Derivative-free optimization and filter methods to solve nonlinear constrained problems

Aldina Correia, João Matias, Pedro Mestre and Carlos Serôdio

In real optimization problems, usually the analytical expression of the objective function is not known, nor its derivatives, or they are complex. In these cases it becomes essential to use optimization methods where the calculation of the derivatives, or the verification of their existence, is not necessary: the Direct Search Methods or Derivative-free Methods are one solution.

When the problem has constraints, penalty functions are often used. Unfortunately the choice of the penalty parameters is, frequently, very difficult, because most strategies for choosing it are heuristics strategies. As an alternative to penalty function appeared the filter methods. A filter algorithm introduces a function that aggregates the constrained violations and constructs a biobjective problem. In this problem the step is accepted if it either reduces the objective function or the constrained violation. This implies that the filter methods are less parameter dependent than a penalty function.

In this work, we present a new direct search method, based on simplex methods, for general constrained optimization that combines the features of the simplex method and filter methods. This method does not compute or approximate any derivatives, penalty constants or Lagrange multipliers. The basic idea of simplex filter algorithm is to construct an initial simplex and use the simplex to drive the search. We illustrate the behavior of our algorithm through some examples. The proposed methods were implemented in Java.

Keywords: nonlinear constrained optimization; filter methods; direct search methods

2000 AMS Subject Classifications: 80M50; 49M37; 90C30

1. Introduction

We consider the general problem of nonlinear constrained optimization (NLP) of the form

$$\begin{array}{ll} (x \in \mathbb{R}^n) \text{ minimize} & f(x) \\ \text{subject to} & C(x) \leq 0, \end{array} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is the objective function and $c : \mathbb{R}^n \to (\mathbb{R} \cup \{\infty\})^m$ are the constraint functions with $C = (c_1, \ldots, c_m)^T$. The feasible region is denoted by X.

Traditionally, these kinds of problem are solved using penalty or merit functions that are a linear combination of the objective function and a measure of the constrained violation.

In recent years, there has been a resurgence of interest in penalty methods, mainly for exact penalty methods, Byrd *et al*. [4], Chen *et al*. [5], Fletcher *et al*. [6], Gould *et al*. [12], Leyffer *et al*. [17], Mongeau *et al*. [18], and Zaslavski [23], because of their ability to handle degenerate problems and inconsistent constrained linearizations.

The penalty methods are designed to solve this problem by solving a sequence of constructed unconstrained problems. They are then seen as a vehicle to solve constrained optimization problems using unconstrained optimization techniques. Unfortunately the choice of suitable penalty parameters is, frequently, very difficult, because most strategies for choosing them are heuristics strategies.

As an alternative to penalty functions, the filter methods appeared, which were introduced by Fletcher and Leyffer [7]. Since then, the filter technique has been mostly applied to SLP (sequential linear programming) and SQP (sequential quadratic programming) type methods. A filter algorithm introduces a function that aggregates constrained violations and constructs a biobjective problem. In this problem the step is accepted if it reduces either the objective function or the constrained violation. This implies that the filter methods are less parameter dependent than a penalty function.

The SQP-filter approach was also applied to interior point algorithms by Ulbrich *et al*. [22]. Audet and Dennis [1] presented a pattern search filter method for derivative-free nonlinear programming. Gould *et al*. [11] introduced a multidimensional filter algorithm for solving nonlinear feasibility problems. Gould *et al*. [13] extended the multidimensional filter techniques to general unconstrained optimization problems. Filter methods were also used in the context of nonsmooth optimization by Fletcher and Leyffer [6] and by Karas *et al*. [15].

A review of the filter methods is presented by Fletcher *et al.* [10]. Global convergence for filter methods in SLP problems was obtained by Fletcher, Leyffer and Toint [8] and a proof of its convergence for SQP was given by Fletcher *et al*. [9]. In both cases, this convergence is only to a point that satisfies the Fritz John optimal conditions. Thus, previous filter algorithms require explicit use of the derivatives of both the objective and the objective constrains. Audet and Dennis [1] present a pattern search filter method for derivative-free nonlinear programming.

Similar to all the derivative free algorithms, the pattern search methods are suitable when some of the functions that define the problem are given as black boxes, not assuring enough precision to approximate derivatives.

