for cases from Rumarsson and Yao (2000)

Problem Name	Number of Variables	Number of Equalities	Number of Inequalities	Solution Type	Remark
A_g01	13	0	9	Global	Matched
B_g02	20	0	2	Best Known	Inferior point
C_g03	10	1	0	Best Known	Matched
D_g04	5	0	6	Best Known	Matched
E_g05	4	3	2	Best Known	Matched
F_g06	2	0	2	Best Known	Matched
G_g07	10	0	8	Best Known	Matched
H_g08	2	0	2	Best Known	Improved solution
I_g09	7	0	4	Best Known	Matched
J_g10	8	0	6	Best Known	Improved solution
K_g11	2	1	0	Best Known	Matched
M_g13	5	3	0	Best Known	Matched

Tab. 6: Problem description for cases from Al et al. (2012)

Problem Name	Number of Variables	Number of Equalities	Number of Inequalities	Solution Type	Remark
P1	6	0	6	Best Known	Matched
P2	10	3	0	Best Known	Improved solution
P3	2	0	2	Best Known	Matched

Tab. 7: Solution summary

Problem Name	Known Solution	Best Solution	No. of Times Found	Worst Solution	No. of Times Found	Mean Samples Size	Mean Filter Points	Mean CPU Time
HS_018	5	5	10	5	10	1852	91	0.450
HS_019	-6961.81	-6961.81	10	-6961.81	10	607	7	0.120
HS_021	-99.96	-99.96	10	-99.96	10	461	1	0.094
HS_023	2	2	10	2	10	851	18	0.180
HS_030	1	1	10	1	10	1268	10	0.310
HS_038	0	0	10	0	10	1108	1	0.320
HS_041	1.92593	1.92593	10	1.92593	10	1887	25	0.570
HS_059	-7.80423	-7.80279	5	-6.74951	10	2465	161	0.860
HS_062	-26272.5	-26272.5	10	-26272.5	10	1217	20	0.300
HS_071	17.014	17.014	10	32.944	1	2076	19	0.610
HS_080	0.05395	0.05395	8	1.00000	2	3171	33	1.070
HS_083	30665.5	-30665.5	10	-30665.5	10	10848	100	3.970
HS_085	-1.90513	-1.90516	10	-1.90516	10	4338	11	1.520
HS_093	135.076	135.076	10	242.710	1	4048	15	1.540
HS_095	0.0156195	0.0156195	10	0.0156195	10	2426	1	0.920
HS_098	3.1358	3.1358	10	4.0712	8	8240	29	3.270
HS_101	1809.76476	4.34370/Fail	1	1809.7648	10	11007	110	5.110
HS_103	543.668	16.324/Fail	1	543.668	10	9590	84	4.410
HS_104	3.951163	3.951163	8	4.218/Sub	4	17042	153	8.800
HS_109	5362.069	711.454/Fail	1	5326.8500	8	14292	100	7.700
HS_114	-1768.81	-1768.81	10	-1768.81	10	11368	82	6.830
HS_118	664.82045	662.52000	10	662.52000	10	31000	36	28.01
HS_119	244.899698	244.90000	10	244.90000	10	16311	31	14.04
2.1.1	-17	-17	1	-2.5	1	2095	2	0.680
2.1.2	-213	-213	10	-213	10	32397	435	15.36
2.1.3	-195	-195	10	-195	10	5526	1	3.690
2.1.4	-11	-11	10	-11	10	26861	346	15.07
2.1.5	-268.015	-268.015	10	-268.015	10	14870	18	8.280
2.1.6	-39.000	-39.000	3	-18.222	1	6093	2	3.310
2.1.7_Case 1	-394.7506	-394.751	10	-135.970	1	39992	87	39.19
2.1.7_Case 2	-884.75058	-884.751	10	-631.617	1	68468	152	68.98
2.1.7_Case 3	-8695.01193	-8695.01	10	-3683.77	1	48621	97	49.80
2.1.7_Case 4	-754.75062	-754.75100	10	-514.10200	1	26616	51	26.58
2.1.7_Case 5	-4150.4101	-4150.4100	10	-904.69500	1	10.472	70	40.20
2.1.8	15639	15639	10	27168	2	29980	56	34.08
2.1.9	-0.375	-0.375	10	-0.333	10	8394	85	4.770
2.1.10	49318	49318	10	133719	2	43835	30	42.55
3.1.1	7049.24	7049.25	10	7049.25	10	10588	10	5.040
3.1.2	-30665.53	-30665.50	10	-30665.50	10	11217	97	4.070
3.1.3	-310	-310	8	-168	1	15807	278	7.050
3.1.4	-4	-4	10	-3.28179	1	1726	17	0.420
5.2.2_Case 1	-400	-400	10	-100	6	7816	22	3.950
5.2.2_Case 2	-600	-600	5	1901.37	1	6635	24	3.320
5.2.2_Case 3	-750	-750	10	782.87	1	6836	22	3.460
5.2.4	-450	-450	8	420.395	1	3769	26	1.570
5.2.5	-3500	-3500	10	2415.34	1	41279	80	61.55
5.3.2	1.86416	1.86416	10	2.21220	1	29566	50	31.83
5.3.3	3.234	1.737/Fail	1	9.621/Fail	1	146809	41	289.8
5.4.2	7512.23	7512.23	10	7512.23	10	12487	9	5.970
5.4.3	4845.00	4845.46	8	5937.44	9	14783	40	12.17
5.4.4	10077.8	10077.8	3	22168.5	1	25454	39	32.64
6.3.1	-0.0202	-0.0202	2	-0.0175	10	6827	28	3.150

