RC24810 (W0906-072) June 17, 2009
Mathematics

IBM Research Report

An Algorithmic Framework for MINLP with Separable Non-convexity

Claudia D'Ambrosio
Department of ECSS
University of Bologna
Italy

Jon Lee, Andreas Wächter
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
USA

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

AN ALGORITHMIC FRAMEWORK FOR MINLP WITH SEPARABLE NON-CONVEXITY

CLAUDIA D'AMBROSIO*, JON LEE†, AND ANDREAS WÄCHTER‡

Abstract. Global optimization algorithms, e.g., spatial branch-and-bound approaches like those implemented in codes such as BARON and COUENNE, have had substantial success in tackling complicated, but generally small scale, non-convex MINLPs (i.e., mixed-integer nonlinear programs having non-convex continuous relaxations). Because they are aimed at a rather general class of problems, the possibility remains that larger instances from a simpler class may be amenable to a simpler approach.

We focus on MINLPs for which the non-convexity in the objective and constraint functions is manifested as the sum of non-convex univariate functions. There are many problems that are already in such a form, or can be brought into such a form via some simple substitutions. In fact, the first step in spatial branch-and-bound is to bring problems into nearly such a form. For our purposes, we shift that burden back to the modeler. We have developed a simple algorithm, implemented at the level of a modeling language (in our case AMPL), to attack such separable problems. First, we identify subintervals of convexity and concavity for the univariate functions using external calls to MATLAB. With such an identification at hand, we develop a convex MINLP relaxation of the problem (i.e., as a mixed-integer nonlinear program having a convex continuous relaxation). Our convex MINLP relaxation differs from those typically employed in spatial branch-and-bound; rather than relaxing the graph of a univariate function on an interval to an enclosing polygon, we work on each subinterval of convexity and concavity separately, using linear relaxation on only the "concave side" of each function on the subintervals. The subintervals are glued together using binary variables. Next, we employ ideas of spatial branch-and-bound, but rather than branching, we repeatedly refine our convex MINLP relaxation by modifying it at the modeling level. We attack our convex MINLP relaxation, to get lower bounds on the global minimum, using the code BONMIN as a black-box convex MINLP solver. Next, by fixing the integer variables in the original non-convex MINLP, and then locally solving the associated non-convex NLP restriction, we get an upper bound on the global minimum, using the code IPOPT. We use the solutions found by BONMIN and IPOPT to guide our choice of further refinements in a way that overall guarantees convergence. Note that our proposed procedure is an exact algorithm, and not just a heuristic.

We have had substantial success in our preliminary computational experiments. In particular, we see very few major iterations occurring, so most of the time is spent in the solution of a small number of convex MINLPs. An advantage of our approach is that it can be implemented easily using existing software components, and that further advances in technology for convex MINLP will immediately give our approach a benefit.

Key words. mixed-integer nonlinear programming, global optimization, spatial branch-and-bound, separable, non-convex

AMS(MOS) subject classifications. 65K05, 90C11, 90C26, 90C30

1. Introduction.

The global solution of practical instances of Mixed-Integer NonLinear Programming (MINLP) problems has been considered for some decades. Over a considerable period of time, technology for the global optimization of convex MINLP (i.e., the continuous relaxation of the problem is a convex program) had matured (see, for example, [8, 17, 9, 3]), and rather recently there has been considerable success in the realm of global optimization of non-convex MINLP (see, for example, [18, 16, 13, 2]).

Global optimization algorithms, e.g., spatial branch-and-bound approaches like those implemented in codes like BARON [18] and COUENNE [2], have had substantial success in tackling complicated, but generally small scale, non-convex MINLPs (i.e., mixed-integer nonlinear programs having non-convex continuous relaxations). Because they are aimed at a rather general class of problems, the possibility remains that larger instances from a simpler class may be amenable to a simpler approach.

We focus on MINLPs for which the non-convexity in the objective and constraint functions is manifested as the sum of non-convex univariate functions. There are many problems that are already in such a form, or can be brought into such a form via some simple substitutions. In fact, the first step in spatial branch-and-bound is to bring problems into nearly such a form. For our purposes, we shift that burden back to the modeler. We have developed a simple algorithm, implemented at the level of a modeling language (in our case AMPL, see [10]), to attack such

*Dept. of ECSS, University of Bologna, Italy. Email: c.dambrosio@unibo.it
†IBM T.J. Watson Research Center, NY, U.S.A. Email: jonlee@us.ibm.com
‡IBM T.J. Watson Research Center, NY, U.S.A. Email: andreasw@us.ibm.com

separable problems. First, we identify subintervals of convexity and concavity for the univariate functions using external calls to `MATLAB` [14]. With such an identification at hand, we develop a convex MINLP relaxation of the problem (i.e., as a mixed-integer nonlinear programs having a convex continuous relaxations). Our convex MINLP relaxation differs from those typically employed in spatial branch-and-bound; rather than relaxing the graph of a univariate function on an interval to an enclosing polygon, we work on each subinterval of convexity and concavity separately, using linear relaxation on only the "concave side" of each function on the subintervals. The subintervals are glued together using binary variables. Next, we employ ideas of spatial branch-and-bound, but rather than branching, we repeatedly refine our convex MINLP relaxation by modifying it at the modeling level. We attack our convex MINLP relaxation, to get lower bounds on the global minimum, using the code `BONMIN` [3, 4] as a black-box convex MINLP solver. Next, by fixing the integer variables in the original non-convex MINLP, and then locally solving the associated non-convex NLP restriction, we get an upper bound on the global minimum, using the code `IPOPT` [19]. We use the solutions found by `BONMIN` and `IPOPT` to guide our choice of further refinements.

We implemented our framework using the modeling language `AMPL`. In order to obtain all of the information necessary for the execution of the algorithm, external software, specifically the tool for high-level computational analysis `MATLAB`, the convex MINLP solver `BONMIN`, and the NLP solver `IPOPT`, are called directly from the `AMPL` environment. A detailed description of the entire algorithmic framework, together with a proof of its convergence, is provided in §2.

We present computational results in §3. Some of the instances arise from specific applications; in particular, Uncapacitated Facility Location problems, Hydro Unit Commitment and Scheduling problems, and Nonlinear Continuous Knapsack problems. We also present computational results on selected instances of GLOBALLib and MINLPLib. We have had significant success in our preliminary computational experiments. In particular, we see very few major iterations occurring, with most of the time being spent in the solution of a small number of convex MINLPs. As we had hoped, our method does particularly well on problems for which the non-convexity is naturally separable. An advantage of our approach is that it can be implemented easily using existing software components and that further advances in technology for convex MINLP will immediately give us a proportional benefit.

Finally, we note that a preliminary shorter version of the present paper appeared as [7].

2. Our algorithmic framework. We focus now on MINLPs, where the non-convexity in the objective and constraint functions is manifested as the sum of non-convex univariate functions. Without loss of generality, we take them to be of the form

$$\begin{array}{l} \min \ \sum_{j\in N} C_j x_j \\ \quad \text{subject to} \\ f(x) \le 0 \ ; \\ r_i(x) + \sum_{k\in H(i)} g_{ik}(x_k) \le 0 \ , \ \forall i \in M \ ; \\ L_j \le x_j \le U_j \ , \ \forall j \in N \ ; \\ x_j \text{ integer}, \forall j \in I \ , \end{array} \qquad (\mathcal{P})$$

where $N := \{1, 2, \ldots, n\}$, $f : \mathbb{R}^n \to \mathbb{R}^p$ and $r_i : \mathbb{R}^n \to \mathbb{R}$ $\forall i \in M$, are convex functions, $H(i) \subseteq N$ $\forall i \in M$, the $g_{ik} : \mathbb{R} \to \mathbb{R}$ are non-convex univariate function $\forall i \in M$, and $I \subseteq N$. Letting $H := \cup_{i\in M} H(i)$, we can take each L_j and U_j to be finite or infinite for $j \in N \setminus H$, but for $j \in H$ we assume that these are finite bounds.

We assume that the problem functions are sufficiently smooth (e.g., twice continuously differentiable) with the exception that we allow the univariate g_{ik} to be continuous functions defined piecewise by sufficiently smooth functions over a finite set of subintervals of $[L_k, U_k]$. Without loss of generality, we have taken the objective function as linear and all of the constraints to be inequalities, and further of the less-then-or-equal variety. Linear equality constraints could be included directly in this formulation, while we assume that nonlinear equalities have been split into two inequality constraints.

Our approach is an iterative technique based on three fundamental ingredients:

- A reformulation method with which we obtain a convex MINLP relaxation $\mathcal{Q}$ of the original problem $\mathcal{P}$. Solving the convex MINLP relaxation $\mathcal{Q}$, we obtain a lower bound of our original problem $\mathcal{P}$;
- A non-convex NLP restriction $\mathcal{R}$ of the original MINLP problem $\mathcal{P}$ obtained by fixing the variables within the set $\{x_j \ : \ j \in I\}$. Locally solving the non-convex NLP restriction $\mathcal{R}$, we obtain an upper bound of our original problem $\mathcal{P}$;
- A refinement technique aimed at improving, at each iteration, the quality of the lower bound obtained by solving the convex MINLP relaxation $\mathcal{Q}$.

