A branch and bound approach for convex semi-infinite programming

Le Thi Hoai An[a*], Mohand Ouanes[b], and A.I.F. Vaz[c]

[a] *Laboratory of Theorical and Applied computer Science (LITA)*
UFR MIM, University of Lorraine
Ile du Saucly, 57045 Metz Cedex, France
[b] *Département de Mathématiques, Faculté des Sciences,*
Université de Tizi-Ouzou, Algérie
[c] *Production and Systems Department, Algoritmi Center,*
Engineering School, University of Minho, Portugal

(*Received 00 Month 200x; in final form 00 Month 200x*)

In this paper we propose an efficient approach for globally solving a class of convex semi-infinite programming (SIP) problems. Under the objective function and constraints (w.r.t. the variables to be optimized) convexity assumption, and appropriate differentiability, we propose a branch and bound exchange type method for SIP. To compute a feasible point for a SIP problem (and check feasibility) we need to solve a global optimization (sub-)problem, which is herein addressed by a branch and bound strategy. The major novelty of the proposed method consists in generating a sequence of feasible points for the SIP problem, obtained by a convex combination of a feasible point and the solution of a discretized finite optimization problem. A branch and bound strategy is also used to address the problem of minimizing the objective function, since we naturally obtain, as a result of the iterative process, bounds for the objective function. Under mild assumptions we prove convergence of the proposed algorithm. To illustrate the proposed approach, we provide some numerical results using some benchmark test problems.

Keywords:
Global optimization; Semi-infinite Programming; Exchange type method; Discretization scheme; Branch and Bound.

AMS Subject Classification: 90C26; 90C34.

1. Introduction

In this paper we consider a semi-infinite programming (SIP) problem of the form

$$\text{(SIP)} \qquad \begin{cases} f_* = \min_{x\in\mathbb{R}^n} f(x) \\ \qquad \text{s.t. } g_j(x,s) \le 0, \;\; j=1,\dots,q, \;\; \forall s \in S \subset \mathbb{R}^m, \;\; |S| = +\infty. \end{cases}$$

In (SIP), the objective function f is expressed in terms of a finite number (n) of optimization variables, x, while it is minimized subject to an infinite number of constraints, which are expressed over a compact set S of infinite cardinality. If S is independent of x then we have a *standard* SIP problem, otherwise (i.e. $S = S(x)$) we have a *generalized* SIP problem. This paper addresses standard SIP problems. Problems of this type arise in several engineering areas, like material

*Corresponding author. Email: hoai-an.le-thi@univ-lorraine.fr stress modeling, design under uncertainty, air pollution control [12], robot trajectory planning [11, 23, 37, 39], optimal signal sets design [1, 14], production planning [19, 40], and in machine learning [2]. Many approaches have been proposed to deal with SIP problems. Traditionally, they can be classified into three main classes: discretization methods, exchange methods, and methods based on local reduction (see e.g. [8, 18, 20, 29, 34], and reference therein). Recently, some works on feasibility enforcing methods have been developed (see, e.g., [33]). For an extensive survey and a complete list of bibliography on semi-infinite programming problems we refer to the paper of Hettich and Kortanek [12]. A recent paper dedicated to standard and generalized SIP can be found in [31].

We consider, in this paper, the particular case where the functions f and $g_j(.,s)$, $j = 1,\dots,q$, are convex and continuously differentiable in $\mathbb{R}^n$ and $\mathbb{R}^{n\times m}$, respectively. We are then faced with a so called convex semi-infinite programming problem. The herein developed theory can be easily extended to a SIP that considers q constraints. So, in order to simplify the notation and without loss of generality, we consider the SIP problem in the form

$$\text{(CSIP)} \qquad \begin{cases} f_* = \min_{x\in\mathbb{R}^n} f(x) \\ \qquad \text{s.t.} \quad g(x,s) \le 0, \;\; \forall s \in S \subset \mathbb{R}^m. \end{cases}$$

We further assume that (CSIP) has a non-empty interior of its feasible set (denoted by Ω), and S is a Cartesian product of intervals on $\mathbb{R}^m$, i.e. $S = [\alpha_1,\beta_1]\times\cdots\times[\alpha_m,\beta_m]$. We also assume that $g(x,.)$ is twice continuously differentiable on S.

Regular inequality constraints (dependent only on x) can also be considered in (CSIP), provided that they are convex.

A natural way to address problem (CSIP) is to consider a discretization of the set S (in an equally spaced grid of points). Usually discretization methods solve a sequence of finite problems for successive grid refinements. The idea is to successively compute an optimal solution to a so called discretized problem, which is a finitely constrained (discretized) approximation to the SIP problem, namely

$$\text{P}[\bar{S}] \qquad \begin{cases} \min_{x\in\mathbb{R}^n} f(x) \\ \text{s.t.} \quad g(x,s_i) \le 0, \;\; s_i \in \bar{S} \subset S, \end{cases}$$

where $|\bar{S}|$ is a finite number.

A conceptual discretization algorithm can be described as follows.

Conceptual discretization algorithm.

1) Define S_0 as the initial (discretized) set of points. Let $\widetilde{S}_0 = S_0$. Solve $\text{P}[\widetilde{S}_0]$ and let x^0 be the found solution. Set $k = 1$.
2) If x^{k-1} is not feasible $\forall s \in S_{k-1}$ (i.e. $\exists \bar{s} \in S_{k-1} : g(x^{k-1},\bar{s}) > 0$),
   - then include all points in S_{k-1} that make x^{k-1} infeasible into $\widetilde{S}_{k-1}$. Solve $\text{P}[\widetilde{S}_{k-1}]$ and let x^{k-1} be the found solution. Go to step 2 (keeping the value of k).
   - else if the maximum number of refinements is attained then stop. Otherwise build another set $\widetilde{S}_k$ from S_k and $\widetilde{S}_{k-1}$. Solve $\text{P}[\widetilde{S}_k]$ and let x^k be the found solution. Set $k = k + 1$ and go to step 2.

While it is easy to implement and a solver for finite optimization can be used, discretization methods for SIP have many drawbacks. These methods are also known as *outer approximation* methods and an infeasible solution for SIP is usually obtained. Also the SIP solution is only obtained when the grid is *close* to the set S (in the sense that the grid considers a huge number of points), leading to a high number of constraints to be considered in the discretized finite problem, if a solution with high accuracy is requested. In such a case an ill posed problem could be attained since a high number of related (by the functional g) constraints are considered.

It is well known that, if the grid density of S_k tends to zero, i.e. $\lim_{k\to+\infty} dist(S_k, S) = 0$, where $dist(S_k, S) = \max_{s\in S} \min_{\bar{s}\in S_k} \|\bar{s} - s\|$, and the level set of $P[S_k]$, at each iteration, is compact, any accumulation point of the sequence of solutions $\{x^k\}$ is a solution to the SIP problem ([28]).

Exchange type methods try to address some of the difficulties in discretization methods. In exchange methods, approximate solutions to the subproblem (we have as many subproblems as constraints in the (SIP))

$$\max_{s\in S} g(\bar{x}, s), \tag{1}$$

are computed, for a given $\bar{x}$.

The key idea is to consider the solution(s) of the (also known as lower level, in opposition to the minimization of f that is called the upper lever optimization problem) subproblem (1) to be added to an auxiliary set $\tilde{S}$ while previous added points may be dropped (exchange of points). As in discretization methods, a sequence of finite optimization problems are to be solved.

A conceptual exchange algorithm follows.

Conceptual exchange algorithm.

1) Let $\tilde{S}_0$ be given and $k = 0$.
2) Solve P$[\tilde{S}_0]$ and let x^0 be the found solution.
3) Approximately solve the lower level subproblem $S_k = arg \max_{s\in S} g(x^k, s)$.
4) If $g(x^k, s) \leq 0$, $\forall s \in S_k$ then stop. We have a feasible x^k that is optimal to P$[S_k]$, and, consequently, optimal to (CSIP).
5) Add the new constraints and eventually drop others ($\tilde{S}_{k+1} \subseteq \tilde{S}_k \bigcup S_k$).
6) Solve P$[\tilde{S}_{k+1}]$ and let x^{k+1} be the solution found.
7) Set $k = k + 1$ and go to step 3.

Since f and $g(., s)$ are convex functions, the discretized problem ($P[S_k]$) is a convex program to which several convex programming methods can be applied. The major difficulty with the mentioned algorithm is with the optimization problem (1), since it has to be solved for a global solution, and with the possible huge number of constraints in problem P$[\tilde{S}_k]$.

Having in mind the computation of a global optimum for (CSIP), we are, therefore, faced with two crucial questions to be studied from an algorithmic point of view: how to check the feasibility of x^k (i.e. how to globally solve problem (1)) and how to find a good SIP feasible solution.

In this paper we propose to use two integrated branch and bound algorithms that address these questions, i.e., one branch and bound algorithm for the upper level problem and another branch and bound algorithm for the lower level problem.

Firstly, we investigate an *inexpensive* adaptive branch and bound scheme applied to problem (1) to check the feasibility of a given point $\bar{x}$ for (CSIP). For a suitable presentation we consider problem (1) in the standard form of a global optimization problem, say the following minimization problem

$$\gamma_{\bar{x}} = \min\{-g(\bar{x}, s) : s \in S\} \ = \min\{\theta_{\bar{x}}(s) : s \in S\} \tag{2}$$

with $\theta_{\bar{x}}(s) = -g(\bar{x}, s)$. Checking the feasibility of $\bar{x}$ is equivalent to check the condition $\gamma_{\bar{x}} \geq 0$. Our aim is to construct tight lower bounds (LB) and upper bounds (UB) of $\gamma_{\bar{x}}$ and improving them via a Branch and Bound procedure. Checking the inequality $\gamma_{\bar{x}} \geq 0$ amounts to checking the conditions $LB \geq 0$ or $UB < 0$. Consequently, we may decide on the feasibility of $\bar{x}$ for (CSIP) before obtaining an optimal (global) solution to (2). We define a convex underestimator for $\theta_{\bar{x}}$ by using a linear interpolan and adding a quadratic term (see [9, 10] for other possible underestimator). In [7, 32] the αBB algorithm also constructs convex relaxations of the lower level problem (2), by adding a quadratic term to the function $\theta_{\bar{x}}$. The herein considered underestimator provides a similar error bound with the extra properties of being quadratic for $m = 2$ and a polynomial of degree m for $m > 2$ ([15, 16]), in opposition to the αBB method that is proposed only for $m = 1$ and the estimator is quadratic only if $\theta_{\bar{x}}$ is, by itself, linear or quadratic. Additionally αBB method is proposed to find first order stationary points. In [26] the lower bounding problem is obtained by a formulation that combines the first- and second-order KKT necessary conditions on the lower level problem. Other methods have also been proposed for SIP (see the ICR method in [3] and further studied in [4], where interval analysis is used. See also [22] where an interval method is used to solve the lower level problem and a genetic algorithm to solve the upper level problem). An exchange type method for convex SIP is proposed in [41]. The proposed method uses similar assumptions on the problem structure (convexity and differentiability) and problem (2) is addressed by using Newton's method taking as initial guesses points in an equally spaced grid of point.

For branch and bound type methods for global optimization the reader is pointed to [30, 36], and [35]. See also [25] where a new procedure for the generation of feasible points is proposed.