Tab. 8: Solution summary

Problem Name	Known Solution	Best Solution	No. of Times Found	Worst Solution	No. of Times Found	Mean Samples Size	Mean Filter Points	Mean CPU Time
6.3.2	-0.03247	-0.03246	3	3.0000E-6	1	2900	33	0.850
6.3.3	-0.3574	-0.3524	10	-0.3242	1	10951	19	7.020
6.3.4	-0.29454	-0.294541	10	8.0Eˆ-7	1	4369	22	1.640
6.4.2	-70.75208	-70.75210	3	-70.5581	10	7183	49	3.670
6.4.3	0	7.0Eˆ-7	10	7.0Eˆ-7	1	1183	10	0.280
6.4.4	-0.16085	-0.16085	10	-0.13839	3	6753	38	3.480
6.4.5	-0.027	-0.027	10	6.0Eˆ-7	1	1281	12	0.310
6.4.7	-0.03407	-0.03406	10	0.15652	10	3749	75	1.150
6.4.8	-3.02954	-3.05198	10	-2.66062	10	9842	133	4.040
6.4.9	0	-2.7Eˆ-6	1	1.6Eˆ-6	10	1127	13	0.280
6.4.11	0.28919	0.28919	10	0.39359	10	7379	159	2.380
6.4.12	-0.25457	-0.21621	4	-0.21620	6	5137	51	1.950
6.4.14	-0.07439	-0.69536	8	0.09867	10	3448	76	1.120
7.2.1	1227.23	1227.23	10	1227.23	10	9965	31	4.330
7.2.2	-0.388000	-0.388811	10	-0.388811	10	4587	38	1.720
7.2.3	7049.25	7049.25	10	7049.25	10	12086	11	5.580
7.2.4	3.95110	3.91801	10	4.20285	6	19872	120	9.210
7.2.5	1.1436	10122.5000	10	10122.5	10	9696	98	3.900
7.2.6	-83.2540	-83.2497	10	-83.2497	10	4261	103	1.140
7.2.7	-5.73980	-5.74376	10	-5.74376	10	6545	101	2.080
7.2.8	-6.04820	-6.04823	10	-5.72294	1	20289	98	10.16
7.2.9	1.14360	1.14362	10	1.14362	10	18054	29	10.93
7.2.10	0.140600	0.140607	10	3.78137	1	16628	45	10.46
8.5.1	0.618570	0.618573	10	0.618573	10	32462	43	38.37
8.5.2	0.485150	0.485152	10	0.485152	10	18855	24	22.37
8.5.3	0.00464972	0.00464971	6	0.005777	1	76456	43	181.7
8.5.4	0.21246	0.21246	10	0.21246	10	15833	38	13.76
8.5.5	0.0003075	0.0003075	2	0.001225	1	18204	34	14.26
8.5.6	0.0011400	0.0016085	1	28434.3	1	15473	49	11.37
8.5.7	29.0473	23.6129	10	23.6129	10	72406	56	172.35
8.5.8	3.32000	3.32185	10	3.32185	10	22804	35	24.74
8.6.1	-0.00988	-5.8Eˆ-7	1	0.0063610	1	7119	23	2.200
8.6.2	0	-9.8Eˆ-9	1	1.9Eˆ-7	1	12282	29	3.860
8.6.3	-0.00400	0.15284	9	1.966/Fail	1	1694	29	0.440
8.6.4	-0.000330	0.134798	10	0.38771	1	1615	28	0.410
8.6.5	-0.00700000	-0.00116737	1	9.9Eˆ-7	1	2615	47	0.690
8.6.6	-0.00120000	-0.00116737	8	1.0Eˆ-7	8	2163	22	0.610
A_g01	-15	-15	10	-12.6562	0	44280	367	35.86
B_g02	-0.803619	-0.740749	0	-0.363375	10	44921	71	31.43
C_g03	-1	-1	10	-1	0	5313	29	2.840
D_g04	-30665.539	-30665.500	10	-30665.5	0	11429	101	3.920
E_g05	5126.4981	5126.5000	10	5126.5	0	3827	60	0.980
F_g06	-6961.81388	-6961.81000	10	-6961.81	0	747	8	0.120
G_g07	24.3062091	24.3062000	10	24.3062	0	17804	34	10.48
H_g08	0.095825	-0.105460	10	1.2Eˆ-8	10	613	9	0.120
I_g09	680.6300573	680.6300000	10	680.6300000	0	22881	247	11.79
J_g10	7049.3307	7049.2480	10	7049.2480	0	10639	10	5.550
K_g11	0.75	0.75	10	1	10	1852	116	0.440
M_g13	0.0539498	0.0539498	10	0.0539498	0	3358	32	1.230
P1	-310	-310	4	-168	6	9640	69	3.780
P2	-47.764888	-47.761100	10	-47.761100	10	25627	299	16.56
P3	-5.508	-5.508	10	-5.508	10	1247	35	0.270