The main idea of our algorithmic framework is to iteratively solve a lower-bounding relaxation $\mathcal{Q}$ and an upper-bounding restriction $\mathcal{R}$ so that, in case the value of the upper and the lower bound are the same, the global optimality of the solution found is proven; otherwise we make a refinement to the lower-bounding relaxation $\mathcal{Q}$. At each iteration, we seek to decrease the gap between the lower and the upper bound, and hopefully, before too long, the gap will be within a tolerance value, or the lower bounding solution is deemed to be sufficiently feasible for the original problem. In this case, or in the case a time/iteration limit is reached, the algorithm stops. If the gap is closed, we have found a global optimum, otherwise we have a heuristic solution (provided that the upper bound is not $+\infty$). The lower-bounding relaxation $\mathcal{Q}$ is a convex relaxation of the original non-convex MINLP problem, obtained by approximating the concave part of the non-convex univariate functions using piecewise linear approximation. The novelty in this part of the algorithmic framework is the new formulation of the convex relaxation: The function is approximated only where it is concave, while the convex parts of the functions are not approximated, but taken as they are. The convex relaxation proposed is described in details in §2.1. The upper-bounding restriction $\mathcal{R}$, described in §2.2, is obtained simply by fixing the variables with integrality constraints. The refinement technique consists of adding one or more breakpoints where needed, i.e., where the approximation of the non-convex function is bad and the solution of the lower-bounding problem lies. Refinement strategies are described in §2.3, and once the ingredients of the algorithmic framework are described in detail, we give a pseudo-code description of our algorithmic framework (see §2.4). Here, we also discuss some considerations about the general framework and the similarities and differences with popular global optimization methods. Theoretical convergence guarantees are discussed in §2.5. In §3, computational experiments are presented, detailing the performance of the algorithm and comparing the approach to other methods.

2.1. The lower-bounding convex MINLP relaxation $\mathcal{Q}$. To obtain our convex MINLP relaxation $\mathcal{Q}$ of the MINLP problem $\mathcal{P}$, we need to locate the subintervals of the domain of each univariate function g_i for which the function is uniformly convex or concave. For simplicity of notation, rather than refer to the constraint $r_i(x) + \sum_{k \in H(i)} g_{ik}(x_k) \leq 0$, we consider a term of the form $g(x_k) := g_{ik}(x_k)$, where $g : \mathbb{R} \to \mathbb{R}$ is a univariate non-convex function of x_k , for some k $(1 \leq k \leq n)$.

We want to explicitly view each such g as a piecewise-defined function, where on each piece the function is either convex or concave. This feature also allows us to handle functions that are already piecewise defined by the modeler. In practice, for each non-convex function g , we compute the points at which the convexity/concavity may change, i.e., the zeros of the second derivative of g , using `MATLAB`. In case a function g is naturally piecewise defined, we are essentially refining the piecewise definition of it in such a way that the convexity/concavity is uniform on each piece.

EXAMPLE 1. *Consider the piecewise-defined univariate function*

$$g(x_k) := \begin{cases} 1 + (x_k - 1)^3 \ , & \text{for } 0 \leq x_k \leq 2 \ ; \\ 1 + (x_k - 3)^2 \ , & \text{for } 2 \leq x_k \leq 4 \ , \end{cases}$$

depicted in FIG. 1. *In addition to the breakpoints $x_k = 0, 2, 4$ of the definition of g, the convexity/concavity changes at $x_k = 1$, so by utilizing an additional breakpoint at $x_k = 1$ the convexity/concavity is now uniform on each piece.*

Now, on each concave piece we can use a secant approximation to give a piecewise-convex lower approximation of g .

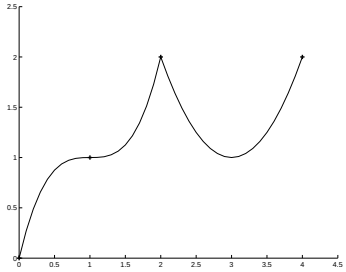


FIG. 1. *A piecewise-defined univariate function*

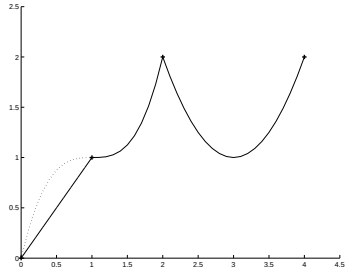


FIG. 2. *A piecewise-convex lower approximation*

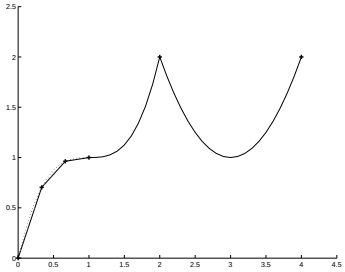


FIG. 3. *Improved piecewise-convex lower approximation*

EXAMPLE 1, continued. *Relative to $g(x_k)$ of* EXAMPLE 1*, we have the piecewise-convex lower approximation*

$$\underline{g}(x_k) := \begin{cases} x_k \ , & \text{for } 0 \le x_k \le 1 \ ; \\ 1 + (x_k - 1)^3 \ , & \text{for } 1 \le x_k \le 2 \ ; \\ 1 + (x_k - 3)^2 \ , & \text{for } 2 \le x_k \le 4 \ , \end{cases}$$

depicted in FIG. 2.

We can obtain a better lower bound by refining the piecewise-linear lower approximation on the concave pieces. We let

$$L_k =: P_0 < P_1 < \cdots < P_{\overline{p}} := U_k$$

be the ordered breakpoints at which the convexity/concavity of g changes, including, in the case of piecewise definition of g, the points at which the definition g changes. We define:

$[P_{p-1}, P_p]$:= the p-th subinterval of the domain of g ($p \in \{1 \dots \overline{p}\}$);
$\breve{H}$:= the set of indices of subintervals on which g is convex;
$\hat{H}$:= the set of indices of subintervals on which g is concave.

On the concave intervals, we will allow further breakpoints. We let B_p be the ordered set of breakpoints for the concave interval indexed by $p \in \hat{H}$. We denote these breakpoints as

$$P_{p-1} =: X_{p,1} < X_{p,2} < \cdots < X_{p,|B_p|} := P_p \ ,$$

and in our relaxation we will view g as lower bounded by the piecewise-linear function that has value $g(X_{p,j})$ at the breakpoints $X_{p,j}$, and is otherwise linear between these breakpoints.

EXAMPLE 1, continued again. *Utilizing further breakpoints, for example at $x_k = 1/3$ and $x_k = 2/3$, we can improve the piecewise-convex lower approximation to instead*

$$\underline{g}(x_k) := \begin{cases} \frac{19}{9} x_k \ , & \text{for } 0 \le x_k \le \frac{1}{3} \ ; \\ \frac{19}{27} + \frac{7}{9} \left(x_k - \frac{1}{3}\right) \ , & \text{for } \frac{1}{3} \le x_k \le \frac{2}{3} \ ; \\ \frac{26}{27} + \frac{1}{9} \left(x_k - \frac{2}{3}\right) \ , & \text{for } \frac{2}{3} \le x_k \le 1 \ ; \\ 1 + (x_k - 1)^3 \ , & \text{for } 1 \le x_k \le 2 \ ; \\ 1 + (x_k - 3)^2 \ , & \text{for } 2 \le x_k \le 4 \ , \end{cases}$$

depicted in FIG. 3.

Next, we define further variables to manage our convexification of g on its domain:

z_p := a binary variable indicating if $x_k \ge P_p$ ($p = 1, \dots, \overline{p} - 1$);
δ_p := a continuous variable assuming a positive value iff $x_k \ge P_{p-1}$ ($p = 1, \dots, \overline{p}$);

$\alpha_{p,b}$:= weight of breakpoint b in the piecewise-linear approximation of the interval indexed by p ($p \in \hat{H}$, $b \in B_p$).

In the convex relaxation of the original MINLP $\mathcal{P}$, we substitute each univariate non-convex term $g(x_k)$ with

$$\sum_{p\in\check{H}} g(P_{p-1}+\delta_p) + \sum_{p\in\hat{H}} \sum_{b\in B_p} g(X_{p,b})\ \alpha_{p,b} - \sum_{p=1}^{\overline{p}-1} g(P_p)\ , \tag{2.1}$$

and we include the following set of new constraints:

$$P_0 + \sum_{p=1}^{\overline{p}} \delta_p - x_k = 0\ ; \tag{2.2}$$

$$\delta_p - (P_p - P_{p-1})z_p \geq 0\ ,\ \forall p \in \check{H}\cup\hat{H}\ ; \tag{2.3}$$

$$\delta_p - (P_p - P_{p-1})z_{p-1} \leq 0\ ,\ \forall p \in \check{H}\cup\hat{H}\ ; \tag{2.4}$$

$$P_{p-1} + \delta_p - \sum_{b\in B_p} X_{p,b}\ \alpha_{p,b} = 0\ ,\ \forall p \in \hat{H}\ ; \tag{2.5}$$

$$\sum_{b\in B_p} \alpha_{p,b} = 1\ ,\ \forall p \in \hat{H}\ ; \tag{2.6}$$

$$\{\alpha_{p,b} : b \in B_p\} := \text{SOS2}\ , \qquad \forall p \in \hat{H}\ ; \tag{2.7}$$

with two dummy variables $z_0 := 1$ and $z_{\overline{p}} := 0$.

