Global minimization using an Augmented Lagrangian method with variable lower-level constraints

E. G. Birgin [*] C. A. Floudas [†] J. M. Martínez [‡]

January 22, 2007 [§]

Abstract

A novel global optimization method based on an Augmented Lagrangian framework is introduced for continuous constrained nonlinear optimization problems. At each outer iteration k the method requires the ε_k-global minimization of the Augmented Lagrangian with simple constraints, where $\varepsilon_k \to \varepsilon$. Global convergence to an ε-global minimizer of the original problem is proved. The subproblems are solved using the αBB method. Numerical experiments are presented.

Key words: deterministic global optimization, Augmented Lagrangians, nonlinear programming, algorithms, numerical experiments.

1 Introduction

Global optimization has ubiquitous applications in all branches of engineering sciences and applied sciences. During the last decade several textbooks addressed different facets of global optimization theory and applications [10, 16, 19, 26, 47, 56, 57, 60]. Recent review papers have also appeared [17, 36].

The Augmented Lagrangian methodology based on the Powell-Hestenes-Rockafellar [25, 39, 41] formula has been successfully used for defining practical nonlinear programming algorithms [6, 7, 12, 14]. Convergence to KKT points was proved using the Constant Positive Linear

[*] Department of Computer Science IME-USP, University of São Paulo, Rua do Matão 1010, Cidade Universitária, 05508-090, São Paulo SP, Brazil. This author was supported by PRONEX-Optimization (PRONEX - CNPq / FAPERJ E-26 / 171.164/2003 - APQ1), FAPESP (Grants 06/53768-0 and 06/51827-9) and CNPq (PROSUL 490333/2004-4). e-mail: egbirgin@ime.usp.br

[†] Department of Chemical Engineering, Princeton University, Princeton, NJ 08544, USA. This author was supported by the National Science Foundation and the National Institute of Health (R01 GM52032). e-mail: floudas@titan.princeton.edu

[‡] Department of Applied Mathematics, IMECC-UNICAMP, University of Campinas, CP 6065, 13081-970 Campinas SP, Brazil. This author was supported by PRONEX-Optimization (PRONEX - CNPq / FAPERJ E-26 / 171.164/2003 - APQ1), FAPESP (Grant 06/53768-0) and CNPq. e-mail: martinez@ime.unicamp.br

[§] Last update: July 17, 2008 (typos corrected on August 17, 2010, and May 22, 2012).

Dependence constraint qualification [5], which strengthens the results based on the classical regularity condition [11, 14].

In this work, we consider the Augmented Lagrangian method introduced in [6] and we modify it in such a way that, at each outer iteration k, we find an ε_k*-global* minimizer of the subproblem. In the definition of the subproblem we introduce an important modification with respect to [6]: besides the lower level constraints we include constraints that incorporate information about the global solution of the nonlinear programming problem. A theorem of convergence to ε-global minimizers is presented.

In the implementation, we consider linear constraints on the lower-level set, and additional valid linear constraints which result from outer approximations of the feasible region and hence incorporate the global optimum information. This allows us to use the αBB [1, 2, 3, 9] method and its convex underestimation techniques [33, 34] for the subproblems, in such a way that the underestimation techniques are applied just to the Augmented Lagrangian function and not to the constraints. The αBB global optimization approach has been applied to various problems that include molecular conformations in protein folding, parameter estimation and phase equilibrium. Mixed-integer nonlinear models arising in process synthesis, design and operations problems represent additional important application areas. See [16] and the references therein for details.

There exist many global optimization techniques for nonlinear programming problems, e.g., [2, 3, 4, 9, 18, 20, 21, 22, 23, 24, 28, 29, 31, 38, 43, 44, 48, 49, 50, 51, 52, 53, 58]. However, up to our knowledge, the proposed approach is the first practical deterministic global optimization method based on the Augmented Lagrangian framework. Moreover, as a consequence of using the Augmented Lagrangian approach combined with the αBB method and its convex α-underestimation techniques, the method introduced in this paper does not rely on the specific form of the functions involved in the problem definition (objective function and constraints), apart from their continuity and differentiability. Interval arithmetic is used to compute bounds on the objective function and to compute the convex α-underestimators. Although the method can take advantage of known underestimators and relaxations for several kinds of functional forms, it can also deal with functional forms for which underestimators and relaxations have not been developed yet. In this sense, the method does not depend on the analysis of expressions involved in the problem definition to identify functional forms for which ad-hoc underestimators are available.

This paper is organized as follows. In Section 2 we describe the Augmented Lagrangian deterministic global optimization algorithm. The convergence to ε-global minimizers is proved in Section 3. In Section 4 we describe the global optimization of the Augmented Lagrangian subproblems. Numerical results are given in Section 5. In Section 6 we draw some conclusions.

Notation. If $v \in I\!\!R^n$, $v = (v_1, \dots, v_n)$, we denote $v_+ = (\max\{0, v_1\}, \dots, \max\{0, v_n\})$. If $K = (k_1, k_2, \dots) \subset I\!\!N$ (with $k_j < k_{j+1}$ for all j), we denote $K \underset{\infty}{\subset} I\!\!N$. The symbol $\|\cdot\|$ will denote the Euclidian norm.

2 The overall algorithm

The problem to be addressed is:

$$\begin{array}{ll} \text{Minimize} & f(x) \\ \text{subject to} & h(x) = 0 \\ & g(x) \leq 0 \\ & x \in \Omega \end{array} \tag{1}$$

where $h : I\!R^n \to I\!R^m, g : I\!R^n \to I\!R^p, f : I\!R^n \to I\!R$ are continuous and $\Omega \subset I\!R^n$ is closed. A typical set Ω consists of "easy" constraints such as linear constraints and box constraints. By easy we mean that a suitable algorithm for local minimization is available.

Assumption 1. From now on we will assume that there exists a global minimizer z of the problem.

We define the following Augmented Lagrangian function [25, 39, 41]:

$$L_\rho(x, \lambda, \mu) = f(x) + \frac{\rho}{2}\left\{ \sum_{i=1}^{m} \left[h_i(x) + \frac{\lambda_i}{\rho} \right]^2 + \sum_{i=1}^{p} \left[\max\left(0, g_i(x) + \frac{\mu_i}{\rho}\right)\right]^2 \right\} \tag{2}$$

for all $x \in \Omega, \rho > 0, \lambda \in I\!R^m, \mu \in I\!R^p_+$.

Algorithm 2.1

Let $\lambda_{\min} < \lambda_{\max}$, $\mu_{\max} > 0$, $\gamma > 1$, $0 < \tau < 1$. Let $\{\varepsilon_k\}$ be a sequence of nonnegative numbers such that $\lim_{k\to\infty} \varepsilon_k = \varepsilon \geq 0$. Let $\lambda_i^1 \in [\lambda_{\min}, \lambda_{\max}], i = 1, \ldots, m$, $\mu_i^1 \in [0, \mu_{\max}], i = 1, \ldots, p$, and $\rho_1 > 0$. Initialize $k \leftarrow 1$.

Step 1. Let $P_k \subset I\!R^n$ be a closed set such that a global minimizer z (the same for all k) belongs to P_k. Find an ε_k-global minimizer x^k of the problem Min $L_{\rho_k}(x, \lambda^k, \mu^k)$ subject to $x \in \Omega \cap P_k$. That is $x^k \in \Omega \cap P_k$ is such that:

$$L_{\rho_k}(x^k, \lambda^k, \mu^k) \leq L_{\rho_k}(x, \lambda^k, \mu^k) + \varepsilon_k \tag{3}$$

for all $x \in \Omega \cap P_k$. The ε_k-global minimum can be obtained using a deterministic global optimization approach, such as the αBB method.

Step 2. Define

$$V_i^k = \max\left\{ g_i(x^k), -\frac{\mu_i^k}{\rho_k} \right\}, i = 1, \ldots, p.$$

If $k = 1$ or

$$\max\{\|h(x^k)\|_\infty, \|V^k\|_\infty\} \leq \tau \ \max\{\|h(x^{k-1})\|_\infty, \|V^{k-1}\|_\infty\}, \tag{4}$$

define $\rho_{k+1} = \rho_k$. Otherwise, define $\rho_{k+1} = \gamma\rho_k$.

Step 3. Compute $\lambda_i^{k+1} \in [\lambda_{\min}, \lambda_{\max}], i = 1, \ldots, m$ and $\mu_i^{k+1} \in [0, \mu_{\max}], i = 1, \ldots, p$. Set $k \leftarrow k+1$ and go to Step 1.

Remark. In the implementation, we will compute $\lambda_i^{k+1} = \min\{\max\{\lambda_{\min}, \lambda_i^k + \rho h_i(x^k)\}, \lambda_{\max}\}$ and $\mu_i^{k+1} = \min\{\max\{0, \mu^k + \rho g_i(x^k)\}, \mu_{\max}\}$. These definitions correspond to safeguarded choices of first-order Lagrange multiplier estimates. After the resolution of each subproblem, the vectors λ^k/ρ_k and μ^k/ρ_k represent shifts of the origin with respect to which infeasibility is penalized. The intutitive idea is that these shifts "correct" the previous decision on the best possible origin and enhance the possibility of achieving feasibility at the present iteration. The theoretical consequence is that, under suitable assumptions, one is able to prove that the penalty parameter does not need to go to infinity [6]. In practice, this implies that the subproblems tend to remain well conditioned.

We emphasize that the deterministic global optimization method αBB will not use the point x^{k-1} as "initial approximation" as most local optimization solvers do. In fact, the concept of "initial point" has no meaning at all in this case. The information used by the outer iteration k is the set of approximate Lagrange multipliers computed after iteration $k-1$, and nothing else.

3 Convergence to an ε-global minimum

In the theorems that follow, we assume that the sequence $\{x^k\}$ is well defined. In other words, the ε_k-global minimizer of the Augmented Lagrangian can always be found. A sufficient condition on the problem that guarantees that this assumption holds is the compactness of Ω. In practical Optimization it is usual to add box constraints to the feasible set of the problem that reflect some previous knowledge on the localization of the solution. Clearly, after intersection with a box, the feasible set becomes compact.

Theorem 1. *Assume that the sequence $\{x^k\}$ is well defined and admits a limit point x^*. Then, x^* is feasible.*

Proof. Since Ω is closed and $x^k \in \Omega$, we have that $x^* \in \Omega$. We consider two cases: $\{\rho_k\}$ bounded and $\rho_k \to \infty$. If $\{\rho_k\}$ is bounded, there exists k_0 such that $\rho_k = \rho_{k_0}$ for all $k \geq k_0$. Therefore, for all $k \geq k_0$, (4) holds. This implies that $\|h(x^k)\| \to 0$ and $\|V^k\| \to 0$. So, $g_i(x^k)_+ \to 0$ for all $i = 1, \ldots, p$. So, the limit point is feasible.