Tab. 9: ICRS-Filter information

Problem Name	Most Feasible Point		Least Feasible Point	
	Mean Objective	Mean Constraint Norm	Mean Objective	Mean Constraint Norm
HS_018	5.04	0.00	0.40	35.26
HS_019	-4939	0.340	-7766	20.89
HS_021	-99.95	0.000	-99.95	0.000
HS_023	3.42	0.00	0.00	11.00
HS_030	20.9300	0.0008	2.2300	1.0300
HS_038	0.15	0.00	0.15	0.000
HS_041	1.9900	0.0002	1.7400	1.9800
HS_059	-7.27	0.00	-15.65	374.20
HS_062	-22540	0.0006	-27901	0.5100
HS_071	70.50	0.0002	21.64	21.2000
HS_080	0.80	0.03	0.00	41.54
HS_083	-30659	0	-31851	2.060
HS_085	-1.81	0.00	-2.03	101.00
HS_093	1605.00	0.030	981.88	2.180
HS_095	0.04	0.00	0.04	0.004
HS_098	3.65	0.00	2.56	12.96
HS_101	2538.40	0.34	11.34	166.10
HS_103	1238.29	0.00	10369.00	158.30
HS_104	4.16	0.007	-4.44	10.56
HS_109	5649	5603.65	178	110769
HS_114	-590.71	10.83	-19233.00	7310
HS_118	734.71	0.00	659.88	78.16
HS_119	2724.820	0.3	1377.5	8.780
2.1.1	-12.01	0.00	-12.12	0.070
2.1.2	-208.36	0.00	-862.3	70.85
2.1.3	-187.63	0.00	-187.63	0.000
2.1.4	-8.54	0.00	-180.64	814.62
2.1.5	-257.62	0.00	-269.30	8.970
2.1.6	-26.34	0.00	-26.71	0.160
2.1.7_Case 1	-187.74	0.00	-31457.00	1156
2.1.7_Case 2	-684.70	0.00	-38367.00	1192
2.1.7_Case 3	-5586	0.00	-691699	1136
2.1.7_Case 4	-616.80	0.00	-25409.00	1224
2.1.7_Case 5	-2884	0.00	-357597	1118
2.1.8	42772	0.00	33328	50.36
2.1.9	0	0.00	-6	4.000
2.1.10	168840	0.00	-800637	26562
3.1.1	9173	0.00	7216	226944
3.1.2	-30657	0.00	-31868	2.000
3.1.3	-253.58	0.00	-129101	115.26
3.1.4	-3.56	0.00	-4.36	2.350
5.2.2_Case 1	157.96	59.14	1806.41	28181.00
5.2.2_Case 2	646.07	31.27	-3978.5	33985.00
5.2.2_Case 3	-4.12	141.45	-4140.80	72824.00
5.2.4	112.83	0.00	-1699.3	31.60
5.2.5	-2030.74	0.02	-8114.8	322.70
5.3.2	4.44	7.86	1.79	950.40
5.3.3	3.283585	218.0682	2.108546	5659
5.4.2	9506.44	0.00	7816.00	241203.00
5.4.3	5692.9	76.4	1774.8	2286.0
5.4.4	4875.6	259.39	2410.98	7608.00