Constraints (2.2–2.4) together with the integrality of the z variables ensure that, given an x_k value, say $x_k^* \in [P_{p^*-1}, P_{p^*}]$:

$$\delta_p = \begin{cases} P_p - P_{p-1}\ , & \text{if } 1 \leq p \leq p^*-1\ ; \\ x_k^* - P_{p-1}\ , & \text{if } p = p^*\ ; \\ 0\ , & \text{otherwise.} \end{cases}$$

Constraints (2.5–2.7) ensure that, for each concave interval, the convex combination of the breakpoints is correctly computed. Finally, (2.1) approximates the original non-convex univariate function $g(x_k)$. Each single term of the first and the second summations, using the definition of δ_p , reduces, respectively, to

$$g(P_{p-1}+\delta_p) = \begin{cases} g(P_p)\ , & \text{if } p \in \{1,\ldots,p^*-1\}\ ; \\ g(x_k^*)\ , & \text{if } p = p^*\ ; \\ g(P_{p-1})\ , & \text{if } p \in \{p^*+1,\ldots,\overline{p}\}\ , \end{cases}$$

and

$$\sum_{b\in B_p} g(X_{p,b})\ \alpha_{p,b} = \begin{cases} g(P_p)\ , & \text{if } p \in \{1,\ldots,p^*-1\}\ ; \\ \sum_{b\in B_{p^*}} g(X_{p^*,b})\ \alpha_{p^*,b}\ , & \text{if } p = p^*\ ; \\ g(P_{p-1})\ , & \text{if } p \in \{p^*+1,\ldots,\overline{p}\}\ , \end{cases}$$

reducing constraint (2.1) to

$$r(x) + \sum_{p=1}^{p^*-1} g(P_p) + \gamma + \sum_{p=p^*+1}^{\overline{p}} g(P_{p-1}) - \sum_{p=1}^{\overline{p}-1} g(P_p) = r(x) + \gamma \leq 0\ ,$$

with

$$\gamma = \begin{cases} g(P_p - x_k^*)\ , & \text{if } p^* \in \check{H}\ ; \\ \sum_{b\in B_{p^*}} g(X_{p^*,b})\ \alpha_{p^*b}\ , & \text{if } p^* \in \hat{H}\ . \end{cases}$$

Constraints (2.7) define $|\hat{H}|$ Special Ordered Sets of Type 2 (SOS2), i.e., ordered sets of positive variables among which at most 2 can assume a non-zero value, and, in this case, they must be consecutive (Beale and Tomlin [1]). Unfortunately, at the moment, convex MINLP solvers

do not typically handle SOS2 like most MILP solvers do (also defining special-purpose branching strategies). For this reason, we substitute constraints (2.7), $\forall p \in \hat{H}$, with new binary variables $y_{p,b}$, with $b \in \{1, \ldots, |B_p| - 1\}$, and constraints:

$$\alpha_{p,b} \leq y_{p,b-1} + y_{p,b} \; \forall b \in B_p \; ; \tag{2.7.a}$$

$$\textstyle\sum_{b=1}^{|B_p|-1} y_{p,b} = 1 \; , \tag{2.7.b}$$

with dummy values $y_{p,0} = y_{p,|B_p|} = 0$. In the future, when convex MINLP solvers will handle the definition of SOS2, variables y and constraints (2.7.a–b) would be not necessary.

It is important to note that if we utilized a very large number of breakpoints at the start, solving the resulting convex MINLP $\mathcal{Q}$ would mean essentially solving globally the original MINLP $\mathcal{P}$ up to some pre-determined tolerance related to the density of the breakpoints. But of course such a convex MINLP $\mathcal{Q}$ would be too hard to be solved in practice. With our algorithmic framework, we dynamically seek a significantly smaller convex MINLP $\mathcal{Q}$, thus generally more easily solvable, which we can use to guide the non-convex NLP restriction $\mathcal{R}$ to a good local solution, eventually settling on and proving global optimality of such a solution to the original MINLP $\mathcal{P}$.

2.2. The upper-bounding non-convex NLP restriction $\mathcal{R}$. Given a solution $\underline{x}$ of the convex MINLP relaxation $\mathcal{Q}$, the upper-bounding restriction $\mathcal{R}$ is defined as the non-convex NLP:

$$\begin{array}{l} \min \; \sum_{j \in N} C_j x_j \\ \quad \text{subject to} \\ f(x) \leq 0 \; ; \\ r_i(x) + \sum_{k \in H(i)} g_{ik}(x_k) \leq 0 \; , \; \forall i \in M \; ; \\ L_j \leq x_j \leq U_j \; , \; \forall j \in N \; ; \\ x_j = \underline{x}_j, \; \forall j \in I \; . \end{array} \tag{$\mathcal{R}$}$$

A solution of this non-convex NLP $\mathcal{R}$ is a heuristic solution of the non-convex MINLP problem $\mathcal{P}$ for two reasons: (i) the integer variables x_j , $j \in I$, might not be fixed to globally optimal values; (ii) the NLP $\mathcal{R}$ is non-convex, and so even if the integer variables x_j , $j \in I$, are fixed to globally optimal values, the NLP solver may only find a local optimum of the non-convex NLP $\mathcal{R}$ or even fail to find a feasible point. This consideration emphasizes the importance of the lower-bounding relaxation $\mathcal{Q}$ for the guarantee of global optimality. The upper-bounding problem resolution could be seen as a "verification phase" in which a solution of the convex MINLP relaxation $\mathcal{Q}$ is tested to be really feasible for the non-convex MINLP $\mathcal{P}$. To emphasis this, the NLP solver for $\mathcal{R}$ is given the solution of the convex MINLP relaxation as starting point.

2.3. The refinement technique. At the end of each iteration, we have two solutions: $\underline{x}$, the solution of the lower-bounding convex MINLP relaxation $\mathcal{Q}$, and $\overline{x}$, the solution of the upper-bounding non-convex NLP restriction $\mathcal{R}$; in case we cannot find a solution of $\mathcal{R}$, e.g., if $\mathcal{R}$ is infeasible, then no $\overline{x}$ is available. If $\sum_{j \in N} C_j \underline{x}_j = \sum_{j \in N} C_j \overline{x}_j$ within a certain tolerance, or if $\underline{x}$ is sufficiently feasible for the original constraints, we return to the user as solution the point $\overline{x}$ or $\underline{x}$, respectively. Otherwise, in order to continue, we want to refine the approximation of the lower-bounding convex MINLP relaxation $\mathcal{Q}$ by adding further breakpoints. We employed two strategies:

- *Based on the lower-bounding problem solution $\underline{x}$: For each $i \in M$ and $k \in H(i)$, if $\underline{x}_k$ lies in a concave interval of g_{ik} , add $\underline{x}_k$ as a breakpoint for the relaxation of g_{ik} .*
  This procedure drives the convergence of the overall method since it makes sure that the lower bounding problem becomes eventually a sufficiently accurate approximation of the original problem in the neighborhood of the global solution. Since adding a breakpoint increases the size of the convex MINLP relaxation, in practice we do not add such a new breakpoint if it would be within some small tolerance of an existing breakpoint for g_{ik} .
- *Based on the upper-bounding problem solution $\overline{x}$: For each $i \in M$ and $k \in H(i)$, if $\overline{x}_k$ lies in a concave interval of g_{ik} , add $\overline{x}_k$ as a breakpoint for the relaxation of g_{ik} .*

The motivation behind this option is to accelerate the convergence of the method. If the solution found by the upper-bounding problem is indeed the global solution, the relaxation should eventually be exact at this point to prove its optimality. Again, to keep the size of the relaxation MINLP manageable, breakpoints are only added if they are not too close to existing ones.

We found that these strategies work well together. Hence, at each major iteration, we add a breakpoint in each concave interval where $\underline{x}$ lies in order to converge and one where $\overline{x}$ lies to speed up the convergence.

2.4. The algorithmic framework. Algorithm 1 details our `SC-MINLP` (Sequential Convex MINLP) Algorithm, while Figure 4 depicts, at a high level, how we designed our implementation of it.