Now, assume that $\rho_k \to \infty$. Let z be as in Step 1. Therefore, z is feasible. So, $\|h(z)\| = \|g(z)_+\| = 0$. Suppose, by contradiction, that x^* is not feasible. Therefore,

$$\|h(x^*)\|^2 + \|g(x^*)_+\|^2 > \|h(z)\|^2 + \|g(z)_+\|^2.$$

Let K be an infinite sequence of indices such that $\lim_{k \in K} x^k = x^*$. Since h and g are continuous, λ^k, μ^k are bounded and $\rho_k \to \infty$, there exists $c > 0$ such that for $k \in K$ large enough:

$$\left\|h(x^k) + \frac{\lambda^k}{\rho_k}\right\|^2 + \left\|\left(g(x^k) + \frac{\mu^k}{\rho_k}\right)_+\right\|^2 > \left\|h(z) + \frac{\lambda^k}{\rho_k}\right\|^2 + \left\|\left(g(z) + \frac{\mu^k}{\rho_k}\right)_+\right\|^2 + c.$$

Therefore,

$$f(x^k) + \frac{\rho_k}{2}\left[\left\|h(x^k) + \frac{\lambda^k}{\rho_k}\right\|^2 + \left\|\left(g(x^k) + \frac{\mu^k}{\rho_k}\right)_+\right\|^2\right]$$

$$> f(z) + \frac{\rho_k}{2}\left[\left\|h(z) + \frac{\lambda^k}{\rho_k}\right\|^2 + \left\|\left(g(z) + \frac{\mu^k}{\rho_k}\right)_+\right\|^2\right] + \frac{\rho_k c}{2} + f(x^k) - f(z).$$

Since $\lim_{k\in K} x^k = x^*$ and f is continuous, for $k \in K$ large enough

$$\frac{\rho_k c}{2} + f(x^k) - f(z) > \varepsilon_k.$$

Therefore,

$$f(x^k)+\frac{\rho_k}{2}\left[\left\|h(x^k)+\frac{\lambda^k}{\rho_k}\right\|^2+\left\|\left(g(x^k)+\frac{\mu^k}{\rho_k}\right)_+\right\|^2\right] > f(z)+\frac{\rho_k}{2}\left[\left\|h(z)+\frac{\lambda^k}{\rho_k}\right\|^2+\left\|\left(g(z)+\frac{\mu^k}{\rho_k}\right)_+\right\|^2\right]+\varepsilon_k.$$

Now, since z is a global minimizer, we have that $z \in \Omega \cap P_k$ for all k. Therefore, the inequality above contradicts the definition of x^k. □

Theorem 2. *Under the same assumptions of Theorem 1, every limit point x^* of a sequence $\{x^k\}$ generated by Algorithm 2.1 is an ε-global minimizer of the problem.*

Proof. Let $K \underset{\infty}{\subset} I\!N$ be such that $\lim_{k\in K} x^k = x^*$. By Theorem 1, x^* is feasible. Let $z \in \Omega$ as as in Step 1. Then, $z \in P_k$ for all k.

We consider two cases: $\rho_k \to \infty$ and $\{\rho_k\}$ bounded.

Case 1 ($\rho_k \to \infty$): By the definition of the algorithm:

$$f(x^k)+\frac{\rho_k}{2}\left[\left\|h(x^k)+\frac{\lambda^k}{\rho_k}\right\|^2+\left\|\left(g(x^k)+\frac{\mu^k}{\rho_k}\right)_+\right\|^2\right] \le f(z)+\frac{\rho_k}{2}\left[\left\|h(z)+\frac{\lambda^k}{\rho_k}\right\|^2+\left\|\left(g(z)+\frac{\mu^k}{\rho_k}\right)_+\right\|^2\right]+\varepsilon_k \tag{5}$$

for all $k \in I\!N$.

Since $h(z) = 0$ and $g(z) \le 0$, we have:

$$\left\|h(z) + \frac{\lambda^k}{\rho_k}\right\|^2 = \left\|\frac{\lambda^k}{\rho_k}\right\|^2 \text{ and } \left\|\left(g(z) + \frac{\mu^k}{\rho_k}\right)_+\right\|^2 \le \left\|\frac{\mu^k}{\rho_k}\right\|^2.$$

Therefore, by (5),

$$f(x^k) \le f(x^k) + \frac{\rho_k}{2}\left[\left\|h(x^k) + \frac{\lambda^k}{\rho_k}\right\|^2 + \left\|\left(g(x^k) + \frac{\mu^k}{\rho_k}\right)_+\right\|^2\right] \le f(z) + \frac{\|\lambda^k\|^2}{2\rho_k} + \frac{\|\mu^k\|^2}{2\rho_k} + \varepsilon_k.$$

Taking limits for $k \in K$ and using that $\lim_{k\in K} \|\lambda^k\|/\rho_k = \lim_{k\in K} \|\mu^k\|/\rho_k = 0$ and $\lim_{k\in K} \varepsilon_k = \varepsilon$, by the continuity of f and the convergence of x^k, we get:

$$f(x^*) \le f(z) + \varepsilon.$$

Since z is a global minimizer, it turns out that x^* is an ε-global minimizer, as we wanted to prove.

Case 2 ($\{\rho_k\}$ bounded): In this case, we have that $\rho_k = \rho_{k_0}$ for all $k \geq k_0$. Therefore, by the definition of Algorithm 2.1, we have:

$$f(x^k)+\frac{\rho_{k_0}}{2}\left[\left\|h(x^k)+\frac{\lambda^k}{\rho_{k_0}}\right\|^2+\left\|\left(g(x^k)+\frac{\mu^k}{\rho_{k_0}}\right)_+\right\|^2\right] \leq f(z)+\frac{\rho_{k_0}}{2}\left[\left\|h(z)+\frac{\lambda^k}{\rho_{k_0}}\right\|^2+\left\|\left(g(z)+\frac{\mu^k}{\rho_{k_0}}\right)_+\right\|^2\right]+\varepsilon_k$$

for all $k \geq k_0$. Since $g(z) \leq 0$ and $\mu^k/\rho_{k_0} \geq 0$,

$$\left\|\left(g(z) + \frac{\mu^k}{\rho_{k_0}}\right)_+\right\|^2 \leq \left\|\frac{\mu^k}{\rho_{k_0}}\right\|^2.$$

Thus, since $h(z) = 0$,

$$f(x^k) + \frac{\rho_{k_0}}{2}\left[\left\|h(x^k) + \frac{\lambda^k}{\rho_{k_0}}\right\|^2 + \left\|\left(g(x^k) + \frac{\mu^k}{\rho_{k_0}}\right)_+\right\|^2\right] \leq f(z) + \frac{\rho_{k_0}}{2}\left[\left\|\frac{\lambda^k}{\rho_{k_0}}\right\|^2 + \left\|\frac{\mu^k}{\rho_{k_0}}\right\|^2\right] + \varepsilon_k$$

for all $k \geq k_0$. Let $K_1 \underset{\infty}{\subset} K$ be such that

$$\lim_{k\in K_1} \lambda^k = \lambda^*, \lim_{k\in K_1} \mu^k = \mu^*.$$

By the feasibility of x^*, taking limits in the inequality above for $k \in K_1$, we get:

$$f(x^*) + \frac{\rho_{k_0}}{2}\left[\left\|\frac{\lambda^*}{\rho_{k_0}}\right\|^2 + \left\|\left(g(x^*) + \frac{\mu^*}{\rho_{k_0}}\right)_+\right\|^2\right] \leq f(z) + \frac{\rho_{k_0}}{2}\left[\left\|\frac{\lambda^*}{\rho_{k_0}}\right\|^2 + \left\|\frac{\mu^*}{\rho_{k_0}}\right\|^2\right] + \varepsilon.$$

Therefore,

$$f(x^*) + \frac{\rho_{k_0}}{2}\left\|\left(g(x^*) + \frac{\mu^*}{\rho_{k_0}}\right)_+\right\|^2 \leq f(z) + \frac{\rho_{k_0}}{2}\left\|\frac{\mu^*}{\rho_{k_0}}\right\|^2 + \varepsilon.$$

Thus,

$$f(x^*) + \frac{\rho_{k_0}}{2}\sum_{i=1}^{p}\left(g_i(x^*) + \frac{\mu_i^*}{\rho_{k_0}}\right)_+^2 \leq f(z) + \frac{\rho_{k_0}}{2}\sum_{i=1}^{p}\left(\frac{\mu_i^*}{\rho_{k_0}}\right)^2 + \varepsilon. \tag{6}$$

Now, if $g_i(x^*) = 0$, since $\mu_i^*/\rho_{k_0} \geq 0$, we have that

$$\left(g_i(x^*) + \frac{\mu_i^*}{\rho_{k_0}}\right)_+ = \frac{\mu_i^*}{\rho_{k_0}}.$$

Therefore, by (6),

$$f(x^*) + \frac{\rho_{k_0}}{2}\sum_{g_i(x^*)<0}\left(g_i(x^*) + \frac{\mu_i^*}{\rho_{k_0}}\right)_+^2 \leq f(z) + \frac{\rho_{k_0}}{2}\sum_{g_i(x^*)<0}\left(\frac{\mu_i^*}{\rho_{k_0}}\right)^2 + \varepsilon. \tag{7}$$

But, by Step 2 of Algorithm 2.1 , $\lim_{k\to\infty} \max\{g_i(x^k), -\mu_i^k/\rho_{k_0}\} = 0$. Therefore, if $g_i(x^*) < 0$ we necessarily have that $\mu_i^* = 0$. Therefore, (7) implies that $f(x^*) \leq f(z)+\varepsilon$. Since z is a global minimizer, the proof is complete. □

4 Global optimization of subproblems

In this section, we address the problem of finding $x^k \in \Omega \cap P_k$ satisfying (3) and we restrict ourselves to the case in which Ω is defined by linear constraints. This problem is equivalent to the problem of finding an ε_k-global solution of the problem:

$$\text{Minimize } L_{\rho_k}(x, \lambda^k, \mu^k) \text{ subject to } x \in \Omega \cap P_k, \tag{8}$$

where $\Omega = \{x \in I\!R^n \mid Ax = b,\ Cx \le d,\ l \le x \le u\}$ and $Ax = b$, $Cx \le d$ and $l \le x \le u$ represent the linear equality, linear inequality and bound constraints of problem (1), respectively. The remaining constraints of problem (1) will be $h(x) = 0$, $g(x) \le 0$. The role of P_k will be elucidated soon.

