European Journal of Operational Research 200 (2010) 658–670

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Continuous Optimization

A linear programming-based optimization algorithm for solving nonlinear programming problems

Claus Still [*], Tapio Westerlund

Process Design and Systems Engineering Laboratory, Åbo Akademi University, Biskopsgatan 8, FIN-20500, Åbo, Finland

ARTICLE INFO

Article history:
Received 7 January 2007
Accepted 21 January 2009
Available online 31 January 2009

Keywords:
Nonlinear programming
Cutting plane algorithms
Sequential linear programming

ABSTRACT

In this paper a linear programming-based optimization algorithm called the Sequential Cutting Plane algorithm is presented. The main features of the algorithm are described, convergence to a Karush–Kuhn–Tucker stationary point is proved and numerical experience on some well-known test sets is showed. The algorithm is based on an earlier version for convex inequality constrained problems, but here the algorithm is extended to general continuously differentiable nonlinear programming problems containing both nonlinear inequality and equality constraints. A comparison with some existing solvers shows that the algorithm is competitive with these solvers. Thus, this new method based on solving linear programming subproblems is a good alternative method for solving nonlinear programming problems efficiently. The algorithm has been used as a subsolver in a mixed integer nonlinear programming algorithm where the linear problems provide lower bounds on the optimal solutions of the nonlinear programming subproblems in the branch and bound tree for convex, inequality constrained problems.

1. Introduction

Kelley's cutting plane method [11] was introduced 1960 to solve nonlinear programming (NLP) problems by solving a sequence of linear programming (LP) problems. Although some other methods based on linear programming exist, such as the method of approximate programming [6], LP techniques were quickly abandoned in favor of sequential quadratic programming (SQP) techniques. After Han proved local and global convergence of SQP methods in [7, 8], a large amount of research papers have been produced on SQP-based techniques. Indeed, many of the NLP solvers today use SQP techniques in one form or the other.

There are some interesting recent papers on successive linear programming (SLP) techniques. In [2], a procedure is presented where linear programming and quadratic programming subproblems are successively solved to find the optimal solution. The linear programming problem provides an estimate of the active constraints within a trust region and a quadratic programming problem is constructed and solved using the active constraints at the optimal solution of the linear problem. However, the method in [2] utilizes linear programming problems mainly to estimate the active constraints in each iteration, and solves a quadratic, equality constrained, problem as well in each iteration.

In this paper it is shown that LP techniques can be applied quite successfully to solve NLP problems efficiently, even without having to solve quadratic subproblems. Indeed, numerical tests on two standard problem test sets show that the described algorithm is a competitive alternative to other NLP solvers. The algorithm described here can be used to solve NLP problems with both nonlinear inequality and equality constraints and global convergence to a Karush–Kuhn–Tucker (KKT) stationary point is shown for nonconvex continuously differentiable problems. The proposed algorithm is an extension of the sequential cutting plane (SCP) algorithm introduced in [19]. The original algorithm only solved convex problems with nonlinear inequality constraints. Note that the acronym SCP should not be confused with sequential convex programming, introduced in [24]. In sequential convex programming, the original NLP problem is solved by solving a sequence of convex, separable nonlinear subproblems. Here, the approach is to use linear subproblems to solve the original NLP problem.

The primary goal with the algorithm described in this paper, as with the convex version of the algorithm, is to optimize performance on problems where the objective and constraints are easy to evaluate. The primary target has not been to minimize the number of function evaluations. If the constraints and objective are time-consuming to evaluate, then some other algorithm may be better suited for such problems.

One application of the algorithm is as a component in a mixed integer nonlinear programming (MINLP) algorithm. For convex, inequality constrained MINLP problems the LP subproblems

[*] Corresponding author.
E-mail addresses: Claus.Still@abo.fi, cstill@abo.fi (C. Still), Tapio.Westerlund@abo.fi (T. Westerlund).

0377-2217/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2009.01.033

conveniently provide a lower bound on the optimal solution of the convex NLP subproblem. Lower bounds are needed in the branch and bound procedure for efficiency reasons. See [21] for more details. Very promising results are reported in [20] for a special set of difficult block optimization problems. The MINLP version of the algorithm found better solutions in one minute compared to the solutions that other commercial solvers found in 12 h.

Solving convex MINLP problems is also important in a global MINLP optimization context as most of the deterministic algorithms are based on solving a sequence of convexified MINLP problems [1,16,23]. The algorithm can also be used for solving general nonconvex MINLP problems as a subsolver in an NLP branch and bound algorithm [4]. Numerical experience indicate that the SCP algorithm can be applied for these types of problems as well as convergence to a stationary point appear to be fast also when iterates are far from the stationary point.

1.1. Overview

Our proposed algorithm solves problems of the form

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & g_j(x) \leqslant 0,\ j = 1,\dots,m_i, \\ & h_r(x) = 0,\ r = 1,\dots,m_e, \\ & x \in R^n, \end{aligned} \qquad (NLP) \tag{1}$$

where the functions $f : R^n \to R$, $g_j : R^n \to R, j = 1,\dots,m_i$ and $h_r : R^n \to R, r = 1,\dots,m_e$ are continuously differentiable over R^n. Contrary to [19], the objective or constraints are not assumed to be convex. It is assumed that the constraints $g_j(x) \leqslant 0, j = 1,\dots,m_i$ include linear constraints defining a bounded region X.

It is also assumed that the extended Mangasarian–Fromovitz constraint qualification (EMFCQ) holds for any $x \in R^n$. The constraint qualification holds at $x \in R^n$ when $\nabla h_r(x),\ r = 1,\dots,m_e$ are linearly independent and there exists a $z \in R^n$ such that

$$\nabla g_j(x)^T z < 0,\ j \in J_+(x) \cup J_0(x),$$
$$\nabla h_r(x)^T z = 0,\ r = 1,\dots,m_e,$$

where J_+ is an index set denoting the violated constraints g_j,

$$J_+(x) := \{j : g_j(x) > 0\},$$

and J_0 is an index set denoting the active constraints g_j,

$$J_0(x) := \{j : g_j(x) = 0\}.$$

The EMFCQ and its relation to exact penalty functions are considered in [15]. In the context of this algorithm, the constraint qualification guarantees that one can, for any infeasible point, find a search direction such that the constraint infeasibilities are reduced.

2. Algorithm

The algorithm is similar to the algorithm presented in [19] in that it will perform a sequence of NLP iterations until a locally optimal solution is found. Each NLP iteration consists of a sequence of LP subiterations. In each LP subiteration an LP problem is solved and a line search performed in the search direction obtained as the solution to the LP problem. At the end of the NLP iteration, the new iterate must reduce a merit function sufficiently in order to guarantee convergence. If not, a new iterate is generated such that the merit function is sufficiently reduced.

2.1. LP subiterations

In each LP subiteration an LP problem is solved. The LP problem is based on forming cutting planes in the current iterate $x^{(i)}$. In LP subiteration (i) of an NLP iteration the LP problem solved is

$$\min \quad \nabla f(x^{(i)})^T d + C\left(\sum_{j=1}^{m_i} t_j^g + \sum_{r=1}^{m_e} t_r^{h^+} + \sum_{r=1}^{m_e} t_r^{h^-}\right),$$

$$\text{s.t.} \quad g_j(x^{(i)}) + \nabla g_j(x^{(i)})^T d \leqslant t_j^g,\ j = 1,\dots,m_i, \tag{1a}$$
$$h_r(x^{(i)}) + \nabla h_r(x^{(i)})^T d = t_r^{h^+} - t_r^{h^-},\ r = 1,\dots,m_e, \tag{1b}$$
$$(d^{(r)})^T H^{(i)} d = 0,\ r = 1,\dots,\ i-1;\ i > 1, \tag{1c}$$
$$d^L \leqslant d \leqslant d^U, \tag{1d}$$
$$0 \leqslant t_j^g \leqslant \max\{g_j(x^{(i)}), 0\},\ j = 1,\dots,m_i, \tag{1e}$$
$$0 \leqslant t_r^{h^+} \leqslant |h_r(x^{(i)})|,\ r = 1,\dots,m_e, \tag{1f}$$
$$0 \leqslant t_r^{h^-} \leqslant |h_r(x^{(i)})|,\ r = 1,\dots,m_e, \tag{1g}$$

which has been generated in the point $x^{(i)}$. Call the problem $LP(x^{(i)})$ and the optimal solution to the problem $(d^{(i)}, t^{g,(i)}, t^{h^+,(i)}, t^{h^-,(i)})$, where $d^{(i)} \in R^n$, $t^{g,(i)} \in R^{m_i}$, $t^{h^+,(i)} \in R^{m_e}$ and $t^{h^-,(i)} \in R^{m_e}$.

Here the constraints (1a) and (1b) are linearizations of the nonlinear constraint functions g and h at $x^{(i)}$ and t^g, t^{h^+} and t^{h^-} are included to relax the constraints such that a solution d exists within the box constraints (1d).

The constraints

$$(d^{(r)})^T H^{(i)} d = 0,\ r = 1,\dots,i-1;\ i > 1,$$

ensure that the solution will be a conjugate direction with respect to the previously obtained search directions within the NLP iteration. The search directions for the previous LP subiterations within the NLP iteration are denoted $d^{(r)}$, $r = 1,\dots,i-1$ and $H^{(i)}$ is the estimate, in LP subiteration (i), of the Hessian of the Lagrangian for (NLP).

Note that for the convergence proof, only one LP subiteration is needed in each NLP iteration. The subsequent LP subiterations are only performed to improve the convergence speed. The subsequent LP subiterations generate conjugate search directions and thus the algorithm resembles the conjugate gradient search method for unconstrained problems.

The lower bounds are assumed to be negative and the upper bounds positive, i.e. $d^L < \mathbf{0}$ and $d^U > \mathbf{0}$.

Note that both the linearizations of the constraints as well as the constraints enforcing conjugacy with respect to the Hessian estimate of the Lagrangian can be viewed as cutting planes. The constraints (1a) form half-spaces that approximate the feasible region of g, (1b) are cutting hyperplanes that approximate the feasible region of h and the constraints (1c) are cutting hyperplanes that limit the search direction d by enforcing it to lie on these hyperplanes and thus be a conjugate direction to the previously obtained search directions within the NLP iteration.

2.1.1. Lagrange multiplier estimates

The optimal values of the dual variables from (1) are used as Lagrange multiplier estimates for subsequent line searches and for the estimate of the Hessian of the Lagrangian.