Algorithm 1 : `SC-MINLP` (Sequential Convex MINLP) Algorithm

Choose tolerances $\varepsilon, \varepsilon_{\text{feas}} > 0$; initialize $LB := -\infty$; $UB := +\infty$;
Find P_p^i , $\hat{H}^i$, $\check{H}^i$, X_{pb}^i $(\forall i \in M, p \in \{1 \dots \overline{p}^i\}, b \in B_p^i)$.
repeat
  Solve the convex MINLP relaxation $\mathcal{Q}$ of the original problem $\mathcal{P}$ to obtain $\underline{x}$;
  if $(\text{val}(\mathcal{Q}) > LB)$ **then**
    $LB := \text{val}(\mathcal{Q})$;
    if ($\underline{x}$ is feasible for the original problem $\mathcal{P}$ (within tolerance $\varepsilon_{\text{feas}}$)) **then**
      return $\underline{x}$
    end if
  end if
  Solve the non-convex NLP restriction $\mathcal{R}$ of the original problem $\mathcal{P}$ to obtain $\overline{x}$;
  if (solution $\overline{x}$ could be computed and $\text{val}(\mathcal{R}) < UB$) **then**
    $UB := \text{val}(\mathcal{R})$; $x_{UB} := \overline{x}$
  end if
  if $(UB - LB > \varepsilon)$ **then**
    Update B_p^i , X_{pb}^i ;
  end if
until $((UB - LB \leq \varepsilon)$ or (time or iteration limited exceeded))
return the current best solution x_{UB}

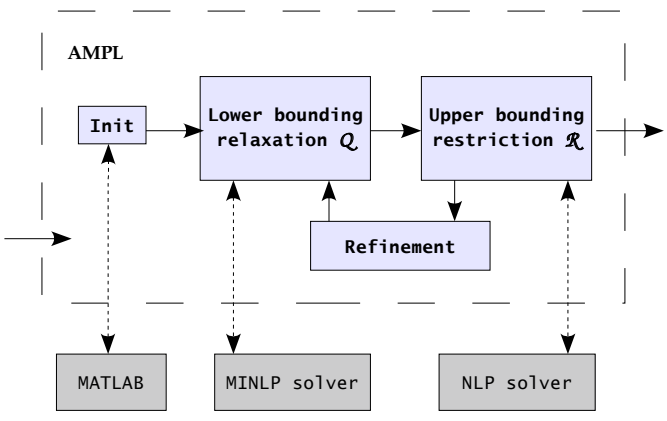


FIG. 4. *The* `SC-MINLP` *(Sequential Convex MINLP) framework*

At each iteration, the lower-bounding MINLP relaxation $\mathcal{Q}$ and the upper-bounding NLP restriction $\mathcal{R}$ are redefined: What changes in $\mathcal{Q}$ are the sets of breakpoints that refine the piecewise-linear approximation of concave parts of the non-convex functions. At each iteration, the number

of breakpoint used increases, and so does the accuracy of the approximation. What may change in $\mathcal{R}$ are the values of the fixed integer variables $\underline{x}_j$, $j \in I$. Moreover, what changes is the starting point given to the NLP solver, derived from an optimal solution of the lower-bounding MINLP relaxation $\mathcal{Q}$.

Our algorithmic framework bears comparison with spatial branch-and-bound, a successful technique in global optimization. In particular:

- during the refining phase, the parts in which the approximation is bad are discovered and the approximation is improved there, but we do it by adding one or more breakpoints instead of branching on a continuous variable as in spatial branch-and-bound;
- like spatial branch-and-bound, our approach is a rigorous global-optimization algorithm rather than a heuristic;
- unlike spatial branch-and-bound, our approach does not utilize an expression tree; it works directly on the broad class of separable non-convex MINLPs of the form $\mathcal{P}$, and of course problems that can be put in such a form;
- unlike standard implementations of spatial branch-and-bound methods, we can directly keep multivariate convex functions in our relaxation instead of using linear approximations;
- unlike spatial branch-and-bound, our method can be effectively implemented at the modeling-language level.

2.5. Convergence Analysis. For the theoretical convergence analysis of Algorithm 1, we make the following assumptions, denoting by l the iteration counter for the **repeat** loop.

A1. The functions $f(x)$ and $r_i(x)$ are continuous, and the univariate functions g_{ik} in $(\mathcal{P})$ are uniformly Lipschitz-continuous with a bounded Lipschitz constant L_g.

A2. The problem $\mathcal{P}$ has a feasible point. Hence, for each l, the relaxation $\mathcal{Q}^l$ is feasible, and we assume its (globally) optimal solution $\underline{x}^l$ is computed.

A3. The refinement technique described in Section 2.3 adds a breakpoint for every lower-bounding problem solution $\underline{x}^l$, even if it is very close to an existing breakpoint.

A4. The feasibility tolerance $\varepsilon_{\text{feas}}$ and the optimality gap tolerance ε are both chosen to be zero, and no iteration limit is set.

THEOREM 2.1. *Under assumptions A1-A4, Algorithm 1 either terminates at a global solution of the original problem $\mathcal{P}$, or each limit point of the sequence $\{\underline{x}^l\}_{l=1}^{\infty}$ is a global solution of $\mathcal{P}$.*

Proof. By construction, $\mathcal{Q}^l$ is always a relaxation of $\mathcal{P}$, and hence val$(\mathcal{Q}^l)$ is always less than or equal to the value val$(\mathcal{P})$ of the objective function at the global solution to $\mathcal{P}$.

If the algorithm terminates in a finite number of iterations, it either returns an iterate $\underline{x}_l$ that is feasible for $\mathcal{P}$ (after the feasibility test for $\mathcal{P}$) and therefore a global solution for $\mathcal{P}$, or it returns a best upper bounding solution x_{UB}, which as a solution for $\mathcal{R}$ is feasible for $\mathcal{P}$ and has the same value as any global solution for $\mathcal{P}$ (since $UB = LB$).

In the case that the algorithm generates an infinite sequence $\{\underline{x}^l\}$ of iterates, let $\underline{x}^*$ be a limit point of this sequence, i.e., there exists a subsequence $\{\underline{x}^{\tilde{l}}\}_{\tilde{l}=1}^{\infty}$ of $\{\underline{x}^l\}_{l=1}^{\infty}$ converging to $\underline{x}^*$. As a solution of $\mathcal{Q}^{\tilde{l}}$, each $\underline{x}^{\tilde{l}}$ satisfies the constraints of $\mathcal{P}$, except for

$$r_i(x) + \sum_{k \in H(i)} g_{ik}(x_k) \leq 0 \ , \ \forall i \in M, \tag{2.8}$$

because the "$g_{ik}(x_k)$" terms are replaced by a piecewise convex relaxation, see (2.1). We denote the values of their approximation by $\tilde{g}_{ik}^{\tilde{l}}(x_k)$. Then, the solutions of $\mathcal{Q}^{\tilde{l}}$ satisfy

$$r_i(\underline{x}^{\tilde{l}}) + \sum_{k \in H(i)} \tilde{g}_{ik}^{\tilde{l}}(\underline{x}_k^{\tilde{l}}) \leq 0 \ , \ \forall i \in M. \tag{2.9}$$

Now choose $\tilde{l}_1, \tilde{l}_2$ with $\tilde{l}_2 > \tilde{l}_1$. Because the approximation $\tilde{g}_{ik}^{\tilde{l}_2}(x_k)$ is defined to coincide with the convex parts of $g_{ik}(x_k)$ and is otherwise a linear interpolation between breakpoints (note that

$\underline{x}_k^{\tilde{l}_1}$ is a breakpoint for $\mathcal{Q}^{\tilde{l}_2}$ if $\underline{x}_k^{\tilde{l}_1}$ is in an interval where $g_{ik}(x_k)$ is concave), the Lipschitz-continuity of the $g_{ik}(x_k)$ gives us

$$\tilde{g}_{ik}^{\tilde{l}_2}(\underline{x}_k^{\tilde{l}_2}) \geq g_{ik}(\underline{x}_k^{\tilde{l}_1}) - L_g|\underline{x}_k^{\tilde{l}_2} - \underline{x}_k^{\tilde{l}_1}|.$$

Together with (2.9) we therefore obtain

$$\begin{aligned} r_i(\underline{x}^{\tilde{l}_2}) + \sum_{k\in H(i)} g_{ik}(\underline{x}_k^{\tilde{l}_1}) &\leq r_i(\underline{x}^{\tilde{l}_2}) + \sum_{k\in H(i)} \tilde{g}_{ik}^{\tilde{l}_2}(\underline{x}_k^{\tilde{l}_2}) + L_g \sum_{k\in H(i)} |\underline{x}_k^{\tilde{l}_2} - \underline{x}_k^{\tilde{l}_1}| \\ &\leq L_g \sum_{k\in H(i)} |\underline{x}_k^{\tilde{l}_2} - \underline{x}_k^{\tilde{l}_1}| \end{aligned}$$

for all $i \in M$. Because $\tilde{l}_1, \tilde{l}_2$ with $\tilde{l}_2 > \tilde{l}_1$ have been chosen arbitrarily, taking the limit as $\tilde{l}_1, \tilde{l}_2 \to \infty$ and using the continuity of r shows that $\underline{x}^*$ satisfies (2.8). The continuity of the remaining constraints in $\mathcal{P}$ ensure that $\underline{x}^*$ is feasible for $\mathcal{P}$. Because $\text{val}(\mathcal{Q}^{\tilde{l}}) \leq \text{val}(\mathcal{P})$ for all $\tilde{l}$, we finally obtain $\text{val}(\mathcal{Q}^*) \leq \text{val}(\mathcal{P})$, so that $\underline{x}^*$ must be a global solution of $\mathcal{P}$. □

3. Computational results. We implemented our algorithmic framework as an `AMPL` script, and we used `MATLAB` as a tool for numerical convexity analysis, `BONMIN` as our convex MINLP solver, and `IPOPT` as our NLP solver.