To solve problem (8), we introduced and implemented the αBB algorithm [2, 3] for the particular case of linear constraints and bounds. The αBB method is a deterministic global optimization method for nonlinear programming problems based on Branch & Bound. For bounding purposes, it uses the convex α-underestimator of the function being minimized that coincides with the function at the bounds of the box and whose maximum separation (distance to the objective function) is proportional to the box dimensions. Therefore, the smaller the box, the tighter the convex α-underestimator.

Based on the last observation, the αBB method consists of splitting the box-constraints domain into smaller subdomains in order to reduce the gap between an upper and a lower bound on the minimum of the problem. The upper bound is given by the smallest functional value obtained through local minimizations within the subdomains, while the lower bound comes from the global minimization of the convex α-underestimators subject to the problem constraints. If, within a subdomain, the lower bound plus the prescribed tolerance ε_k is above the upper bound, the subdomain can be discarded as it clearly does not contain the solution of the problem (considering the tolerance ε_k). The same argument applies if, via interval analysis, it is shown the subdomain does not contain any promising point.

The constraints of the problem, or any other valid constraint, can also be used to substitute a subdomain $[\bar{l}, \bar{u}]$ for any other proper subdomain $[\hat{l}, \hat{u}]$. In our implementation, we considered just linear constraints in the process of reducing a subdomain $[\bar{l}, \bar{u}]$. Three sources of linear constraints were used, namely (a) linear constraints of the original problem; (b) linear relaxations (valid within the subdomain $[\bar{l}, \bar{u}]$) of the nonlinear penalized constraints; and (c) linear "cuts" of the form $L^U_{\rho_k}(x, \lambda^k, \mu^k) \le L_{\text{ub}}$, where $L^U_{\rho_k}(\cdot, \lambda^k, \mu^k)$ is a linear relaxation of $L_{\rho_k}(\cdot, \lambda^k, \mu^k)$ within $[\bar{l}, \bar{u}]$. Constraints of type (a) and (b) can be used to discard regions that do not contain feasible points of problem (1) and play a role in the definition of P_k. Constraints of type (c) eliminate regions that do not contain the global solution of (8).

Let us call B the original box of the problem and L the original polytope defined by the linear constraints. As a result of the αBB process, the original box is covered by t "small" boxes $B_1, \ldots, B_t$. For each small box B_i a polytope Q_i (defined by the relaxations) is given (perhaps $Q_i = I\!R^n$) and a new small box $\hat{B}_i$ such that $(Q_i \cap B_i) \subset \hat{B}_i \subset B_i$ is constructed. By construction, the set $P_k = \cup_{i=1}^t \hat{B}_i$ contains the feasible set of the problem. So, P_k contains the global minimizers of the problem. The αBB algorithm guarantees an ε_k-global minimizer on $L \cap P_k$, as required by the theory.

The algorithm starts with a list S of unexplored subdomains that initially has as unique element the original box domain of the problem. Then, for each subdomain in the list it does the following tasks: (i) reduce the subdomain; (ii) try to discard the subdomain via interval analysis or computing the global solution of the underestimating problem; (iii) if the subdomain cannot be discarded; perform a local minimization within the subdomain; (iv) finally, split the subdomain and add the new subdomains to S. The method stops when the list S is empty.

The description of the αBB algorithm, that follows very closely the algorithm introduced in [3], is as follows.

Algorithm 4.1: αBB

Step 1. *Initialization*

Set $S = \{[l, u]\}$ and $L_{\mathrm{ub}} = +\infty$.

Step 2. *Stopping criterion*

If S is empty, stop.

Step 3. *Choose a subdomain*

Choose $[\bar{l}, \bar{u}] \in S$ and set $S \leftarrow S \setminus [\bar{l}, \bar{u}]$.

Step 4. *Subdomain reduction and possible discarding*

Let $W_{[\bar{l},\bar{u}]}$ be a set of linear constraints valid within the subdomain $[\bar{l}, \bar{u}]$ plus linear constraints satisfied by the global solution of (8). For $i = 1, \dots, n$, compute $\hat{l}_i$ and $\hat{u}_i$ as

$$arg \min \pm x_i \text{ subject to } Ax = b,\ Cx \le d, \bar{l} \le x \le \bar{u},\ x \in W_{[\bar{l},\bar{u}]}, \tag{9}$$

respectively. If the feasible set of (9) is empty, discard the subdomain $[\bar{l}, \bar{u}]$ and go to Step 2.

Step 5. *Reduced subdomain discarding*

Step 5.1 *Via interval analysis*

Step 5.1.1 Using interval analysis, compute $[L^{\min}, L^{\max}]$ such that

$$L^{\min} \le L_{\rho_k}(x, \lambda^k, \mu^k) \le L^{\max},\ \forall\, x \in [\hat{l}, \hat{u}].$$

If $L^{\min} + \varepsilon_k \ge L_{\mathrm{ub}}$ then discard the reduced subdomain $[\hat{l}, \hat{u}]$ and go to Step 2.

Step 5.1.2 For each equality constraint $h_i(x) = 0$, compute $[h_i^{\min}, h_i^{\max}]$ such that

$$h_i^{\min} \le h_i(x) \le h_i^{\max},\ \forall\, x \in [\hat{l}, \hat{u}].$$

If $0 \notin [h_i^{\min}, h_i^{\max}]$ then discard the reduced subdomain $[\hat{l}, \hat{u}]$ and go to Step 2. The same reasoning applies to each inequality constraint $g_i(x) \le 0$ if $[-\infty, 0] \cap [g_i^{\min}, g_i^{\max}] = \emptyset$.

Step 5.2. *Via minimization of a convex underestimator*

Compute a convex underestimator $U_{[\hat{l},\hat{u}]}(x)$ of $L_{\rho_k}(x, \lambda^k, \mu^k)$ and find

$$y_1 \leftarrow arg \min U_{[\hat{l},\hat{u}]}(x) \text{ subject to } Ax = b,\ Cx \leq d,\ \hat{l} \leq x \leq \hat{u}.$$

If $U_{[\hat{l},\hat{u}]}(y_1) + \varepsilon_k \geq L_{\text{ub}}$ then discard the reduced subdomain $[\hat{l}, \hat{u}]$ and go to Step 2.

Step 6. *Local optimization within subdomain*

Using y_1 as initial guess, compute

$$y_2 \leftarrow arg \min L_{\rho_k}(x, \lambda^k, \mu^k) \text{ subject to } Ax = b,\ Cx \leq d,\ \hat{l} \leq x \leq \hat{u}.$$

If $L_{\rho_k}(y_2, \lambda^k, \rho^k) < L_{\text{ub}}$ then set $L_{\text{ub}} \leftarrow L_{\rho_k}(y_2, \lambda^k, \rho^k)$ and $x_{\text{ub}} \leftarrow y_2$.

Step 7. *Split subdomain*

Split $[\hat{l}, \hat{u}]$ in at least 2 proper subdomains. Add the new subdomains to S and go to Step 3.

Remarks.

1. The set S is implemented as a queue (see Steps 3 and 7). It is a very simple and problem-independent strategy that, in contrast to the possibility of using a stack, provides to the method a diversification that could result in fast improvements of L_{ub}.

2. At Step 7, we divide the subdomain in two subdomains splitting the range of the variable selected by the least reduced axis rule [3] in its middle point. Namely, we choose x_i such that $i = arg \min_j \{(\hat{u}_j - \hat{l}_j)/(u_j - l_j)\}$. If a variable appears linearly in the objective function of the subproblem (8) then its range does not need to be split at all and it is excluded from the least reduced axis rule. The variables that appear linearly in the Augmented Lagrangian are the ones that appear linearly in the objective function of the original problem (1) and do not appear in the penalized nonlinear constraints.

3. Set $W_{[\bar{l},\bar{u}]}$ at Step 4 is composed by linear relaxations (valid within the subdomain $[\bar{l}, \bar{u}]$) of the penalized nonlinear constraints. We considered the convex and concave envelopes for bilinear terms and the convex envelope for concave functions [16], as well as tangent hyperplanes to the convex constraints.

4. By the definition of the Augmented Lagrangian function (2), it is easy to see that $f(x) \leq L_\rho(x, \lambda, \mu)\ \forall x$. So, any linear relaxation of the objective function $f(x)$ is also a linear relaxation of the Augmented Lagrangian function $L_\rho(x, \lambda, \mu)$. Therefore, a constraint of the form $f^U(x) \leq L_{\text{ub}}$, where $f^U(\cdot)$ is a linear relaxation of $f(\cdot)$ for all $x \in [\bar{l}, \bar{u}]$ was also included in the definition of $W_{[\bar{l},\bar{u}]}$ at Step 4.

5. As a consequence of the definition of Ω in (8) (associated to the choice of the lower-level constraints), the optimization subproblems at Steps 5.2 and 6 are linearly constrained optimization problems. Moreover, by the definition of Ω and the fact that $W_{[\bar{l},\bar{u}]}$ at Step 4 is described by linear constraints, the $2n$ optimization problems at Step 4 are linear programming problems.

6. At Step 5.2, it is not necessary to complete the minimization process. It would be enough to find z such that $U_{[\hat{l},\hat{u}]}(z) + \varepsilon_k < L_{\text{ub}}$ to realize that the subdomain can not be discarded.

There is an alternative that represents a trade-off between effectiveness and efficiency. Step 4 can be repeated, substituting $\bar{l}$ and $\bar{u}$ by $\hat{l}$ and $\hat{u}$ in (9), respectively, while $\hat{l} \neq \bar{l}$ or $\hat{u} \neq \bar{u}$. Moreover, within the same loop, $\bar{l}_i$ and $\bar{u}_i$ can be replaced by $\hat{l}_i$ and $\hat{u}_i$ as soon as they are computed. Doing that, the order in which the new bounds are computed might influence the final result, and a strategy to select the optimal sequence of bounds updates might be developed.

The computation of the convex underestimator $U_{[\hat{l},\hat{u}]}(x)$ uses the convex α-underestimator for general nonconvex terms introduced in [33] and defined as follows:

$$U_{[\hat{l},\hat{u}]}(x) = L_{\rho_k}(x, \lambda^k, \mu^k) - \sum_{i=1}^{n} \alpha_i(\hat{u}_i - x_i)(x_i - \hat{l}_i), \tag{10}$$

where $\alpha_i, i = 1, \dots, n$, are positive scalars large enough to ensure the convexity of $U_{[\hat{l},\hat{u}]}(x)$. It is worth mentioning that, although the theory of the convex α-underestimator was developed for twice-continuously differentiable functions, the continuity of the second derivatives, which does not hold in the Augmented Lagrangian (2), is not necessary. A detailed explanation follows.