Secondly, we investigate a procedure to compute a feasible solution to (CSIP), in the case where $\bar{x}$ is not feasible for (CSIP). We first propose a way to find a feasible point for (CSIP). Knowing a feasible point we can improve it w.r.t. the objective function of (CSIP), during the iterations of the discretization scheme. So, the lower bound Lf and the upper bound Uf of f_* may be improved at each iteration, and the algorithm terminates when either we get a solution to (CSIP) or $Uf - Lf \leq \epsilon$, where ϵ is a given tolerance. In both cases the obtained solution is always feasible to (CSIP).

Our approach may be considered as an exchange type algorithm, as it considers a branch and bound algorithm to solve problem (1). The main algorithm is divided in two phases. The first phase is used to compute a feasible point for (CSIP). The second phase consists in a procedure that computes successive lower and upper bounds on the objective function f. For the upper-bounding procedure for (CSIP) we propose a simple formula to compute a better feasible point to (CSIP) from a feasible point and an optimal solution of the current discretized problem. Thanks to this procedure, the algorithm may find an optimal solution to (CSIP) before the termination of the standard discretization scheme (i.e. as soon as $\bar{x}$ is feasible for (CSIP)).

The main advantages of our algorithm are that in the first phase it furnishes, by a very simple calculation, a feasible solution to (CSIP). In the second phase it computes successive feasible approximation to the (CSIP) solution, meaning that we can stop the algorithm prematurely and still obtain a feasible approximation to the (CSIP) solution. We prove convergence, in this second phase, to a global optimum.

The rest of the paper is organized as follows. In Section 2 we propose an *adaptive branch and bound scheme* applied to the global optimization of problem (1), in order to check the (CSIP) feasibility of a given point $\bar{x}$. Section 3 is devoted to computing upper bounds of f_*. More precisely, we first present a procedure to find a feasible solution to (CSIP), and then show how to get a better (CSIP) feasible solution from a feasible solution. The description of our main algorithm for solving (CSIP) and its convergence to an optimal point are presented in Section 4. Finally, computational experiments are presented in the last section.

2. An adaptive Branch and Bound (B&B) algorithm for checking (CSIP) feasibility

We propose, in this section, an *adaptive* branch and bound scheme for problem (1) in its equivalent form (2), allowing to check the (CSIP) feasibility of a given point $\bar{x} \in \mathbb{R}^n$. Our algorithm is based on the following result, whose proof is trivial and therefore omitted.

Proposition 2.1: *Let LB and UB be real numbers such that $LB \leq \gamma_{\bar{x}} \leq UB$.*

i) If $LB \geq 0$, then $\gamma_{\bar{x}} \geq 0$ and consequently $\bar{x}$ is a feasible point for (CSIP).

ii) If $UB < 0$, then $\gamma_{\bar{x}} < 0$ and therefore $\bar{x}$ is not a feasible point for (CSIP).

iii) If $UB - LB \leq \epsilon$, for a sufficiently small positive number ϵ, and $LB < 0 \leq UB$, then $-\epsilon \leq LB < 0$ and we say that $\bar{x}$ is an ϵ-feasible point for (CSIP).

Before describing the B&B algorithm for problem (2) we present the three core operations: lower bounding – to compute a lower bound on $\theta_{\bar{x}}(s)$, upper bounding – to compute an upper bound on $\theta_{\bar{x}}(s)$ and branching – to branch a given subset of the feasible region.

To simplify our notation we use $\theta(s)$ instead of $\theta_{\bar{x}}(s)$, implicitly knowing that $\theta(s)$ also depends on $\bar{x}$.

2.1. *Lower bounding*

From the assumptions on the function g we known that $\theta(s)$ is twice continuous differentiable and we further assume that the Hessian of $\theta(s)$ can be evaluated at any element in S. Under these assumptions, we develop, in this section, an underestimator $LB_{\theta(s)}$ for $\theta(s)$. For the clarity of exposition we separate the univariate case from the multivariate case.

The lower bound on g can then be obtained by solving the following convex optimization problem,

$$\min_{s \in S} LB_{\theta(s)}, \tag{3}$$

whose analytic solution can be trivially obtained from the first order optimality conditions for the $m = 1, 2$ cases.

2.1.1. Univariate global optimization case

We first consider the case where $m = 1$. Consider the given bounded closed interval $\left[s^0, s^1\right] \subseteq [\alpha, \beta] \subset \mathbb{R}$. Let $w^j(s) : \mathbb{R} \to \mathbb{R}$, $j \in \{0, 1\}$, be the functions defined as

$$w^0(s) = \begin{cases} \frac{s^1-s}{s^1-s^0} & \text{if} \quad s^0 \le s \le s^1 \\ 0 & \text{otherwise} \end{cases}, \quad w^1(s) = \begin{cases} \frac{s-s^0}{s^1-s^0} & \text{if} \quad s^0 \le s \le s^1 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Clearly, we have

$$w^0(s) + w^1(s) = 1, \quad \forall s \in [s^0, s^1]$$

and, for $i, j \in \{0, 1\}$,

$$w^j(s^i) = \begin{cases} 0 & \text{if} \quad j \ne i \\ 1 & \text{otherwise.} \end{cases}$$

Let $L\theta(s)$ be the linear interpolant of $\theta(s)$ at points s^0, s^1 ([6]), given by

$$L\theta(s) = \sum_{i=0}^{1} \theta(s^i) w^i(s). \tag{5}$$

In [15] a tight quadratic underestimator for $\theta(s)$, on the interval $[s^0, s^1]$, is proposed, which is better than the well-known linear underestimator of g ([6]). The underestimator is given by

$$LB_{\theta(s)} = L\theta(s) - \frac{1}{2} K \left(s - s^0\right)\left(s^1 - s\right),$$

where K is a positive number such that $|\theta''(s)| \le K, \forall s \in [s^0, s^1]$, and $\theta''(s)$ denotes the $\theta(s)$ second derivative.

2.1.2. Multivariable global optimization case

In this section, we extend to the multivariate case the previous underestimator for $\theta(s)$. Let $\mathcal{S}$ be the bound and closed product of intervals $\Pi_{i=1}^{m}[s_i^0, s_i^1]$, whose vertex set is denoted by $V(\mathcal{S})$. An element in $V(\mathcal{S})$ is denoted by $v = (s_1^{i_1}, \ldots, s_m^{i_m})$ with $i_j \in \{0, 1\}$, $j = 1, \ldots, m$.

The w functions also need to be adapted for the multivariate case. Let $w_i^j(s)$: $\mathbb{R} \to \mathbb{R}$, $j \in \{0, 1\}$, $i = 1, \ldots, m$, be the functions defined as

$$w_i^0(s_i) = \begin{cases} \frac{s_i^1-s_i}{s_i^1-s_i^0} & \text{if} \quad s_i^0 \le s_i \le s_i^1 \\ 0 & \text{otherwise} \end{cases}, \quad w_i^1(s_i) = \begin{cases} \frac{s_i-s_i^0}{s_i^1-s_i^0} & \text{if} \quad s_i^0 \le s_i \le s_i^1 \\ 0 & \text{otherwise.} \end{cases}$$

The function $LB_{\theta(s)} : \mathbb{R}^m \to \mathbb{R}$ is then defined as

$$LB_{\theta(s)} = LB_{\theta(s_1,\ldots,s_m)} = \sum_{i_m=0}^{1}\left(\ldots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\ldots,s_m^{i_m})w_1^{i_1}(s_1)\right)\ldots\right)w_m^{i_m}(s_m) - \frac{1}{2}K\left(\sum_{i=1}^{m}(s_i - s_i^0)(s_i^1 - s_i)\right), \quad (6)$$

where K is a positive number such that

$$||H_{\theta(s)}||_\infty \le K, \ \forall s \in \mathcal{S}$$

and $H_{\theta(s)}$ denotes the function $\theta(s)$ Hessian matrix.

The next proposition is a generalization of the result in the univariate case. Let $\theta''_{s_is_j}(s)$ be the second derivative of $\theta(s)$ w.r.t. the variables s_i and s_j, $i,j = 1,\ldots,m$.

Proposition 2.2:

i) The functions $LB_{\theta(s)}$ and $\theta(s)$ agree for all the vertices v on the vertex set $V(\mathcal{S})$ of $\mathcal{S}$, i.e., $LB_{\theta(v)} = \theta(v)$, $\forall v \in V(\mathcal{S})$.

ii) If

$$K \ge \max_{s\in\mathcal{S}} \max_{i=1,\ldots,m} \left|\theta''_{s_is_i}(s)\right|,$$

then $LB_{\theta(s)}$ is a minorization of $\theta(s)$ on $\mathcal{S}$, i.e. $LB_{\theta(s)} \le \theta(s)$, $\forall s \in \mathcal{S}$.

iii) The function $LB_{\theta(s)}$ is convex on $\mathcal{S}$ if

$$K \ge \max_{s\in\mathcal{S}} \max_{i=1,\ldots,m} \sum_{j=1,j\neq i}^{m} \left|\theta''_{s_is_j}(s)\right|.$$

Remark 2.1: Proposition 2.2 is a consequence of Theorem 1 and 2 available in [16]. For a matter of completeness of the present paper, and since [16] is not widely available, we reproduce these results in section A.1.

2.2. *Upper bounding*

In global optimization, a good way to compute upper bounds for a given function is by using efficient local approaches or, for example, the overestimator described in [15, 16].

In this paper we just use the simplest way to determine an upper bound, which consists in computing the objective value function at known feasible points.

2.3. *Branching*

For box constrained optimization the adaptive w-subdivision developed in [27] has been shown to be efficient in several problems (see, e.g., [13, 17]). Thus, we are motivated to use this procedure in our B&B algorithm. A simple description follows.

Let $\mathcal{S}_l = \Pi_{i=1}^m \left[s_i^{0,l}, s_i^{1,l}\right]$ be the box chosen to be divided at iteration l, using the side $\left[s_{i_l}^{0,l}, s_{i_l}^{1,l}\right]$, where $i_l \in \arg\max\{s_i^{1,l} - s_i^{0,l}\}$. We divide $\left[s_{i_l}^{0,l}, s_{i_l}^{1,l}\right]$ at $s^{L,l} \in \arg\min\left\{LB_{\theta(s)} : s \in \mathcal{S}_l\right\}$, i.e., we get $\mathcal{S}_l^1 = \left[s_{i_l}^{0,l}, s_{i_l}^{L,l}\right] \times \Pi_{i=1,i\neq i_l}^m \left[s_i^{0,l}, s_i^{1,l}\right]$ and $\mathcal{S}_l^2 = \left[s_{i_l}^{L,l}, s_{i_l}^{1,l}\right] \times \Pi_{i=1,i\neq i_l}^m \left[s_i^{0,l}, s_i^{1,l}\right]$. Whenever $s_{i_l}^{L,l}$ equals $s_{i_l}^{0,l}$ or $s_{i_l}^{1,l}$ we consider $s_{i_l}^{L,l} = \frac{s_{i_l}^{1,l} + s_{i_l}^{0,l}}{2}$ for the branching procedure.

2.4. *Adaptive B&B algorithm for checking CSIP feasibility*

We are now in position to describe the full B&B algorithm, used to check for (CSIP) feasibility of a given point $x \in \Omega$. Whenever x is not feasible and as a sub-product, the algorithm provides a point $s^* \in S$ such that $g(x, s^*) > 0$.