If the LP problem is generated in a stationary point x^* to (NLP), then the dual variables λ and μ of the constraints (1a) and (1b) for the solution to $LP(x^*)$ will be Lagrange multipliers for x^* in (NLP).

2.1.2. Infeasible LP problems

The first LP problem within the NLP iteration cannot be infeasible, since the variables t^g, t^{h^+} and t^{h^-} will relax the problem sufficiently to allow a solution $d = \mathbf{0}$ for any problem (when the relaxing variables are equal to the maximum constraint violation). Thus, there is no need to deal with infeasible LP problems as was the case for the convex version of the algorithm presented in [19].

Note, however, that the constant C should be chosen large enough to guarantee, for infeasible iterates, that an optimal solution is obtained where $d^{(i)} \neq \mathbf{0}$ and the constraint infeasibility is reduced.

The constraint infeasibility is reduced when there is either a $j \in \{1,\ldots,m_i\}$ such that $t_j^g < g_j(x^k)$ or there is an $r \in \{1,\ldots,m_e\}$ such that $t_r^{h^+} < |h_r(x^k)|$ and $t_r^{h^-} < |h_r(x^k)|$ for the optimal solution to $LP(x^k)$.

If such a solution cannot be obtained in the first LP subiteration, then the original problem (*NLP*) is considered infeasible. Generally, this is not true, as the algorithm may have converged to a locally infeasible point. However, as it was assumed that the EMFCQ holds, the constraint qualification guarantees that such solutions can be found.

2.1.3. Line search

The optimal solution $d^{(i)}$ to $LP(x^{(i)})$ provides a search direction, which is used in a line search to minimize the function

$$\widetilde{L}(x,\lambda,\mu) = f(x) + \sum_{j=1}^{m_i}\lambda_j g_j(x)^+ + \sum_{r=1}^{m_e}|\mu_r h_r(x)| + \rho\sum_{j=1}^{m_i}\lambda_j(g_j(x)^+)^2 + \rho\sum_{r=1}^{m_e}|\mu_r|(h_r(x))^2.$$

Here λ and μ are Lagrange multiplier estimates obtained from the dual variables of the previously solved LP problem and $g_j(x)^+ = \max\{g_j(x),0\}$. The parameter ρ is a penalty parameter that is kept fixed during the optimization process.

The line search looks for the minimum of $\widetilde{L}^{(i)}(x) = \widetilde{L}(x,\lambda^{(i)},\mu^{(i)})$, where $\lambda^{(i)}$ and $\mu^{(i)}$ are the Lagrange multiplier estimates obtained in LP subiteration (i). The line search is performed in direction $d^{(i)}$ starting from $x^{(i)}$, that is

$$\alpha^{(i)} = \underset{0\leqslant\alpha\leqslant 1}{\operatorname{argmin}}\,\widetilde{L}^{(i)}(x^{(i)} + \alpha d^{(i)}).$$

The next iterate is then set to $x^{(i+1)} = x^{(i)} + \alpha^{(i)}d^{(i)}$ and used as the starting point in the next LP subiteration. Note that an exact line search is not necessarily required, one could also use sufficient decrease criteria for the line search.

2.1.4. Hessian estimates

The standard Broyden–Fletcher–Goldfarb–Shanno (BFGS) update formula was used to provide estimates for the Hessian, although other methods could be used as well. The Hessian estimate is based on the Lagrangian function

$$L(x,\lambda,\mu) = f(x) + \sum_{j=1}^{m_i}\lambda_j g_j(x) + \sum_{r=1}^{m_e}\mu_r h_r(x).$$

2.1.5. Subiteration termination criteria

The above described steps are then repeated until a termination criterion is met. A number of criteria are used to determine when to stop doing LP subiterations. Contrary to the algorithm described in [19], the first LP subproblem in each NLP iteration cannot be infeasible as there are variables t^g, t^{h^+} and t^{h^-} that relaxes the LP subproblem. Instead, for the first subiteration, an optimal solution to the LP subproblem where the constraint infeasibility is not reduced indicates that the original problem may be infeasible or that the point is close to a locally infeasible point. In subsequent subiterations, if any of the relaxation variables are greater than zero, the equality constraints (1c) may be too constraining and the subiterations are terminated.

The algorithm stops doing more LP subiterations if one of the following criteria is met:

(1) Terminate if $i > n$.
(2) Terminate if the norm of $d^{(i)}$ is close to zero.
(3) Terminate if the constraint infeasibility is not reduced, i.e. there is not a $j \in \{1,\ldots,m_i\}$ such that $t_j^g < g_j(x^k)$ and there is not an $r \in \{1,\ldots,m_e\}$ such that $t_r^{h^+} < |h_r(x^k)|$ and $t_r^{h^-} < |h_r(x^k)|$ for the optimal solution to $LP(x^k)$.
(4) Terminate if $i > 1$ and any of relaxation variables t^g, t^{h^+} and t^{h^-} are greater than zero.
(5) Terminate if $\alpha^{(i)}$ is close to one.

2.2. NLP iterations

Each NLP iteration consists of a sequence of LP subiterations. Thus, several LP problems are solved and several line searches performed in each NLP iteration until one of the subiteration termination criteria described in 2.1.5 is met. At the end of the NLP iteration, the new iterate must reduce a merit function sufficiently in order to guarantee convergence to a KKT stationary point. If not, the new iterate must be replaced with an iterate that does reduce the merit function sufficiently. Such an iterate can be found by restarting from the previously accepted iterate and minimizing the merit function, rather than the function $\widetilde{L}^{(i)}$, in a descent direction for the merit function.

The merit function used here is

$$M(x) = f(x) + p(x),$$

where $p(x)$ is a penalty term defined by

$$p(x) = \sum_{j=1}^{m_i}\bar{\lambda}_j g_j(x)^+ + \sum_{r=1}^{m_e}\bar{\mu}_r|h_r(x)|.$$

The parameters $\bar{\lambda}$ and $\bar{\mu}$ should be chosen such that they are greater than the absolute value of any Lagrange multiplier estimate, thus

$$\bar{\lambda}_j > \lambda_j,\ j = 1,\ldots,m_i \tag{2}$$

and

$$\bar{\mu}_r > |\mu_r|,\ r = 1,\ldots,m_e \tag{3}$$

should hold for any Lagrange multiplier estimate λ and μ obtained during the optimization process.

In practice, numerical experience shows that it is better to use a procedure that updates $\bar{\lambda}$ and $\bar{\mu}$ dynamically during the optimization process whenever estimates of the Lagrange multipliers are greater than the current $\bar{\lambda}$ and $\bar{\mu}$ (or use some similar scheme for estimating these parameters). Assumptions (2) and (3) are, however, needed for the convergence proof.

2.2.1. Sufficient reduction test

The new iterate at the end of the NLP iteration needs to satisfy a sufficient reduction test. Note first that the directional derivatives of $g_j(x)^+$ and $|\,h_r(x)\,|$ in direction d are

$$D_d\,g_j(x)^+ = \begin{cases}\nabla g_j(x)^T d & \text{if } g_j(x) > 0,\\ (\nabla g_j(x)^T d)^+ & \text{if } g_j(x) = 0,\\ 0 & \text{if } g_j(x) < 0,\end{cases}$$

and

$$D_d|h_r(x)| = \begin{cases}\nabla h_r(x)^T d & \text{if } h_r(x) > 0,\\ |\nabla h_r(x)^T d| & \text{if } h_r(x) = 0,\\ -\nabla h_r(x)^T d & \text{if } h_r(x) < 0.\end{cases}$$

Consequently, the directional derivative of M in direction d is

$$D_d\,M(x) = \nabla f(x)^T d + \sum_{j=1}^{m_i}\bar{\lambda}_j D_d\,g_j(x)^+ + \sum_{r=1}^{m_e}\bar{\mu}_r D_d\,|h_r(x)|.$$

The new iterate x^{k+1}, at the end of NLP iteration k, should satisfy

$$M(x^{k+1}) \leqslant M(x^k + \alpha^k d^k) \tag{4}$$

and $x^k + \alpha^k d^k$ should satisfy

$$M(x^k + \alpha^k d^k) - M(x^k) \leqslant \sigma\alpha^k D_{d^k}M(x^k),\ \sigma \in (0,1) \tag{5}$$

and

$$D_{d^k}M(x^k + \alpha^k d^k) \geqslant \eta D_{d^k}M(x^k),\ \eta \in (\sigma, 1). \tag{6}$$

Here x^k is the current iterate in the first LP subiteration for NLP iteration k, d^k is the search direction obtained as a solution to $LP(x^k)$ and α^k is the result for the corresponding line search. Since the line search is restricted to $0 \leqslant \alpha^k \leqslant 1$, the iterate can be accepted if $\alpha^k = 1$ even if (6) does not hold.

The requirements (4)–(6) are theoretical requirements needed when proving convergence of the algorithm. The requirement (5) ensures that the merit function M is sufficiently reduced and (6) ensures that large enough steps are taken in each iteration, i.e. it does not allow arbitrarily small α^k. Therefore, $\alpha^k = 1$ can alternatively be accepted. Finally, (4) ensures that the iterate at the end of the NLP iteration does not increase the merit function value above the merit function value obtained after the first LP subiteration in the NLP iteration. Note that it is proved later in Theorem 6 that $D_{d^k} M(x^k) < 0$ unless x^k is a KKT stationary point.

Note also that if d^k is a descent direction for M and an exact line search starting from x^k minimizing M is performed, then $x^k + \alpha^k d^k$ will satisfy the sufficient reduction criteria (4) and (6), but not necessarily (5). However, in practice this can be avoided by choosing σ sufficiently close to zero. Alternatively, an inexact line search algorithm that finds a step satisfying (5) and (6), and hence also (4) if $x^{k+1} = x^k + \alpha^k d^k$, is described in [13].

2.2.2. Generating acceptable iterates

If the new iterate x^{k+1} does not satisfy the sufficient reduction test defined by (4)–(6), a new iterate that satisfies the test must be generated. It is shown in Theorem 6 that the solution $d^{(1)}$ to the first LP subproblem in an NLP iteration is a descent direction for the merit function M. Thus, a new point that satisfies the sufficient reduction test can be generated by restarting the LP subiterations from x^k and repeating the line searches in the directions $d^{(i)}$ obtained in the LP subiterations, but rather minimizing M instead of $\widetilde{L}^{(i)}$ in each line search. Since at least $d^{(1)}$ is a descent direction for the merit function, an acceptable iterate can be generated by using this procedure.