Black box problems occur frequently in science and engineering, where the evaluation of the objective function usually requires complex deterministic simulations. Those simulations are required to properly describe the underlying physical phenomena. Also the computational noise, associated with the simulations, indicates that obtaining derivatives is difficult and unreliable.

Direct search methods are nonlinear optimization methods that neither require explicitly or approximate derivatives to solve the problem. Instead, at each iteration a set of trial points is generated and their function values are compared with the best previously obtained solution. This information is then used to determine the next set of trial points.

Results presented by Audet and Dennis in [1] are not very conclusive for general constraints. However they do not compute or approximate any derivatives. Instead, they present and analyse a pattern search method, for general constrained optimization, based on filter methods for step acceptance. The filter mentioned in [1] differs in three important aspects from the filters described earlier. The authors extend the usual pattern-search convergence results to filter methods.

Our objective is to apply the method, of Audet and Dennis, to the other direct search methods.

The article is divided as follows. In Section 2 we present a brief description of direct search methods. In Section 3 we briefly present the filter method of Audet and Dennis in [1] and their notation. Then, in Section 4 we present a new simplex method for general constrained optimization.

This method combines the features of the simplex method and filter methods. This method does not compute or approximate any derivatives, penalty constants or Lagrange multipliers. In Section 5, we illustrate the behaviour of our algorithm through some examples.

2. Direct search methods

In the last decade derivative-free methods or direct search methods have attracted more attention from the optimization community. Those methods are especially effective when Newton-like methods are inappropriate or inapplicable. They are effective for minimization of functions with one or more of the following properties:

- Calculation of objective function, f, is very expensive or time consuming.
- Exact first partial derivatives of f cannot be calculated.
- Numerical approximation of the gradients of f is impractically expensive or slow.
- The values of f are "noisy".

We can organize, in accordance with Lewis *et al*. [16], the most popular direct search methods for unconstrained minimization into tree basic categories:

- Pattern search methods.
- Simplex methods.
- Methods with adaptive sets of search directions.

Pattern search methods include methods such as Hooke and Jeeves method 14, Nelder and Mead method [19] (which is a good example of a Simplex method) and the Powell method [21]. This last one is a method with adaptive sets of search directions.

In this study we just refer the Pattern search methods and the Simplex methods, because our objective is to apply the method of Audet and Denis, which uses the Pattern Search, to the simplex method.

2.1 *Pattern search methods*

Pattern search methods are characterized by a series of exploratory moves that consider the performance of the objective function. This performance is evaluated at a pattern of points, all lying on a rational lattice. The exploratory moves consist on a systematic strategy of visiting the points in the lattice, in the instant neighborhood of the current iterate. One of the first direct search methods was the Hooke and Jeeves Method, introduced in 1961 by Hooke and Jeeves [14].

Algorithm consists in choosing an initial point, a step-length, for the respective variables, after f is evaluated in the initial point and the method proceeds by a sequence of *exploratory* and *p*attern moves.

If an exploratory move leads to a decrease in the value of f it is called a *success*; otherwise it is called *failure*. A pattern move is not tested for success or failure.

The aim of an exploratory move is to acquire information about the function f in the neighborhood of the current base point and to find a descent direction. A Pattern move attempts to speed up the search by using information already acquired about f . It is invariably followed by a sequence of exploratory moves for finding an improved direction of search in which to make another pattern move.

2.2 *Simplex methods*

Simplex methods are characterized by the simple reason that they use to guide the search. The basic idea of simplex search is to construct a nondegenerate simplex in R^n and use the simplex to drive the search.

A simplex is a set of $n + 1$ points in the R^n. Thus in R^2 , a simplex is a triangle, and in R^3 is a tetrahedron, etc.

Simplex method is introduced by Spendley in 1962. In this simplex a single move is specified, the reflection. It consists in replacing a vertex by reflecting it through the centroid of the opposite face, resulting also in a simplex. This move identifies the " worst" vertex in the simplex (which are the least desirable objective value) and then reflects the vertex. If the reflected vertex is still the worst vertex, then next chose the "second worst" vertex and repeat the process.