Algorithm 2.1: Checking the (CSIP) feasibility

- **Initialization.** Given x (a point to check for (CSIP) feasibility) and let ϵ be a given sufficiently small positive number. Compute K, an upper bound of $\|H_{\theta(s)}\|$. Set $l = 0$, $\mathcal{S} = S = \prod_{i=1}^m [\alpha_i, \beta_i]$ and $\mathcal{M} = \{\mathcal{S}\}$. Compute $LB^0 = LB_{\theta(\bar{s}^0)}$, where $\bar{s}^0$ is obtained by solving the box constrained convex program

$$\bar{s}^0 = \arg\min\left\{LB_{\theta(s)} : s \in \mathcal{S}\right\}.$$

  Set $s^0 = \arg\min\left\{\theta(s) : s \in \left(V(\mathcal{S}) \cup \{\bar{s}^0\}\right) \subset S\right\}$ and $UB^0 = \theta(s^0)$.
- **While** $LB^l < 0$ and $UB^l \geq -\epsilon$ and $UB^l - LB^l > \epsilon$ **do**
  - Let $\mathcal{S}_l \in \mathcal{M}$ be the set such that LB^l equals its lower bound.
  - Bisect $\mathcal{S}_l$ into two sets: $\mathcal{S}_l^1$ and $\mathcal{S}_l^2$, accordingly to the strategy described in the previous section.
  - Compute $LB_1^l = LB_{\theta(\bar{s}^1)}$ and $LB_2^l = LB_{\theta(\bar{s}^2)}$, by solving the box constrained convex programs

$$\bar{s}^1 = \arg\min\left\{LB_{\theta(s)} : s \in \mathcal{S}^1\right\}, \qquad \bar{s}^2 = \arg\min\left\{LB_{\theta(s)} : s \in \mathcal{S}^2\right\}.$$

  - Update current best solution $s^{l+1} = \arg\min\left\{UB^l, \theta(\bar{s})\right\}$, with $\bar{s} \in (V(\mathcal{S}^1) \cup V(\mathcal{S}^2) \cup \{\bar{s}_1, \bar{s}_2\})$ and the upper bound $UB^{l+1} = \theta(s^{l+1})$.
  - Set $\mathcal{M} = \mathcal{M} \cup \{\mathcal{S}_l^i : LB_i^l < UB^{l+1} + \epsilon, i = 1, 2\} \backslash \{\mathcal{S}_l\}$. (Additionally, the sets from $\mathcal{M}$ whose estimated lower bound is greater than $UB^{l+1} + \epsilon$ can be removed).
  - Update the lower bound by setting $LB^{l+1} = \min\{LB_{\theta(\bar{s})}\}$, $\bar{s} \in \mathcal{S}$, $\mathcal{S} \in \mathcal{M}$ (i.e. set LB^{l+1} to the minimum lower bound for all sets in $\mathcal{M}$).
  - Set $l = l + 1$.
- **End while**

Algorithm **2.1** outputs $LB_x^* = LB^l$ and $UB_x^* = UB^l$ as approximations to the lower and upper bounds on $\theta(s)$, $s \in S$, respectively, and $s^* = s^l$ as the point where the upper bound is attained. Clearly the algorithm ends with one of the following three possible cases.

i) $LB_x^* \geq 0$, meaning that x is feasible for (CSIP).

ii) $LB_x^* < 0$, $UB_x^* \geq -\epsilon$ and $UB_x^* - LB_x^* \leq \epsilon$, meaning that s^* is an $\epsilon-$optimal solution of (1) and then x is $\epsilon-$feasible for (CSIP).

iii) $UB_x^* < -\epsilon$, meaning that x is not feasible for (CSIP). The algorithm ends with s^*, a point that violates the (CSIP) constraint and therefore s^* should be included in a discretized problem of (CSIP).

A proof of the finite termination of Algorithm **2.1** or the generation of a bounded sequence converging to a global solution of (2) is presented in [16] (again and for a matter of completeness of the present paper we reproduce this results in Appendix A.2).

3. Bounding procedures for the (CSIP) objective function

In this section we propose a procedure to compute a feasible point with a lower objective function value for (CSIP), starting with a feasible point, i.e. a point in the interior of the (CSIP) feasible set. We begin by presenting an algorithm to find a (CSIP) feasible point.

3.1. *Upper bounding: finding a feasible point for (CSIP)*

We determine a feasible point for (CSIP) by computing centers of several boxes T_k, generated during the discretization scheme. To check feasibility we use Algorithm **2.1**. The optimality can be checked by using lower bounds obtained while solving the discretized problems and once the feasibility holds. Let S_k and x^k denote, respectively, the finite (discretized) set of points in S at iteration k in the discretization scheme and the solution of the discretized problem. At iteration k, we use Uf_k and Lf_k to denote the best known upper bound and lower bound, respectively, of f_*, the optimal value of (CSIP).

Algorithm 3.1: Finding a feasible point of (CSIP)

- Set $k = 0$, *stop* $=$ *false*, *feasible* $=$ *false*, $Uf_0 = +\infty$. Let S_0 be a given finite set of chosen points in S such that problem $P([S_0])$ is bounded. Let x^0_{feas}, x^0_{disc}, and ϵ be given.
- **While** not *stop* and not *feasible* **do**

  1) Solve the following $2n$ convex programs for $j = 1, \ldots, n$,

$$\left\{\begin{array}{l} l_j^k = \min\limits_{x \in \mathcal{X}} x_j \\ \\ \quad \text{s.t.} \quad g(x, s_i) \leq 0, \\ \\ \quad s_i \in S_k \end{array}\right. \quad \text{and} \quad \left\{\begin{array}{l} u_j^k = \max\limits_{x \in \mathcal{X}} x_j \\ \\ \quad \text{s.t.} \quad g(x, s_i) \leq 0, \\ \\ \quad s_i \in S_k \end{array}\right. \tag{7}$$

  where $\mathcal{X}$ is set of bound constraints to prevent the problems unboundedness. The set $\mathcal{X}$ considered is a neighborhood (in the infinite norm) of x^k_{disc} with radius $\rho > 0$, i.e., we have $\mathcal{X} = \left\{x \in \mathrm{I\!R}^n : \left\|x - x^k_{disc}\right\|_\infty \leq \rho\right\}$.
  Set $T_k = \Pi_{j=1}^n \left[l_j^k, u_j^k\right]$.

  2) Compute the center c^k of T_k, i.e. $c^k = \frac{u^k + l^k}{2}$.

3) Apply Algorithm **2.1** to check the feasibility of c^k (considering x replaced by c^k).
   - If $LB^*_{c^k} \geq -\epsilon$ and $f(c_k) < Uf_k$ then set $Uf_k = f(c_k)$ and $x^{k+1}_{feas} = c_k$, else set $x^{k+1}_{feas} = x^k_{feas}$.
   - If $LB^*_{c^k} \geq 0$ then $feasible = true$, and go to step 8.
4) Solve the discretized problem P($[S_k]$) to get an optimal solution x^k_*. If problem P($[S_k]$) is unbounded then go to step 7.
5) Apply Algorithm **2.1** to check the feasibility of x^k_*.
   - Set $Lf_k = f(x^k_*)$ and $x^{k+1}_{disc} = x^k_*$. If x^k_* is feasible ($LB^*_{x^k_*} \geq -\epsilon$) then set $stop = true$ and go to step 8.
6) If $Uf_k - Lf_k \leq \epsilon$ then set $stop = true$, $x^{k+1}_{disc} = x^{k+1}_{feas}$ (x^{k+1}_{feas} is an (ϵ-)optimal solution of (CSIP)), and go to step 8.
7) Set $S_{k+1} = S_k \cup \left\{s^*_{c^k}\right\}$ if $s^*_{x^k_*}$ is not defined in step 4 otherwise set $S_{k+1} = S_k \cup \left\{s^*_{c^k}\right\} \cup \left\{s^*_{x^k_*}\right\}$, and set $k = k + 1$.
8) Continue.

- **End while**

Algorithm **3.1** ends if a feasible point or an optimal solution to (CSIP) is attained. In the case of $feasible = true$ (and $stop = false$) the algorithm outputs $x^*_{feas} = x^k_{feas}$, a feasible point to (CSIP) and $x^*_{disc} = x^k_{disc}$, the solution (not feasible to (CSIP)) of the discretized optimization problem P($[S_k]$). In the case of $stop = true$ the algorithm outputs $x^*_{disc} = x^k_{disc}$ as an optimal solution to (CSIP), since we have a solution to P($[S_k]$) that is feasible to (CSIP) or, by chance, a central point is optimal do (CSIP) (controlled by the upper and lower bounds on f). As additional output of the algorithm we have $Lf = Lf_k$ and $Uf = Uf_k$, representing lower and upper bounds on f, respectively.

Proposition 3.1: *Convergence of Algorithm* **3.1**. *Under the assumption of Ω to be a bounded compact set with non empty interior and given ρ big enough, Algorithm* **3.1** *generates a sequence of sets T_k such that $T_k \supset T_{k+1} \supset \cdots \supset \Omega$, $\forall k$, where Ω is the feasible set of problem (CSIP). Hence, there exists an iteration k_0, such that the center c^{k_0} of T_{k_0} is feasible.*

Proof: The results follows trivially by noting that $S_k \subset S_{k+1}$ whenever c^k is not feasible. The ρ radius is necessary to prevent problems (7) to be unbounded and ρ big enough is requested so problems (7) includes all (CSIP) feasible points in the feasible region, i.e., $T_k \supset \Omega$, $\forall k$. □

Remark 3.1:
Algorithm **3.1** can also be used for solving (CSIP), provided that the stopping criteria is changed to ignore the feasibility test. Such an algorithm differs from the standard discretization scheme due to steps 1, 2, 3, and 6, which consists in finding a feasible point to (CSIP) and the upper bound on f_*.

3.2. *Improving upper bounds: finding a better feasible point of (CSIP) from a feasible point.*

In this section we develop a scheme to compute a better (with lower objective function value) feasible point for (CSIP). Algorithm **3.1** allows us to assume that we have x^*_{feas}, a feasible point to (CSIP), and x^*_{disc}, an optimal point to the discretized optimization problem P$([S_*])$. As previously stated, x^*_{disc} is not feasible to (CSIP), otherwise we would be in the presence of an optimal point to (CSIP), and all the optimization process could be stopped.

Clearly we have $LB^*_{x^*_{feas}} \geq 0$, since x^*_{feas} is feasible to (CSIP), and $LB^*_{x^*_{disc}} < 0$, since x^*_{disc} is not feasible to (CSIP).

The following result establishes a way to obtain a feasible point whose objective function value is lower than for x^*_{disc}.