We used `MATLAB` to detect the subintervals of convexity and concavity for the non-convex univariate functions in the model. In particular, `MATLAB` reads a text file generated by the `AMPL` script, containing the constraints with univariate non-convex functions, together with the names and bounds of the independent variables. With this information, using the Symbolic Math Toolbox, `MATLAB` first computes the formula for the second derivative of each univariate non-convex function, and then computes its zeros to split the function into subintervals of convexity and concavity. The zeros are computed in the following manner. On points of a rather fine uniform discretization, we evaluate the second derivative. Then, between pairs of adjacent points for which the second derivative changes sign, we compute precisely the associated zero using the `MATLAB` function "fzero". For each univariate non-convex function, we use `MATLAB` to return the number of subintervals, the breakpoints, and associated function values in a text file which is read by the `AMPL` script.

In this section we present computational results for four problem categories. The tests were executed on a single processor of an Intel Core2 CPU 6600, 2.40 GHz with 1.94 GB of RAM, using a time limit of 2 hours per instance. The relative optimality gap and feasibility tolerance used for all the experiments is 10^{-4}, and we do not add a breakpoint if it would be within 10^{-5} of an existing breakpoint.

For each set of problems, we describe the non-convex MINLP model $\mathcal{P}$. Two tables with computational results exhibit the behavior of our algorithm on some instances of each problem class. The first table presents the iterations of our `SC-MINLP` Algorithm, with the columns labeled as follows:

- instance: the instance name;
- var/int/cons: the total number of variables, the number of integer variables, and the number of constraints in the convex relaxation $\mathcal{Q}$;
- iter #: the iteration count;
- LB: the value of the lower bound;
- UB: the value of the upper bound;
- int change: indicated whether the integer variables in the lower bounding solution $\underline{x}$ are different compared to the previous iteration;
- time MINLP: the CPU time needed to solve the convex MINLP relaxation $\mathcal{Q}$ to optimality (in seconds);
- # br added: the number of breakpoints added at the end of the previous iteration.

The second table presents comparisons of our `SC-MINLP` Algorithm with `COUENNE` and `BONMIN`. `COUENNE` is an open-source Branch-and-Bound algorithm aimed at the global solution of MINLP

problems [2, 6]. It is an exact method for the problems we address in this paper. BONMIN is an open-source code for solving general MINLP problems [3, 4], but it is an exact method only for convex MINLPs. Here, BONMIN's nonlinear branch-and-bound option was chosen. When used for solving non-convex MINLPs, the solution returned is not guaranteed to be a global optimum. However, a few heuristic options are available in BONMIN, specifically designed to treat non-convex MINLPs. Here, we use the option that allows solving the root node with a user-specified number of different randomly-chosen starting points, continuing with the best solution found. This heuristic use of BONMIN is in contrast to its use in SC-MINLP, where BONMIN is employed only for the solution of the *convex* MINLP relaxation $\mathcal{Q}$.

The columns in the second table have the following meaning:

- instance: the instance name;
- var/int/cons: the total number of variables, the number of integer variables, and the number of constraints;
- for each approach, in particular SC-MINLP, COUENNE, BONMIN 1, BONMIN 50, we report:
  - time (LB): the CPU time (or the value of the lower bound (in parentheses) if the time limit is reached);
  - UB: the value of the upper bound.

BONMIN 1 and BONMIN 50 both refer to the use of BONMIN, but they differ in the number of multiple solutions of the root node; in the first case, the root node is solved just once, while in the second case, 50 randomly-generated starting points are given to the root-node NLP solver. If BONMIN reached the time limit, we do not report the lower bound because BONMIN cannot determine a valid lower bound for a non-convex problem.

3.1. Uncapacitated Facility Location (UFL) problem. The UFL application is presented in [12]. The set of customers is denoted with T and the set of facilities is denoted with K (w_{kt} is the fraction of demand of customer t satisfied by facility k for each $t \in T, k \in K$). Univariate non-convexity in the model arises due to nonlinear shipment costs. The UFL model formulation is as follows:

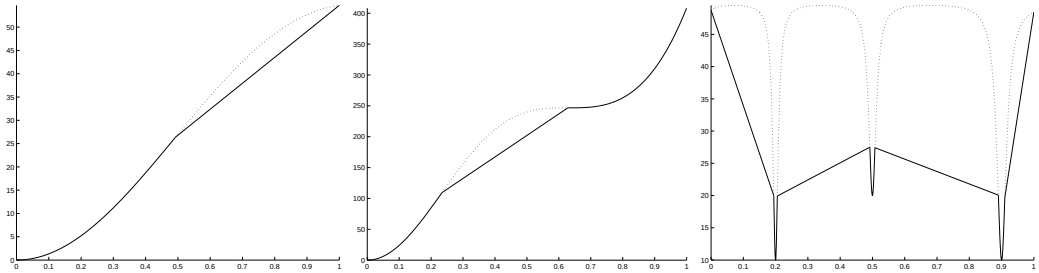
$$
\begin{array}{l}
\min \ \sum_{k\in K} C_k y_k + \sum_{t\in T} v_t \\
\quad \text{subject to} \\
v_t \geq -\sum_{k\in K} S_{kt} s_{kt} \ , \ \forall\, t \in T \ ; \\
s_{kt} \leq g_{kt}(w_{kt}) \ , \ \forall\, t \in T \ ; \\
w_{kt} \leq y_k \ , \ \forall\, t \in T, k \in K \ ; \\
\sum_{k\in K} w_{kt} = 1 \ , \ \forall\, t \in T \ ; \\
w_{kt} \geq 0 \ , \ \forall\, t \in T, k \in K \ ; \\
y_k \in \{0,1\} \ , \ \forall\, k \in K \ .
\end{array}
$$

Figure 5 depicts the three different nonlinear functions $g_{kt}(w_{kt})$ that were used for the computational results presented in Tables 1 and 2. The dashed line depicts the non-convex function, while the solid line indicates the initial piecewise-convex underestimator. Note that the third function was intentionally designed to be pathological and challenging for SC-MINLP. The remaining problem data was randomly generated.

TABLE 1
Results for Uncapacitated Facility Location problem

instance	var/int/cons	iter #	LB	UB	int change	time MINLP	# br added
ufl_1	153/39/228	1	4,122.000	4,330.400	-	1.35	-
	...	2	4,324.780	4,330.400	no	11.84	11
	...	3	4,327.724	4,330.400	no	19.17	5
	...	4	4,328.993	4,330.400	no	30.75	5
	205/65/254	5	4,330.070	4,330.400	no	45.42	5
ufl_2	189/57/264	1	27,516.600	27,516.569	-	4.47	-
ufl_3	79/21/101	1	1,947.883	2,756.890	-	2.25	-
	...	2	2,064.267	2,756.890	no	2.75	2
	87/25/105	3	2,292.743	2,292.777	no	3.06	2

In Table 1 the performance of SC-MINLP is shown. For the first instance, the global optimum is found at the first iteration, but 4 more iteration are needed to prove global optimality. In the second



(a) Instance `ufl_1` (b) Instance `ufl_2` (c) Instance `ufl_3`

FIG. 5. *UFL: Shapes of $-g_{kt}(w_{kt})$ for the three instances.*

TABLE 2
Results for Uncapacitated Facility Location problem

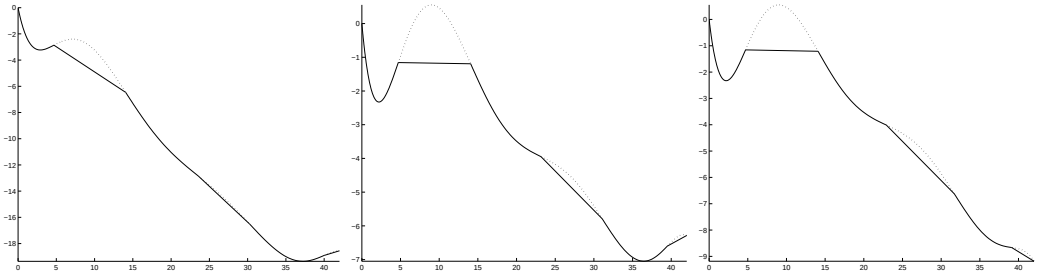
		SC-MINLP		COUENNE		BONMIN 1		BONMIN 50	
instance	var/int/cons original	time (LB)	UB	time (LB)	UB	time	UB	time	UB
ufl_1	45/3/48	116.47	4,330.400	529.49	4,330.400	0.32	4,330.400	369.85	4,330.39
ufl_2	45/3/48	17.83	27,516.569	232.85	27,516.569	0.97	27,516.569	144.06	27,516.569
ufl_3	32/2/36	8.44	2,292.777	0.73	2,292.775	3.08	2,292.777	3.13	2,292.775

instance, only one iteration is needed. In the third instance, the first feasible solution found is not the global optimum which is found at the third (and last) iteration. Table 2 demonstrates good performance of SC-MINLP. In particular, instance `ufl_1` is solved in about 117 seconds compared to 530 seconds needed by COUENNE, instance `ufl_2` in less than 18 seconds compared to 233 seconds. In instance `ufl_3`, COUENNE performs better than SC-MINLP, but this instance is really quite easy for both algorithms. BONMIN 1 finds solutions to all three instances very quickly, and these solutions turn out to be globally optimal (but note, however, that BONMIN 1 is a heuristic algorithm and no guarantee of the global optimality is given). BONMIN 50 also finds the three global optima, but in non-negligible time (greater than the one needed by SC-MINLP in 2 out of 3 instances).