The gradient of the Augmented Lagrangian, $\nabla_x L_{\rho_k}(x, \lambda^k, \mu^k)$, is continuous. The Hessian of the Augmented Lagrangian, which exists and is continuous at any point x such that $\mu_i^k + \rho_k \, g_i(x) \neq 0$ for $i \in \{1, \dots, p\}$, is given by

$$\nabla_{xx} L_{\rho_k}(x, \lambda^k, \mu^k) = H_{\rho_k}(x, \lambda^k, \mu^k) + \rho_k \sum_{i \in I(x)} \nabla g_i(x) \nabla g_i(x)^T,$$

where $I(x) = \{i \in \{1, \dots, p\} \mid \mu_i^k + \rho_k g_i(x) > 0\}$ and $H_{\rho_k}(x, \lambda^k, \mu^k)$ is defined by

$$H_{\rho_k}(x, \lambda^k, \mu^k) = \nabla^2 f(x) \; + \; \sum_{i=1}^{m} [\lambda_i^k + \rho_k h_i(x)] \nabla^2 h_i(x) \; + \; \sum_{i=1}^{p} [\mu_i^k + \rho_k g_i(x)]_+ \nabla^2 g_i(x) + \rho_k \sum_{i=1}^{m} \nabla h_i(x) \nabla h_i(x)^T. \tag{11}$$

We used the Scaled Gerschgorin method [2, 3] to compute the α's in (10). However, instead of applying it to the interval Hessian of $L_{\rho_k}(x, \lambda^k, \mu^k)$, we apply it to $[H_{[\hat{l},\hat{u}]}]$, the interval matrix for $x \in [\hat{l}, \hat{u}]$ associated to $H_{\rho_k}(x, \lambda^k, \mu^k)$, as defined in (11), that is a continuous function of x. So, given $d \in I\!R_{++}$ we have

$$\alpha_i = \max \{0, -\frac{1}{2}(h_{ii}^{\min} - \sum_{j \neq i} |h|_{ij} \frac{d_j}{d_i})\}, i = 1, \dots, n, \tag{12}$$

where $|h|_{ij} = \max\{|h_{ij}^{\min}|, |h_{ij}^{\max}|\}$ and $(h_{ij}^{\min}, h_{ij}^{\max})$ denotes element at position (i, j) of $[H_{[\hat{l},\hat{u}]}]$. As suggested in [2], we choose $d = \hat{u} - \hat{l}$.

It remains to prove the convexity of the underestimator $U_{[\hat{l},\hat{u}]}(x)$ defined in (10), that inherits the second-derivative discontinuity from $L_{\rho_k}(x, \lambda^k, \mu^k)$ and for which the α's in (12) were computed applying the Scaled Gerschgorin method to $H_{\rho_k}(x, \lambda^k, \rho^k)$ instead of to $\nabla_{xx} L_{\rho_k}(x, \lambda^k, \rho^k)$.

Given $x \in [\hat{l}, \hat{u}]$ and $w \in I\!\!R^n$, let us define

$$\varphi(t) = L_{\rho_k}(x + tw, \lambda^k, \mu^k). \tag{13}$$

By the continuity of $\nabla L_{\rho_k}(x, \lambda^k, \rho^k)$,

$$\varphi'(t) = \nabla L_{\rho_k}(x + tw, \lambda^k, \rho^k)^T w$$

always exists and is continuous. The second derivative $\varphi''(t)$ also exists and is continuous when, for all $i = 1, \ldots, p$, one of the following three posssibilities hold:

- $g_i(x + tw) < 0$;
- $g_i(x + tw) > 0$;
- $g_i(x + tw) = 0$ and $\nabla g_i(x + tw)^T w = 0$.

Moreover, when $\varphi''(t)$ exists, it is given by

$$\varphi''(t) = w^T H_{\rho_k}(x + tw, \lambda^k, \mu^k) w + \rho_k \sum_{i \in I(x)} (\nabla g_i(x + tw)^T w)^2. \tag{14}$$

In order to verify that $U_{[\hat{l},\hat{u}]}(x)$ is convex we will use a mild sufficient assumption: for all $x, y \in [\hat{l}, \hat{u}]$ and $t \in [0, 1]$, the function φ defined by (13) has only a finite number of second-derivative discontinuities. (Since second-derivative discontinuities are related with changes of sign of the functions g_i, the above assumption holds except in very pathological situations.) Assume that $t_1 < \ldots < t_q$ are the discontinuity points of $\varphi''(t)$ in $[0, 1]$. Let $x, y \in [\hat{l}, \hat{u}]$ and define $w = y - x$. By the continuity of φ',

$$\varphi'(1) - \varphi'(0) = \int_0^{t_1} \varphi''(t) dt + \ldots + \int_{t_i}^{t_{i+1}} \varphi''(t) dt + \ldots + \int_{t_q}^1 \varphi''(t) dt.$$

Then, by (14),

$$\varphi'(1) - \varphi'(0) \geq \int_0^1 w^T H_{\rho_k}(x + tw, \lambda^k, \mu^k) w \; dt.$$

Therefore, by (10), (13) and the definition of the α's, we have:

$$\begin{aligned} [\nabla U_{[\hat{l},\hat{u}]}(y) - \nabla U_{[\hat{l},\hat{u}]}(x)]^T w &\geq \int_0^1 w^T H_{\rho_k}(x + tw, \lambda^k, \mu^k) w \; dt - \sum_{i=1}^n \alpha_i w_i^2 \\ &= \int_0^1 w^T [H_{\rho_k}(x + tw, \lambda^k, \mu^k) - \mathrm{diag}(\alpha_1, \ldots, \alpha_n)] w \; dt \geq 0. \end{aligned}$$

This implies that the underestimator $U_{[\hat{l},\hat{u}]}$ is convex.

The *piecewise* convex α-underestimator [34] can also be used instead of the convex α-underestimator to compute the Augmented Lagrangian underestimator at Step 5.2. The latter one may be significantly tighter than the α-underestimator, while its computation is more time consuming. Its computation deserves further explanations. Considering, for each interval

$[\hat{l}_i, \hat{u}_i], i = 1, \dots, n$, a partitioning in N_i subintervals with endpoints $\hat{l}_i \equiv \hat{v}_i^0, \hat{v}_i^1, \dots, \hat{v}_i^{N_i} \equiv \hat{u}_i$, a definition of the piecewise convex α-underestimator, equivalent to the one introduced in [34], follows:

$$\begin{array}{lcl} \Phi_{[\hat{l},\hat{u}]}(x) & = & L_{\rho_k}(x, \lambda^k, \mu^k) - \sum_{i=1}^n q_i(x), \\ q_i(x) & = & q_i^j(x), \text{ if } x \in [\hat{v}_i^{j-1}, \hat{v}_i^j], i = 1, \dots, n, j = 1, \dots, N_i, \\ q_i^j(x) & = & \alpha_i^j(\hat{v}_i^j - x_i)(x_i - \hat{v}_i^{j-1}) + \beta_i^j x_i + \gamma_i^j, i = 1, \dots, n, j = 1, \dots, N_i. \end{array}$$

In the definition above, α's, β's and γ's are such that $q_i(x), i = 1, \dots, n$, are continuous and smooth and $\Phi_{[\hat{l},\hat{u}]}(x)$ is a convex underestimator of $L_{\rho_k}(x, \lambda^k, \mu^k)$ within the box $[\hat{l}, \hat{u}]$.

One way to compute the α's is to compute one $\alpha \in I\!R^n$ for each one of the $\prod_{i=1}^n N_i$ subdomains and set $\alpha_i^j, i = 1, \dots, n, j = 1, \dots, N_i$, as the maximum over all the $[\alpha]_i$'s of the subdomains included in the "slice" $[(\hat{l}_1, \dots, \hat{l}_{i-1}, \hat{v}_i^{j-1}, \hat{l}_{i+1}, \dots, \hat{l}_n), (\hat{u}_1, \dots, \hat{u}_{i-1}, \hat{v}_i^j, \hat{u}_{i+1}, \dots, \hat{u}_n)]$. In this way, $\prod_{i=1}^n N_i$ α's must be computed.

After having computed the α's, and considering that each interval $[\hat{l}_i, \hat{u}_i]$ is partitioned in N_i identical subintervals of size $s_i = (\hat{u}_i - \hat{l}_i)/N_i$, β's and γ's can be computed as follows: for each $i = 1, \dots, n$,

$$\begin{array}{lcll} \beta_i^1 & = & -\frac{s_i}{N_i} \sum_{j=2}^{N_i} (N_i - j + 1)(\alpha_i^{j-1} + \alpha_i^j), & \\ \gamma_i^1 & = & -\beta_i^1 \hat{l}_i, & \\ \beta_i^j & = & \beta_i^{j-1} - s_i(\alpha_i^{j-1} + \alpha_i^j), & j = 2, \dots, N_i, \\ \gamma_i^j & = & \gamma_i^{j-1} + (\hat{l}_i + (j-1)s_i) s_i (\alpha_i^{j-1} + \alpha_i^j), & j = 2, \dots, N_i. \end{array}$$

Note that all the β's and γ's can be computed in $O(\sum_{i=1}^n N_i)$ operations using the formulae above, while the direct application of formulae present in [34] would imply in $O(\sum_{i=1}^n N_i^2)$ operations.

5 Numerical experiments

The method requires from the user subroutines to compute the objective function, the constraints and their first and second derivatives. In addition, it also requires subroutines to compute the interval objective function and the interval Hessian of the objective function, as well as the interval constraints and the interval Jacobian and interval Hessians of the constraints. (For interval analysis calculations we use the INTLIB library [27].) Finally, the user must also provide two extra subroutines: (i) a first subroutine to indicate which variables appear nonlinearly in the objective function or appear in a nonlinear constraint; and (ii) a second subroutine that computes the linear relaxations of the nonlinear constraints. This second subroutine is not mandatory and even an empty subroutine is a valid option. Note that the implemented method does not perform any kind of algebraic manipulation on the problems, which are solved as stated by the user. On the other hand, the implemented algorithm provides several subroutines to empirically verify the correctness of the derivatives, the interval calculations and the relaxations of the constraints.

For the practical implementation of Algorithms 2.1 and 4.1, we set $\tau = 0.5$, $\gamma = 10$, $\lambda_{\min} = -10^{20}$, $\mu_{\max} = \lambda_{\max} = 10^{20}$, $\lambda^1 = 0$, $\mu^1 = 0$ and

$$\rho_1 = \max\left\{10^{-6}, \min\left\{10, \frac{2|f(x^0)|}{\|h(x^0)\|^2 + \|g(x^0)_+\|^2}\right\}\right\},$$

where x_0 is an arbitrary initial point. As stopping criterion we used $\max(\|h(x^k)\|_\infty, \|V^k\|_\infty) \leq 10^{-4}$. The choice of the optimality gaps ε_k for the ε_k-global solution of the subproblems and ε for the ε-global solution of the original problem will be detailed below.