Proposition 3.2: *Let x^*_{feas} be a feasible point of (CSIP) and x^*_{disc} be an optimal solution of the discretized problem $P[S_*]$, which is not feasible for (CSIP). Let $LB^*_{x^*_{feas}} \geq 0$ and $LB^*_{x^*_{disc}} < 0$ be the lower bounds of $\gamma_{x^*_{feas}}$ and $\gamma_{x^*_{disc}}$, respectively, obtained at the end of Algorithm* **2.1***, while checking for the feasibility of x^*_{feas} and x^*_{disc}. Let*

$$\mu = \frac{-LB^*_{x^*_{disc}}}{-LB^*_{x^*_{disc}} + LB^*_{x^*_{feas}}}. \tag{8}$$

We have then,

i) The point $\widehat{x} = x^*_{disc} + \mu(x^*_{feas} - x^*_{disc}) = (1-\mu)x^*_{disc} + \mu x^*_{feas}$ is feasible for (CSIP).

ii) Either $\widehat{x}$ is an optimal solution for (CSIP) or $\widehat{x}$ is a better feasible point than x^*_{feas}, i.e. $f(\widehat{x}) < f(x^*_{feas})$.

Proof:

i) By construction we have that $LB^*_{x^*_{disc}} < 0$ and $LB^*_{x^*_{disc}} \leq \theta_{x^*_{disc}}(s) = -g(x^*_{disc}, s)$, $\forall s \in S$. The same applies to $LB^*_{x^*_{feas}} \geq 0$ and $LB^*_{x^*_{feas}} \leq \theta_{x^*_{feas}}(s) = -g(x^*_{feas}, s)$, $\forall s \in S$.
Using the convexity of g, in the variable x, we have

$$\begin{aligned} g(\widehat{x}, s) &= g((1-\mu)x^*_{disc} + \mu x^*_{feas}, s) \leq (1-\mu)g(x^*_{disc}, s) + \mu g(x^*_{feas}, s) \quad (9)\\ &\leq (1-\mu)\ (-LB^*_{x^*_{disc}}) + \mu(-LB^*_{x^*_{feas}}) \quad (10)\\ &= \mu(LB^*_{x^*_{disc}} - LB^*_{x^*_{feas}}) - LB^*_{x^*_{disc}} = 0, \quad (11) \end{aligned}$$

getting that $\widehat{x}$ is feasible for (CSIP).

ii) We will argue by contradiction. Suppose that we have $f(\widehat{x}) \geq f(x^*_{feas})$.
Using f convexity, we have

$$f(\widehat{x}) = f((1-\mu)x^*_{disc} + \mu x^*_{feas}) \leq (1-\mu)f(x^*_{disc}) + \mu f(x^*_{feas}).$$

Using $f(\widehat{x}) \geq f(x^*_{feas})$ it follows that

$$f(\widehat{x}) \leq (1-\mu_k)f(x^*_{disc}) + \mu_k f(x^*_{feas}) \leq (1-\mu)f(x^*_{disc}) + \mu f(\widehat{x}),$$

and therefore

$$f(\widehat{x}) \leq f(x^*_{disc}).$$

Noting that $\widehat{x}$ is feasible to (CSIP), x^*_{disc} is infeasible to (CSIP), and $f(x^*_{disc})$ is a lower bound of f_*, the optimal value of (CSIP), we arrive to:

a) $f(\widehat{x}) = f(x^*_{disc})$ and $\widehat{x}$ is optimal to (CSIP).

b) $f(\widehat{x}) < f(x^*_{disc})$, a contradiction (noting that $f(x^*_{disc})$ is a lower bound)

□

Remark 3.2 :
When $q > 1$ constraints are present the $LB^*_{x^*_{feas}}$ and $LB^*_{x^*_{disc}}$ in (8) are replaced by the lowest lower bounds for all the q constraints. It is an easy exercise to show that equations (9)-(11) are still valid for all g_i, $i = 1, \dots, q$, constraints.

3.3. *Lower bounding*

Traditionally we get lower bounds of f_* while solving the discretized problems.

4. The main algorithm

We are now in position to describe the main algorithm for (CSIP). The algorithm takes advantage of Algorithm **2.1** to check for iterates feasibility (and to provide a new point on S that violates the constraints). Algorithm **3.1** is used to obtain a feasible point (and as a sub-product a point that is optimal to the discretized problem, but is not feasible to (CSIP)). From the two points obtained by Algorithm **3.1**, we proceed iteratively, by computing a feasible point with better objective function value, until an optimal point for (CSIP) is obtained. We consider the main algorithm to be divided in two phases.

i) Phase 1: finding a feasible point for (CSIP) during the iterative discretization scheme.

ii) Phase 2: finding an optimal point for (CSIP), using a procedure to find a better feasible point during the iterative discretization scheme.

In both phases the adaptive branch and bound algorithm (Algorithm **2.1**) is used for checking the SIP feasibility. The main algorithm is terminated when the current best upper bound and lower bound coincide or optimality is reached.

Algorithm 4.1: A two phase algorithm for (CSIP)

1) **Initialization**. Choose D_0, a finite set of points in S, set $k = 0$ and $stop = false$. Let x^0 be a given initial guess.

2) **Phase 1**. Apply Algorithm **3.1** to find a feasible point x_{feas} and x^0_{disc}, Uf, Lf, and S_0. Get from Algorithm **2.1** (used in Algorithm **3.1**) $LB^*_{x_{feas}}$, $LB^*_{x^0_{disc}}$ and $s_{x^0_{disc}}$.
   If we got a $stop = true$ in Algorithm **3.1** then $x^* = x^0_{feas}$ is an optimal solution to (CSIP) and stop.
   Set $S_{k+1} = S_k \cup \{s_{x^k_{disc}}\}$ and $k = k + 1$.

3) **Phase 2**. Improve upper and lower bounds on f.

   While not *stop* **do**

   a) Set $\widehat{x}_k = x^k_{disc} + \frac{-LB^*_{x^k_{disc}}}{-LB^*_{x^k_{disc}} + LB^*_{x_{feas}}}(x_{feas} - x^k_{disc})$, $Uf_{k+1} = Uf_k$ and $Lf_{k+1} = Lf_k$.
      If $f(\widehat{x}_k) < Uf_k$ then set $Uf_{k+1} = f(\widehat{x}_k)$ and $x^{best} = \widehat{x}_k$.

b) If $Uf_{k+1} - Lf_{k+1} \leq \epsilon$ then set *stop* = *true* and $x^* = x^{best}$, an optimal solution to (CSIP). Stop the algorithm.

c) Solve the discretized problem P$[S_k]$ to get an optimal solution x_{disc}^{k+1}.

d) Apply Algorithm **2.1** to check x_{disc}^{k+1} feasibility. Get also $s_{x_{disc}^{k+1}}$ and $LB^*_{x_{disc}^{k+1}}$.

If x_{disc}^{k+1} is feasible then set *stop* = *true*, $Uf_{k+1} = Lf_{k+1} = f(x_{disc}^{k+1})$, $x^* = x_{disc}^{k+1}$, and stop the algorithm. Else set $Lf_{k+1} = f(x_{disc}^{k+1})$.

e) Set $S_{k+1} = S_k \cup \left\{ s_{x_{disc}^{k+1}} \right\}$ and $k = k + 1$.

End while

Algorithm **4.1** outputs x^*, an optimal solution to (CSIP).

Theorem 4.1 : *(Convergence theorem.) The two phase algorithm generates a sequence of points* $\{x_{disc}^k\}$ *and two sequences of values* $\{Uf_k\}$ *and* $\{Lf_k\}$ *such that*

i) The sequence $\{Uf_k\}$ *is decreasing and the sequence* $\{Lf_k\}$ *is increasing.*

ii) Either there is an iteration k_* *such that* $Uf_{k_*} = Lf_{k_*}$ *(getting an optimal solution for (CSIP)) or*

$$\lim_{k\to+\infty}(Uf_k - Lf_k) = 0.$$

In the last case any accumulation point of $\{x_{disc}^k\}$ *solves the problem (CSIP).*

Proof :

i) This result is immediate from the construction of Uf_k and Lf_k.

ii) If there is an iteration k_* such that $Uf_{k_*} = Lf_{k_*}$, then x^* is an optimal solution to (CSIP).

Just recall that if $\{x_{disc}^k\}$, for some k, is feasible for (CSIP) then it would also be optimal. Therefore we have that all points in the sequence $\{x_{disc}^k\}$ are not feasible for (CSIP), i.e. $LB^*_{x_{disc}^k} < 0$, $\forall k$.

Considering that $\mu_k = \frac{-LB^*_{x_{disc}^k}}{-LB^*_{x_{disc}^k} + LB^*_{x_{feas}}}$, we have that

$$\begin{aligned} Uf_k - Lf_k &= f(x^{best}) - f(x_{disc}^k) \\ &\leq f(\widehat{x}_k) - f(x_{disc}^k) \\ &= f(x_{disc}^k + \mu_k(x_{feas} - x_{disc}^k)) - f(x_{disc}^k) \\ &\leq \mu_k f(x_{feas}) + (1-\mu_k) f(x_{disc}^k) - f(x_{disc}^k) \\ &= \mu_k(f(x_{feas}) - f(x_{disc}^k)). \end{aligned}$$

Since $\lim_{k\to+\infty} -LB^*_{x_{disc}^k} = 0$ we have that $\lim_{k\to+\infty} \mu_k = 0$ and therefore $\lim_{k\to+\infty}(Uf_k - Lf_k) = 0$.

$\square$

5. Numerical experiments

5.1. *Implementation details*

We choose to implement Algorithm **4.1** using MATLAB [24]. While other, more efficient (w.r.t. CPU time), platform could be selected (e.g. using the C programming language), we choose to provide a more simple and widely available implementation in a high level programming language. We also choose to provide a general implementation of our proposed algorithm. An implementation for a given SIP optimization problem may lead to a somehow more efficient implementation. For example, estimates of K are difficult to obtain for a generic problem while they can be analytically obtained for the majority of the problems in our test set.

The first practical issue is the computation of K, where $\|H_{\theta(s)}\|_\infty \leq K$, $\forall s \in \mathcal{S}$, for a given x, requested by the underestimator function $LB_{\theta(s)}$. We choose to initialize K to be the maximum value of the Hessian infinite norm at all the vertices that define the feasible region and $K \geq 1$, i.e.,

$$K = \max\left\{\max_{s\in V(S)} \|H_{\theta(s)}\|, 1\right\}.$$

The value of K is then updated whenever new Hessian values are available during the B&B procedure. When K is updated all the B&B procedure has to restart (as some sets may have been erroneously removed from the set $\mathcal{M}$ in Algorithm **2.1**). A more sophisticated algorithm could be used to estimate K, as the one reported in [32] based on interval analysis.

Another implementation detail is related with Algorithm **3.1** initial set S_0. As stated in Remark **3.1**, steps 1, 2, and 3 are not crucial for the convergence of Algorithm **3.1** and, consequently, for the Algorithm **4.1** convergence. Not being able to obtain a feasible c^k will drive Algorithm **3.1**. Nevertheless, not being able to obtain a feasible c^k and to compute an x_*^k, $\forall k$, would lead Algorithm **3.1** to an infinite cycle. Therefore, the set S_0 is dynamically computed as described in the following algorithm, in the hope to obtain a suitable initial set S_0.

Algorithm 5.1:

(1) Set $\bar{h}_i^0 = (\beta_i - \alpha_i)/n_h$, $i = 1, \dots, m$, and set S_0^0 to be an equally spaced grid of points of step size $\bar{h}^0$. Let $r = 0$.
(2) While $P([S_0^r])$ is unbounded then set $\bar{h}^{r+1} = \bar{h}^r/2$ and $r = r + 1$.