2.3. Updating trust region bounds

The bounds d^L and d^U form a trust region for the solution d to the LP subproblem. This trust region may either be increased if the current trust region is too small or reduced if the current trust region is too large. A simple procedure to update the trust region based on the step length taken in each NLP iteration was used. Let

$$\delta_l = \max\left\{\left|\frac{x_l^{k+1} - x_l^k}{d_l^L}\right|, \left|\frac{x_l^{k+1} - x_l^k}{d_l^U}\right|\right\},\ l = 1, \ldots, n$$

and $\delta^{\max} = \max_{1 \leqslant l \leqslant n}\{\delta_l\}$. Here $\delta^{\max}$ measures the step between two NLP iterates relative to the lower and upper bounds. Note that $\delta^{\max}$ may be larger than one, since multiple LP subiterations are performed in an NLP iteration.

Let further $0 < \delta^{\text{dec}} < \frac{1}{2}$ and $\frac{1}{2} < \delta^{\text{inc}} < 1$ be given tolerances for how small and large a step may be without decreasing or increasing the trust region bounds.

Then, if $\delta^{\max} < \delta^{\text{dec}}$, the trust region bounds are decreased,

$$d^U := \frac{\delta^{\max}}{\delta^{\text{dec}}}\, d^U \quad \text{and } d^L := \frac{\delta^{\max}}{\delta^{\text{dec}}}\, d^L,$$

and if $\delta^{\max} > \delta^{\text{inc}}$, the trust region bounds are increased,

$$d^U := 2\delta^{\max}\, d^U \quad \text{and } d^L := 2\delta^{\max}\, d^L.$$

2.4. NLP iteration termination criteria

The NLP iterations are continued until the current iterate is a stationary point. Here it has been assumed that the problems always have stationary points. Generally, the algorithm may stop at points where the constraint infeasibility cannot be reduced, i.e. the relaxation variables t^g, t^{h^+} and t^{h^-} are close to the upper bounds of these variables for the first LP subproblem in an NLP iteration. In that case it must be assumed that the original NLP problem is infeasible, although it may be that the algorithm has converged to a locally infeasible point.

The current iterate x^k is a stationary point if it satisfies the first-order Karush–Kuhn–Tucker conditions, i.e.

(1) $g_j(x^k) \leqslant 0,\ j = 1, \ldots, m_i$.
(2) $h_r(x^k) = 0,\ r = 1, \ldots, m_e$.
(3) $\lambda_j g_j(x^k) = 0,\ j = 1, \ldots, m_i$.
(4) $\lambda_j \geqslant 0,\ j = 1, \ldots, m_i$.
(5) $\nabla f(x^k) + \sum_{j=1}^{m_i}\lambda_j \nabla g_j(x^k) + \sum_{r=1}^{m_e}\mu_r \nabla h_r(x^k) = \mathbf{0}$.

These criteria are easy to evaluate using the approximations of λ and μ, which are obtained from the dual problem of the LP subproblem. Note that (4) is always satisfied as the approximation of λ is based on the dual problem of the LP subproblem, which contains the constraint $\lambda \geqslant \mathbf{0}$.

Note also that the algorithm may run into problems in cases where the stationary point is not a Karush–Kuhn–Tucker stationary point as the criteria above will not hold in such a point. In these cases, however, initial numerical experience shows that the algorithm may still find a Karush–Kuhn–Tucker stationary point within the accepted solution tolerance in a neighbourhood of the true solution.

2.5. SCP algorithm

The SCP algorithm is summarized below.

```
1. Initialize: $H^1 = I$, $k = 1$, $x^1$ = initial starting point.
2. Do LP subiterations.
   2.1 Initialize: $i = 1$, $x^{(1)} = x^k$, $H^{(1)} = H^k$.
   2.2 Generate $LP(x^{(i)})$ and solve it to obtain search direction
       $d^{(i)}$ and Lagrange multiplier estimates $\lambda^{(i)}$ and $\mu^{(i)}$ (dual
       optimal solution).
   2.3 Check whether current iterate is a stationary point (Section 2.4).
   2.4 Perform line search to minimize $\widetilde{L}^{(i)}(x^{(i)} + \alpha d^{(i)})$ (Section
       2.1.3). Let $x^{(i+1)} = x^{(i)} + \alpha^{(i)} d^{(i)}$.
   2.5 Update the estimate of the Hessian of the Lagrangian
       using the BFGS update formula and call it $H^{(i+1)}$ (Section
       2.1.4).
   2.6 If some subiteration termination criterion satisfied (Section 2.1.5)
       Then exit LP subiterations (go to 3).
       Else let $i := i + 1$ and go to 2.2.
3. Store the current iterate and Hessian estimate from the LP sub-
   iterations, i.e. $x^{k+1} = x^{(i)}$ and $H^{k+1} = H^{(i)}$.
4. If not sufficient decrease for merit function (Section 2.2.1)
      Then find a new iterate $x^{k+1}$ with sufficient decrease (Section 2.2.2).
5. If trust region too small or too large
      Then update trust region (Section 2.3).
6. If iterate not a stationary point (Section 2.4)
      Then let $k := k + 1$ and start new NLP iteration (go to 2).
```

3. Convergence

In this section it is shown that the algorithm has global convergence properties. The section is organized as follows: in Theorem 2 it is shown that a constant C may be found such that the constraint infeasibility is reduced in the first LP subiteration.

In Theorem 6 it is shown that the algorithm will, in the first LP subiteration, generate directions that are descent directions for the merit function M when solving LP problem $LP(x^k)$.

Finally, in Theorem 8 it is shown that any limit point of an infinite sequence of iterates is a KKT stationary limit point. Note that Theorem 6 already states that if the solution to the LP problem solved in the first subiteration of an NLP iteration is not a descent direction for the merit function, then the current iterate is a KKT stationary point for (NLP).

First it is shown that it is possible in the first LP subiteration to find constants C for any $x^k \in R^n$ such that the constraint infeasibility is reduced for the linearized problem $LP(x^k)$. The proof needs the following lemma.

Lemma 1. *Take any x^k such that EMFCQ holds. Then there is a d such that*

$$\nabla g_j(x^k)^T d \leqslant -g_j(x^k),\ j \in J_+(x^k) \cup J_0(x^k),$$
$$\nabla h_r(x^k)^T d = -h_r(x^k),\ r = 1,\dots,m_e.$$

Proof. Let $b_r = -h_r(x^k)/\|g(x^k)^+\|_\infty$, where $\|g(x^k)^+\|_\infty$ is the maximum constraint violation for the inequality constraints. Replace $\|g(x^k)^+\|_\infty$ with one if there are no inequality constraints or none of the inequality constraints are violated. A slight generalization of Theorem 2.2 in [9] can be used to see that there is a $\tilde{d}$ such that

$$\nabla g_j(x^k)^T \tilde{d} \leqslant -1,\ j \in J_+(x^k) \cup J_0(x^k),$$
$$\nabla h_r(x^k)^T \tilde{d} = b_r,\ r = 1,\dots,m_e.$$

Let $d = \tilde{d}\ \|g(x^k)^+\|_\infty$. Then

$$\nabla g_j(x^k)^T d \leqslant -\|g(x^k)^+\|_\infty \leqslant -g_j(x^k),\ j \in J_+(x^k) \cup J_0(x^k)$$

and $\nabla h_r(x^k)^T d = -h_r(x^k),\ r = 1,\dots,m_e.$ □

The following Theorem shows that there is a big enough C such that the optimal solution of $LP(x^k)$ reduces the constraint infeasibility for $LP(x^k)$.

Theorem 2. *For any $x^k \in R^n$ there is a big enough C such that the constraint infeasibility is reduced for at least one of the infeasible constraints, i.e. there is either a $j \in \{1,\dots,m_i\}$ such that $t_j^g < g_j(x^k)$ or there is an $r \in \{1,\dots,m_e\}$ such that $t_r^{h^+} < |h_r(x^k)|$ and $t_r^{h^-} < |h_r(x^k)|$ for the optimal solution to $LP(x^k)$.*

Proof. By Lemma 1 there is a $\tilde{d}$ such that

$$\nabla g_j(x^k)^T \tilde{d} \leqslant -g_j(x^k),\ j \in J_+(x^k) \cup J_0(x^k),$$
$$\nabla h_r(x^k)^T \tilde{d} = -h_r(x^k),\ r = 1,\dots,m_e.$$

It is possible to find an $\tilde{\alpha}$ such that $0 < \tilde{\alpha} < 1$, $d^L \leqslant \tilde{\alpha}\tilde{d} \leqslant d^U$ and $\nabla g_j(x^k)^T(\tilde{\alpha}\tilde{d}) \leqslant -g_j(x^k),\ j \in \{j : g_j(x^k) < 0\}$. Choose $(\hat{d}, \hat{t}^g, \hat{t}^{h^+}, \hat{t}^{h^-})$ such that

$$(\hat{d}, \hat{t}^g, \hat{t}^{h^+}, \hat{t}^{h^-}) = (\tilde{\alpha}\tilde{d},\ (1-\tilde{\alpha})\max\{\mathbf{0}, g(x^k)\},$$
$$(1-\tilde{\alpha})\max\{\mathbf{0}, h(x^k)\},\ (1-\tilde{\alpha})\max\{\mathbf{0}, -h(x^k)\}).$$

Then $(\hat{d}, \hat{t}^g, \hat{t}^{h^+}, \hat{t}^{h^-})$ is a feasible solution for $LP(x^k)$. Choose $\widehat{C}$ such that

$$-\tilde{\alpha}\widehat{C}\sum_{j=1}^{m_i}\max\{0, g_j(x^k)\} - \tilde{\alpha}\widehat{C}\sum_{r=1}^{m_e}\max\{0, h_r(x^k)\}$$
$$-\tilde{\alpha}\widehat{C}\sum_{r=1}^{m_e}\max\{0, -h_r(x^k)\} < \min_{d^L \leqslant d \leqslant d^U}\{\nabla f(x^k)^T d\} - \nabla f(x^k)^T \hat{d}.$$

Looking at the objective function value in the point $(\hat{d}, \hat{t}^g, \hat{t}^{h^+}, \hat{t}^{h^-})$ and letting $C = \widehat{C}$, it can be seen that

$$\nabla f(x^k)^T\hat{d} + \widehat{C}\sum_{j=1}^{m_i}\hat{t}_j^g + \widehat{C}\sum_{r=1}^{m_e}\hat{t}_r^{h^+} + \widehat{C}\sum_{r=1}^{m_e}\hat{t}_r^{h^-}$$
$$= \nabla f(x^k)^T\hat{d} + (1-\tilde{\alpha})\widehat{C}\sum_{j=1}^{m_i}\max\{0, g_j(x^k)\}$$
$$+ (1-\tilde{\alpha})\widehat{C}\sum_{r=1}^{m_e}\max\{0, h_r(x^k)\} + (1-\tilde{\alpha})\widehat{C}\sum_{r=1}^{m_e}\max\{0, -h_r(x^k)\}$$
$$< \min_{d^L \leqslant d \leqslant d^U}\{\nabla f(x^k)^T d\} + C\sum_{j=1}^{m_i}\max\{0, g_j(x^k)\}$$
$$+ C\sum_{r=1}^{m_e}\max\{0, h_r(x^k)\} + C\sum_{r=1}^{m_e}\max\{0, -h_r(x^k)\},$$

from the way the constant $\widehat{C}$ was chosen. Thus, the solution $(\hat{d}, \hat{t}^g, \hat{t}^{h^+}, \hat{t}^{h^-})$ is better than any solution $(d, t^g, t^{h^+}, t^{h^-})$ to $LP(x^k)$ where $t^g = \max\{\mathbf{0}, g(x^k)\}$ and $\max\{t^{h^+}, t^{h^-}\}$ is equal to $|h(x^k)|$. Hence the constraint infeasibility is reduced for at least one of the infeasible constraints. □

Next consider the direction d^k found in the first LP problem solved in each NLP iteration, i.e. $d^k = d^{(1)}$. It can be shown that this search direction is a descent direction for the merit function M.