Codes are written in Fortran 77 (double precision) and they are free for download in *http://www.ime.usp.br/∼egbirgin/*. All the experiments were run on a 3.2 GHz Intel(R) Pentium(R) with 4 processors, 1Gb of RAM and Linux Operating System. Compiler option "-O" was adopted.

5.1 Choice of the optimality gaps

Several tests were done in order to analyze the behavior of the method in relation to the choice of the optimality gaps ε_k and ε for the solution of the k-th Augmented Lagrangian subproblem and the original problem, respectively. For this purpose, we selected a set of 16 small problems from the literature (see Appendix). All the problems were taken from [16] and their common characteristic is that the task of coding all the required subroutines mentioned at the beginning of the present section was not very hard.

As the selected problems were small, we choose two dense solvers for linear programming and linearly constrainted nonlinear programming problems. For solving linear programming problems we use subroutine `simplx` from the Numerical Recipes in Fortran [40]. To solve the linearly constrained optimization problems, we use Genlin [8], an active-set method for linearly constrained optimization based on a relaxed form of Spectral Projected Gradient iterations intercalated with internal iterations restricted to faces of the polytope. Genlin modifies and generalizes recently introduced box-constraint optimization algorithms [13].

In this first experiment, we solved the problems using the convex α-underestimator and using a variable $\varepsilon_k = \max\{\varepsilon, 10^{-k}\}$ and a fixed $\varepsilon_k = \varepsilon$. In both cases we considered $\varepsilon \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. Table 1 shows the results. In Table 1, n is the number of variables and m is the number of constraints. The number within parentheses is the number of linear constraints. "It" is the number of iterations of Algorithm 2.1 (outer iterations) which is equal to the number of subproblems (8) that are solved to ε_k-global optimality using the αBB approach, and "#Nodes" is the total number of iterations of Algorithm 4.1 (inner iterations).

The method with the eight combinations of ε_k and ε found the same global minimum in all the problems. In all the cases, the total CPU time, proportional to #Nodes, was very small. As expected, tighter gaps require more computations. However, the effort increase is not the same in all the problems, as it strongly depends on the tightness of the convex α-underestimator in relation to the Augmented Lagrangian function and the difficulty in closing that gap for each particular problem. On the other hand, the number of outer iterations of the Augmented Lagrangian method does not necessarily increase when the required gap is tightened. The reason why in the second part of Table 1 the number of outer iterations is the same, independently of the value of ε is the following: The number of iterations is, in this cases, determined exclusively by the necessity of getting feasibility on the penalized constraints (always with the same tolerance). The reason why in the first part of Table 1 the number of outer iterations sometimes increases when we reduce ε is related, not to the variation of ε but to the fact of using a variable ε_k gap for the subproblems, and can be better understood when we compare the $\varepsilon = 0.1$ column with the $\varepsilon = 10^{-4}$ column. In the first we require optimality gap $\varepsilon_k = 0.1$ in all the subproblems. On the other hand in the $\varepsilon = 10^{-4}$ column we require gaps $\varepsilon_k = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$ in the first four subproblems, Since the gap required for declaring final convergence is $\varepsilon = 10^{-4}$ in this case, it turns out that, except in very lucky situations, at least four outer iterations will be needed for completing the process. Of course, in the $\varepsilon = 0.1$ column convergence with only one outer iteration is possible, since $\varepsilon = \varepsilon_1 = 0.1$ are the gap tolerances for the original problem as well as for the first subproblem.

Figure 1 illustrates the numerical results on Table 1 using performance profiles [15]. As in [37], the number of nodes was used as a performance measurement. Figures 1(a–b) correspond to variable and fixed gaps for the Augmented Lagrangian subproblems, respectively. Both graphics compare the performance of the methods for tightening values of the required optimality gap. As expected, looser gaps allow the method to prune nodes faster and to solve a smaller total number of nodes. Figures 1(c–e) correspond to $\varepsilon = 10^{-2}, 10^{-3}, 10^{-4}$, respectively. Each graphic compares the version that uses $\varepsilon_k = \varepsilon \ \forall \ k$ with the version that uses a dynamic choice of ε_k. (For $\varepsilon = 10^{-1}$ both versions coincide.) As already mentioned, looser gaps reduce the effort to solve the subproblems while it may increase the number of subproblems that need to be solved to achieve the required gap for the original problem. The performance profiles show that there is no clear advantage on using one or the other strategy.

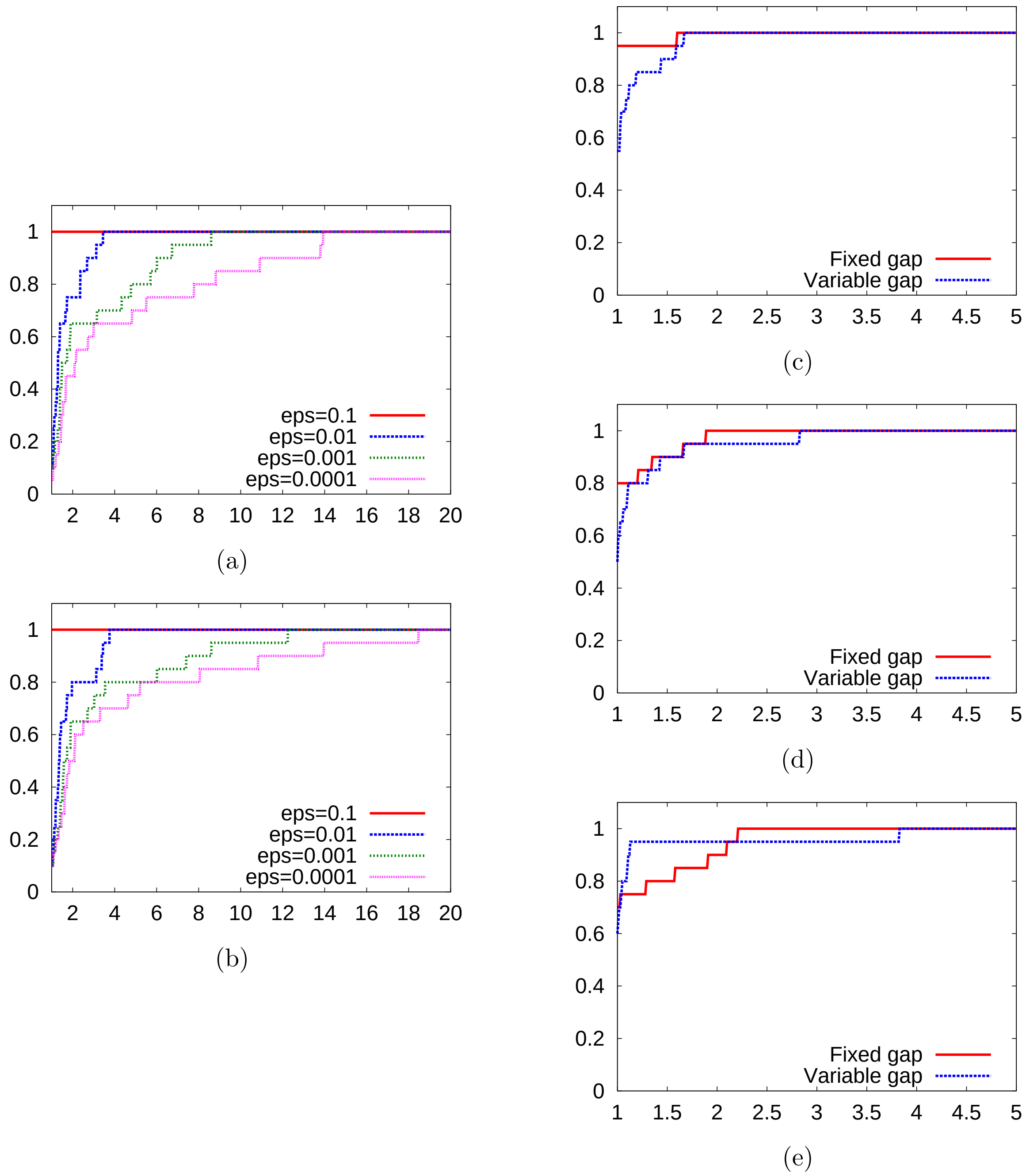
5.2 Convex α-underestimator versus piecewise convex α-underestimator

In a second set of experiments, we set $\varepsilon_k = \max\{\varepsilon, 10^{-k}\}$ and $\varepsilon = 10^{-4}$ (which provides the tightest global optimality certification) and compare the performance of the method using the convex α-underestimator and the *piecewise* convex α-underestimator with $N_i \equiv N \equiv 2 \ \forall i$. Table 2 shows the results. In Table 2, $f(x^*)$ is the global minimum. As expected, both versions of the method found the global minimum reported in the literature up to the prescribed tolerances. The version that uses the convex α-underestimator required less CPU time while the version that used the piecewise convex α-underestimator used a smaller number of inner iterations.