3.2. Hydro Unit Commitment and Scheduling problem. The Hydro Unit Commitment and Scheduling problem is described in [5]. Univariate non-convexity in the model arises due to the dependence of the power produced by each turbine on the water flow passing through the turbine. The following model was used for the computational results of Tables 3 and 4:

$$
\begin{aligned}
&\min\ -\textstyle\sum_{j\in J}\sum_{t\in T}\Big(\Delta t\,\Pi_t\,p_{jt} - C_j\,\widetilde{w}_{jt} - (D_j+\Pi_t E_j)\widetilde{y}_{jt}\Big)\\
&\text{subject to}\\
&v_{\overline{t}} - V_{\overline{t}} = 0\ ;\\
&v_t - v_{t-1} - 3600\,\Delta t\,(I_t - \textstyle\sum_{j\in J} q_{jt} - s_t) = 0\ ,\ \forall t\in T\ ;\\
&q_{jt} - (Q_j^-\,u_{jt} + \underline{Q}_j\,g_{jt}) \geq 0\ ,\ \forall j\in J, t\in T\ ;\\
&q_{jt} - (Q_j^-\,u_{jt} + \overline{Q}_j\,g_{jt}) \leq 0\ ,\ \forall j\in J, t\in T\ ;\\
&\textstyle\sum_{j\in J}(q_{jt} - q_{j(t-1)}) + \Delta q^- \geq 0\ ,\ \forall t\in T\ ;\\
&\textstyle\sum_{j\in J}(q_{jt} - q_{j(t-1)}) - \Delta q^+ \leq 0\ ,\ \forall t\in T\ ;\\
&s_t - \textstyle\sum_{j\in J}(W_j\,\widetilde{w}_{jt} + Y_j\,\widetilde{y}_{jt}) \geq 0\ ,\ \forall t\in T\ ;\\
&\textstyle\sum_{j\in J} q_{jt} + s_t - \underline{\Theta} \geq 0\ ,\ \forall t\in T\ ;\\
&g_{jt} - g_{j(t-1)} - (\widetilde{w}_{jt} - w_{jt}) = 0\ ,\ \forall j\in J, t\in T\ ;\\
&\widetilde{w}_{jt} + w_{jt} \leq 1\ ,\ \forall j\in J, t\in T\ ;\\
&u_{jt} - u_{j(t-1)} - (\widetilde{y}_{jt} - y_{jt}) = 0\ ,\ \forall j\in J, t\in T\ ;\\
&\widetilde{y}_{jt} + y_{jt} \leq 1\ ,\ \forall j\in J, t\in T\ ;\\
&g_{jt} + u_{kt} \leq 1\ ,\ \forall j,k\in J, t\in T\ ;\\
&\textstyle\sum_{j\in J} u_{jt} \leq \overline{n} - 1\ ,\ \forall t\in T\ ;\\
&p_{jt} - \varphi(q_{jt}) = 0\ ,\ \forall j\in J, t\in T.
\end{aligned}
$$

Figure 6 shows the non-convex functions $\varphi(q_{jt})$ used for the three instances. The remaining problem data was chosen according to [5].



(a) Instance `hydro_1` (b) Instance `hydro_2` (c) Instance `hydro_3`

FIG. 6. *Hydro UC: Shapes of $-\varphi(q_{jt})$ for the three instances*

TABLE 3
Results for Hydro Unit Commitment and Scheduling problem

instance	var/int/cons	iter #	LB	UB	int change	time MINLP	# br added
hydro_1	324/142/445	1	-10,231.039	-10,140.763	-	18.02	-
	332/146/449	2	-10,140.760	-10,140.763	no	23.62	4
hydro_2	324/142/445	1	-3,950.697	-3,891.224	-	21.73	-
	...	2	-3,950.583	-3,891.224	no	21.34	2
	...	3	-3,950.583	-3,891.224	no	27.86	2
	336/148/451	4	-3,932.182	-3,932.182	no	38.20	2
hydro_3	324/142/445	1	-4,753.849	-4,634.409	-	59.33	-
	...	2	-4,719.927	-4,660.189	no	96.93	4
	336/148/451	3	-4,710.734	-4,710.734	yes	101.57	2

TABLE 4
Results for Hydro Unit Commitment and Scheduling problem

		SC-MINLP		COUENNE		BONMIN 1		BONMIN 50	
instance	var/int/cons original	time (LB)	UB	time (LB)	UB	time	UB	time	UB
hydro_1	124/62/165	107.77	-10,140.763	(-11,229.80)	-10,140.763	5.03	-10,140.763	5.75	-7,620.435
hydro_2	124/62/165	211.79	-3,932.182	(-12,104.40)	-2,910.910	4.63	-3,928.139	7.02	-3,201.780
hydro_3	124/62/165	337.77	-4,710.734	(-12,104.40)	-3,703.070	5.12	-4,131.095	13.76	-3,951.199

Our computational results are reported in Tables 3 and 4. We observe good performance of `SC-MINLP`. It is able to find the global optimum of the three instances within the time limit, but `COUENNE` does not solve to global optimality any of the instances. Also, `BONMIN 1` and `BONMIN 50` show good performance. In particular, often a good solution is found in few seconds, and `BONMIN 1` finds the global optimum in one case.

3.3. Nonlinear Continuous Knapsack problem. In this subsection, we present results for the Nonlinear Continuous Knapsack problem. This purely continuous problem can be motivated as a continuous resource-allocation problem. A limited resource of amount C (such as advertising dollars) has to be partitioned for different categories $j \in N$ (such as advertisements for different products). The objective function is the overall return from all categories. The non-convexity arises because a small amount of allocation provides only a small return, up to a threshold, when the advertisements are noticed by the consumers and result in substantial sales. On the other hand, at some point, saturation sets in, and an increase of advertisement no longer leads to a significant increase in sales.

Our model is the non-convex NLP problem:

$$\begin{array}{l} \min \ -\sum_{j\in N} p_j \\ \quad \text{subject to} \\ p_j - g_j(x_j) \le 0 \ , \ \forall j \in N \ ; \\ \sum_{j\in N} x_j \le C \ ; \\ 0 \le x_j \le U \ , \ \forall j \in N \ . \end{array}$$

where $g_j(x_j) = c_j/(1 + b_j \exp^{-a_j(x_j+d_j)})$.

Note that the sigmoid function $g_j(x_j)$ is increasing, and it is convex up to the inflection point at $x = -d_j + \ln(b_j)/a_j$, whereupon it becomes concave.

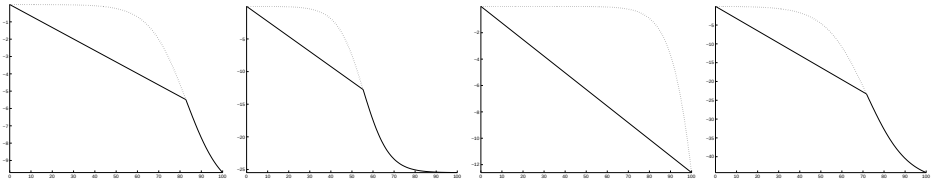


FIG. 7. *Example shapes of* $-g_j(x_j)$ *for instance* `nck_20_100`

The instances were generated randomly. In particular, independently for each $j \in N$, a_j was uniformly generated in the interval $[0.1, 0.2]$, b_j and c_j in the interval $[0, U]$ and d_j in the interval $[-U, 0]$. The name of each instance contains information about the values $|N|$ and C, namely, `nck_`$|N|$`_`C.

Our computational results are reported in Tables 5 and 6. `SC-MINLP` finds the global optimum for all the 6 instances in less than 3 minutes. `COUENNE` is able to close the gap for only 2 instances within the time limit. `BONMIN 1` and `BONMIN 50` terminate quickly, but the global optimum is found only for 1 instance for `BONMIN 1` and 2 instances for `BONMIN 50`.