The direction d^k is the solution to the LP problem (1) generated in the current point x^k in the first LP subiteration, i.e. the solution to the problem $LP(x^k)$:

$$\min \quad \nabla f(x^k)^T d + C\sum_{j=1}^{m_i} t_j^g + C\sum_{r=1}^{m_e} t_r^{h^+} + C\sum_{r=1}^{m_e} t_r^{h^-},$$

$$\text{s.t.} \quad \nabla g_j(x^k)^T d - t_j^g \leqslant -g_j(x^k),\ j = 1,\dots,m_i, \tag{7a}$$
$$\nabla h_r(x^k)^T d - t_r^{h^+} + t_r^{h^-} = -h_r(x^k),\ r = 1,\dots,m_e, \tag{7b}$$
$$d - d^U \leqslant \mathbf{0}, \tag{7c}$$
$$d^L - d \leqslant \mathbf{0}, \tag{7d}$$
$$0 \leqslant t_j^g \leqslant \max\{g_j(x^k), 0\},\ j = 1,\dots,m_i, \tag{7e}$$
$$0 \leqslant t_r^{h^+} \leqslant |h_r(x^k)|,\ r = 1,\dots,m_e, \tag{7f}$$
$$0 \leqslant t_r^{h^-} \leqslant |h_r(x^k)|,\ r = 1,\dots,m_e. \tag{7g}$$

In order to prove that the directions d^k are descent directions a few lemmas are needed. The first lemma states that the current iterate is a stationary point to the original problem (NLP) if the optimal value for the linearized problem is zero and the current iterate x^k is feasible.

Lemma 3. *Assume the current iterate x^k is feasible in (NLP). Assume further that the optimal value of (7), i.e. $LP(x^k)$, is zero for an optimal solution to $LP(x^k)$. Then x^k is a KKT stationary point for (NLP).*

Proof. Assume the optimal solution is d^k. Assume further that $d^k \neq \mathbf{0}$. Since x^k is feasible in (NLP), it also holds that $\bar{d} = \mathbf{0}$ is a feasible solution to $LP(x^k)$. The optimal value of $LP(x^k)$ is zero by assumption and thus $\bar{d} = \mathbf{0}$ is an optimal solution to $LP(x^k)$ as well. Furthermore, $\bar{d} = \mathbf{0}$ is an optimal solution to the problem

$$\begin{aligned} \min \quad & \nabla f(x^k)^T d, \\ \text{s.t.} \quad & \nabla g_j(x^k)^T d \leqslant -g_j(x^k),\ j = 1,\dots,m_i, \\ & \nabla h_r(x^k)^T d \leqslant -h_r(x^k),\ r = 1,\dots,m_e, \\ & -\nabla h_r(x^k)^T d \leqslant h_r(x^k),\ r = 1,\dots,m_e, \end{aligned} \tag{8}$$

where the relaxing variables t^g, t^{h^+} and t^{h^-} as well as the simple bounds on d have been dropped and each equality constraint has been divided into two inequality constraints. Note here that the relaxing variables t^g, t^{h^+} and t^{h^-} are all zero since x^k is feasible and can, therefore, be dropped.

Note also that if there would be an optimal solution $\tilde{d}$ to (8) such that $\nabla f(x^k)^T\tilde{d} < 0$, then it would be possible to find a constant $\tilde{\alpha}$ such that $d^L \leqslant \tilde{\alpha}\tilde{d} \leqslant d^U$ and $\nabla f(x^k)^T(\tilde{\alpha}\tilde{d}) < 0$. Since x^k is feasible in (*NLP*), $\tilde{\alpha}\tilde{d}$ will also satisfy the constraints in $LP(x^k)$. Thus, $d = \tilde{\alpha}\tilde{d}$ would be a solution to $LP(x^k)$. This contradicts the fact that $\bar{d}$ is an optimal solution to $LP(x^k)$ and $\nabla f(x^k)^T\bar{d} = 0$.

The dual problem of (8) is

$$\begin{aligned} \max \quad & \sum_{j=1}^{m_i} \lambda_j g_j(x^k) + \sum_{r=1}^{m_e} \mu_r^+ h_r(x^k) - \sum_{r=1}^{m_e} \mu_r^- h_r(x^k), \\ \text{s.t.} \quad & \sum_{j=1}^{m_i} \lambda_j \nabla g_j(x^k) + \sum_{r=1}^{m_e} \mu_r^+ \nabla h_r(x^k) - \sum_{r=1}^{m_e} \mu_r^- \nabla h_r(x^k) = -\nabla f(x^k), \\ & \lambda \geqslant \mathbf{0},\ \mu^+ \geqslant \mathbf{0},\ \mu^- \geqslant \mathbf{0}, \end{aligned} \tag{9}$$

where λ, μ^+ and μ^- are the dual variables corresponding to the constraints of (8). See [12], for instance, for more information on linear duality.

Assume $\lambda^k, \mu^{+,k}, \mu^{-,k}$ is an optimal solution to (9). By the duality theorem, the optimal value of the primal and dual problems are equal and thus,

$$\sum_{j=1}^{m_i} \lambda_j^k g_j(x^k) = 0 \tag{10}$$

since $h(x^k) = \mathbf{0}$ by assumption.

It was also assumed that $g_j(x^k) \leqslant 0,\ j = 1,\dots,m_i$ and, from the constraints in (9), it holds that $\lambda_j^k \geqslant 0,\ j = 1,\dots,m_i$. Thus,

$$\lambda_j^k g_j(x^k) \leqslant 0,\ j = 1,\dots,m_i. \tag{11}$$

From (10) and (11) one then gets

$$\lambda_j^k g_j(x^k) = 0,\ j = 1,\dots,m_i.$$

Let $\mu^k = \mu^{+,k} - \mu^{-,k}$. Then, from the constraints in (9),

$$\nabla f(x^k) + \sum_{j=1}^{m_i} \lambda_j^k \nabla g_j(x^k) + \sum_{r=1}^{m_e} \mu_r^k \nabla h_r(x^k) = \mathbf{0}$$

and

$$\lambda^k \geqslant \mathbf{0}.$$

Thus, the current iterate x^k satisfies the first-order Karush–Kuhn–Tucker conditions for (*NLP*) and x^k is a KKT stationary point for (*NLP*). □

The next lemma provides a practical upper bound for the directional derivative of the penalty term in the merit function.

Lemma 4. *Let $(d^k, t^{g,k}, t^{h^+,k}, t^{h^-,k})$ be an optimal solution to $LP(x^k)$, i.e. (7), with dual variables $\lambda^{g,k}$, μ^k, $\lambda^{U,k}$ and $\lambda^{L,k}$ for the constraints 7a,7b,7c,7d, respectively. Assume further that x^k is infeasible in (NLP), $\bar{\lambda}_j > \lambda_j^{g,k}$, $j = 1,\dots,m_i$ and $\bar{\mu}_r > |\mu_r^k|$, $r = 1,\dots,m_e$. Then*

$$D_{d^k}\, p(x^k) < \sum_{j=1}^{m_i} \lambda_j^{g,k} \nabla g_j(x^k)^T d^k + \sum_{r=1}^{m_e} \mu_r^k \nabla h_r(x^k)^T d^k.$$

Proof. By applying the definition of the directional derivatives, one gets

$$\begin{aligned} D_{d^k}\, p(x^k) &= \sum_{j=1}^{m_i} \bar{\lambda}_j D_{d^k} g_j(x^k)^+ + \sum_{r=1}^{m_e} \bar{\mu}_r D_{d^k} |h_r(x^k)| \\ &= \sum_{j: g_j(x^k)>0} \bar{\lambda}_j \nabla g_j(x^k)^T d^k + \sum_{j: g_j(x^k)=0} \bar{\lambda}_j (\nabla g_j(x^k)^T d^k)^+ \\ &\quad + \sum_{r: h_r(x^k)>0} \bar{\mu}_r \nabla h_r(x^k)^T d^k - \sum_{r: h_r(x^k)<0} \bar{\mu}_r \nabla h_r(x^k)^T d^k \\ &\quad + \sum_{r: h_r(x^k)=0} \bar{\mu}_r |\nabla h_r(x^k)^T d^k|. \end{aligned} \tag{12}$$

Since $t_j^{g,k} \leqslant \max\{g_j(x^k), 0\}$, $j = 1,\dots,m_i$, and from (7a), it holds that $\nabla g_j(x^k)^T d^k \leqslant 0$ whenever $g_j(x^k) > 0$ and thus

$$\bar{\lambda}_j \nabla g_j(x^k)^T d^k \leqslant \lambda_j^{g,k} \nabla g_j(x^k)^T d^k \tag{13}$$

for any j such that $g_j(x^k) > 0$. Furthermore, for any $j = 1,\dots,m_i$ such that $g_j(x^k) = 0$, it holds that $\nabla g_j(x^k)^T d^k \leqslant 0$ and thus

$$\bar{\lambda}_j (\nabla g_j(x^k)^T d^k)^+ = 0.$$

Also note that whenever $h_r(x^k) < 0$, $r = 1,\dots,m_e$, the constraints (7b) and (7g) ensure that $\nabla h_r(x^k)^T d^k \geqslant 0$ and thus

$$-\bar{\mu}_r \nabla h_r(x^k)^T d^k \leqslant \mu_r^k \nabla h_r(x^k)^T d^k. \tag{14}$$