The radius ρ is set to $\max(\|x_{disc}^k\|, 10)$ in Algorithm **3.1**, and the default value for n_h is 8.

While Algorithm **2.1** is not designed to provided a good lower bound on $\theta(s)$ when a feasible x is provided, $LB_{x_{feas}^*}$ plays an important role in Algorithm **4.1**. Algorithm **2.1** is forced to perform at least two interval subdivisions in order to provide an acceptable lower bound when x is feasible.

In order to keep the B&B algorithm manageable in time and computationally affordable a branch of a set when $h_i < 0.01 \times 10^{m-1}$, $i = 1, \dots, m$, is not allowed.

In spite off all the optimization subproblems considered are convex, due to our convexity assumptions on the CSIP problem, we choose to use the MATLAB `fmincon` solver. While efficient algorithm exist for convex optimization (e.g. [5]) we choose to make our implementation independent of other third party software.

The implemented solver is publicly available at `www.norg.uminho.pt/aivaz/csip.html`.

Table 1. Test problems. 'Problem' is the problem name, n is the number of finite variables, m is the number of infinite variables, and q is the number of infinite constraints.

Problem	n	m	q	Problem	n	m	q	Problem	n	m	q
andreson1	3	2	1	leon1	4	1	2	lin1	6	2	1
coopeN	2	1	1	leon2	6	1	2	liu1	2	1	1
fang1	50	1	1	leon3	6	1	2	liu2	2	1	1
fang2	50	1	1	leon4	7	1	2	liu3	16	1	2
fang3	50	1	1	leon5	8	1	2	polak1	4	2	2
ferris1	7	1	2	leon6	5	1	2	powell1	2	1	1
ferris2	7	1	1	leon7	5	1	2	priceK	2	1	1
goerner4	7	2	2	leon8	7	1	2	reemtsen1	11	3	2
goerner5	7	2	2	leon9	7	1	2	reemtsen2	10	2	2
goerner6	16	2	2	leon10	3	1	2	reemtsen3	10	2	2
hettich2	3	1	2	leon11	3	1	2	reemtsen4	37	2	2
hettich4	2	1	2	leon12	2	1	1	reemtsen5	11	3	2
hettich5	3	2	2	leon13	2	1	1	still1	2	1	1
hettich6	7	2	2	leon14	2	1	1	watson1	2	1	1
hettich7	7	2	2	leon15	2	1	1	watson3	3	1	1
hettich8	5	1	2	leon16	3	1	1	watson4a	3	1	1
hettich9	11	2	2	leon17	3	1	1	watson4b	6	1	1
hettich10	2	1	2	leon18	2	1	1	watson4c	8	1	1
hettich12	16	2	2	leon19	5	1	1	watson5	3	1	1
hettich13	2	2	1	leon20	2	1	1	watson6	2	1	1
hettich14*	2	2	1	leon21	2	1	2	watson7	3	2	1
kortanek1	2	1	1	leon22	2	1	1	watson8	6	2	1
kortanek2	2	2	1	leon23	3	1	4	watson9	6	2	1
kortanek3	7	1	1	leon24	4	1	5	zhou1	2	1	1
kortanek4	8	1	1								

*This problem has an additional finite constraint.

5.2. *Test problems*

We tested our implementation on 73 differentiable convex test problems from the test set available in SIPAMPL [38] that match our assumptions. The test set is reported in Table 1, where 'Problem' is the problem name under the SIPAMPL. The problems mathematical formulation can be obtained from the references in [38] or by downloading our own database from `www.norg.uminho.pt/aivaz/csip.html`.

SIPAMPL provides an interface between AMPL [21] and MATLAB, which is used in our implementation. AMPL provides automatic differentiation (first and second derivatives for the objective and constraints), allowing a fast and efficient coding of optimization problems. AMPL also provides a presolver phase which attempts to transform the problem into an equivalent one that is smaller and easier to solve.

Whenever possible we use the initial guess provided by (SIP)AMPL (proposed in the papers where the problems were first published). Please note that our proposed algorithm does not request a feasible initial guess. For problems that do not have an initial guess we just start with a vector of ones, i.e. $x^0 = (1, \ldots, 1)^T$.

5.3. *Numerical results*

In the absent of a solver that addresses all the problems proposed herein we just report our numerical results with a compare between the obtained objective function values against the previously reported (also available in the publicly available problem mathematical formulation as described in Section 5.2), whenever they are available. See [32] for a generalized semi-infinite programming solver, where a design centering problem is address.

The tolerance ϵ is chosen to be $\epsilon = 10^{-6}$, both for feasibility and for optimality.

We report the numerical results in Table 2 obtained with an 8 GB memory laptop Intel(R) Core(TM) i7 CPU running a Windows 7 operative system. We report for each problem the number of iterations in phase one (iterations performed in Algorithm **2.1**), the number of iterations in phase two (iterations performed in Algorithm **3.1**), the CPU time taken to obtain the solution, and the objective function value attained at the reported solution. CPU time is merely given as an indication of running times, since these times are computer load and programming language dependent. An efficient C programming language implementation can significantly reduce the presented figures. To allow a comparison we include the objective function value previously reported by other authors, whenever they are available. These objective function values where obtained under several types of methods for SIP, where an infeasible point may be obtained. Higher objective function values are possible for our implementation as the proposed method enforces feasibility (until a accuracy of $\epsilon = 10^{-6}$).

A further analysis and explanation of the results in Table 2 are now in order.

For the test problems considered, our implementation takes on average one iteration on phase one and 3 iterations on phase two to reach an optimal solution. The average CPU time is 3.73 seconds.

For the majority of the test problems (more than 60%) we achieved an absolute value of the difference between our obtained objective function and the previous reported objective function value (reported on column 'Diff.') lower than 10^{-4}. We have a maximum of 1,328612E-01 in 'Diff.'. We get an exact match (for the provided accuracy) between objective function values for 7 problems.

For 60 problems only one iteration on phase one is performed, meaning that a feasible or optimal point is immediately obtained. From these, 42 problems need to enter phase two, where the phase one obtained feasible point is driven to optimality. For 18 problems we get an optimal solution in one iteration of phase one.

A feasible point in not obtained in phase one until optimality is reached for 10 problems (phase two is not entered and more than one iteration is performed in phase one), confirming in practice Remark **3.1**.

Finally, from Table 2 we can conclude that the implementation of Algorithm **4.1** solved all the test problems in the SIPAMPL database that matched our assumptions, proving its robustness in getting a CSIP solution.

6. Conclusion

In this paper we propose and implement an exchange type algorithm for convex semi-infinite programming problems (CSIP). By assuming differentiability of the considered problem we are able to prove convergence of the proposed algorithm to a CSIP optimal point. The algorithm considers two phases. The first one is devoted to obtain a feasible point for CSIP. The second phase uses a strategy based on a feasible point and on an optimal point (not feasible to CSIP) for a discretized optimization problem in order to drive a sequence of iterates to CSIP optimality. A B&B algorithm is used to check the CSIP feasibility of a given point and another B&B algorithm is used in the second phase in order to obtain a CSIP optimal point.

We implemented the proposed algorithm in MATLAB and we provide extensive numerical results with a set of 73 test problems from the SIPAMPL database. Numerical results allows us to conclude that the proposed solver is robust (always getting a CSIP solution) and we provide directions on how to obtain a more efficient implementation.

As future work we plan to extend the proposed algorithm for non-convex SIP optimization problems.

Table 2. Numerical results.

Problem	Phase 1 Iter.	Phase 2 Iter.	CPU (seconds)	$f(x^*)$	Reported [38] $f(x^*)$	Diff.
andreson1	7	0	7.11	-3.333333E-01	-3.333333E-01	3.333333E-08
coopeN	1	0	1.19	-9.434727E-09	0.000000E+00	9.434727E-09
fang1	1	2	21.73	4.794255E-01	4.792677E-01	1.578200E-04
fang2	1	6	23.07	6.931482E-01	6.931481E-01	1.500000E-07
fang3	1	1	21.28	1.718282E+00	1.718536E+00	2.544600E-04
ferris1	1	5	5.99	4.881276E-04	4.880000E-04	1.276000E-07
ferris2	1	23	7.46	-1.785873E+00	-1.786880E+00	1.007000E-03
goerner4	1	2	8.25	5.331155E-02	5.332400E-02	1.245000E-05
goerner5	1	16	22.00	2.720494E-02	2.727500E-02	7.006000E-05
goerner6	1	9	46.58	1.082269E-03	1.077000E-03	5.269000E-06
hettich2	7	0	2.68	5.382455E-01	5.380000E-01	2.455000E-04
hettich4	1	0	0.36	1.000000E+00		(1)
hettich5	6	0	119.95	5.382415E-01	5.380000E-01	2.415000E-04
hettich6	1	8	55.04	2.805990E-02	2.810000E-02	4.010000E-05
hettich7	1	4	49.76	1.776457E-01	1.780000E-01	3.543000E-04
hettich8	1	3	2.29	5.643852E-03	(2)	
hettich9	1	12	84.65	3.468757E-03	3.470000E-03	1.243000E-06
hettich10	1	0	0.28	1.000000E+00	(2)	
hettich12	1	22	78.44	1.154055E-03	(2)	
hettich13	2	26	10.31	-2.236286E+00	(2)	
hettich14*	1	14	3.82	-2.121382E+00	(2)	
kortanek1	1	6	48.02	3.221170E+00	3.221175E+00	5.040000E-06
kortanek2	1	0	1.11	6.862915E-01	5.857864E-01	1.005051E-01
kortanek3	1	0	1.50	2.169838E-04	1.470768E-02	1.449070E-02
kortanek4	1	3	26.66	2.712498E-05	5.208333E-03	5.181208E-03
leon1	1	2	1.29	4.505040E-03	4.505000E-03	4.000000E-08
leon2	1	2	11.11	4.178053E-05	4.188000E-05	9.947000E-08
leon3	1	6	5.12	5.217055E-04	5.219000E-04	1.945000E-07
leon4	1	14	13.81	2.602607E-03	2.602800E-03	1.930000E-07
leon5	1	36	47.22	1.425647E-02	1.425650E-02	3.000000E-08
leon6	1	3	4.12	1.552448E-04	1.554000E-04	1.552000E-07
leon7	1	6	4.37	2.099721E-03	2.099700E-03	2.100000E-08
leon8	1	20	21.39	5.422198E-02	5.422210E-02	1.200000E-07
leon9	1	19	16.96	1.633809E-01	1.633810E-01	1.000000E-07
leon10	7	0	2.65	5.382455E-01	5.382453E-01	1.820000E-07
leon11	2	0	1.28	4.841439E-02	4.841440E-02	1.000000E-08
leon12	1	5	0.55	-9.999997E-01	-1.000000E+00	3.000000E-07
leon13	1	11	1.26	2.360679E-01	2.360680E-01	7.749979E-08
leon14	2	12	1.40	6.666670E-01	6.666666E-01	4.000000E-07
leon15	1	7	0.95	-6.666667E-01	-6.666667E-01	3.333333E-08
leon16	1	0	0.22	1.859141E+00	1.726280E+00	1.328612E-01
leon17	1	0	0.19	-2.000000E+00	-2.000000E+00	0.000000E+00
leon18	18	0	3.63	-1.750000E+00	(2)	
leon19	1	12	2.12	7.858409E-01	(2)	
leon20	3	1	1.68	3.238012E-01	3.238015E-01	3.000000E-07
leon21	4	0	3.73	-9.966068E+01	-9.966067E+01	8.000000E-06
leon22	1	3	0.59	-1.047214E+01	-1.047214E+01	4.040000E-06
leon23	1	0	0.51	-3.085714E+01	-3.085714E+01	2.000000E-06
leon24	1	0	1.54	-1.199868E+01	(2)	
lin1	1	45	29.80	-1.824437E+00	(2)	
liu1	1	0	0.19	-4.653846E+00	-4.653850E+00	4.000000E-06
liu2	8	0	2.23	-3.363442E+00	-3.363500E+00	5.800000E-05
liu3	7	0	27.36	1.541178E+02	1.541176E+02	2.200000E-04
polak1	1	18	32.93	5.443703E+00	(2)	
powell1	1	6	0.92	-1.000000E+00	-1.000000E+00	0.000000E+00
priceK	1	3	0.51	-3.000000E+00	-3.000000E+00	0.000000E+00
reemtsen1	1	0	126.89	1.516059E-01	1.524860E-01	8.801000E-04
reemtsen2	1	8	101.45	5.833739E-02	5.835897E-02	2.158000E-05
reemtsen3	1	19	166.33	7.348724E-01	7.354679E-01	5.955000E-04
reemtsen4	1	52	451.09	2.090823E-04	1.140057E-02	1.119149E-02
reemtsen5	1	0	145.13	8.867492E-02	8.893175E-02	2.568300E-04
still1	1	0	0.39	1.000000E+00	1.000000E+00	0.000000E+00
watson1	1	0	0.22	-2.500016E-01	-2.500000E-01	1.600000E-06
watson3	1	0	0.53	5.334687E+00	5.334690E+00	3.000000E-06
watson4a	1	11	1.78	6.490420E-01	6.490420E-01	0.000000E+00
watson4b	1	10	2.22	6.160851E-01	6.168760E-01	7.909000E-04
watson4c	1	7	2.82	6.156891E-01	6.166070E-01	9.179000E-04
watson5	1	5	0.84	4.301184E+00	4.301191E+00	7.000000E-06
watson6	1	0	0.41	9.715891E+01	9.715890E+01	1.000000E-05
watson7	1	0	1.23	1.000000E+00	1.000000E+00	0.000000E+00
watson8	1	20	21.89	2.435598E+00	2.435644E+00	4.600000E-05
watson9	1	0	16.44	-1.200000E+01	-1.200000E+01	0.000000E+00
zhou1	14	0	3.09	1.783937E-01	2.360538E-01	5.766011E-02