The contribution of Nelder and Mead [19] was to turn simplex search into an optimization algorithm with additional moves designed to accelerate the search. Four basic operations added are Reflect; Expand; Contract and Shrink.

Expansion and contraction were the first moves to be added. The expansion step allows a more aggressive move by doubling the length of the step from the centroid to the reflection point, whereas the contraction steps allows more conservative moves by halving the length of the step from the centroid to either the reflection point or the worst vertex.

But these steps were not sufficient, because there was nothing to do when no step produces a significant improvement in the objective function value. Later, Nelder and Mead also resolved this issue by adding the shrink step. That step consists in reduce the lengths of the edges adjacent to the current best vertex by half.

The Nelder–Mead simplex algorithm is, of all the direct search methods, most often found in numerical software packages because it is effective and computationally compact.

3. Filter method of Audet and Dennis

3.1 *Notation and definitions*

Filter methods treat the optimization problem as a biobjective attempt to minimize the objective function and a continuous function h, that aggregate constraint violation function. The priority must be given to h, at least until a feasible iterate is found [1].

h must satisfy:

$$h(x) \geq 0 \quad \text{with } h(x) = 0 \quad \text{if and only if} \quad x \text{ is feasible.}$$

The function h is often set to $h(x) = \|C(x)_+\|$, where $\|.\|$ is a vector norm and $C(x)_+$ is the vector of constrained violations at x, i.e, for $i = 1, 2, \ldots, m$,

$$C(x)_+ = \begin{cases} C_i(x) & \text{if } C_i(x) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

In [1] Audet and Dennis define a second constrained violation function $h_X = h + \Psi_x$, where Ψ_x is the indicator function for the feasible region X,

$$\Psi_x = \begin{cases} 0 & \text{on } X, \\ +\infty & \text{elsewhere.} \end{cases}$$

Others definitions are needed here.

Definition 1 *A point $x \in \mathbb{R}^n$ is said to dominate $y \in \mathbb{R}^n$, writen $x \prec y$, if $f(x) \leq f(y)$ and $h_X(x) \leq h_X(y)$ with either $f(x) < f(y)$ or $h_X(x) < h_X(y)$.*

Definition 2 *A filter, denoted F, is a finite set of points in the domain of f and h such that no pair of points x and y in the set have the relation $x \prec y$.*

The filter in [1] has two additional restrictions on F:

- Set a bound $h_{\max}$ on aggregate constraint violation, so that each point $y \in F$ satisfies $h_X(x) < h_{\max}$;
- Include only infeasible points in the filter and track feasible points separately.

With these two modifications we can define:

Definition 3 *A point x is said to be filtered by a filter F if any of the following properties hold*:

- *There exists a point $y \in F$ such that $y \prec x$ or $y = x$*;
- $h_X(x) \geq h_{\max}$;
- $h_X(x) = 0$ *and* $f(x) \geq f^F$ *, where* f^F *is the objective function value of the best feasible point found thus far.*

Definition 4 *The point x is said to be unfiltered by F if is not filtered by F.*

Thus, the set of unfiltered points, denoted by $\bar{F}$, is given by

$$\bar{F} = x \in F \cup \{y : y \prec x \text{ or } y = x\} \cup \{y : h_X(y) = 0, f(y) \geq f^F\}.$$

Note that, with this notation, if a new trial point has the same function values as those of any point in the filter, then the trial point is filtered. Thus, only the first point with such values is accepted into the filter.

3.2 *Pattern search algorithm*

With the notation and definitions of the previous section we can now present the pattern search algorithm of Audet and Dennis:

1. INITIALIZATION
   Let x_0 be an undominated point of a set of initial solutions. Include all these solutions in the filter F_0, together with $h_{\max} > h(x_0)$. Fix the mesh size parameter $ll_0 > 0$, $(ll_k \to 0, k \to +\infty)$ and set the iteration counter k to 0.

2. DEFINITION OF INCUMBENT SOLUTIONS
   Define (if possible)
   f_k^F: the smallest objective function value for all feasible solutions found so far;
   $h_k^I > 0$: the least positive constrained violation function value found so far;
   f_k^I: the smallest objective function value of the points found so far whose constraint violation function value are equal to h_k^I.