Applying a similar reasoning to the rest of the terms in (12) one obtains

$$\bar{\mu}_r \nabla h_r(x^k)^T d^k \leqslant \mu_r^k \nabla h_r(x^k)^T d^k \tag{15}$$

whenever $h_r(x^k) > 0$, $r = 1,\dots,m_e$, and

$$\bar{\mu}_r |\nabla h_r(x^k)^T d^k| = 0$$

when $h_r(x^k) = 0$. Combining the inequalities and equalities above, one obtains

$$\begin{aligned} D_{d^k}\, p(x^k) \leqslant & \sum_{j: g_j(x^k)>0} \lambda_j^{g,k} \nabla g_j(x^k)^T d^k + \sum_{r: h_r(x^k)>0} \mu_r^k \nabla h_r(x^k)^T d^k \\ & + \sum_{r: h_r(x^k)<0} \mu_r^k \nabla h_r(x^k)^T d^k. \end{aligned}$$

As $\lambda_j^{g,k} > 0$ only if the corresponding constraint is active, it holds that

$$\nabla g_j(x^k)^T d^k \geqslant 0$$

for all j where $g_j(x^k) \leqslant 0$ and $\lambda_j^{g,k} > 0$. Also, $\nabla h_r(x^k)^T d^k = 0$ when $h_r(x^k) = 0$. Finally, from Theorem 2, it is known that strict inequality holds for at least one of the inequality constraints (13)–(15) and thus

$$D_{d^k}\, p(x^k) < \sum_{j=1}^{m_i} \lambda_j^{g,k} \nabla g_j(x^k)^T d^k + \sum_{r=1}^{m_e} \mu_r^k \nabla h_r(x^k)^T d^k,$$

which is the desired result. □

The following lemma is also needed.

Lemma 5. *Let $(d^k, t^{g,k}, t^{h^+,k}, t^{h^-,k})$ be an optimal solution to (7) with dual variables $\lambda^{g,k}$, μ^k, $\lambda^{U,k}$ and $\lambda^{L,k}$ for the constraints 7a,7b,7c,7d respectively. Assume further that x^k is feasible in (NLP). Then*

$$D_{d^k}\, p(x^k) = 0.$$

Proof. From (12) it can be seen that the directional derivative will be

$$D_{d^k}\, p(x^k) = \sum_{j: g_j(x^k)=0} \bar{\lambda}_j (\nabla g_j(x^k)^T d^k)^+ + \sum_{r: h_r(x^k)=0} \bar{\mu}_r |\nabla h_r(x^k)^T d^k| \tag{16}$$

whenever x^k is feasible. As noted in the proof of Lemma 4,

$$\bar{\lambda}_j (\nabla g_j(x^k)^T d^k)^+ = 0 \tag{17}$$

when $g_j(x^k) = 0$ and

$$\bar{\mu}_r |\nabla h_r(x^k)^T d^k| = 0 \tag{18}$$

when $h_r(x^k) = 0$. By combining (16)–(18) one gets $D_{d^k}\, p(x^k) = 0$. □

It can now be proven that the search directions generated by the SCP algorithm are descent directions for the merit function M.

Theorem 6. *Let $(d^k, t^{g,k}, t^{h^+,k}, t^{h^-,k})$ be an optimal solution to (7) with dual variables $\lambda^{g,k}$, μ^k, $\lambda^{U,k}$ and $\lambda^{L,k}$ for the constraints 7a,7b,7c,7d* *respectively. Assume further that* $\bar{\lambda}_j > \lambda_j^{g,k}$, $j = 1, \dots, m_i$ *and* $\bar{\mu}_r > |\mu_r^k|$, $r = 1, \dots, m_e$. *Then* $D_{d^k}\, M(x^k) < 0$ *or* x^k *is a KKT stationary point for* (NLP).

Proof. Assume first that x^k is infeasible in (NLP). Using Lemma 4 one gets

$$D_{d^k}\, M(x^k) = \nabla f(x^k)^T d^k + D_{d^k}\, p(x^k) < \nabla f(x^k)^T d^k + \sum_{j=1}^{m_i} \lambda_j^{g,k} \nabla g_j(x^k)^T d^k + \sum_{r=1}^{m_e} \mu_r^k \nabla h_r(x^k)^T d^k.$$

Utilizing the fact that for the dual variables of the linear problem (7) it holds that

$$\nabla f(x^k) + \sum_{j=1}^{m_i} \lambda_j^{g,k} \nabla g_j(x^k) + \sum_{r=1}^{m_e} \mu_r^k \nabla h_r(x^k) + \lambda^{U,k} - \lambda^{L,k} = \mathbf{0},$$

one obtains

$$D_{d^k}\, M(x^k) < (\lambda^{L,k} - \lambda^{U,k})^T d^k \leqslant 0,$$

whenever x^k is infeasible.

The last inequality is due to the fact that for $l = 1, \dots, n$, $\lambda_l^{L,k} > 0$ only if d_l^k is at the lower bound, i.e. $d_l^k < 0$ and $\lambda_l^{U,k} > 0$ only if d_l^k is at the upper bound, i.e. $d_l^k > 0$.

It remains to look at the cases when x^k is feasible. From Lemma 5, one has that $D_{d^k}\, p(x^k) = 0$ and thus

$$D_{d^k}\, M(x^k) = \nabla f(x^k)^T d^k.$$

If $\nabla f(x^k)^T d^k < 0$, then $D_{d^k}\, M(x^k) < 0$.

Finally, if $\nabla f(x^k)^T d^k = 0$, then Lemma 3 shows that x^k is a KKT stationary point for (NLP).

Note that $\nabla f(x^k)^T d^k > 0$ is clearly not possible since $d = \mathbf{0}$ is a feasible solution to (7), whenever x^k is feasible. □

The next lemma states that any limit point $\bar{x}$ with corresponding search direction $\bar{d}$ of an infinite sequence of points $\{x^k\}$ will have a zero directional derivative in direction $\bar{d}$. The proof is similar to the one for the inequality constrained case in [19]. Since different termination criteria was used in [19] based on the Goldstein test for differentiable functions, the proof is provided here as well. For more information on the Goldstein test see, for instance, [14].

Lemma 7. *Assume the SCP algorithm generates an infinite sequence of points* $\{x^k\}$ *with corresponding search directions* d^k *and line search solutions* α^k. *Here* d^k *is the optimal solution to the first LP subproblem in the NLP iteration starting from* x^k, *i.e.* $d^k = d^{(1)}$. *The variable* α^k *is the corresponding solution to the line search in direction* d^k *starting from the point* x^k. *Then, for any limit point* $\bar{x}$ *of the sequence with corresponding search direction* $\bar{d}$, *it holds that*

$$D_{\bar{d}}\, M(\bar{x}) = 0.$$

Proof. By the way d^k was chosen, it is known from Theorem 6 that $D_{d^k}\, M(x^k) \leqslant 0$. Assume there exists a limit point $\bar{x}$ with corresponding search direction $\bar{d}$ such that $D_{\bar{d}}\, M(\bar{x}) < 0$. Since $\{M(x^k)\}$ is monotonically decreasing and M continuous, $\{M(x^k)\}$ converges to $M(\bar{x})$. Thus,

$$M(x^k) - M(x^{k+1}) \to 0$$

when $k \to \infty$. From (4) and (5), one has that $M(x^{k+1}) \leqslant M(x^k + \alpha^k d^k) \leqslant M(x^k)$ and thus

$$M(x^k) - M(x^k + \alpha^k d^k) \to 0,$$

when $k \to \infty$. From (5) and $D_{d^k}\, M(x^k) \leqslant 0$ one further has

$$M(x^k) - M(x^k + \alpha^k d^k) \geqslant -\sigma \alpha^k D_{d^k}\, M(x^k) \geqslant 0,$$

and thus $\alpha^k D_{d^k}\, M(x^k) \to 0$ when $k \to \infty$. Since it was assumed that $D_{\bar{d}}\, M(\bar{x}) < 0$ and $\alpha^k \geqslant 0$, it further holds that $\alpha^k \to 0^+$ when $k \to \infty$. As $g_j, j = 1, \dots, m_i$ and $h_r, r = 1, \dots, m_e$ are continuously differentiable,

$$D_{d^k}\, M(x^k + \alpha^k d^k) \to D_{d^k}\, M(x^k),$$

when $\alpha^k \to 0^+$. This contradicts (6) and thus $D_{\bar{d}}\, M(\bar{x}) = 0$ for any limit point $\bar{x}$. □

It is now possible to show that any limit point of the infinite sequence $\{x^k\}$ is a KKT stationary limit point for (NLP).

Theorem 8. *Assume the SCP algorithm generates an infinite sequence of points* $\{x^k\}$. *Then any limit point of this sequence is a KKT stationary limit point for* (NLP).

Proof. From Lemma 7 it is known that for any limit point $\bar{x}$, with search direction $\bar{d}$, it holds that $D_{\bar{d}}\, M(\bar{x}) = 0$. Furthermore, from the proof of Theorem 6, it is known that $D_{\bar{d}}\, M(\bar{x}) < 0$ whenever $\bar{x}$ is infeasible. Thus, the limit point must be feasible. Furthermore, as the limit point is feasible, one has from Lemma 5 that $D_{\bar{d}}\, p(\bar{x}) = 0$ and thus $D_{\bar{d}}\, M(\bar{x}) = \nabla f(\bar{x})^T \bar{d} = 0$. Consequently, the optimal value of $LP(\bar{x})$ is zero and one may use Lemma 3 to see that the limit point is a KKT stationary point for (NLP). □

4. Numerical experience

In order to get experience of the performance of the algorithm, numerical experiments were performed on known test sets. For the experiments, Hock and Schittkowski [10] and Schittkowski [17] test sets were chosen. Model formulations in AMPL publicly available on the home pages of professor Robert Vanderbei, see [22], were used. The Hock and Schittkowski test set contained 115 compiled problems and the Schittkowski test set 188 problems. Thus, the numerical experiments covered a total of 303 problems.

Two well-known NLP solvers, LANCELOT [3] version 20010227 and DONLP2 [18] version 20020506 were compared to the SCP algorithm. The LANCELOT and DONLP2 solvers were chosen as they are based on different, but comparable, concepts to the SCP algorithm. Both solvers use first-order information from the problems, thus enabling a comparison of algorithm performance based on the number of gradient evaluations. Also, source codes were available for the algorithms, which made it easy to integrate them into the test environment.