TABLE 5
Results for Nonlinear Continuous Knapsack problem

instance	var/int/cons	iter #	LB	UB	int change	time MINLP	# br added
nck_20_100	144/32/205	1	-162.444	-159.444	-	0.49	-
	146/33/206	2	-159.444	-159.444	-	0.94	1
nck_20_200	144/32/205	1	-244.015	-238.053	-	0.67	-
	...	2	-241.805	-238.053	-	0.83	1
	...	3	-241.348	-238.053	-	1.16	1
	...	4	-240.518	-238.053	-	1.35	1
	...	5	-239.865	-238.053	-	1.56	1
	...	6	-239.744	-238.053	-	1.68	1
	156/38/211	7	-239.125	-239.125	-	1.81	1
nck_20_450	144/32/205	1	-391.499	-391.337	-	0.79	-
	146/32/206	2	-391.364	-391.337	-	0.87	1
nck_50_400	356/78/507	1	-518.121	-516.947	-	4.51	-
	...	2	-518.057	-516.947	-	14.94	2
	...	3	-517.837	-516.947	-	23.75	2
	...	4	-517.054	-516.947	-	25.07	2
	372/86/515	5	-516.947	-516.947	-	31.73	2
nck_100_35	734/167/1035	1	-83.580	-79.060	-	3.72	-
	...	2	-82.126	-81.638	-	21.70	2
	...	3	-82.077	-81.638	-	6.45	2
	744/172/1040	4	-81.638	-81.638	-	11.19	1
nck_100_80	734/167/1035	1	-174.841	-171.024	-	6.25	-
	...	2	-173.586	-172.631	-	24.71	2
	742/171/1039	3	-172.632	-172.632	-	12.85	2

TABLE 6
Results for Nonlinear Continuous Knapsack problem

		SC-MINLP		COUENNE		BONMIN 1		BONMIN 50	
instance	var/int/cons original	time (LB)	UB	time (LB)	UB	time	UB	time	UB
nck_20_100	40/0/21	15.76	-159.444	3.29	-159.444	0.02	-159.444	1.10	-159.444
nck_20_200	40/0/21	23.76	-239.125	(-352.86)	-238.053	0.03	-238.053	0.97	-239.125
nck_20_450	40/0/21	15.52	-391.337	(-474.606)	-383.149	0.07	-348.460	0.84	-385.546
nck_50_400	100/0/51	134.25	-516.947	(-1020.73)	-497.665	0.08	-438.664	2.49	-512.442
nck_100_35	200/0/101	110.25	-81.638	90.32	-81.638	0.04	-79.060	16.37	-79.060
nck_100_80	200/0/101	109.22	-172.632	(-450.779)	-172.632	0.04	-159.462	15.97	-171.024

3.4. GLOBALLib and MINLPLib instances. Our algorithm is suitable for problems with functions having all non-convexities manifested as sums of univariate non-convex functions, and moreover the variables in the univariate non-convex functions should be bounded. We selected 16 instances from GLOBALLib [11] and 9 from MINLPLib [15] to test our algorithm. These instances were selected because they were easily reformulated to have these properties. Note that we are not advocating solution of general global optimization problems via reformulation as problems of the form $\mathcal{P}$, followed by application of our algorithm `SC-MINLP`. We simply produced these reformulation to enlarge our set of test problems.

To reformulate these instances in order to have univariate non-convex functions, we used as a starting point the reformulation performed by `COUENNE` [2]. It reformulates MINLP problems in

order to have the following "basic operators": sum, product, quotient, power, exp, log, sin, cos, abs. The only non-univariate basic operators are product and quotient (power x^y is converted to $e^{y\log(x)}$). In these cases we modified the reformulation of COUENNE in order to have only univariate non-convexities:

- product: xy is transformed into $(w_1^2 - w_2^2)/4$ with $w_1 = x + y$ and $w_2 = x - y$;
- quotient: x/y is transformed into xw and $w = 1/y$ and xw is then treated as any other product.

We report in the first part of the table the GLOBALLib results. The second part of the table reports the MINLPLib results. The instances were selected among the ones reported in the computational results of a recent paper [2]. Some of these particular MINLPLib instances are originally convex, but their reformulation is non-convex. It is clear that the employment of specific solvers for convex MINLPs such as BONMIN to solve this kind of problem is much more efficient and advisable. However, we used the non-convex reformulation of these instances to extend our test bed. In Table 8, the second column reports the number of variables, integer variables and constraints of the reformulated problem.

Table 8 shows that COUENNE performs much better than SC-MINLP on the GLOBALLib instances that we tested. But on such small instances, COUENNE behaves very well in general, so there cannot be much to recommend any alternative algorithm. Also BONMIN 1 and BONMIN 50 perform well. They find global optima in 10 (resp. 13) out of the 16 GLOBALLib instances rather quickly. However, in 3 (resp. 1) instances they fail to produce a feasible solution.

Concerning the MINLPLib instances, 4 out of 9 instances are solved to global optimality by SC-MINLP. COUENNE finishes within the imposed time limit on precisely the same 4 instances (in one case, COUENNE with default settings incorrectly returned a solution with a worse objective function). In the 5 instances not solved (by both solvers) within the imposed time limit, the lower bound given by SC-MINLP is always better (higher) than the one provided by COUENNE. This result emphasizes the quality of the lower bound computed by the solution of the convex MINLP relaxation $\mathcal{Q}$. Moreover, the upper bound computed by SC-MINLP is better in 2 instances out of 5. Concerning BONMIN 1 and BONMIN 50 performance, when they terminate before the time limit is reached (4 instances out of 9), the solution computed is globally optimal for 3 instances. Note that in these cases, the CPU time needed by BONMIN 1 and BONMIN 50 is often greater than that needed by SC-MINLP. When the time limit is reached, BONMIN 1 computes a better solution than SC-MINLP in only 1 instance out of 5 (for 1 instance they provide the same solution) and BONMIN 50 computes a better solution than SC-MINLP in 1 only instance out of 5.

Finally, we note that for several instances of applying SC-MINLP, in just the second iteration, the convex MINLP solver BONMIN was in the midst of working when our self-imposed time limit of 2 hours was reached. In many of these cases, much better lower bounds could be achieved by increasing the time limit. Generally, substantial improvements in BONMIN would produce a corresponding improvement in the results obtained with SC-MINLP.

4. Conclusions. In this paper, we proposed an algorithm for solving to global optimality a broad class of separable MINLPs. Our simple algorithm, implemented within the AMPL modeling language, works with a lower-bounding convex MINLP relaxation and an upper-bounding non-convex NLP restriction. For the definition of the lower-bounding problem, we identify subintervals of convexity and concavity for the univariate functions using external calls to MATLAB; then we develop a convex MINLP relaxation of the problem approximating the concave intervals of each non-convex function with the linear relaxation. The subintervals are glued together using binary variables. We iteratively refine our convex MINLP relaxation by modifying it at the modeling level. The upper-bound is obtained by fixing the integer variables in the original non-convex MINLP, then locally solving the associated non-convex NLP restriction. We presented preliminary computational experiments on models from a variety of application areas, including problems from GLOBALLib and MINLPLib. We compared our algorithm with the open-source solvers COUENNE as an exact approach, and BONMIN as a heuristic approach, obtaining significant success.

TABLE 7
Results for GLOBALLib and MINLPLib

reformulated instance	var/int/cons	iter #	LB	UB	int change	time MINLP	# br added
ex14_2_1	239/39/358	1	0.000	0.000	-	5.35	-
ex14_2_2	110/18/165	1	0.000	0.000	-	3.33	-
ex14_2_6	323/53/428	1	0.000	0.000	-	7.02	-
ex14_2_7	541/88/808	1	0.000	0.000	-	1.06	-
ex2_1_1	27/5/38	1	-18.900	-16.500	-	0.01	-
	...	2	-18.318	-16.500	-	0.06	1
	...	3	-18.214	-16.500	-	0.12	1
	...	4	-18.000	-16.500	-	0.15	1
	...	5	-17.625	-17.000	-	0.22	2
	39/11/44	6	-17.000	-17.000	-	0.26	1
ex2_1_2	29/5/40	1	-213.000	-213.000	-	0.01	-
ex2_1_3	36/4/41	1	-15.000	-15.000	-	0.00	-
ex2_1_4	15/1/16	1	-11.000	-11.000	-	0.00	-
ex2_1_5	50/7/72	1	-269.453	-268.015	-	0.01	-
	54/9/74	2	-268.015	-268.015	-	0.15	2
ex2_1_6	56/10/81	1	-44.400	-29.400	-	0.01	-
	...	2	-40.500	-39.000	-	0.16	2
	...	3	-40.158	-39.000	-	0.25	1
	66/15/86	4	-39.000	-39.000	-	0.52	2
ex2_1_7	131/20/181	1	-13,698.362	-4,105.278	-	0.07	-
		2	-10,643.558	-4,105.278	-	1.34	11
		3	-8,219.738	-4,105.278	-	2.68	5
		4	-6,750.114	-4,105.278	-	4.66	10
		5	-5,450.142	-4,105.278	-	16.42	11
		6	-5,014.019	-4,105.278	-	32.40	6
		7	-4,740.743	-4,105.278	-	38.61	15
		8	-4,339.192	-4,150.410	-	86.47	4
		9	-4,309.425	-4,150.410	-	133.76	11
		10	-4,250.248	-4,150.410	-	240.50	6
		11	-4,156.125	-4,150.410	-	333.08	5
	309/109/270	12	-4,150.411	-4,150.410	-	476.47	5
ex9_2_2	68/14/97	1	55.556	100.000	-	0.00	-
	...	2	78.679	100.000	-	0.83	10
	...	3	95.063	100.000	-	5.77	19
	...	4	98.742	100.000	-	18.51	8
	...	5	99.684	100.000	-	37.14	11
	184/72/155	6	99.960	100.000	-	56.00	10
ex9_2_3	90/18/125	1	-30.000	0.000	-	0.00	-
	...	2	-30.000	0.000	-	2.29	12
	...	3	-30.000	0.000	-	6.30	12
	...	4	-30.000	0.000	-	8.48	12
	...	5	-24.318	0.000	-	30.44	22
	...	6	-23.393	0.000	-	16.25	10
	...	7	-21.831	0.000	-	31.92	12
	...	8	-19.282	0.000	-	26.53	12
	...	9	-7.724	0.000	-	203.14	17
	332/139/246	10	-0.000	0.000	-	5,426.48	12
ex9_2_6	105/22/149	1	-1.500	3.000	-	0.02	-
	...	2	-1.500	1.000	-	9.30	28
	...	3	-1.500	0.544	-	34.29	15
	...	4	-1.500	-1.000	-	48.33	12
	...	5	-1.500	-1.000	-	130.04	22
	...	6	-1.500	-1.000	-	50.29	24
	...	7	-1.500	-1.000	-	53.66	13
	...	8	-1.500	-1.000	-	67.93	22
	...	9	-1.500	-1.000	-	103.13	14
	...	10	-1.366	-1.000	-	127.60	12
	...	11	-1.253	-1.000	-	1,106.78	22
	...	12	-1.116	-1.000	-	3,577.48	13
	...	13	-1.003	-1.000	-	587.11	15
	557/248/375	14	-1.003	-1.000	-	1,181.12	14
ex5_2_5	802/120/1,149	1	-34,200.833	-3,500.000	-	0.10	-
	1,280/359/1,388	2	-23,563.500	-3,500.000	-	7,192.22	239
ex5_3_3	1,025/181/1,474	1	-67,499.021	-	-	0.17	-
	1,333/335/1,628	2	-16,872.900	3.056	-	7,187.11	154
du-opt	458/18/546	1	3.556	3.556	-	5.30	-
du-opt5	455/15/545	1	8.073	8.073	-	9.38	-
fo7	366/42/268	1	8.759	22.518	-	7,200	-
m6	278/30/206	1	82.256	82.256	-	182.72	-
no7_ar2_1	421/41/325	1	90.583	127.774	-	7,200	-
no7_ar3_1	421/41/325	1	81.539	107.869	-	7,200	-
no7_ar4_1	421/41/325	1	76.402	104.534	-	7,200	-
o7_2	366/42/268	1	79.365	124.324	-	7,200	-
stockcycle	626/432/290	1	119,948.675	119,948.676	-	244.67	-

