On the approximation of separable non-convex optimization programs to an arbitrary numerical precision

Claudio Contardo, Sandra Ulrich Ngueveu

On the approximation of separable non-convex optimization programs to an arbitrary numerical precision

Claudio Contardo[1] Sandra Ulrich Ngueveu[2]

[1]IBM, Toronto, Canada
[2]Université de Toulouse, CNRS, INP, LAAS, Toulouse, France
August 30, 2021

Abstract

We consider the problem of minimizing the sum of a series of univariate (possibly non-convex) functions on a polyhedral domain. We introduce an iterative method with optimality guarantees to approximate this problem to an arbitrary numerical precision. At every iteration, our method replaces the objective by a lower bounding piecewise linear approximation to compute a dual bound. A primal bound is computed by evaluating the cost function on the solution provided by the approximation. If the difference between these two values is deemed as not satisfactory, the approximation is locally tightened and the process repeated. By keeping the scope of the update local, the computational burden is only slightly increased from iteration to iteration. The convergence of the method is assured under very mild assumptions, and no NLP nor MINLP solver/oracle is required to ever be invoked to do so. As a consequence, our method presents very nice scalability properties and is little sensitive to the desired precision. We provide a formal proof of the convergence of our method, and assess its efficiency in approximating the non-linear variants of three problems: the transportation problem, the capacitated facility location problem, and the multi-commodity network design problem. Our results indicate that the overall performance of our method is superior to five state-of-the-art mixed-integer nonlinear solvers by a significant margin, and scales better than a naive variant of the method that avoids performing successive iterations in exchange of solving a much larger mixed-integer linear program.

1 Introduction

We consider the problem of solving the following mixed-integer non-linear program (MINLP) with a separable objective:

$$x \in \arg\min \left\{ \sum_{i=1}^{n} f_i(x_i) : Ax = b, x \in \mathcal{X} \subseteq \mathbb{R}_{n-p} \times \mathbb{N}_p \right\}, \tag{1}$$

where the functions $f_i : \mathbb{R} \longrightarrow \mathbb{R}^+, i = 1 \dots n$ are piecewise differentiable and the set $\mathcal{X}$ is used to represent the possibility of restricting some variables to be integer-valued. For the sake of simplicity, we also assume that the problem is well defined and that admits an optimal solution (although perhaps not an unique one). If $(f_i)_{i=1\dots n}$ are all affine linear functions, the problem is a conventional mixed-integer linear problem for which state-of-the-art algorithms and commercial software can scale and solve problems with millions of variables and constraints IBM (2020). This is no longer true when some of the functions f_i are not linear. While it is possible to efficiently handle problems for some specific forms of f_i —namely when they are quadratic and/or convex (IBM, 2020; Stellato et al., 2020)—, general forms of the functions f_is make the optimization problems much less tractable.

One possible way to approximate problem (1) is by replacing functions f_is by piecewise linear functions. This procedure results in a mixed-integer problem (MIP) with additional binary variables, and can be tackled using state-of-the-art machinery from the integer programming literature. This is perhaps the most efficient way known to handle problems with this structure. By properly choosing the piecewise linear approximations, this approach can provide guarantees on the quality of the solutions achieved. These guarantees, however, come at the extent of potentially very large piecewise linear approximations, and remain only practical for very rough numerical precisions.

This article addresses the issue of solving problem (1) to an arbitrary numerical precision by solving a series of piecewise linear approximations of the problem in a way that the tractability of the resulting MIPs is not compromised along the process. Our method relies on the existence of a tractable piecewise linear approximation for a reasonably good (but probably not good enough) precision to derive primal and dual bounds, and on a refinement method used to achieve a better numerical precision by tightening the approximation. The key in the success of our method lies in the fact that the refinement procedure has very local scope, and the successive approximations, while becoming tighter, do not lose their tractability, allowing for the primal and dual bounds to converge quickly.

The remainder of this manuscript is organized as follows. In Section 2 we present a literature review that focuses on the numerical approximation of mixed integer non-linear programs using piecewise linear approximations. In Section 3 we present our method and provide the formal background to justify its convergence. In Section 4 we describe the application of our method to approximating the non-linear variants of three optimization problems relevant in practice: the transportation problem (TP), the uncapacitated facility location problem (UFLP), and the multi-commodity network design problem (MCNDP). We also provide computational evidence of the efficiency of our method. Section 5 feeds upon the computational campaign and some preliminary experiments to provide a discussion about the overall performance and limits of the proposed method. Section 6 concludes this manuscript.

2 Literature review / Related works

Two problems need to be addressed when building MILP-based approximations or relaxations of nonconvex MINLPs with a predefined accuracy : 1) that of obtaining good piecewise linear approximations of the nonlinear functions; and 2) that of efficiently constructing and solving the resulting MILP.