A remarkable situation happens for problem 2(a): the fact of the piecewise convex α-underestimator being tighter than the convex α-underestimator provides a neglegible reduction in the number of nodes from 104 to 102. On the other hand, for this particular problem, the piecewise convex α-underestimator requires much more time to be computed than the convex α-underestimator (see the discussion at the end of Section 4), making its usage counterproductive for the overall performance of the method. A similar behavior can be observed for other problems in Table 2. However, the piecewise convex α-underestimator results in a reduction of nodes in problems 3(a), 3(b) and 5, among others. Moreover, its usage may be profitable in critical situations in which memory (for saving the list of open nodes) may be a limited resource. The number of outer iterations employed by the methods remains the same, independently of the convex underestimator used in the αBB for solving the subproblems. This is not surprising, since the number of outer iterations depends only on the quality of the solution of the subproblems

Variable $\varepsilon_k = \max\{\varepsilon, 10^{-k}\}$										
Problem	n	m	$\varepsilon = 10^{-1}$		$\varepsilon = 10^{-2}$		$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-4}$	
			It	#Nodes	It	#Nodes	It	#Nodes	It	#Nodes
1	5	3	9	59861	9	82167	9	103919	9	124897
2(a)	11	8(6)	8	78	8	88	8	100	8	104
2(b)	11	8(6)	13	563	13	613	13	651	13	671
2(c)	11	8(6)	8	60	8	72	8	84	8	88
2(d)	12	9(7)	2	20	2	28	3	63	4	110
3(a)	6	5	6	1476	6	5078	6	12688	6	20366
3(b)	2	1	2	1294	2	3046	3	8719	4	18016
4	2	1	2	22	2	24	3	41	4	60
5	3	3	7	765	7	797	7	817	7	817
6	2	1	5	343	5	427	5	467	5	493
7	2	4(2)	3	149	3	247	3	283	4	446
8	2	2(1)	6	2340	6	3032	6	3496	6	3918
9	6	6(6)	1	3	1	3	1	3	1	5
10	2	2	3	95	3	123	3	139	4	206
11	2	1	2	22	2	38	3	105	4	194
12	2	1	8	240	8	314	8	334	8	370
13	3	2(1)	8	866	8	2048	8	3750	8	4178
14	4	3(3)	1	1	1	1	1	1	1	1
15	3	3(1)	1	99	2	266	3	565	4	1080
16	5	3(1)	6	116	6	362	6	698	6	902
Fixed $\varepsilon_k = \varepsilon$										
Problem	n	m	$\varepsilon = 10^{-1}$		$\varepsilon = 10^{-2}$		$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-4}$	
			It	#Nodes	It	#Nodes	It	#Nodes	It	#Nodes
1	5	3	9	59861	9	82167	9	103919	9	124897
2(a)	11	8(6)	8	78	8	88	8	100	8	104
2(b)	11	8(6)	13	563	13	613	13	651	13	671
2(c)	11	8(6)	8	60	8	72	8	84	8	88
2(d)	12	9(7)	2	20	2	28	2	38	2	50
3(a)	6	5	6	1476	6	5078	6	12694	6	20598
3(b)	2	1	2	1294	2	4382	2	9580	2	14012
4	2	1	2	22	2	26	2	34	2	38
5	3	3	7	765	7	797	7	819	7	825
6	2	1	5	343	5	443	5	515	5	557
7	2	4(2)	3	149	3	293	3	403	3	493
8	2	2(1)	6	2340	6	3100	6	3692	6	4304
9	6	6(6)	1	5	1	5	1	5	1	5
10	2	2	3	95	3	137	3	181	3	201
11	2	1	2	22	2	38	2	78	2	102
12	2	1	8	240	8	322	8	342	8	386
13	3	2(1)	8	866	8	3250	8	10608	8	15992
14	4	3(3)	1	1	1	1	1	1	1	1
15	3	3(1)	1	99	1	167	1	299	1	515
16	5	3(1)	6	116	6	362	6	698	6	934

Table 1: Performance of the global Augmented Lagrangian algorithm using different gaps for the global optimality of the subproblems and the original problem. In all the cases the method found the same global minimum.



(a)

(b)

(c)

(d)

(e)

Figure 1: Performance profile comparing the different choices for the gap tolerances of the Augmented Lagrangian subproblems and the original problem.

Problem	n	m	αBB			Piecewise αBB (N=2)			$f(x^*)$
			Time	It	#Nodes	Time	It	#Nodes	
1	5	3	18.86	9	124897	42.57	9	117679	2.9313E−02
2(a)	11	8(6)	0.13	8	104	1.30	8	102	−4.0000E+02
2(b)	11	8(6)	0.76	13	671	2.23	13	671	−6.0000E+02
2(c)	11	8(6)	0.16	8	88	1.09	8	88	−7.5000E+02
2(d)	12	9(7)	0.23	4	110	3.44	4	110	−4.0000E+02
3(a)	6	5	12.07	6	20366	72.83	6	15264	−3.8880E−01
3(b)	2	1	2.90	4	18016	5.62	4	13954	−3.8881E−01
4	2	1	0.00	4	60	0.00	4	58	−6.6666E+00
5	3	3	0.04	7	817	0.04	7	561	2.0116E+02
6	2	1	0.01	5	493	0.02	5	443	3.7629E+02
7	2	4(2)	0.02	4	446	0.02	4	372	−2.8284E+00
8	2	2(1)	0.15	6	3918	0.17	6	3522	−1.1870E+02
9	6	6(6)	0.00	1	5	0.01	1	5	−1.3402E+01
10	2	2	0.01	4	206	0.01	4	170	7.4178E−01
11	2	1	0.01	4	194	0.01	4	160	−5.0000E−01
12	2	1	0.01	8	370	0.01	8	350	−1.6739E+01
13	3	2(1)	0.47	8	4178	1.47	8	3820	1.8935E+02
14	4	3(3)	0.00	1	1	0.00	1	1	−4.5142E+00
15	3	3(1)	0.06	4	1080	0.18	4	1558	0.0000E+00
16	5	3(1)	0.15	6	902	0.22	6	758	7.0492E−01

Table 2: Performance of the global Augmented Lagrangian algorithm. As expected, the global solutions were found (up to the prescribed tolerance) in all the problems. While using the piecewise convex α-underestimator reduces the number of nodes, it increases the CPU time when compared with using the convex α-underestimator.

obtained, and not on the procedures used to obtain it. The same results (in terms of number of outer iterations) would be expected if a completely different robust global optimization method were used for solving the subproblems.

5.3 Influence of linear relaxations of the penalized constraints

In a third set of experiments we selected three problems (prodpl0, prodpl1 and optmass) from the Coconut Benchmark [46] with $m + n \geq 100$. For these medium-size problems we use Minos [35], that uses sparse linear algebra, for solving the linear programming as well as the linearly constrained nonlinear programming subproblems.

We solve these problems using the convex α-underestimator and $\varepsilon_k = \varepsilon = 10^{-1}$. In a first run, we set $W_{[\bar{l},\bar{u}]} = \emptyset$, which is equivalent to skip Step 4 of Algorithm 4.1, where the subdomains are shrinked using valid linear inequalities related to the penalized constraints. In a second run we made full use of the shrinking feature as described at the Remark 3 of Algorithm 4.1. Table 3 shows the results. In Table 3, n is the number of variables and m is the number of constraints (recall that the number within parentheses is the number of linear constraints), "It" is the number of iterations of Algorithm 2.1 (outer iterations), "#Nodes" is the total number of iterations of Algorithm 4.1 (inner iterations), and $f(x^*)$ is the global minimum. Both versions found the global minimum reported in the literature [46]. The feature being evaluated is related to the performance of the αBB method for solving the Augmented Lagrangian subproblems. Then, as expected, the number of outer iterations is the same for the two different runs. On the other hand, looking at the number of nodes needed to solve the subproblems in both cases, it is

Problem	n	m	Without shrinking subdomains		Shrinking subdomains with the help of linear relaxations of penalized constraints		$f(x^*)$
			It	#Nodes	It	#Nodes	
prodpl0	68	37(33)	9	673573	9	269	6.0917E+01
prodpl1	68	37(33)	5	2230001	5	681	5.3037E+01
optmass	70	55(44)	2	–	2	5703816	−1.8954E−01

Table 3: Evaluation of the box-shrinking feature of the αBB method by considering linear relaxations of the nonlinear penalized constraints in the Augmented Lagrangian framework.

easy to see that the shrinking feature greatly speeds up the convergence of the αBB method.

A consideration about CPU time is needed at this point. For the small problems considered in the previous subsections, the CPU time varies from a fraction of a second to a few seconds, and is directly related to the number of nodes in the Branch and Bound tree multiplied by the time needed to solve the $2n$ LP problems from Step 4 plus the two NLP problems from Steps 5.2 and 6 of Algorithm 4.1. So, the total amount of CPU time is small if the number of nodes is small or if n (the number of variables) is small. This is the case for all the problems of the previous subsections and for problems prodpl0 and propl1, for which the method used 7.35 and 21.43 seconds of CPU time, respectively. However, to solve the almost 840 millions of LP and the 12 millions of NLP subproblems of problem optmass, the method used around four days of CPU time. Several improvements may be suggested to alleviate this work. Probably, using a warm start procedure considering the solution of one PL to initialize another one at Step 4 of Algorithm 4.1 would help. Definitely, the precision required for the optimality of each NLP subproblem has a huge impact in the computational effort needed to satisfy the stopping criteria. Therefore, further improvement is necessary in order to address large problems in reasonable time. Moreover, there must be a balance between computational effort and the allowed tolerance for truly finding the global optimum.

5.4 Discusion

It is well known that the performance of a global optimization method for solving a problem strongly depends on the particular way the problem is modeled. So, it is worth mentioning that all the test problems, apart from Problems 2(a–d), were solved using the models shown in the Appendix without any kind of algebraic manipulation or reformulation. One of the key problem features that affects the behavior of the present approach is the number of variables that appear nonlinearly in the objective function or in a nonlinear constraint. Those are the variables that need to be branched in the αBB method. In the formulation of Problems 2(a–c) presented in the Appendix, *all* the nine variables need to be branched. As a result, the direct application of Algorithm 2.1 to those problems takes several seconds of CPU time. On the other hand, the addition of new variables $w_{78} = x_7 x_8$ and $w_{79} = x_7 x_9$ reduces the number of variables that need to be branched from nine to five and the CPU time used by the method to a fraction of a second. Almost identical reasoning applies to Problem 2(d).

Finally, note that, as Problems 11 and 16 have just lower-level constraints, the Augmented Lagrangian framework is not activated at all and its resolution is automatically made through the direct application of the αBB method.

These numerical results are the first step towards corroboration of the practical reliability of our method. The Augmented Lagrangian ideas are different from the ones that support current available global nonlinear programming solvers [36], in particular, the ones involved in the broad numerical study [37]. However, most of the interest of the Augmented Lagrangian approach relies on the fact that one is not restricted to the use of a particular linear-constraint solver. The Augmented Lagrangian method may use linear-constraint solvers quite different from αBB, if this is required by particular characteristics of the problem. Moreover, if the constraints of a particular problem can be divided into two sets, *Easy* and *Difficult*, and problems with Easy constraints (not necessarily linear) may be efficiently solved by some other solver, an appealing idea is to define the Augmented Lagrangian only with respect to the Difficult constraints and employ the other solver for the subproblems. This is the approach of [6] for constrained local optimization. External Penalty methods share this characteristic of the Augmented Lagrangian algorithm, but they tend to produce very difficult subproblems for big penalty parameters. In the Augmented Lagrangian method, the penalty parameter tends to remain bounded [6].

6 Final remarks

As already mentioned in [6], one of the advantages of the Augmented Lagrangian approach for solving nonlinear programming problems is its intrinsic adaptability to the global optimization problem. Namely, if one knows how to solve globally simple subproblems, the Augmented Lagrangian methodology allows one to globally solve the original constrained optimization problem. In this paper, we proved rigorously this fact, with an additional improvement of the Augmented Lagrangian method: the subproblems are redefined at each iteration using an auxiliary constraint set P_k that incorporates information obtained on the flight about the solution. Using the αBB algorithm for linearly constrained minimization subproblems, we showed that this approach is reliable. As a result, we have a practical Augmented Lagrangian method for constrained optimization that provably obtains ε-global minimizers. Moreover, the Augmented Lagrangian approach has a modular structure thanks to which one may easily replace subproblem global optimization solvers. This means that our method will be automatically improved as long as new global optimization solvers will be developed for the simple subproblems.