3. SEARCH AND POLL STEPS
   Perform the SEARCH and possibly the POLL step (or only part of the steps) until an unfiltered trial point x_{k+1} is found, or when it is shown that all trial points are filtered by F_k:
   – SEARCH STEP: Evaluate the function h and f on a set of trial points on the current mesh M_k (the strategy that gives the set of points is usually provided by the user).

– POLL STEP: Evaluate the function h and $\boldsymbol{f}$ on the poll set around p_k, where p_k satisfies either $(h(p_k), \boldsymbol{f}(p_k)) = (0, f_k^F)$ or $(h(p_k), \boldsymbol{f}(p_k)) = (h_k^I, f_k^I)$ a set of trial points on the current mesh M_k (the strategy that gives the set of points is usually provided by the user).

4. PARAMETER UPDATE
   If the SEARCH or POLL step produced an unfiltered iterate $x_{k+1} \in F_{k+1}$, then declare the iteration *successful* and update $ll_{k+1} \geq ll_k$.
   Otherwise, set $x_{k+1} = x_k$, declare the iteration *unsuccessful* and update $ll_{k+1} < ll_k$.
   Increase $k \leftarrow k + 1$ and go back the definition of the incumbents (Step 2).

4. Simplex filter algorithm

In this section we present a new derivative-free algorithm to solve nonlinear constrained problems, like the Audet and Dennis method, that combines the features of a derivative free method and filter methods. It uses the pattern search derivative-free method. Our algorithm combines filter methods with a simplex method, so it consists in a simplex method for general constrained optimization. This method does not compute or approximate any derivatives, penalty constants or Lagrange multipliers.

4.1 *Notation and definitions*

The terminology used by Audet and Dennis differs from the usual terminology of filter methods. That terminology is more adjusted to derivative-free methods and so more simple to understand in this context. In this work we adopted the same notation and definitions.

4.2 *Simplex search algorithm*

As in the simplex methods presented in Section 2.2, the basic idea of simplex filter algorithm is to construct an initial simplex and use the simplex to drive the search. The difference is that now we have a constrained problem so we introduce a filter to accept or reject the step.

In this method feasible points, in each iteration, were tested by the filter and we use the four basic operations of Nelder and Mead for the infeasible points. We call this procedure the *simplex search*. Each operation produce a new vertex that can be a *good* (which are a unfiltered trial point) or *bad* (which are filtered and leave the filter unmodified) vertex in the present simplex.

If the simplex search produces an unfiltered iterate, then the iteration is accepted and we can find the usual best solution in the actual filter. The solution is the vertex that has the smallest objective function value of the unfiltered points found so far, $\boldsymbol{f}_k^F$, whose constraint violation function value are equal to zero.

Otherwise we reduce the lengths of the edges adjacent to the current best vertex by half and repeat the process.

The simplex search algorithm is

1. INITIALIZATION
   Let x_0 be an undominated point of a set of initial solutions. Include all these solutions in the filter F_0, together with $h_{\max} > h(x_0)$. Fix the parameters s, length of the edge step, α, β, γ, the reflection, contraction and expansion parameters and set the iteration counter k to 1.

2. DEFINITION OF INCUMBENT SOLUTIONS
   Define (if possible)

f_k^F: the smallest objective function value for all feasible solutions found so far;
$h_k^I > 0$: the least positive constrained violation function value found so far;
f_k^I: the smallest objective function value of the points found so far whose constraint violation function value are equal to h_k^I.

2. SIMPLEX SEARCH STEPS
   Perform the SIMPLEX SEARCH STEPS until an unfiltered trial point x_k is found, or when it is shown that all trial points are filtered by F_{k-1}
   – Construct the simplex;
   – Compute the function values and constraint violation value of all vertex of the simplex;
   – Test the feasible vertexes by the filter;
   – Implement the four basics operations of Nelder and Mead for the infeasible vertexes.

3. ACEPTANCE STEP
   If the result is an unfiltered iterate $x_k \in F_k$, then:
   – Accept the iteration;
   – Find the usual best solution in the actual filter;
   – Repeat the process until the stop criterion pre-established is verified.
   Otherwise, reduce the lengths of the edges adjacent to the current best vertex by half and repeat the process.