(1) Infinite number of solutions with objective function values in the set $[0.75, 1]$.

(2) Objective function value not available at the SIPAMPL problems database.

Acknowledgements

The first author acknowledges the support from the FEDER of Lorraine under the project INNOMAD. The third author acknowledges the support from the FEDER under COMPETE program, by FCT under project FCOMP-01-0124-FEDER-022674, and from the University of Lorraine. This work was partially done under a visit to the University of Lorraine, France.

References

[1] M.G. Anthony and A.J. Kearsley. Optimal signal sets for non-gaussian detectors. *SIAM J. Optim.*, 9:316–326, 1997.

[2] K.P. Bennett and E. Parrado-Hernández. The interplay of optimization and machine learning research. *J. Mach. Learn. Res.*, 7:1265–1281, 2006.

[3] B. Bhattacharjee, W.H. Green, and P. Barton. Interval methods for semi-infinite programs. *Comput. Optim. Appl.*, 30:63–93, 2005.

[4] B. Bhattacharjee, P. Lemonidis, W.H. Green Jr., and P.I. Barton. Global solution of semi-infinite programs. *Math. Program.*, 103:283–307, 2005.

[5] Inc. CVX Research. CVX: Matlab software for disciplined convex programming, version 2.0 beta. `http://cvxr.com/cvx`, 2012.

[6] C. de Boor. *A Practical Guide to Splines, revised edition*, volume 27 of *Applied Mathematical Sciences*. Springer-Verlag, 2001.

[7] C.A. Floudas and O. Stein. The adaptive convexification algorithm: A feasible point method for semi-infinite programming. *SIAM J. Optim.*, 18:1187–1208, 2007.

[8] M.A. Goberna and M.A. Lopez, editors. *Semi-infinite programming: recent advances*, volume 57 of *Nonconvex Optimization and Its Applications*. Kluwer, 2001.

[9] C.E. Gounaris and C.A. Floudas. Tight convex underestimators for C-2-continuous problems: I. univariate functions. *Journal of Global Optimization*, 42:51–67, 2008.

[10] C.E. Gounaris and C.A. Floudas. Tight convex underestimators for C-2-continuous problems: IU. multivariate functions. *J. Global Optim.*, 42:69–89, 2008.

[11] E. Haaren-Retagne. *A Semi-Infinite Programming Algorithm for Robot Trajectory Planning*. PhD thesis, University of Trier, 1992.

[12] R. Hettich and K. Kortanek. Semi-infinite programming: Theory, methods and applications. *SIAM Rev.*, 35:380–429, 1993.

[13] H. Konno and A. Wijayanayake. Portfolio optimization under dc transaction costs and minimal transaction unit constraints. *J. Global Optim.*, 22:137–154, 2002.

[14] C.T. Lawrence. *A Computationally Efficient Feasible Sequencial Quadratic Programming Algorithm*. PhD thesis, Institute for Systems Research, 1998.

[15] H.A. Le Thi and M. Ouanes. Convex quadratic underestimation and branch and bound for univariate global optimization with one nonconvex constraint. *RAIRO Oper. Res.*, 40:285–302, 2006.

[16] H.A. Le Thi, M. Ouanes, and T.P. Nguyen. A branch and bound method for multivariate global optimization with box constraints. In *Proceedings du colloque International sur l'Optimisation et les Systèmes d'Information COSI'06*, pages 325–336, Alger, June 11-13 2006.

[17] H.A. Le Thi and T. Pham Dinh. Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. *J. Global Optim.*, 11:253–285, 1997.

[18] E. Levitin and R. Tichatschke. A branch-and-bound approach for solving a class of generalized semi-infinite programming problems. *J. Global Optim.*, 13:299–315, 1998.

[19] Y. Li and D. Wang. A semi-infinite programming model for earliness/tardiness production planning with simulated annealing. *Math. Comput. Modelling*, 26:35–42, 1997.

[20] Y. Liu, S. Ito, H.W.J. Lee, and K.L. Teo. Semi-infinite programming approach to continuously-constrained linear-quadratic optimal control problems. *J. Optim. Theory Appl.*, 108(3):617–632, 2001.

[21] AMPL Optimization LLC. AMPL: A modeling language for mathematical programming. `http://www.ampl.com`, 2012.

[22] C.G. Lo Bianco and A. Piazzi. A hybrid algorithm for infinitely constrained optimization. *Internat. J. Systems Sci.*, 32:91–102, 2001.

[23] S.P. Marin. Optimal parametrization of curves for robot trajectory design. *IEEE Trans.*

Automat. Control, 33:209–214, 1988.
[24] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.
[25] A. Mitsos. Global optimization of semi-infinite programs via restriction of the right-hand side. *Optimization*, 60:1291–1308, 2011.
[26] A. Mitsos, P. Lemonidis, C.K. Lee, and P.I. Barton. Relaxation-based bounds for semi-infinite programs. *SIAM J. Optim.*, 19:77–113, 2008.
[27] T.Q. Phong, H.A. Le Thi, and P. Dinh Tao. On globally solving linearly constrained indefinite quadratic minimization problems by decomposition branch and bound method. *RAIRO Oper. Res.*, 30:31–49, 1996.
[28] R. Reemsten and S. Gorner. Numerical methods for semi-infinite programming: A survey. In R. Reemsten and J. Ruckmann, editors, *Semi-infinite Programming*, volume 25 of *Nonconvex Optimization and Its Applications*, pages 195–275, Dordrecht, Netherlands, 1998. Kluwer Academic Publishers.
[29] R. Reemtsen and J.-J. Rückmann, editors. *Semi-infinite programming*, volume 25 of *Nonconvex Optimization and Its Applications*, Dordrecht, Netherlands, 1998. Kluwer Academic Publishers.
[30] H.S. Ryoo and N.V. Sahinidis. A branch-and-reduce approach to global optimization. *J. Global Optim.*, 8:107–138, 1996.
[31] O. Stein. How to solve a semi-infinite optimization problem. *European J. Oper. Res.*, 223:312–320, 2012.
[32] O. Stein and P. Steuermann. The adaptive convexification algorithm for semi-infinite programming with arbitrary index sets. *Math. Program.*, 136:183–207, 2012.
[33] O. Stein and A. Winterfeld. Feasible method for generalized semi-infinite programming. *J. Optim. Theory Appl.*, 146:419–443, 2010.
[34] G. Still. Discretization in semi-infinite programming: the rate of the convergence. *Math. Program.*, 91:53–69, 2001.
[35] M. Tawarmalani and N.V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming*, volume 65 of *Nonconvex Optimization and its Applications.* Kluwer Academic Publishers, Boston, 2002.
[36] M. Tawarmalani and N.V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Math. Program.*, 103:225–249, 2005.
[37] A.I.F. Vaz, E.M.G.P. Fernandes, and M.P.S.F. Gomes. Robot trajectory planning with semi-infinite programming. *European J. Oper. Res.*, 153:607–617, 2004.
[38] A.I.F. Vaz, E.M.G.P. Fernandes, and M.P.S.F. Gomes. SIPAMPL: Semi-infinite programming with AMPL. *ACM Trans. Math. Software*, 30:47–61, 2004.
[39] O. von Stryk and M. Schlemmer. Computational optimal control. volume 115 of *International Series of Numerical Mathematics*, chapter Optimal control of the industrial robot manutec R3, pages 367–382. Birkhauser Verlag, 1994.
[40] D. Wang and S.-C. Fang. A semi-infinite programming model for earliness/tardiness production planning with a genetic algorithm. *Comput. Math. Appl.*, 31:95–106, 1996.
[41] L. Zhang, S. Wu, and M. Lopez. A new exchange method for convex semi-infinite programming. *SIAM Journal on Optimization*, 20:2959–2977, 2010.

Appendix A. Theoretical results from [16]

We provide in this section some results from [16] related with the underestimating function $LB_{\theta(s)}$ for the multidimensional case and with the convergence of the B&B Algorithm **2.1**.