The challenge is improving efficiency. There are lots of unconstrained and constrained global optimization problems in Engineering, Physics, Chemistry, Economy, Computational Geometry and other areas that are not solvable with the present computer facilities. Our experiments seem to indicate that there is little to improve in the Augmented Lagrangian methodology, since the number of outer iterations is always moderate. The field for improvement is all concentrated in the global optimization of the subproblems. So, much research is expected in the following years in order to be able to efficiently solve more challenging practical problems.

The αBB global optimization algorithm is fully parallelizable in at least two levels: (i) subproblems with different subdomains can be solved in parallel; and (ii) the piecewise convex α-underestimator can be computed in parallel, reducing the number of nodes in the Branch & Bound algorithm without increasing the CPU time. Also the development of tighter underestimators for general nonconvex terms would be the subject of further research.

Acknowledgements

The authors are also indebted to the associated editor and an anonymous referee, whose comments helped a lot to improve the quality of the paper.

References

[1] C. S. Adjiman, I. P. Androulakis, C. D. Maranas and C. A. Floudas, A Global Optimization Method αBB for Process Design, *Computers and Chemical Engineering* 20, pp. S419–424, 1996.

[2] C. S. Adjiman, S. Dallwig, C. A. Floudas and A. Neumaier, A global optimization method, αBB, for general twice-differentiable constrained NLPs – I. Theoretical Advances, *Computers & Chemical Engineering* 22, pp. 1137–1158, 1998.

[3] C. S. Adjiman, I. P. Androulakis and C. A. Floudas, A global optimization method, αBB, for general twice-differentiable constrained NLPs – II. Implementation and computational results, *Computers & Chemical Engineering* 22, pp. 1159–1179, 1998.

[4] F. A. Al-Khayyal and H. D. Sherali, On finitely terminating Branch-and-Bound algorithms for some global optimization problems, *SIAM Journal on Optimization* 10, 1049–1057, 2000.

[5] R. Andreani, J. M. Martínez and M. L. Schuverdt, On the relation between the Constant Positive Linear Dependence condition and quasinormality constraint qualification, *Journal of Optimization Theory and Applications* 125, pp. 473–485, 2005.

[6] R. Andreani, E. G. Birgin, J. M. Martínez and M. L. Schuverdt, On Augmented Lagrangian methods with general lower-level constraints, *SIAM Journal on Optimization* 18, pp. 1286–1309, 2007.

[7] R. Andreani, E. G. Birgin, J. M. Martínez and M. L. Schuverdt, Augmented Lagrangian methods under the Constant Positive Linear Dependence constraint qualification, *Mathematical Programming* 111, pp. 5–32, 2008.

[8] M. Andretta, E. G. Birgin and J. M. Martínez, Partial Spectral Projected Gradient Method with Active-Set Strategy for Linearly Constrained Optimization, submitted, 2008.

[9] I. P. Androulakis, C. D. Maranas and C. A. Floudas, αBB: A Global Optimization Method for General Constrained Nonconvex Problems, *Journal of Global Optimization* 7, pp. 337–363, 1995.

[10] J. F. Bard, *Practical bilevel optimization. Algorithms and applications*, Kluwer Book Series: Nonconvex Optimization and its Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[11] D. P. Bertsekas, *Nonlinear Programming*, 2nd edition, Athena Scientific, Belmont, Massachusetts, 1999.

[12] E. G. Birgin, R. Castillo and J. M. Martínez, Numerical comparison of Augmented Lagrangian algorithms for nonconvex problems, *Computational Optimization and Applications* 31, pp. 31–56, 2005.

[13] E. G. Birgin and J. M. Martínez, Large-scale active-set box-constrained optimization method with spectral projected gradients, *Computational Optimization and Applications* 23, pp. 101–125, 2002.

[14] A. R. Conn, N. I. M. Gould and Ph. L. Toint, *Trust Region Methods*, MPS/SIAM Series on Optimization, SIAM, Philadelphia, 2000.

[15] E. D. Dolan and J. J. Moré, Benchmarking optimization software with performance profiles, *Mathematical Programming* 91, pp. 201–213, 2002.

[16] C. A. Floudas, *Deterministic global optimization: theory, methods and application*, Kluwer Academic Publishers, DorDrecht, Boston, London, 1999.

[17] C. A. Floudas, I. G. Akrotirianakis, S. Caratzoulas, C. A. Meyer and J. Kallrath, *Global optimization in the 21st century: Advances and challenges*, *Computers and Chemical Engineering* 29, pp. 1185–1202, 2005.

[18] C. A. Floudas and V. Visweeswaran, A global optimization algorithm (GOP) for certain classes of nonconvex NLPs – I. Theory, *Computers & Chemical Engineering* 14, pp. 1397–1417, 1990.

[19] D.Y. Gao, *Duality Principles in Nonconvex Systems: Theory, Methods, and Applications*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[20] D. Y. Gao, Canonical dual transformation method and generalized triality theory in nonsmooth global optimization, *Journal of Global Optimization* 17, pp. 127–160, 2000.

[21] D. Y. Gao, Perfect duality theory and complete solutions to a class of global optimization problems, generalized triality theory in nonsmooth global optimization, *Optimization* 52, pp. 467–493, 2003.

[22] D. Y. Gao, Canonical duality theory and solutions to constrained nonconvex quadratic programming, *Journal of Global Optimization* 29, pp. 337–399, 2004.

[23] D. Y. Gao, Complete solutions and extremality criteria to polynomial optimization problems, *Journal of Global Optimization* 35, pp. 131–143, 2006.

[24] D. Y. Gao, Solutions and optimality to box constrained nonconvex minimization problems, *Journal of Industry and Management Optimization* 3, pp. 1–12, 2007.

[25] M. R. Hestenes, Multiplier and gradient methods, *Journal of Optimization Theory and Applications* 4, pp. 303–320, 1969.

[26] R. Horst, P. M. Pardalos and M. V. Thoai, *Introduction to Global Optimization*, Kluwer Book Series: Nonconvex Optimization and its Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[27] R.B. Kearfott, M. Dawande, K. Du and C. Hu, Algorithm 737: INTLIB: A portable Fortran 77 interval standard-function library, *ACM Transactions on Mathematical Software* 20, pp. 447–459, 1994.

[28] L. Liberti, Reduction constraints for the global optimization of NLPs, *International Transactions in Operational Research* 11, pp. 33–41, 2004.

[29] L. Liberti and C. C. Pantelides, An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms, *Journal of Global Optimization* 36, pp. 161–189, 2006.

[30] J. Liebman, L. Lasdon, L. Schrage and A. Waren, *Modeling and Optimization with GINO*, The Scientific Press, Palo Alto, CA, 1986.

[31] H. Z. Luo, X. L. Sun and D. Li, On the Convergence of Augmented Lagrangian Methods for Constrained Global Optimization, *SIAM Journal on Optimization* 18, pp. 1209–1230, 2007.

[32] M. Manousiouthakis and D. Sourlas, A global optimization approach to rationally constrained rational programming, *Chemical Engineering Communications* 115, pp. 127–147, 1992.

[33] C. D. Maranas and C. A. Floudas, Global minimum potencial energy conformations for small molecules, *Journal of Global Optimization* 4, pp. 135–170, 1994.

[34] C. A. Meyer and C. A. Floudas, Convex underestimation of twice continuously differentiable functions by piecewise quadratic perturbation: spline αBB underestimators, *Journal of Global Optimization* 32, pp. 221–258, 2005.

[35] B. A. Murtagh and M. A. Saunders, *MINOS 5.4 User's Guide.* System Optimization Laboratory, Department of Operations Research, Standford University, CA.

[36] A. Neumaier, Complete search in continuous global optimization and constraints satisfaction, *Acta Numerica* 13, pp. 271–369, 2004.

[37] A. Neumaier, O. Shcherbina, W. Huyer and T. Vinkó, A comparison of complete global optimization solvers, *Mathematical Programming* 103, pp. 335–356, 2005.

[38] G. Pacelli and M. C. Recchioni, An interior point algorithm for global optimal solutions and KKT points, *Optimization Methods & Software* 15, pp. 225–256, 2001.

[39] M. J. D. Powell, A method for nonlinear constraints in minimization problems, in *Optimization*, R. Fletcher (ed.), Academic Press, New York, NY, pp. 283–298, 1969.

[40] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in Fortran: The Art of Scientific Computing*, Cambridge University Press, New York, 1992.

[41] R. T. Rockafellar, Augmented Lagrange multiplier functions and duality in nonconvex programming, *SIAM Journal on Control and Optimization* 12, pp. 268–285, 1974.

[42] H. S. Ryoo and N. V. Sahinidis, Global optimization of nonconvex NLPs and MINLPs with applications in process design, *Computers & Chemical Engineering* 19, pp. 551–566, 1995.

[43] H. S. Ryoo and N. V. Sahinidis, A branch-and-reduce approach to global optimization *Journal of Global Optimization* 8, pp. 107–138, 1996.

[44] N. V. Sahinidis, BARON: A general purpose global optimization software package, *Journal of Global Optimization* 8, pp. 201–205, 1996.

[45] N. V. Sahindis and I. E. Grossmann, Reformulation of the multi-period MILP model for capacity expansion of chemical processes, *Operations Research* 40, S127–S140, 1992.

[46] O. Shcherbina, A. Neumaier, D. Sam-Haroud, Xuan-Ha Vu and Tuan-Viet Nguyen, Benchmarking global optimization and constraint satisfaction codes, *Lecture Notes in Computer Science* 2861, pp. 211–222, 2003.

[47] H. D. Sherali and W. P. Adams, *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, Kluwer Book Series: Nonconvex Optimization and its Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[48] H. D. Sherali, A. Alameddine and T. S. Glickman, Biconvex models and algorithms for risk management problems, *American Journal of Mathematical and Management Science* 14, 197–228, 1995.

[49] H. D. Sherali and J. Desai, A global optimization RLT-based approach for solving the hard clustering problem, *Journal of Global Optimization* 32, 281–306, 2005.

[50] H. D. Sherali and J. Desai, A global optimization RLT-based approach for solving the fuzzy clustering problem, *Journal of Global Optimization* 33, 597–615, 2005.

[51] H. D. Sherali and C. H. Tuncbilek, A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique, *Journal of Global Optimization* 2, 101-112, 1992.