Results are visualized using performance profiles as the number of problems is quite large.

4.1. LANCELOT

LANCELOT is a trust region-based algorithm used for solving equality constrained problems. Therefore, inequality constraints must be converted to equality constraints using slack variables.

In each iteration of the algorithm, the augmented Lagrangian function

$$L(x, \mu, \rho, S) = f(x) + \sum_{r=1}^{m_e} \mu_r h_r(x) + \frac{1}{2\rho} \sum_{r=1}^{m_e} s_{rr} (h_r(x))^2$$

is approximately minimized subject to simple bounds with a trust region-based subsolver. Here μ are Lagrange multiplier estimates, ρ is a penalty parameter and the entries s_{rr} of the diagonal matrix S are positive scaling factors. Then, either the penalty parameter is updated or a new Lagrange multiplier estimate μ is calculated and the constraint scaling factors s_{rr} are updated. The process is repeated until the current iterate is considered a stationary point. The iterate is considered to be a stationary point when the iterate is sufficiently feasible with respect to the original problem and the projected Lagrangian gradient is sufficiently small.

4.2. DONLP2

DONLP2 is an SQP method. In each iteration, the quadratic programming (QP) problem

$$\begin{aligned} \min \quad & \nabla f(x^k)^T d + \tfrac{1}{2} d^T H^k d, \\ \text{s.t.} \quad & g_j(x^k) + \nabla g_j(x^k)^T d \leqslant 0,\ j = 1, \ldots, m_i, \\ & h_r(x^k) + \nabla h_r(x^k)^T d = 0,\ r = 1, \ldots, m_e, \end{aligned} \tag{19}$$

is solved. The matrix H^k is an estimate of the Hessian of the augmented Lagrangian function for (NLP).

A step $x^{k+1} = x^k + \alpha^k d^k$ is then taken in the direction d^k obtained as a solution to (19). The algorithm ensures that the computed step reduces a merit function sufficiently. The merit function used is

$$M(x) = f(x) + \sum_{j=1}^{m_i} \bar{\lambda}_j g_j(x)^+ + \sum_{r=1}^{m_e} \bar{\mu}_r |h_r(x)|.$$

Table 1
Number of successfully solved test problems (tolerance $\epsilon = 10^{-3}$).

	Hock and Schittkowski (Total: 115)	Schittkowski (Total: 188)
SCP	108	176
DONLP2	103	168
LANCELOT	95	172

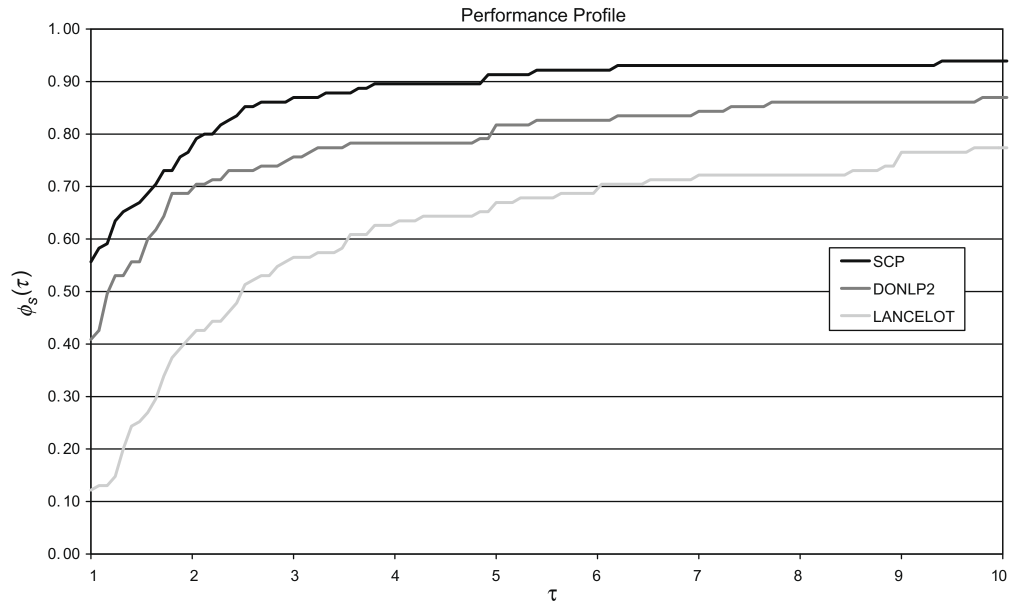

Fig. 1. Test results for the Hock and Schittkowski test set ($\epsilon = 10^{-3}$).

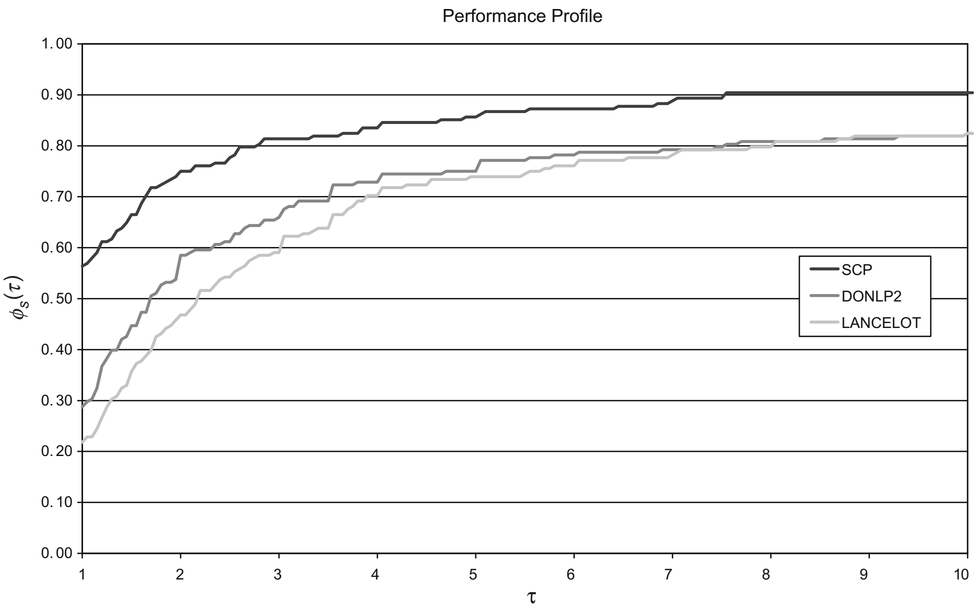

Fig. 2. Test results for the Schittkowski test set ($\epsilon = 10^{-3}$).

The algorithm terminates when the current iterate satisfies the Karush–Kuhn–Tucker conditions for the problem or no significant progress can be made. Note the similarities with the proposed SCP algorithm. The main difference to the proposed algorithm is that the subproblem solved is a quadratic programming problem with linear constraints rather than a linear programming subproblem. Surprisingly, as can be seen in the numerical results, the SCP algorithm required fewer gradient evaluations than DONLP2 for several test problems. Consequently, the number of LP problems solved in the SCP algorithm was fewer than the number of QP problems solved in the DONLP2 algorithm for many of the test problems.

4.3. Comparing solver performance

The number of times the gradients were evaluated was used as the metric for comparing the algorithms. Providing a fair comparison of these algorithms is a nontrivial task as they are quite different. However, the number of times the gradients are evaluated is roughly equal to the number of subproblems solved, being an LP problem for the SCP algorithm, a linearly constrained QP problem for DONLP2 and a simple bound constrained NLP problem for LANCELOT. The amount of work to solve each of these types of subproblems was assumed to be roughly equal, in which case the number of gradient evaluations provides a fair picture of algorithm performance.

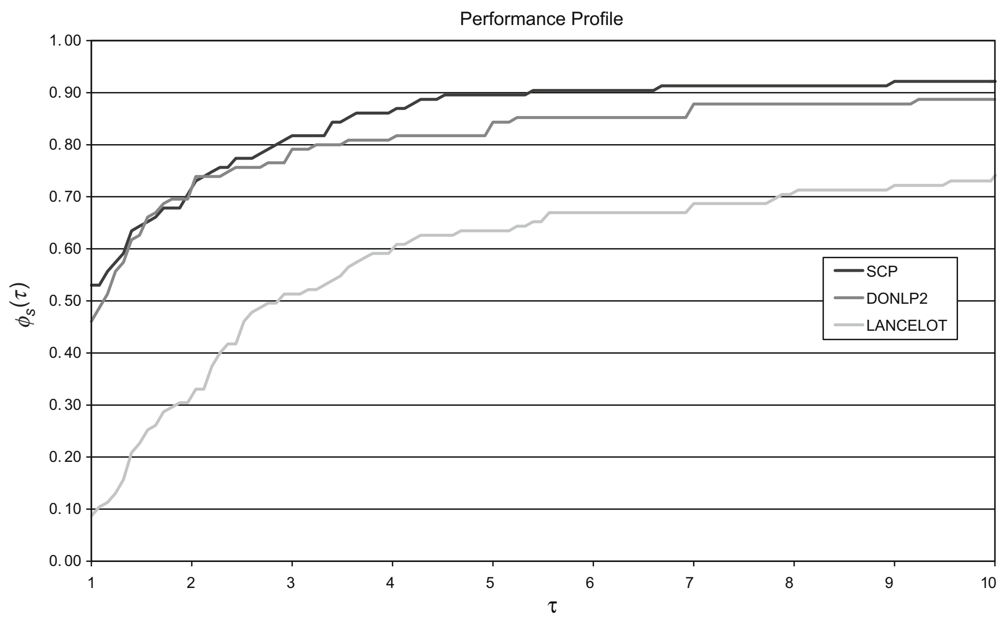

Fig. 3. Test results for the Hock and Schittkowski test set ($\epsilon = 10^{-6}$).