A.1. *A multidimensional convex underestimator*

Theorem A.1 : *(Theorem 1 in [16]) Let $\mathcal{S}$ be the bound and closed set defined by a product of intervals $\Pi_{i=1}^m [s_i^0, s_i^1]$, whose vertex set is denoted by $V(\mathcal{S})$. Let $LB_{\theta(s)}$ be the multidimensional function defined in* (6)*.*

i) The functions $LB_{\theta(s)}$ and $\theta(s)$ agree for all vertices $v \in V(\mathcal{S})$.

ii) $LB_{\theta(s)}$ is a minorization of $\theta(s)$ on $\mathcal{S}$, i.e. $LB_{\theta(s)} \leq \theta(s)$, $\forall s \in \mathcal{S}$, if

$$K \geq \max_{s\in\mathcal{S}} \max_{i=1,\dots,m} |\theta''_{s_is_i}(s)|.$$

Proof:

i) By definition we have

$$LB_{\theta(s)} = \sum_{i_m=0}^{1}\left(\dots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\dots,s_m^{i_m})w_1^{i_1}(s_1)\right)\dots\right)w_m^{i_m}(s_m) - \frac{1}{2}K\left(\sum_{i=1}^{m}(s_i-s_i^0)(s_i^1-s_i)\right). \quad \text{(A1)}$$

Let $v = (s_1^{v_1},\dots,s_m^{v_m})$, $v_i \in \{0,1\}$, $i = 1,\dots,m$ be a vertex of $\mathcal{S}$, i.e. $v \in V(\mathcal{S})$. We have

$$LB_{\theta(s_1^{v_1},\dots,s_m^{v_m})} = \sum_{i_m=0}^{1}\left(\dots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\dots,s_m^{i_m})w_1^{i_1}(s_1^{v_1})\right)\dots\right)w_m^{i_m}(s_m^{v_m}) - \frac{1}{2}K\left(\sum_{i=1}^{m}(s_i^{v_i}-s_i^0)(s_i^1-s_i^{v_i})\right). \quad \text{(A2)}$$

Clearly,

$$\frac{1}{2}K\left(\sum_{i=1}^{m}(s_i^{v_i}-s_i^0)(s_i^1-s_i^{v_i})\right) = 0, \quad \text{(A3)}$$

since $v_i \in \{0,1\}$, $i = 1,\dots,m$, which makes each term in the sum to be zero.

Considering the definition of $w_j^{i_j}(s)$, $i_j \in \{0,1\}$, $j = 1,\dots,m$, it follows that

$$w_j^0(s_j^0) = w_j^1(s_j^1) = 1 \quad \text{and} \quad w_j^0(s_j^1) = w_j^1(s_j^0) = 0, \quad j = 1,\dots,m.$$

Therefore, noting that each $w_j^{i_j}(s_j^{v_j})$, $j = 1,\dots,m$, is zero (when $i_j \neq v_j$) or one (when $i_j = v_j$) it follows that

$$\sum_{i_1=0}^{1}\theta(s_1^{i_1},\dots,s_m^{i_m})w_1^{i_1}(s_1^{v_1}) = \theta(s_1^{v_1},s_2^{i_2},\dots,s_m^{i_m}).$$

Using the same reasoning we obtain the following rule for each sum in the left hand side of equation (A1),

$$\sum_{i_k=0}^{1}\theta(s_1^{v_1},\dots,s_{k-1}^{v_{k-1}},s_k^{i_k},\dots,s_m^{i_m})w_k^{i_k}(s_k^{v_k}) = \theta(s_1^{v_1},\dots,s_k^{v_k},s_{k+1}^{i_{k+1}},\dots,s_m^{i_m}), \quad \forall k = 2,\dots,m, \quad \text{(A4)}$$

noting that, when $k = m$, $\{s_m^{i_m}, \dots, s_m^{i_m}\}$ degenerates in $s_m^{i_m}$ and $\{s_{m+1}^{i_{m+1}}, \dots, s_m^{i_m}\}$ is an empty set of parameters.

Hence using (A3) and applying the results in (A4) to the sequence of sums in (A2) we have $LB_{\theta(s_1^{v_1},\dots,s_m^{v_m})} = \theta(s_1^{v_1}, \dots, s_m^{v_m})$, getting the desired result.

ii) We start by computing an error bound between the function $\theta(s)$ and the left hand side of equation (A1).

$$\left|\theta(s_1,\dots,s_m) - \sum_{i_m=0}^{1}\left(\dots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\dots,s_m^{i_m})w_1^{i_1}(s_1)\right)\dots\right)w_m^{i_m}(s_m)\right| =$$

$$\begin{aligned}&\Bigg|\theta(s_1,\dots,s_m) - \sum_{i_1=0}^{1}\theta(s_1^{i_1},s_2,\dots,s_m)w_1^{i_1}(s_1) + \\ &\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},s_2,\dots,s_m)w_1^{i_1}(s_1) - \sum_{i_2=0}^{1}\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},s_2^{i_2},s_3,\dots,s_m)w_1^{i_1}(s_1)\right)w_2^{i_2}(s_2)\right) + \\ &\left(\sum_{i_2=0}^{1}\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},s_2^{i_2},s_3,\dots,s_m)w_1^{i_1}(s_1)\right)w_2^{i_2}(s_2) - \dots\right) + \dots + \\ &\left(\sum_{i_{m-1}=0}^{1}\left(\dots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\dots,s_{m-1}^{i_{m-1}},s_m)w_1^{i_1}(s_1)\right)\dots\right)w_m^{i_{m-1}}(s_{m-1}) - \right.\\ &\left.\sum_{i_m=0}^{1}\left(\dots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\dots,s_m^{i_m})w_1^{i_1}(s_1)\right)\dots\right)w_m^{i_m}(s_m)\right)\Bigg| \le\end{aligned}$$

$$\begin{aligned}&\left|\theta(s_1,\dots,s_m) - \sum_{i_1=0}^{1}\theta(s_1^{i_1},s_2,\dots,s_m)w_1^{i_1}(s_1)\right| + \\ &\left|\sum_{i_1=0}^{1}\theta(s_1^{i_1},s_2,\dots,s_m)w_1^{i_1}(s_1) - \sum_{i_2=0}^{1}\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},s_2^{i_2},s_3,\dots,s_m)w_1^{i_1}(s_1)\right)w_2^{i_2}(s_2)\right| + \\ &\left|\sum_{i_2=0}^{1}\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},s_2^{i_2},s_3,\dots,s_m)w_1^{i_1}(s_1)\right)w_2^{i_2}(s_2) - \dots\right| + \dots + \\ &\left|\sum_{i_{m-1}=0}^{1}\left(\dots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\dots,s_{m-1}^{i_{m-1}},s_m)w_1^{i_1}(s_1)\right)\dots\right)w_{m-1}^{i_{m-1}}(s_{m-1}) - \right.\\ &\left.\sum_{i_m=0}^{1}\left(\dots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\dots,s_m^{i_m})w_1^{i_1}(s_1)\right)\dots\right)w_m^{i_m}(s_m)\right|\end{aligned}$$

$$\leq \frac{1}{2}K_1(s_1-s_1^0)(s_1^1-s_1)+\frac{1}{2}K_2(s_2-s_2^0)(s_2^1-s_2)+\cdots+\frac{1}{2}K_m(s_m-s_m^0)(s_m^1-s_m)$$
$$\leq \frac{1}{2}K\sum_{i=1}^{m}(s_i-s_i^0)(s_i^1-s_i),$$

where $K_i \geq |\theta''_{s_is_i}(s)|$, $s \in \mathcal{S}$, $i = 1, \ldots, m$ and $K \geq \max_{i=1,\ldots,m} K_i$.

The second to last inequality is obtained by a similar reasoning as in [16, Theorem 2] proof. We provide a sketch of the proof for the first term in the sum. The remaining terms are proved in a similar way and therefore we omit the proof. Define the function $\phi_1(s_1)$ in the following way

$$\phi_1(s_1) = \theta(s_1, s_2, \ldots, s_m) - \sum_{i_1=0}^{1}\theta(s_1^{i_1}, s_2, \ldots, s_m)w_1^{i_1}(s_1) + \frac{1}{2}K_1(s_1-s_1^0)(s_1^1-s_1),$$

for a given constant K_1. Noting that $\phi_1(s_1)$ is a concave one dimensional function and by using a similar argument as in [16, Theorem 2] we get that $\phi_1(s_1) \geq 0$ for $s_1 \in [s_1^0, s_1^1]$, where $K_1 \geq |\theta''_{s_1s_1}(s)|$, $s \in \mathcal{S}$.

Using the provided error bound and the $LB_{\theta(s)}$ definition allows to conclude that $LB_{\theta(s)} \leq \theta(s)$, $\forall s \in \mathcal{S}$, where $K \geq \max_{i=1,\ldots,m} |\theta''_{s_is_i}(s)|$, $s \in \mathcal{S}$.

□

As an immediate consequence of Theorem A.1 we have the following result.

Corollary A.2:

$$\left|\theta(s_1,\ldots,s_m) - \sum_{i_m=0}^{1}\left(\cdots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\ldots,s_m^{i_m})w_1^{i_1}(s_1)\right)\cdots\right)w_m^{i_m}(s_m)\right|$$
$$\leq \frac{1}{2}K\left(h_1^2+\cdots+h_m^2\right)$$

where $h_i = s_i^1 - s_i^0$, $i = 1, \ldots, m$.

Proof: This results can be simple checked by noting that,

$$\max_{s\in\mathcal{S}}(s_i-s_i^0)(s_i^1-s_i) = \frac{1}{4}(s_i^1-s_i^0)^2 \leq h_i^2, \quad i = 1, \ldots, m.$$

□

Theorem A.3: *(Theorem 2 in [16]) The function $LB_{\theta(s)}$ is convex on $\mathcal{S}$ if*

$$K \geq \max_{s\in\mathcal{S}}\max_{i=1,\ldots,m}\sum_{j=1,j\neq i}^{m}|\theta''_{s_is_j}(s)|.$$

Proof: Let

$$L\theta(s_1,\ldots,s_m) = \sum_{i_m=0}^{1}\left(\cdots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\ldots,s_m^{i_m})w_1^{i_1}(s_1)\right)\cdots\right)w_m^{i_m}(s_m).$$

We then express $LB_{\theta(s)}$ in the form

$$LB_{\theta(s)} = L\theta(s_1, \dots, s_m) - \frac{1}{2}K\left(\sum_{i=1}^{m}(s_i - s_i^0)(s_i^1 - s_i)\right)$$

$$= L\theta(s_1, \dots, s_m) + \frac{1}{2}K\sum_{i=1}^{m} s_i^2 - \frac{1}{2}K\sum_{i=1}^{m}\left((s_i^1 + s_i^0)s_i - s_i^0 s_i^1\right).$$

Since the part

$$\frac{1}{2}K\sum_{i=1}^{n}((s_i^1 + s_i^0)s_i - s_i^0 s_i^1)$$

is linear, it suffices to prove that the function $\Phi(s)$, defined by

$$\Phi(s_1, \dots, s_m) = L\theta(s_1, \dots, s_m) + \frac{1}{2}K\sum_{i=1}^{m} s_i^2$$

is convex. This amounts to show that the Hessian matrix of $\Phi(s)$, denoted by $H_{\Phi(s)}$, is semi-definite positive.

From the definition of w_i^j, $j \in \{0, 1\}$, it is easy to see that all elements $(L\theta(s))''_{s_i s_i}$ are zero, for $i = 1, \dots, m$. Hence, $H_{\Phi(s)}$ takes the form

$$H_{\Phi(s)} = \begin{pmatrix} K & L_{12} & L_{13} & \dots & L_{1n} \\ L_{21} & K & L_{23} & \dots & L_{2n} \\ \vdots & & & & \vdots \\ L_{n1} & L_{n2} & L_{n3} & \dots & K \end{pmatrix},$$

where $L_{ij} = (L\theta(s_1, \dots, s_n))''_{s_i s_j}$ is the second mixed derivative of $L\theta(s)$ with respect to the variables s_i and s_j.