Tab. 10: ICRS-Filter information

Problem Name	Most Feasible Point		Least Feasible Point	
	Mean Objective	Mean Constraint Norm	Mean Objective	Mean Constraint Norm
6.3.1	0.01	54846.000	-0.20	0.870
6.3.2	-0.03	0.340	-0.44	1.240
6.3.3	0.25	0.830	0.12	0.990
6.3.4	0.04	0.540	-0.42	1.040
6.4.2	-61.93	0.01	-115.04	66.44
6.4.3	0.27	0.000	0.03	0.310
6.4.4	-0.09	0.000	-0.21	0.530
6.4.5	0.04	0.000	-0.02	0.190
6.4.7	0.16	0.000	-0.01	0.480
6.4.8	-2.66	0.000	-3.97	0.480
6.4.9	0.34	0.000	0.04	0.320
6.4.11	0.39	0.000	0.01	0.510
6.4.12	0.22	0.000	-0.48	0.420
6.4.14	0.099	0.000	-0.67	0.420
7.2.1	1286.29	0.000	598.84	0.410
7.2.2	-0.30	0.000	-0.94	1.530
7.2.3	7873.88	0.000	6632.34	5.350
7.2.4	4.13	0.000	-3.37	5.730
7.2.5	10133	0.000	8958.2	0.500
7.2.6	-83.21	0.000	-98.69	0.910
7.2.7	-5.73	0.000	-9.07	2.330
7.2.8	-5.82	0.000	-14.10	6.960
7.2.9	3.42	0.00	2.31	49.04
7.2.10	4.03	0.04	0.11	978.7
8.5.1	1.49	0.13	0	15.60
8.5.2	1.08	2.71	0	15.60
8.5.3	2.17	0.03	0	86.97
8.5.4	0.55	0.016	0	1.880
8.5.5	0.0008	0.004	0	1.030
8.5.6	153466	0.04	1.28	7.020
8.5.7	104.77	0.003	0	0.960
8.5.8	41.32	0.009	0	0.720
8.6.1	0.77	0.0001	-0.52	0.2300
8.6.2	0.35	0.0006	-1.02	0.3200
8.6.3	1.59	0.008	-0.13	1.040
8.6.4	0.69	0.020	-0.16	1.080
8.6.5	0.019	0.0002	-0.5	0.3750
8.6.6	0.24	0.0005	-0.24	0.6000
A_g01	-5.38	0.00	-170.07	644.10
B_g02	-0.50	0.000	-0.51	0.450
C_g03	-0.02	0.000	-203.00	2.220
D_g04	-30659	0.000	-25578	2.170
E_g05	5354.93	1.350	1607.87	1120.800
F_g06	-4961.55	0.00	-7806.89	12.64
G_g07	26.15	0.00	7.85	63.08
H_g08	-0.105	0.000	-87.150	4.390
I_g09	680.98	0.000	299.860	7974.400
J_g10	7788.80	0.00	7140.52	1470812.00
K_g11	1	0.000	0.019	0.900
M_g13	0.84	0.02	0.00	42.73
P1	-263.710	0.000	-421.110	2.610
P2	-42.48	0.006	-1156.46	94.720
P3	-5.49	0.000	-6.83	3.740

Problem number HS_109 from Hock and Schittkowski (1981b)

Reported solution in literature	Solution vector obtained from the study
$(\ 674.8881 \quad 1134.170 \quad 0.1335691 \quad -0.3711526$ $252 \quad 252 \quad 201.465 \quad 426.661 \quad 368.494 \ )^T$	$(\ 669.10976 \quad 1131.66315 \quad 0.132959 \quad -0.36042$ $252 \quad 252 \quad 209.21247 \quad 386.44246 \quad 327.99049 \ )^T$

Problem number HS_118 from Hock and Schittkowski (1981b)

Reported solution in literature	Solution vector obtained from the study
$\left(\ 8 \quad 49 \quad 3 \quad 1 \quad 56 \quad 0 \quad 1 \quad 63 \quad 6 \quad 3 \quad 70 \quad 12 \quad 5 \quad 77 \quad 18 \ \right)^T$	$\left(\ 8 \quad 49 \quad 3 \quad 0 \quad 56 \quad 0 \quad 1 \quad 63 \quad 6 \quad 3 \quad 70 \quad 12 \quad 5 \quad 77 \quad 18 \ \right)^T$