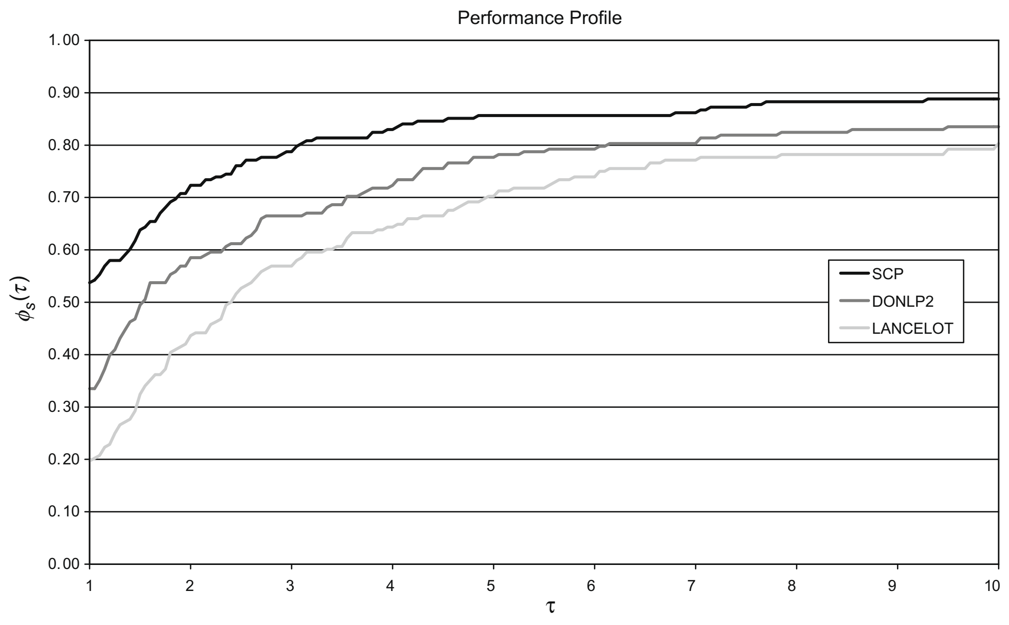

Fig. 4. Test results for the Schittkowski test set ($\epsilon = 10^{-6}$).

Note that comparing CPU times on the same platform would neither be an exact comparison of algorithm performance as memory handling, choice of programming language and other coding related issues affect the CPU time quite significantly.

4.4. Performance profiles

The results from the tests are visualized using performance profiles, a concept introduced by Dolan and Moré in [5]. Performance profiles are constructed by computing an estimate of the probability that an algorithm performs within a multiple of the chosen metric, which may, for instance, be the running time, iteration count or the number of gradient evaluations. Here the number of gradient evaluations was used as the criterion for computing the performance profile.

To illustrate performance profiles, assume the performance of n_s solvers on n_p problems is compared. Let $g_{p,s}$ be the number of gradient evaluations for solver s to solve problem p. The performance ratio $r_{p,s}$ for solver s on problem p is defined to be

$$r_{p,s} = \frac{g_{p,s}}{\min_{1\leqslant s\leqslant n_s}\{g_{p,s}\}}.$$

If solver s could not solve problem p, let $r_{p,s} = C$, where C is a big number.

Table 2
Number of successfully solved test problems for SLP version (tolerance $\epsilon = 10^{-3}$).

	Hock and Schittkowski (Total: 115)	Schittkowski (Total: 188)
SCP	108	176
SCP(SLP)	98	122

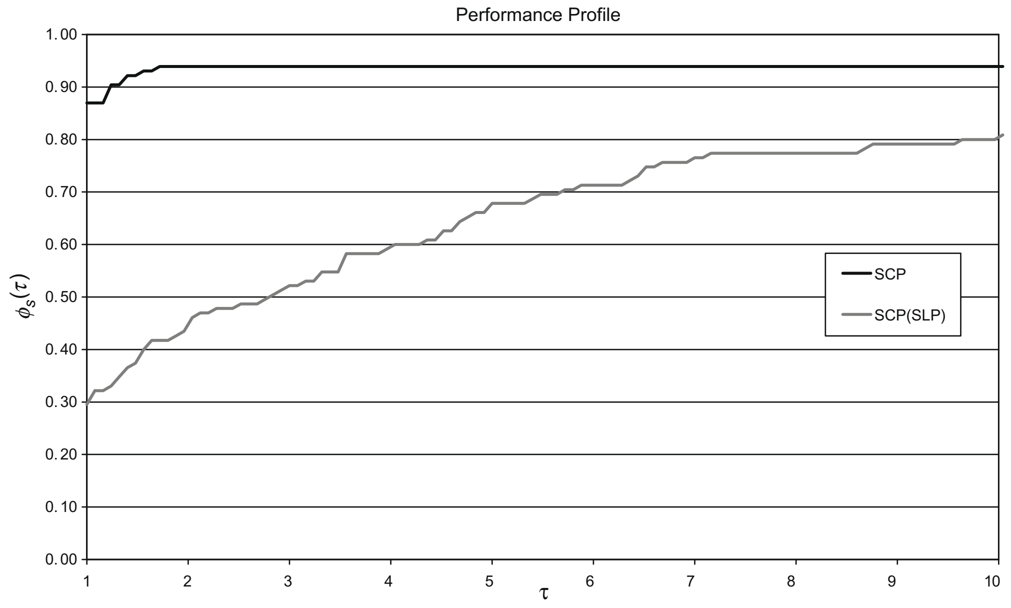

Fig. 5. Comparison of SCP algorithm with version without conjugate directions on Hock and Schittkowski collection.

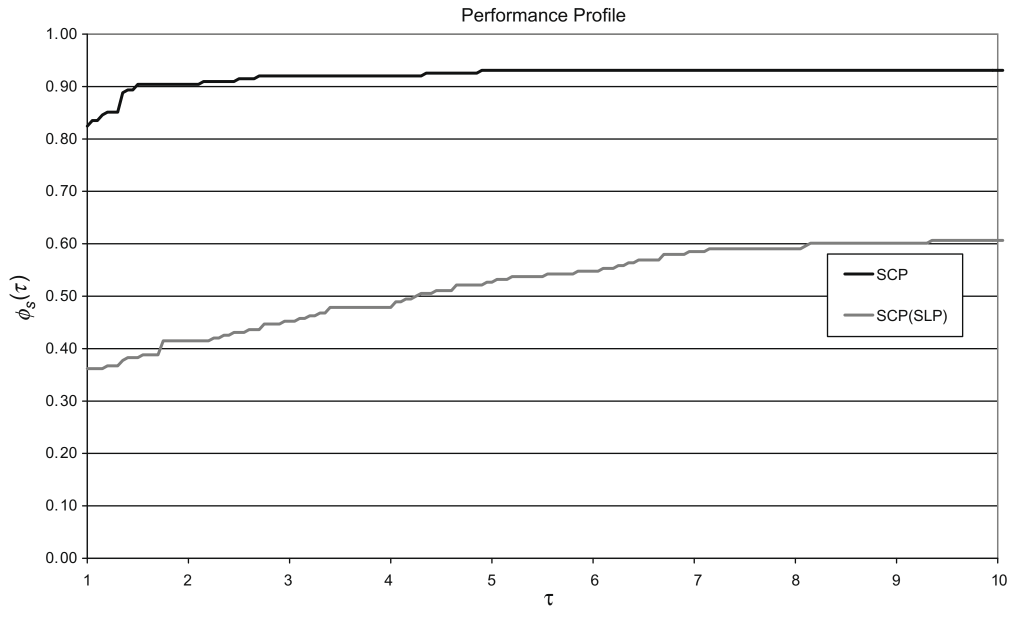

Fig. 6. Comparison of SCP algorithm with version without conjugate directions on Schittkowski collection.

The performance of solver s on the entire test set is defined by the quantity

$$\phi_s(\tau) = \frac{1}{n_p}\,\mathrm{card}\{p : 1 \leqslant p \leqslant n_p, r_{p,s} \leqslant \tau\},$$

where the function $\phi_s : R \to [0,1]$ is called the performance profile and represents the cumulative distribution function of the performance ratio. For a particular value of τ, the larger the value of $\phi_s(\tau)$, the better the performance of solver s.

4.5. Test results

The problems were solved within a precision tolerance $\epsilon = 10^{-3}$. Additionally, for the SCP algorithm, a value $\rho = 0.1$ was used as the value for the penalty parameter, $\delta^{\mathrm{dec}} = 0.2$ and $\delta^{\mathrm{inc}} = 0.8$ for updating the trust region and an initial trust region $d^L = -\mathbf{50}$ and $d^U = \mathbf{50}$. CPLEX was used as the solver for the LP subproblems.

For the other algorithms default option values were used, except for the precision tolerance, which was set to $\epsilon = 10^{-3}$.

A problem was considered successfully solved if the objective value of the solution found by the solver was within a 2% relative tolerance to the reported optimal value of the problem and the solution was feasible within the precision tolerance. Thus, if solver s found the solution x^s and the reported optimal solution is x^*, then the problem was considered successfully solved if $g(x^s) \leqslant \epsilon$, $|\,h(x^s)\,| \leqslant \epsilon$ and

$$\frac{f(x^s) - f(x^*)}{\max\{|f(x^*)|, 1\}} \leqslant 0.02.$$

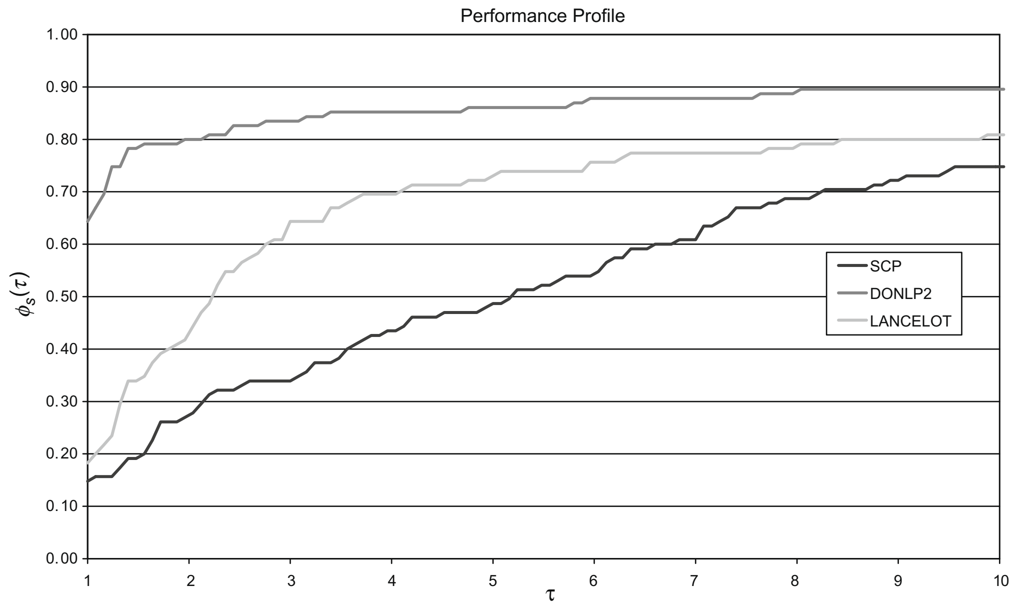

Fig. 7. Test results for equivalent function calls on the Hock and Schittkowski test set ($\epsilon = 10^{-6}$).