We will prove that $H_{\Phi(s)}$ is semi-positive definite by using the sufficient condition that $H_{\Phi(s)}$ is diagonally dominant.

For a matter of simplicity of exposition we just provide the analytic formula for L_{12}, since the remaining values of L_{ij} can be obtained in a similar way. We start by providing the fist derivative of $L\theta(s)$ with respect to s_1.

$$(L\theta(s_1, \dots, s_m))'_{s_1} = \left(\sum_{i_m=0}^{1}\left(\dots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1}, \dots, s_m^{i_m})w_1^{i_1}(s_1)\right)\dots\right)w_m^{i_m}(s_m)\right)'_{s_1}$$

$$= \sum_{i_m}^{1}\left(\dots\left(\sum_{i_2=0}^{1}\left(\theta(s_1^0, s_2^{i_2}, \dots, s_m^{i_m})(w_1^0(s_1))'_{s_1} + \right.\right.\right.$$

$$\left.\left.\left.\theta(s_1^1, s_2^{i_2}, \dots, s_m^{i_m})(w_1^1(s_1))'_{s_1}\right)w_2^{i_2}(s_2)\right)\dots\right)w_m^{i_m}(s_m)$$

$$= \sum_{i_m}^{1} \left(\dots \left(\sum_{i_2=0}^{1} \left(\theta(s_1^0, s_2^{i_2}, \dots, s_n^{i_n}) \frac{-1}{s_1^1 - s_1^0} + \theta(s_1^1, s_2^{i_2}, \dots, s_n^{i_n}) \frac{1}{s_1^1 - s_1^0} \right) w_2^{i_2}(s_2) \right) \dots \right) w_m^{i_m}(s_m)$$

$$= \sum_{i_m}^{1} \left(\dots \left(\sum_{i_2=0}^{1} \left(\frac{\theta(s_1^1, s_2^{i_2}, \dots, s_n^{i_n}) - \theta(s_1^0, s_2^{i_2}, \dots, s_n^{i_n})}{s_1^1 - s_1^0} \right) w_2^{i_2}(s_2) \right) \dots \right) w_m^{i_m}(s_m)$$

$$= \sum_{i_m}^{1} \left(\dots \left(\sum_{i_2=0}^{1} \left(\theta'_{s_1}(\xi, s_2^{i_2}, \dots, s_n^{i_n}) \right) w_2^{i_2}(s_2) \right) \dots \right) w_m^{i_m}(s_m),$$

with $s_1^0 < \xi < s_1^1$.

The second derivative, with respect to s_2, can be computed in the following way.

$$(L\theta(s_1, \dots, s_m))''_{s_1 s_2} = \left(\sum_{i_m=0}^{1} \left(\dots \left(\sum_{i_2=0}^{1} \left(\theta'_{s_1}(\xi, s_2^{i_2}, \dots, s_n^{i_n}) \right) w_2^{i_2}(s_2) \right) \dots \right) w_m^{i_m}(s_m) \right)'_{s_2}$$

$$= \sum_{i_m=0}^{1} \left(\dots \left(\sum_{i_3=0}^{1} \left(\theta'_{s_1}(\xi, s_2^0, s_3^{i_3}, \dots, s_m^{i_m}) (w_2^0(s_2))'_{s_2} + \theta'_{s_1}(\xi, s_2^1, s_3^{i_3}, \dots, s_m^{i_m}) (w_2^1(s_2))'_{s_2} \right) w_3^{i_3}(s_3) \right) \dots \right) w_m^{i_m}(s_m)$$

$$= \sum_{i_m=0}^{1} \left(\dots \left(\sum_{i_3=0}^{1} \left(\theta'_{s_1}(\xi, s_2^0, s_3^{i_3} \dots, s_m^{i_m}) \frac{-1}{s_2^1 - s_2^0} + \theta'_{s_1}(\xi, s_2^1, s_3^{i_3}, \dots, s_m^{i_m}) \frac{1}{s_2^1 - s_2^0} \right) w_3^{i_3}(s_3) \right) \dots \right) w_m^{i_m}(s_m)$$

$$= \sum_{i_m=0}^{1} \left(\dots \left(\sum_{i_3=0}^{1} \left(\frac{\theta'_{s_1}(\xi, s_2^1, s_3^{i_3}, \dots, s_m^{i_m}) - \theta'_{s_1}(\xi, s_2^0, s_3^{i_3}, \dots, s_m^{i_m})}{s_2^1 - s_2^0} \right) w_3^{i_3}(s_3) \right) \dots \right) w_m^{i_m}(s_m)$$

$$= \sum_{i_m=0}^{1} \left(\dots \left(\sum_{i_3=0}^{1} \left(\theta''_{s_1 s_2}(\xi, \eta, s_3^{i_3}, \dots, s_m^{i_m}) \right) w_3^{i_3}(s_3) \right) \dots \right) w_m^{i_m}(s_m)$$

with $s_1^0 < \xi < s_1^1$ and $s_2^0 < \eta < s_2^1$.

Since $w_i^0(s_i) + w_i^1(s_i) = 1$ for all $s_i \in [s_i^0, s_i^1]$, $i = 1, \dots, m$, we have

$$\sum_{i_3=0}^{1} \theta''_{s_1 s_2}(\xi, \eta, s_3^{i_3}, \dots, s_m^{i_m}) w_3^{i_3}(s_3) \leq \max_{s_3 \in [s_3^0, s_3^1]} \theta''_{s_1 s_2}(\xi, \eta, s_3, s_4^{i_4}, \dots, s_m^{i_m}),$$

and therefore

$$\sum_{i_m=0}^{1} \left(\cdots \left(\sum_{i_3=0}^{1} \left(\theta''_{s_1 s_2}(\xi, \eta, s_3^{i_3}, \dots, s_m^{i_m}) \right) w_3^{i_3}(s_3) \right) \cdots \right) w_m^{i_m}(s_m)$$
$$\leq \max_{(s_1, \dots, s_m) \in \mathcal{S}} \theta''_{s_1 s_2}(s_1, \dots, s_m).$$

Since $L\theta''_{s_1 s_2}(s)$ is a linear interpolator for each s_i, $i = 1, \dots, m$, we conclude that

$$\left| L\theta''_{s_1 s_2}(s_1, \dots, s_m) \right| \leq \max_{(s_1, \dots, s_m) \in \mathcal{S}} \left| \theta''_{s_1 s_2}(s_1, \dots, s_m) \right|.$$

Likewise, for each pair $i \neq j$, $i, j = 1, \dots, m$, we have

$$|L\theta''_{s_i s_j}(s_1, \dots, s_m)| \leq \max_{(s_1, \dots, s_m) \in \mathcal{S}} |\theta''_{s_i s_j}(s_1, \dots, s_m)|.$$

Therefore, provided that

$$K \geq \max_{s \in \mathcal{S}} \max_{i=1,\dots,m} \sum_{j=1, j \neq i}^{m} \left| \theta''_{s_i s_j}(s) \right|,$$

we have

$$K \geq \max_{s \in \mathcal{S}} \max_{i=1,\dots,m} \sum_{j=1, j \neq i}^{m} \left| L\theta''_{s_i s_j}(s) \right|,$$

making $H_{\Phi(s)}$ to be diagonally dominant and consequently semi-definite positive.
$\square$

Remark *A1***:** The inequality

$$K \geq \max_{s \in \mathcal{S}} \max_{i=1,\dots,m} \sum_{j=1}^{m} \left| (\theta)''_{s_i s_j}(s) \right|,$$

i.e. $K \geq \|H_{\theta(s)}\|_\infty$, implies that

$$K \geq \max_{s \in \mathcal{S}} \sum_{i=1}^{m} \left| (\theta_x)''_{s_i s_i}(s) \right|,$$

a sufficient condition for $LB_{\theta(s)}$ to be a underestimator of $\theta(s)$ (see Theorem A.1) and

$$K \geq \max_{s\in\mathcal{S}} \max_{i=1,\dots,m} \sum_{j=1,i\neq j}^{m} \left|(\theta)''_{s_i s_j}(s)\right|,$$

a sufficient condition for $LB_{\theta(s)}$ to be convex on $\mathcal{S}$.

A.2. *Convergence of the B&B Algorithm* 2.1

Theorem A.4: *(Theorem 4 in [16]) Either Algorithm* **2.1** *is finite or it generates a bounded sequence* $\{s^l\}$*, where every accumulation point is a global optimal solution of* (2)*, and*

$$LB^l \nearrow \gamma_x, \qquad UB^l \searrow \gamma_x.$$

Proof: For each iteration l let $h_i^l = s_i^{1,l} - s_i^{0,l}$, for $i = 1,\dots,m$, and $\mathcal{S}_l = \Pi_{i=1}^m \left[s_i^{0,l}, s_i^{1,l}\right]$. We have

$$\min_{i_k\in\{0,1\},k=1,\dots,m} \theta(s_1^{i_1},\dots,s_m^{i_m}) - \frac{1}{2}K\left(\left(h_1^l\right)^2 + \cdots + \left(h_m^l\right)^2\right)$$

$$\leq \min_{s\in\mathcal{S}_l}\left(\sum_{i_m=0}^{1}\left(\cdots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\dots,s_m^{i_m})w_1^{i_1}(s_1)\right)\cdots\right)w_m^{i_m}(s_m)\right.$$
$$\left. -\frac{1}{2}K\left(\sum_{i=1}^{m}(s_i - s_i^{0,l})(s_i^{1,l} - s_i)\right)\right)$$

$$\leq \min_{s\in\mathcal{S}_l} \theta(s_1,\dots,s_m).$$

Additionally let

$$LB_1^l = \min_{i_k\in\{0,1\},k=1,\dots,m} \theta(s_1^{i_1},\dots,s_m^{i_m}) - \frac{1}{2}K\left(\left(h_1^l\right)^2 + \cdots + \left(h_m^l\right)^2\right),$$

$$LB^l = \min_{s\in\mathcal{S}}\left(\sum_{i_m=0}^{1}\left(\cdots\left(\sum_{i_1=0}^{1}\theta(s_1^{i_1},\dots,s_m^{i_m})w_1^{i_1}(s_1)\right)\cdots\right)w_m^{i_m}(s_m)\right.$$
$$\left. -\frac{1}{2}K\left(\sum_{i=1}^{m}(s_i - s_i^{0})(s_i^{1} - s_i)\right)\right),$$

and

$$UB^l = \min_{i_k\in\{0,1\},k=1,\dots,m} \theta(s_1^{i_1},\dots,s_m^{i_m}).$$

Then we have

$$0 \leq \lim_{l\to\infty} \left(UB^l - LB^l\right) \leq \lim_{l\to\infty} \left(UB^l - LB_1^l\right)$$
$$\leq \lim_{l\to\infty} K\left(\left(h_1^l\right)^2 + \cdots + \left(h_m^l\right)^2\right) = 0,$$

which implies that $\lim_{l\to\infty} UB^l - LB^l = 0$, since we have $h_i^l \to 0$, $i = 1, \ldots, m$, as a consequence of the branching procedure.

Moreover, since $s^l \in \mathcal{S}$ and $UB^l = \theta(s^l)$, any cluster point of the sequence $\{s^l\}$ belongs to $\mathcal{S}$ and has the function value γ_x, i.e., it solves problem (2). $\square$