5. Numerical results

5.1 *Java language*

The proposed methods were implemented in Java. Java is a multi-platform technology capable of running on multiple processors or Operating Systems. It has a rich variety of classes that abstracts the Internet protocols such as HTTP, FTP, IP, TCP-IP, SMTP, DNS, making it suitable for networked applications. In fact it was the Internet that made it so popular, embeded in applets, running in web pages.

One of the key features of Java is the fact that it is compiled once and can run anywhere. An application, independently of the platform (processor and Operating System) where it is compiled, will run on any other platform capable of running Java.

Although Java uses interpreted code it has good performance. Some comparisons between Java and other programming languages have been made to test its performance. Its performance is comparable to the performance of C, in finite element analysis [20]. It was also compared against C and FORTRAN in scientific applications [3], and the conclusion was that Java had a good relative performance.

5.2 *Problems*

To test the implemented methods, we solved several problems using the developed Java applications. The first problem was presented by Lewis and Torczon and used by Audet and Dennis in [1] to exemplify their method.

Problem A:

$$\begin{array}{ll} (x \in \mathbb{R}^2)\ \text{minimize} & -x_1 - 2x_2 \\ \text{subject to} & 0 \le x_1 \le 1, \\ & x_2 \le 0. \end{array} \qquad (2)$$

This problem is a linear program, whose optimal solution is $x^* = (1,0)^{\mathrm{T}}$.

Similar to Audet and Dennis in [1], we consider the initial point $x_0 = (0, 0)^T$, the mesh size parameter $ll_0 = 1$ and the four directions $\pm(1, 1)^T$ and $\pm(1, -1)^T$, to implement the Pattern search algorithm, and the same initial point, the step size $s_0 = 1$ and the directions $(1, 0)^T$ and $(0, 1)^T$, to implement the Simplex search algorithm.

Second and third problems were selected in the collection Cute [2].

Problem C-801:

$$\begin{array}{ll} (x \in \mathbb{R}^2) \text{ minimize} & 6x_1^2 + x_2^2 - 60x_1 - 8x_2 + 166 \\ \text{subject to} & 0 \le x_1 \le 10, \\ & 0 \le x_2 \le 10, \\ & x_1 + x_2 - x_1x_2 \ge 0, \\ & x_1 + x_2 - 3 \ge 0. \end{array} \tag{3}$$

This problem is a nonlinear program, whose optimal solution x^* is unknown, but the function value is $f(x^*) = 7.563$. We consider in this issue the feasible initial point $x_0 = (5, 1)^T$.

Problem C-802:

$$\begin{array}{ll} (x \in \mathbb{R}^2) \text{ minimize} & 7x_1^2 + 3x_2^2 - 84x_1 - 34x_2 + 300 \\ \text{subject to} & 0 \le x_1 \le 10, \\ & 0 \le x_2 \le 10, \\ & x_1x_2 - 1 \ge 0, \\ & 9 - x_1^2 - x_2^2 \ge 0. \end{array} \tag{4}$$

This problem is a nonlinear program, whose optimal solution x^* is unknown, but the function value is $f(x^*) = -97.30952$. We consider in this problem the feasible initial point $x_0 = (2.5,\ 1.5)^T$.

We consider, in these two problems the mesh size parameter $ll_0 = 1$ and the four directions $\pm(1, 1)^T$ and $\pm(1, -1)^T$, to implement the Pattern search algorithm, and the same initial point, the step size $s_0 = 1$ and the directions $(1, 0)^T$ and $(0, 1)^T$, to implement the Simplex search algorithm.

5.3 *Numerical results*

We consider in these tests tolerances $T1$ and $T2$ ($T1 = T2 < 0.0001$) where

$$T1 = \frac{\|x_{k+1} - x_k\|_2}{\|x_{k+1}\|_2}$$

and

$$T2 = |f_{k+1} - f_k|$$

(x_k is the iteration k, x_{k+1} is the iteration $k + 1$ and f_k and f_{k+1} are they function values, respectively) and $k_{\max} = 40$.

Problem A

Implementation of algorithms to the problem A (2) produces the following results.