Problem number 7.2.4 from Floudas et al. (1999)

Reported solution in literature	Solution vector obtained from the study
$\left(\ 6.4747 \quad 2.234 \quad 0.6671 \quad 0.5957 \quad 5.931 \quad 5.5271 \quad 1.0108 \quad 0.4004 \ \right)^T$	$\left(\ 6.34578 \quad 2.34101 \quad 0.670869 \quad 0.534746 \quad 5.95279 \quad 5.3164 \quad 1.04399 \quad 0.420086 \ \right)^T$

Problem number 8.5.7 from Floudas et al. (1999)

Reported solution in literature	Solution obtained from the study
$\theta = \begin{pmatrix} 0.0168 \\ 12.4332 \end{pmatrix} \ z = \begin{pmatrix} 0.9985 & 0.8826 & 0.1159 & 547.84 & 663.78 \\ 0.9878 & 0.8437 & 0.1441 & 531.49 & 675.63 \\ 1.0011 & 0.8282 & 0.1729 & 512.06 & 684.95 \\ 0.9920 & 0.7874 & 0.2046 & 490.84 & 695.44 \\ 1.0058 & 0.7660 & 0.2398 & 464.80 & 704.62 \\ 1.0005 & 0.7242 & 0.2763 & 438.64 & 714.62 \\ 0.9995 & 0.6822 & 0.3173 & 407.66 & 724.94 \\ 0.9997 & 0.6377 & 0.3600 & 375.35 & 735.32 \\ 1.0013 & 0.5962 & 0.4052 & 340.39 & 745.55 \\ 1.0071 & 0.5587 & 0.4484 & 306.56 & 754.96 \end{pmatrix}$	$\theta = \begin{pmatrix} 0.0970966 \\ 9.51457 \end{pmatrix} \ z = \begin{pmatrix} 0.996757 & 0.880943 & 0.115815 & 547.568 & 663.382 \\ 0.989826 & 0.845474 & 0.144352 & 531.729 & 676.081 \\ 1.00115 & 0.828251 & 0.172897 & 512.061 & 684.959 \\ 0.991248 & 0.786752 & 0.204495 & 490.782 & 695.278 \\ 1.0021 & 0.762805 & 0.23929 & 464.535 & 703.825 \\ 1.00205 & 0.725549 & 0.276503 & 438.434 & 714.936 \\ 1.00587 & 0.687725 & 0.31815 & 407.99 & 726.14 \\ 0.999182 & 0.639018 & 0.360163 & 375.418 & 735.582 \\ 1.00144 & 0.596263 & 0.405174 & 340.393 & 745.567 \\ 1.00153 & 0.553772 & 0.447756 & 306.367 & 754.12 \end{pmatrix}$

Problem number H_g08 from Rumarsson and Yao (2000)

Reported solution in literature	Solution obtained from the study
$\left(\ 1.2279713 \quad 4.2453733 \ \right)^T$	$\left(\ 1.227816471 \quad 3.74490789 \ \right)^T$

Problem number J_g10 from Rumarsson and Yao (2000)

Reported solution in literature	Solution obtained from the study
$(\ 579.3167 \quad 1359.943 \quad 5110.071 \quad 182.0174$ $295.5985 \quad 217.9799 \quad 286.4162 \quad 395.5979 \ )^T$	$(\ 579.3067 \quad 1359.9707 \quad 5109.9707 \quad 182.0177$ $295.6012 \quad 217.9823 \quad 286.4165 \quad 395.6012 \ )^T$

Problem number P2 from Al et al. (2012)

Reported solution in literature	Solution obtained from the study
$(0.04066 \quad 0.1477212 \quad 0.7832057 \quad 0.001414339 \quad 0.48529363$ $0.000693138 \quad 0.074052 \quad 0.017950966 \quad 0.0373268186 \quad 0.09688446 \ )^T$	$(\ 0.04067 \quad 0.1477304 \quad 0.7831533 \quad 0.001414220 \quad 0.48524665$ $0.0006931726 \quad 0.02739932 \quad 0.01794728 \quad 0.03731437 \quad 0.09688446 \ )^T$