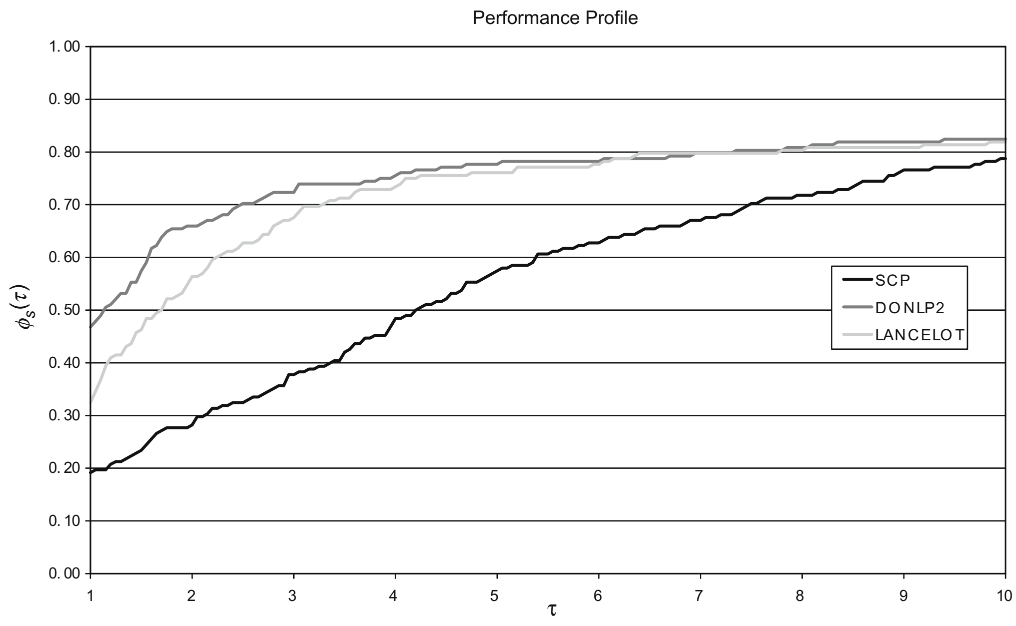

Fig. 8. Test results for equivalent function calls on the Schittkowski test set ($\epsilon = 10^{-6}$).

Table 1 contains the number of problems solved by each solver out of the 115 problems defined in the Hock and Schittkowski and the 188 problems defined in the Schittkowski test collections.

As can be seen from the results, the SCP algorithm performed well on the test problems, solving most problems from the Hock and Schittkowski test collection and also performing very well on the Schittkowski test collection. It was the most robust solver on both test collections.

Fig. 1 contains the performance profile for the solvers on the Hock and Schittkowski test collection. SCP performed best on these problems with respect to the number of gradient evaluations needed. Also DONLP2 performed well on the test problems, but DONLP2 had some difficulties with the harder problems and needed more gradient evaluations than the SCP algorithm for the more difficult problems.

Fig. 2 shows the performance profile for the Schittkowski test collection. Here LANCELOT performed a lot better than on the Hock and Schittkowski collection, but was still slower than the other two solvers. The SCP algorithm clearly performed best on the test collection being both the fastest solver as well as the most robust solver on the more difficult problems in the collection, when comparing the algorithms in terms of the gradient evaluations needed.

Finally, in Figs. 3 and 4 the performance profile is shown for the Hock and Schittkowski and the Schittkowski test collections respectively when the precision tolerance has been set to $\epsilon = 10^{-6}$. As can be seen from the results, the SCP algorithm performed better for a lower precision tolerance. This may indicate either that the algorithm has problems with convergence near the stationary point or that there are some numerical difficulties in the current implementation. The DONLP2 algorithm clearly performed better for the higher precision tolerance than for the lower.

The numerical results clearly indicate that the SCP algorithm has good convergence properties and may therefore, for instance, be considered as a subsolver in a branch and bound-based MINLP algorithm.

4.6. Results without conjugate directions

Although the conjugate directions, see (1c), in the LP subproblems are not needed to prove convergence to a KKT stationary point, they are essential to improve the convergence speed. As a comparison, the implementation of the original SCP algorithm described in this paper was compared with a version where only one LP subiteration is done in each NLP master iteration. In other words, an NLP iteration is stopped after solving one LP subproblem and no further LP problems are solved with constraints based on conjugate directions. The alternative version is denoted SCP(SLP) since restricting the subiterations to one for each NLP iteration will make the algorithm resemble a traditional sequential linear programming (SLP) algorithm. The performance of the SLP version was significantly worse. In Table 2 the number of successfully solved problems for the original algorithm is compared with the number of successfully solved problems for the SLP version.

As can be seen from the table, the original version is much more robust than the SLP version, particularly on the Schittkowski test collection. In addition, the algorithms are compared with performance profiles, see Figs. 5 and 6. The performance profiles clearly indicate that the conjugate directions in the LP subproblems are significant for the performance of the algorithm.

4.7. Results on function evaluations

Finally, the algorithms were compared to each other with regards to the number of function evaluations needed to solve the problems. As noted in the introduction, the primary goal when designing the algorithm has not been to minimize the number of function evaluations. It is assumed that the objective and constraints are fast to evaluate. However, for completeness it is interesting to see how the algorithms perform also with regards to the number of function evaluations.

In Figs. 7 and 8, the equivalent function calls required by each algorithm are compared. The equivalent function calls are defined as the number of function evaluations plus n times the number of gradient evaluations, where n is the size of the problem.

As can be seen from the figures, the DONLP2 algorithm was the fastest algorithm with respect to the number of equivalent function calls.

5. Conclusions

In this paper, a new LP-based algorithm for solving continuously differentiable NLP problems has been presented. Numerical experience on two standard test sets indicate that the algorithm is competitive with two existing NLP algorithms.

Potential applications for the algorithm is as a subsolver in an MINLP algorithm. Particularly for convex MINLP problems, the algorithm is attractive as it provides lower bounds on the optimal solutions for the convex NLP subproblems in the branch and bound tree. In [21], promising results for a selection of convex MINLP problems are reported. Very good results for a special set of difficult block optimization problems are found in [20].

Another application could be as a solver for large-scale NLP problems. More work on the algorithm is, however, needed for such applications.

Acknowledgements

Financial support from the Research Institute of the Foundation of Åbo Akademi is gratefully acknowledged. We would also like to thank the referees for their insightful comments that helped us improve the paper significantly.

References

[1] K.-M. Björk, P.O. Lindberg, T. Westerlund, Some Convexifications in Global Optimization of Problems Containing Signomial Terms, Computers and Chemical Engineering 27 (2003) 669–679.
[2] R.H. Byrd, N.I.M. Gould, J. Nocedal, R.A. Waltz, An algorithm for nonlinear optimization using linear programming and equality constrained subproblems, Mathematical Programming B 100 (2004) 27–48.
[3] A.R. Conn, N.I. M Gould, P.L. Toint, A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds, SIAM Journal on Numerical Analysis 28 (1991) 545–572.
[4] R.J. Dakin, A tree-search algorithm for mixed integer programming problems, Computer Journal 8 (1965) 250–255.
[5] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Mathematical Programming 91 (2002) 201–213.
[6] R.E. Griffith, R.A. Stewart, A nonlinear programming technique for the optimization of continuous processing systems, Management Science 7 (1961) 379–392.
[7] S.-P. Han, Superlinearly convergent variable metric algorithms for general nonlinear programming problems, Mathematical Programming 11 (1976) 263–282.
[8] S.-P. Han, A globally convergent method for nonlinear programming, Journal of Optimization Theory and Applications 22 (1977) 297–309.
[9] S.-P. Han, O.L. Mangasarian, Exact penalty functions in nonlinear programming, Mathematical Programming 17 (1979) 251–269.
[10] W. Hock, K. Schittkowski, Test Examples for Nonlinear Programming Codes, Lecture Notes in Economics and Mathematical Systems, vol. 187, Springer-Verlag, Berlin, 1981.
[11] J.E. Kelley, The cutting plane method for solving convex programs, Journal of SIAM 8 (1960) 703–712.
[12] B. Kolman, R.E. Beck, Elementary Linear Programming with Applications, Second ed., Academic Press, San Diego, 1995.
[13] C. Lemaréchal, A view of line searches, in: A. Auslender, W. Oettli, J. Stoer (Eds.), Optimization and optimal control, Lecture Notes in Control and Information Science, vol. 30, Springer-Verlag, Berlin, 1981, pp. 59–78.

[14] D.G. Luenberger, Linear and Nonlinear Programming, Second ed., Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.
[15] G. Di Pillo, Exact penalty methods, in: E. Spedicato (Ed.), Algorithms for Continuous Optimization: The State of the Art, Kluwer Academic Publishers, Dordrecht, 1994, pp. 209–253.
[16] R. Pörn, Mixed Integer Nonlinear Programming: Convexification Techniques and Algorithm Development, PhD Thesis, Åbo Akademi University, Åbo, 2000.
[17] K. Schittkowski, More Test Examples for Nonlinear Programming Codes, Lecture Notes in Economics and Mathematical Systems, vol. 282, Springer-Verlag, Berlin, 1987.
[18] P. Spellucci, DONLP2 Short Users Guide, Technical Report, Department of Mathematics, Darmstadt University of Technology, Darmstadt, 1999.
[19] C. Still, T. Westerlund, A sequential cutting plane algorithm for solving convex NLP problems, European Journal of Operational Research 173 (2006) 444–464.
[20] C. Still, T. Westerlund, Sequential cutting plane algorithm, in: C.A. Floudas, P.M. Pardalos (Eds.), Encyclopedia of Optimization, second ed., Springer, 2009, pp. 3471–3476.
[21] C. Still, T. Westerlund, Solving convex MINLP optimization problems using a sequential cutting plane algorithm, Computational Optimization and Applications 34 (2006) 63–83.
[22] R.J. Vanderbei, Nonlinear Optimization Models. <http://www.princeton.edu/~rvdb/ampl/nlmodels>.
[23] T. Westerlund, Some transformation techniques in global optimization, in: L. Liberti, N. Maculan (Eds.), Global Optimization: From Theory to Implementation, Springer, New York, 2006, pp. 45–74.
[24] Ch. Zillober, K. Schittkowski, K. Moritzen, Very large scale optimization by sequential convex programming, Optimization Methods and Software 19 (2004) 103–120.