Results	Pattern search algorithm	Simplex search algorithm
Number of evaluations of h	157	129
Number of evaluations of f	195	158
Unsuccessful iterations	39	29
Successful iterations	1	1
Last successful iteration	2th	2th
Best solution (x_k)	$(1.0;0.0)^T$	$(1.0;0.0)^T$
Function value ($f(x_k)$)	-1.0	-1.0
Total time	6 seconds	5 seconds

Implementation to the problem A

Comparison of numerical results

Comparing the previous results we can say the following:

- Both methods have found the solution in the second iteration;
- The number of functions' evaluations is higher in the first method;
- The total execution time is similar in both methods; however, simplex search is faster.

Problem C-801

Implementation of algorithms to the problem C-801 (3) produces the following results.

Results	Pattern search algorithm	Simplex search algorithm
Number of evaluations of h	138	118
Number of evaluations of f	138	146
Unsuccessful iterations	1	29
Successful iterations	29	1
Last successful iteration	4th	4th
Best solution (x_k)	$(5.0;1.25)^T$	$(5.0;1.25)^T$
Function value ($f(x_k)$)	7.5625	7.5625
Total time	0 s	0 s

Implementation to the problem C-801

Comparison of numerical results

Comparing the previous results we can say the following:

- Both methods have found the solution in the fourth iteration;
- The number of h function evaluations is higher in the first method and the number of f function evaluations is higher in the second method;
- The execution is very small, below one second, in the two methods.

Problem C-802

Implementation of algorithms to the problem C-802 (4) produces the following results.

Results	Pattern search algorithm	Simplex search algorithm
Number of evaluations of h	154	153
Number of evaluations of f	154	189
Unsuccessful iterations	34	37
Successful iterations	6	3
Last successful iteration	36th	24th
Best solution (x_k)	$(2.50006103515625;1.6582202911376953)^T$	(2.5;1.658203125)
Function value ($f(x_k)$)	85.61660300656149	85.6200065612793
Total time	0 s	0

s Implementation to the problem C-802

Comparison of numerical results

Comparing the previous results we can say that:

- None of the methods can find the solution. The two methods have the same best value of the objective function (approximately 85);
- The number of functions evaluations is similar, but the number of f function evaluations is higher in the second method;
- The execution time is very small, approximately zero seconds, in the two methods.

In problems C-801 and C-802 ((3) and (4), respectively), execution times are shown as 0 seconds, because when applications take less than one second to execute and exit, the used Java IDE (Integrated Development Environment) shows the execution time has 0 seconds. Taking into account the obtained results, presented in this section, we can say that the methods have very similar behaviour, for these problems. In this implementation we also can see that the solutions quality is very dependent of the initial point. Therefore, we chose feasible points to start the methods. Actually if the proposed starting points in [2] were used to solve (3) and (4), the methods could not find the solution. We can conclude that Simplex Search Algorithm can be considered as another alternative to solve Optimization Nonlinear Constrained Problems.

6. Conclusions and future work

In this work we present a new direct search method, based on simplex methods, for general constrained optimization that combines the features of the simplex method and filter methods. This method does not compute or approximate any derivatives.

Traditionally, this kind of problems is solved using penalty or merit functions that are a linear combination of the objective function and a measure of the constrained violation. These methods are designed to solve this problem by instead solving a sequence of constructed unconstrained problems, so they were seen as vehicle for solving constrained optimization problems by means of unconstrained optimization techniques.

In this article we presented an alternative to this kind of methods and another way to solve constrained nonlinear problems. The proposed methods were implemented in Java, which is a

multi-platform technology. It will allow, in the future, to embed in a web page an applet that can solve any constrained and unconstrained nonlinear problems. Obtained numerical data show that the proposed method can be used to solve the proposed problems.

References

[1] C. Audet and J.E. Dennis, *A pattern search filter method for nonlinear programming without derivatives*, SIAM J. Optim. 14(4) (2004), pp. 980–1010.

[2] I. Bongartz, A.R. Conn, N.I. Gould, and P.L. Toint, *CUTE: Constrained and Unconstrained Testing Environment*, ACM Transactions on Mathematical Software: 21 (1995) pp. 123–160.

[3] J. M. Bull, L.A. Smith, C. Ball, L. Pottage, and R. Freeman, *Benchmarking Java against C and Fortran for scientific applications*, Concurrency and Computation: Practice and Experience 15(3–5) (2003), pp. 417–430.