Acknowledgment. This work was partially developed when the first author was visiting the IBM T.J. Watson Research Center, and their support is gratefully acknowledged. We also thank Pietro Belotti for discussions about the modification of COUENNE to reformulate GLOBALLib and MINLPLib instances.

REFERENCES

[1] E. BEALE AND J. TOMLIN, *Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables*, in Proc. of the 5^{th} Int. Conf. on Operations Research, J. Lawrence, ed., 1970, pp. 447–454.

[2] P. BELOTTI, J. LEE, L. LIBERTI, F. MARGOT, AND A. WÄCHTER, *Branching and bounds tightening techniques*

TABLE 8
Results for GLOBALLib and MINLPLib

reformulated instance	var/int/cons original	SC-MINLP time (LB)	SC-MINLP UB	COUENNE time (LB)	COUENNE UB	BONMIN 1 time	BONMIN 1 UB	BONMIN 50 time	BONMIN 50 UB
ex14_2_1	122/0/124	21.81	0.000	0.18	0.000	0.13	0.000	46.25	0.000
ex14_2_2	56/0/57	12.86	0.000	0.10	0.000	0.05	0.000	28.02	0.000
ex14_2_6	164/0/166	42.56	0.000	1.03	0.000	1.25	0.000	138.02	0.000
ex14_2_7	277/0/280	128.70	0.000	0.63	0.000	0.28	0.000	170.62	0.000
ex2_1_1	12/0/8	2.84	-17.000	0.16	-17.000	0.04	-11.925	0.89	-17.000
ex2_1_2	14/0/10	1.75	-213.000	0.06	-213.000	0.02	-213.000	0.66	-213.000
ex2_1_3	24/0/17	1.62	-15.000	0.04	-15.000	0.02	-15.000	0.36	-15.000
ex2_1_4	12/0/10	1.28	-11.000	0.04	-11.000	0.03	-11.000	0.58	-11.000
ex2_1_5	29/0/30	2.21	-268.015	0.13	-268.015	0.03	-268.015	0.96	-268.015
ex2_1_6	26/0/21	3.31	-39.000	0.12	-39.000	0.04	-39.000	1.11	-39.000
ex2_1_7	71/0/61	1388.00	4,150.410	3.98	4,150.410	0.04	4,150.410	13.64	4,150.410
ex9_2_2	38/0/37	1,355.47	100.000	0.36	100.000	0.12	-	2.65	100.000
ex9_2_3	54/0/53	5,755.76	0.000	0.73	0.000	0.08	15.000	2.85	5.000
ex9_2_6	57/0/53	(-1.003)	-1.000	0.78	-1.000	0.21	-	4.04	-1.000
ex5_2_5	442/0/429	(-23,563.500)	-3,500.000	(-11,975.600)	-3,500.000	2.06	-3,500.000	315.53	-3,500.000
ex5_3_3	563/0/550	(-16,872.900)	3.056	(-16,895.400)	3.056	20.30	-	22.92	-
du-opt	242/18/230	13.52	3.556	38.04	3.556	41.89	3.556	289.88	3.556
du-opt5	239/15/227	17.56	8.073	37.96	8.073	72.32	8.118	350.06	8.115
fo7	338/42/437	(8.759)	22.518	(1.95)	22.833	-	22.518	-	24.380
m6	254/30/327	185.13	82.256	54.13	82.256	154.42	82.256	211.49	82.256
no7_ar2_1	394/41/551	(90.583)	127.774	(73.78)	111.141	-	122.313	-	107.871
no7_ar3_1	394/41/551	(81.539)	107.869	(42.19)	113.810	-	121.955	-	119.092
no7_ar4_1	394/41/551	(76.402)	104.534	(54.85)	98.518	-	117.992	-	124.217
o7_2	338/42/437	(79.365)	124.324	(5.85)	133.988	-	126.674	-	130.241
stockcycle	578/480/195	251.54	119,948.676	84.35	119, 948.676*	321.89	119,948.676	328.03	119,948.676

* This time and correct solution were obtained with non-default options of Couenne (which failed with default settings).

for non-convex MINLP, 2008. IBM Research Report RC24620; to appear in: Optimization Methods and Software.

[3] P. BONAMI, L. BIEGLER, A. CONN, G. CORNUÉJOLS, I. GROSSMANN, C. LAIRD, J. LEE, A. LODI, F. MARGOT, N. SAWAYA, AND A. WÄCHTER, *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optimization, 5 (2008), pp. 186–204.

[4] BONMIN. `projects.coin-or.org/Bonmin`, v. 1.0.

[5] A. BORGHETTI, C. D'AMBROSIO, A. LODI, AND S. MARTELLO, *An MILP approach for short-term hydro scheduling and unit commitment with head-dependent reservoir*, IEEE Transactions on Power Systems, 23 (2008), pp. 1115–1124.

[6] COUENNE. `projects.coin-or.org/Couenne`, v. 0.1.

[7] C. D'AMBROSIO, J. LEE, AND A. WÄCHTER, *A global-optimization algorithm for mixed-integer nonlinear programs having separable non-convexity*, in Proc. of 17th Annual European Symposium on Algorithms (ESA), Copenhagen, Denmark. *Lecture Notes in Computer Science*, A. Fiat, ed., 2009 (to appear).

[8] M. DURAN AND I. GROSSMANN, *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Mathematical Programming, 36 (1986), pp. 307–339.

[9] R. FLETCHER AND S. LEYFFER, *Solving mixed integer nonlinear programs by outer approximation*, Mathematical Programming, 66 (1994), pp. 327–349.

[10] R. FOURER, D. GAY, AND B. KERNIGHAN, *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press/Brooks/Cole Publishing Co., second ed., 2003.

[11] GLOBALLIB. `www.gamsworld.org/global/globallib.htm`.

[12] O. GÜNLÜK, J. LEE, AND R. WEISMANTEL, *MINLP strengthening for separable convex quadratic transportation-cost UFL*, 2007. IBM Research Report RC24213.

[13] L. LIBERTI, *Writing global optimization software*, in Global Optimization: From Theory to Implementation, L. Liberti and N. Maculan, eds., Springer, Berlin, 2006, pp. 211–262.

[14] MATLAB. `www.mathworks.com/products/matlab/`, R2007a.

[15] MINLPLIB. `www.gamsworld.org/minlp/minlplib.htm`.

[16] I. NOWAK, H. ALPERIN, AND S. VIGERSKE, *LaGO – an object oriented library for solving MINLPs*, in Global Optimization and Constraint Satisfaction, vol. 2861 of Lecture Notes in Computer Science, Springer, Berlin Heidelberg, 2003, pp. 32–42.

[17] I. QUESADA AND I. GROSSMANN, *An LP/NLP based branch and bound algorithm for convex MINLP optimization problems*, Computer & Chemical Engineering, 16 (1992), pp. 937–947.

[18] N. SAHINIDIS, *BARON: A general purpose global optimization software package*, J. Global Opt., 8 (1996), pp. 201–205.

[19] A. WÄCHTER AND L. T. BIEGLER, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, Mathematical Programming, 106 (2006), pp. 25–57.