[52] H. D. Sherali and C. H. Tuncbilek, New reformulation-linearization/convexification relaxations for univariate and multivariate polynomial programming problems, *Operations Research Letters* 21, pp. 1–10, 1997.

[53] H. D. Sherali and H. Wang, Global optimization of nonconvex factorable programming problems, *Mathematical Programming* 89, 459–478, 2001.

[54] G. Stephanopoulos and A. W. Westerberg, The use of Hestenes' method of multipliers to resolve dual gaps in engineering system optimization, *Journal of Optimization Theory and Applications* 15, pp. 285–309, 1975.

[55] R. E. Swaney, Global solution of algebraic nonlinear programs, *AIChE Annual Meeting*, Chicago, IL, 1990.

[56] M. Tawarmalani and N. V. Sahinidis, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming*, Kluwer Book Series: Nonconvex Optimization and its Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[57] H. Tuy, *Convex Analysis and Global Optimization*, Kluwer Book Series: Nonconvex Optimization and its Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[58] V. Visweeswaran and C. A. Floudas, A global optimization algorithm (GOP) for certain classes of nonconvex NLPs – II. Application of theory and test problems, *Computers & Chemical Engineering* 14, pp. 1419–1434, 1990.

[59] A. W. Westerberg and J. V. Shah, Assuring a global optimum by the use of an upper bound on the lower (dual) bound, *Computers & Chemical Engineering* 2, pp. 83–92, 1978.

[60] Z. B. Zabinsky, *Stochastic Adaptive Search for Global Optimization*, Kluwer Book Series: Nonconvex Optimization and its Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003.

7 Appendix

In this Appendix we describe the global optimization test problems considered in the numerical experiments.

Problem 1. [35]

$$\begin{array}{ll} \text{Minimize} & (x_1-1)^2+(x_1-x_2)^2+(x_2-x_3)^3+(x_3-x_4)^4+(x_4-x_5)^4 \\ \text{subject to} & x_1+x_2^2+x_3^3=3\sqrt{2}+2 \\ & x_2-x_3^2+x_4=2\sqrt{2}-2 \\ & x_1x_5=2 \\ & -5\le x_i\le 5,\ i=1,\dots,5 \end{array}$$

Problem 2. *Haverly's pooling problem [3].*

$$\begin{array}{ll} \text{Minimize} & -9x_1-15x_2+6x_3+c_1x_4+10(x_5+x_6) \\ \text{subject to} & x_7x_8+2x_5-2.5x_1\le 0 \\ & x_7x_9+2x_6-1.5x_2\le 0 \\ & 3x_3+x_4-x_7(x_8+x_9)=0 \\ & x_8+x_9-x_3-x_4=0 \\ & x_1-x_8-x_5=0 \\ & x_2-x_9-x_6=0 \\ & (0,\dots,0)\le x\le(c_2,200,500,\dots,500) \end{array}$$

(a) $c_1=16$ and $c_2=100$; **(b)** $c_1=16$ and $c_2=600$; **(c)** $c_1=13$ and $c_2=100$.

Problem 2(d). A very similar version of the problem above but with an additional variable and different bounds [42].

$$\begin{array}{ll}\text{Minimize} & -9x_5 - 15x_9 + 6x_1 + 16x_2 + 10x_6 \\ \text{subject to} & x_{10}x_3 + 2x_7 - 2.5x_5 \leq 0 \\ & x_{10}x_4 + 2x_8 - 1.5x_9 \leq 0 \\ & 3x_1 + x_2 - x_{10}(x_3 + x_4) = 0 \\ & x_1 + x_2 - x_3 - x_4 = 0 \\ & x_3 + x_7 - x_5 = 0 \\ & x_4 + x_8 - x_9 = 0 \\ & x_7 + x_8 - x_6 = 0 \\ & (0, \ldots, 0, 1) \leq x \leq (300, 300, 100, 200, 100, 300, 100, 200, 200, 3)\end{array}$$

Problem 3. *Reactor network design [32].*

$$\begin{array}{ll}\text{Minimize} & -x_4 \\ \text{subject to} & x_1 + k_1x_1x_5 = 1 \\ & x_2 - x_1 + k_2x_2x_6 = 0 \\ & x_3 + x_1 + k_3x_3x_5 = 1 \\ & x_4 - x_3 + x_2 - x_1 + k_4x_4x_6 = 0 \\ & \sqrt{x_5} + \sqrt{x_6} \leq 4 \\ & (0, 0, 0, 0, 10^{-5}, 10^{-5}) \leq x \leq (1, 1, 1, 1, 16, 16)\end{array}$$

$k_1 = 9.755988\ 10^{-2}$; $k_2 = 0.99\ k_1$; $k_3 = 3.919080\ 10^{-2}$; $k_4 = 0.90\ k_3$.

Problem 3(b). A reformulation of Problem 3 eliminating some variables [16].

$$\begin{array}{ll}\text{Minimize} & -\left(\frac{k_1x_1}{(1+k_1x_1)(1+k_3x_1)(1+k_4x_2)} + \frac{k_2x_2}{(1+k_1x_1)(1+k_2x_2)(1+k_4x_2)}\right) \\ \text{subject to} & \sqrt{x_1} + \sqrt{x_2} \leq 4 \\ & (10^{-5}, 10^{-5}) \leq x \leq (16, 16)\end{array}$$

$k_1 = 9.755988\ 10^{-2}$; $k_2 = 0.99\ k_1$; $k_3 = 3.919080\ 10^{-2}$; $k_4 = 0.90\ k_3$.

Problem 4. [45]

$$\begin{array}{ll}\text{Minimize} & -x_1 - x_2 \\ \text{subject to} & x_1x_2 \leq 4 \\ & (0, 0) \leq x \leq (6, 4)\end{array}$$

Problem 5. *Water pumping system [42].*

$$\begin{array}{ll}\text{Minimize} & x_3 \\ \text{subject to} & 30x_1 - 6x_1^2 - x_3 = -250 \\ & 20x_2 - 12x_2^2 - x_3 = -300 \\ & 0.5(x_1 + x_2)^2 - x_3 = -150 \\ & (0, 0, 0) \leq x \leq (9.422, 5.903, 267.42)\end{array}$$

Problem 6. *Design of a reinforced concrete beam [30].*

$$\begin{array}{ll} \text{Minimize} & 29.4x_1 + 18x_2 \\ \text{subject to} & -x_1 + 0.2458x_1^2/x_2 \le -6 \\ & (0, 10^{-5}) \le x \le (115.8, 30) \end{array}$$

Problem 7. [58]

$$\begin{array}{ll} \text{Minimize} & x_1 + x_2 \\ \text{subject to} & x_1^2 + x_2^2 \le 4 \\ & -x_1^2 - x_2^2 \le -1 \\ & x_1 - x_2 \le 1 \\ & -x_1 + x_2 \le 1 \\ & (-2, -2) \le x \le (2, 2) \end{array}$$

Problem 8. [32]

$$\begin{array}{ll} \text{Minimize} & x_1^4 - 14x_1^2 + 24x_1 - x_2^2 \\ \text{subject to} & x_2 - x_1^2 - 2x_1 \le -2 \\ & -x_1 + x_2 \le 8 \\ & (-8, 0) \le x \le (10, 10) \end{array}$$

Problem 9. *Design of three-stage process system with recycle [54].*

$$\begin{array}{ll} \text{Minimize} & x_1^{0.6} + x_2^{0.6} + x_3^{0.4} - 4x_3 + 2x_4 + 5x_5 - x_6 \\ \text{subject to} & -3x_1 + x_2 - 3x_4 = 0 \\ & -2x_2 + x_3 - 2x_5 = 0 \\ & 4x_4 - x_6 = 0 \\ & x_1 + 2x_4 \le 4 \\ & x_2 + x_5 \le 4 \\ & x_3 + x_6 \le 6 \\ & (10^{-5}, 10^{-5}, 10^{-5}, 0, 0, 0) \le x \le (3, 4, 4, 2, 2, 6) \end{array}$$

Problem 10. [55]

$$\begin{array}{ll} \text{Minimize} & 2x_1 + x_2 \\ \text{subject to} & -16x_1x_2 \le -1 \\ & -4x_1^2 - 4x_2^2 \le -1 \\ & (0, 0) \le x \le (1, 1) \end{array}$$

Problem 11. [55]

$$\begin{array}{ll} \text{Minimize} & -2x_1x_2 \\ \text{subject to} & 4x_1x_2 + 2x_1 + 2x_2 \le 3 \\ & (0, 0) \le x \le (1, 1) \end{array}$$

Problem 12. [54]

$$\begin{array}{ll} \text{Minimize} & -12x_1 - 7x_2 + x_2^2 \\ \text{subject to} & -2x_1^4 - x_2 = -2 \\ & (0, 0) \le x \le (2, 3) \end{array}$$

Problem 13. [59]

$$\begin{array}{ll}\text{Minimize} & 35x_1^{0.6} + 35x_2^{0.6} \\ \text{subject to} & 600x_1 - 50x_3 - x_1x_3 = -5000 \\ & 600x_2 + 50x_3 = 15000 \\ & (10^{-5}, 10^{-5}, 100) \leq x \leq (34, 17, 300)\end{array}$$

Problem 14. *Design of two-stage process system [54].*

$$\begin{array}{ll}\text{Minimize} & x_1^{0.6} + x_2^{0.6} - 6x_1 - 4x_3 + 3x_4 \\ \text{subject to} & -3x_1 + x_2 - 3x_3 = 0 \\ & x_1 + 2x_3 \leq 4 \\ & x_2 + 2x_4 \leq 4 \\ & (10^{-5}, 10^{-5}, 0, 0) \leq x \leq (3, 4, 2, 1)\end{array}$$

Problem 15. *Chemical equilibrium problem [42].*

$$\begin{array}{ll}\text{Minimize} & 0 \\ \text{subject to} & x_3^2/(x_1x_2^3) = 0.000169 \\ & x_2/x_1 = 3 \\ & x_1 + x_2 + x_3 = 50 \\ & (10^{-5}, 10^{-5}, 0) \leq x \leq (12.5, 37.5, 50)\end{array}$$

Problem 16. *Heat exchanger network design [30].*

$$\begin{array}{ll}\text{Minimize} & x_1 + x_2 + x_3 \\ \text{subject to} & (x_4 - 1) - 12x_1(3 - x_4) = 0 \\ & (x_5 - x_4) - 8x_2(4 - x_5) = 0 \\ & (5 - x_5) - 4x_3 = 0 \\ & (0, 0, 0, 1, 1) \leq x \leq (1.5834, 3.6250, 1, 3, 4)\end{array}$$