[4] R.H. Byrd, J. Nocedal and R.A. Waltz, *Steering exact penalty methods for optimization*, Tech. Rep., Optimization Technology Center, Northwestern University, Evanston, IL 60208, USA, 2006.

[5] L. Chen and D. Goldfarb, *Interior-point l2-penalty methods for nonlinear programming with strong global convergence properties*, Mathematical Programming, Tech. Rep., IEOR Department, Columbia University, New York, 2005.

[6] R. Fletcher and S. Leyffer, *A bundle filter method for nonsmooth nonlinear optimization*, Tech. Rep. NA/195, Dundee University, Department of Mathematics, 1999.

[7] R. Fletcher and S. Leyffer, *Nonlinear programming without a penalty function*, Math. Program.: Ser. A, 91(2) (2002) pp. 239–269.

[8] R. Fletcher, S. Leyffer, and P. L. Toint, *On the global convergence of an SLP-filter algorithm*, Tech. Rep. NA/183, Dundee University, Department of Mathematics, 1998.

[9] R. Fletcher, N. Gould, S. Leyffer, P.L. Toint, and A. Wachter, *Global convergence of trust region and SQP-filter algorithms for general nonlinear programming*, SIAM J. Optim. 13(3) (2002), pp. 635–659.

[10] R. Fletcher, S. Leyffer, and P.L. Toint, *A brief history of filter method*, Tech. Rep. ANL/MCS-P1372-0906, Argonne National Laboratory, Mathematics and Computer Science Division, 2006.

[11] N.I.M. Gould, D. Orban, and P.L. Toint, *An interior-point l1-penalty method for nonlinear optimization*, Tech. Rep. RAL-TR-2003-022 Rutherford Appleton Laboratory Chilton, Oxfordshire, UK, November 2003.

[12] N.I.M. Gould, S. Leyffer, and P.L. Toint, *A multidimensional filter algorithm for nonlinear equations and nonlinear least-squares*, SIAM J. Optim. 15(1) (2005), pp. 17–38.

[13] N.I.M. Gould, C. Sainvitu, and P.L. Toint, *A filter-trust-region method for unconstrained optimization*, SIAM J. Optim. 16(2) (2006), pp. 341–357.

[14] R. Hooke and T. Jeeves, *Direct search solution of numerical and statistical problems*, J. Assoc. Comput. Mach. 8 (1961), pp. 212–229.

[15] E. Karas, A. Ribeiro, C. Sagastizábal, and M. Solodov, *A bundle-filter method for nonsmooth convex constrained optimization*, Math. Program. 116(1) (2006), pp. 297–320.

[16] R.M. Lewis, V. Trosset, and M.W. Trosset, *Direct Search Methods: Then and Now*, NASA Langley Research Center, ICASE Report No. 2000-26, Virginia, 2000.

[17] S. Leyffer, G. López-Calva, and J. Nocedal, *Interior Methods for Mathematical Programs with Complementarity Constraints*, SIAM J. Optim. 17(1) (2006), pp. 52–77.

[18] M. Mongeau and A. Sartenaer, *Automatic decrease of the penalty parameter in exact penalty function methods*, Eur. J. Oper. Res. 83(3) (1995), pp. 686–699.

[19] J.A. Nelder and R. Mead, *A simplex method for function minimization*, Comput. J. 7 (1965), pp. 308–313.

[20] G.P. Nikishkov, Yu.G. Nikishkov, and V.V. Savchenko, *Comparison of C and Java performance in finite element computations*, Comput. Struct. 81 (24–25)(2003), pp. 2401–2408.

[21] M.J.D. Powell, *An efficient method for finding the minimum of a function of several variables without calculating derivatives*, Comput. J. 7 (1964), pp. 155–162.

[22] M. Ulbrich, S. Ulbrich, and L.N. Vicente, *A globally convergent primal-dual interior-point filter method for nonlinear programming*, Math. Program. Ser. A, 100(2) (2004), pp. 379–410.

[23] A.J. Zaslavski, *A sufficient condition for exact penalty in constrained optimization*, SIAM J. 16(1) (2005), pp. 250–262.