Tab. 11: Improved solutions

Tab. 12: Problem 5.3.3 ICRS-Filter information for multiple runs

Run	Filter Most Feasible Point		Filter Least Feasible Point	
	Objective Value	Constraint Norm	Objective Value	Constraint Norm
1	1.959	206.080	1.827	1010.07
2	2.545	207.717	2.031	6845.00
3	3.899	262.990	1.968	8296.00
4	3.199	185.753	2.000	3369.80
5	3.449	220.061	1.889	6652.00
6	3.729	231.500	2.622	7707.00
7	2.620	257.828	2.486	5118.00
8	3.790	150.625	2.17	9643.00
9	2.514	233.207	1.878	2403.60
10	5.130	224.921	2.214	5545.20

6 Conclusions and future work

This paper presents a new approach towards obtaining improved solutions for generally constrained nonconvex optimization problems. By combining the ICRS Algorithm of Casares and Banga (1987) with the concept of a dynamic filter, the new method ICRS-Filter is produced.

Our computational results suggest that the method works well with "small" problems (*i.e.* up to 52 variables and 25 constraints). Failures of the method are attributed to the local optimization solver which was unable to converge. However, it should be noted that the ICRS-Filter method gives the most feasible points whose objective values are very close to the best known solution.

The future work following from the very encouraging results obtained in this study would be to use an alternative local optimization solver and scaling up the algorithm by implementing it in C++. Large-scale applications with many more variables and constraints will be examined to evaluate the performance of the algorithm.

References

Al, M., Golalikham, M., Zhuang, J., 2012. A computational study on different penalty approaches for solving constrained global optimization problems with the

electromagnetism-like method. Optimization, 1–17.

Banga, J. R., Seider, W. D., 1996. Global optimization of chemical processes using stochastic algorithms. In: Floudas, C. A., Pardalos, P. M. (Eds.), State of Art in Global Optimization. Kluwer Academic Publishers, pp. 563–583.

Box, G., Muller, M., 1958. A note on the generation of random normal deviates. The Annals of Mathematical Statistics 29, 610–611.

Casares, J., Banga, J., 1987. Icrs: application to a wastewater treatment plant model. Process Optimisation, 183–192.

Correia, A., Matias, J., Mestre, P., Serodio, C., 2010. Derivative-free nonlinear optimization filter simplex. International Journal of Applied Mathematics and Computer Science 20, 679–688.

Edgar, T., Himmelblau, D., Lasdon, L., 2001. Optimization of Chemical Process. McGraw-Hill Higher Education.

Fletcher, R., Leyffer, S., 2000. Nonlinear programming without a penalty function. Mathematical Programming 91, 239–269.

Fletcher, R., Leyffer, S., Toint, P., 2006. A Brief History of Filter Methods. Tech. rep., Argonne National Laboratory, Mathematics and Computer Science Division.

Floudas, C., 1999. Deterministic Global Optimization: Theory, Methods and Applications. Springer.

Floudas, C., Misener, R., 2009. Advances for the pooling problems: Modelling, global optimization, and computational studies. Appl. Comput. Math. 1, 3–22.

Floudas, C., Pardalos, P., Adjiman, C., Esposito, W., Gumus, Z., Harding, S., Klepeis, J., C.A., M., C.A., S., 1999. Handbook of Test Problems in Local and Global Optimization. Kluwer Academic Publishers.

Fouskakis, D., Draper, D., 2002. Stochastic optimization: a review. International Statistical Review 70, 315–349.

Hickernell, F. J., Yuan, Y.-X., 1997. A simple multistart algorithm for global optimization. OR Transactions 1, 1–12.

Hock, W., Schittkowski, K., 1981a. Test examples for nonlinear programming codes. Vol. 187 of Lecture Notes in Economics and Mathematical Systems. Springer Verlag.

Hock, W., Schittkowski, K., 1981b. Test Examples for Nonlinear Programming Codes. Vol. 187. Springer.

Karas, E., Ribeiro, A., Sagastizabal, C., Solodov, M., 2006. A bundle-filter method for nonsmooth convex constrained optimization. Mathematical Programming 1, 297–320.

Nelder, J., Mead, R., 1965. A simplex method for function minimization. The Computer Journal 7, 308–313.

Park, S. K., Miller, K. W., 1988. Random number generators: good ones are hard to find. Commun. ACM 31 (10), 1192–1201.

Powell, M. J. D., 1969. A method for nonlinear constraints in minimization problems. In: Fletcher, R. (Ed.), Optimization. Academic Press, New York, pp. 283–298.

Rumarsson, T., Yao, X., 2000. Stochastic ranking for constrained evolutionary optimization. IEEE 4, 284–294.

Torn, A., 1978. A search clustering approach to global optimization. North-Holland, Amsterdam.