The quality of piecewise linear approximations is typically evaluated using two conflicting criteria: the approximation error evaluated with a relevant metric, and the size of the approximation, measured as the number of linear pieces necessary to achieve the desired precision (Ertel and Fowlkes, 1976). To ensure that the desired MINLP accuracy will be achieved, a bound is set on pointwise approximation errors, i.e that can be expressed in function of the maximal difference between each nonlinear function and its approximation (Geißler et al., 2012). The fewer the number of pieces the smaller and easier to solve the resulting MILP. Therefore, it is of interest to obtain (near-)optimal piecewise linearizations with respect to the objective of minimizing the number of linear pieces given a predefined pointwise error bound. Among the few publications that tackle this version of the piecewise linearization problem with formal models and exact algorithms to ensure optimality of its solutions, Rebennack and Kallrath (2015) and Rebennack and Krasko (2019) showed that distributing breakpoints freely and allowing shifts from the nonlinear function at breakpoints leads to an order of magnitude less linear pieces compared to equidistant breakpoints that interpolate the nonlinear univariate function. Ngueveu (2019) authorizes discontinuity in the piecewise linear function even if the original nonlinear function is continuous, yielding an additional degree of freedom to obtain a breakpoint system or equal or less linear segments. Codsi et al. (2021) propose a geometric approach that can solve the problem in quasi-logarithmic time on a very broad class of pointwise error metrics.

Simply replacing each nonlinear function with its piecewise linear approximation may lead to a

MILP approximation whose solution is not guaranteed to be feasible for the MINLP. The objective value of that MILP approximation is also not guaranteed to provide an upper bound or a lower bound for the optimal MINLP solution value. In the case of linearly constrained MINLP, such guarantee may be ensured by computing a piecewise linear underestimation or overestimation of the nonlinear objective-function, as in Ngueveu (2019).

For general nonconvex MINLP with a linear objective-function and nonlinear constraints, which can be obtained after reformulation of any nonconvex MINLP with a nonlinear objective-function, Geißler et al. (2012) present a general methodology to construct a mixed integer piecewise polyhedral (MIP) relaxation of a MINLP, instead of a mixed integer piecewise linear approximation, provided that the piecewise linear functions interpolate the nonlinear functions at the breakpoints and if the maximum linearization error has been calculated beforehand for each linear piece. The MIP relaxation produces lower bounds for the MINLP. Then the authors use an NLP solver to produce feasible solutions for the MINLP once its integer variables have been fixed to their values in the MIP relaxation solution. As a consequence, it is straightforward to implement a branch-and-bound algorithm to solve the MINLP, which can prove optimality and prune infeasible or suboptimal parts of the search tree by using MILP techniques.

The drawback of any MINLP solution method based on piecewise linear approximations is that small approximation errors lead to large MILPs, which become difficult to solve. Burlacu et al. (2020) build on the work of Geißler et al. (2012) and develop an iterative algorithm to find a global optimal solution of the MINLP by solving a series of MIP relaxations with gradually increasing accuracy, based on piecewise linear functions that are adaptively refined from one iteration to another. A critical component concerns the way the piecewise linear functions are defined and their refinement procedure. The authors need piecewise linear functions that interpolate the nonlinear function at the breakpoints and that completely contain the graph of the function. They provide rather general convergence conditions for MINLP solution algorithms that rely on the adaptive refinement of their piecewise linear relaxations. They show that the refinement strategy adding solely points with maximal approximation error on a simplex does not fulfill these conditions and thus may not converge in certain cases. In contrast, the refinement strategy adding linearization breakpoints on the longest edge of a simplex, such as the classical longest-edge bisection, fulfills these convergence conditions and therefore is suitable for the solution framework proposed.

Burlacu (2021) extends the iterative algorithm of Burlacu et al. (2020) with another refinement strategy for n-dimensional simplices: the generalized red refinement introduced by Freudenthal (1942). Their procedure is to some extent an n-dimensional generalization of the well-known red-green refinement, which is used for two-dimensional simplices. However, for the one-dimensional domains we focus on in this paper, i.e univariate functions, the red refinement and the longest edge refinement are identical and simply split the domain of the active linear pieces in two equal halves by adding, midway through the domain, a breakpoint that interpolates the function. Burlacu (2021) and Burlacu et al. (2020) do not compute any upper bounds, and thus do not require NLP solvers for such task. However, the authors assume that there is an oracle that optimizes the difference between a nonlinear and linear function over a simplex, in order to compute the linearization errors resulting from the piecewise linear function refinements. Such an oracle may be an NLP solver if the solution analytical formula is not available.

Exact solution methods that solve an instance of an NP-hard problem as a series of smaller instances of the same NP-hard problem have been investigated recently in relation to decremental and sampling mechanisms, to increase the size of instances solved to proven optimality for various MiniMax or MaxiMin combinatorial optimization problems. Chen and Chen (2009) and Contardo et al. (2019) propose decremental relaxation mechanisms to ignore some node allocation constraints of the vertex p-center problem (VPCP), which are only added as needed. The relaxed problems can thus be modeled as smaller VPCPs in an iterative manner, allowing the solution to proven optimality of problems containing up to $1M$ nodes. Aloise and Contardo (2018) consider the problem of clustering a set of points so as to minimize the maximum intra-cluster dissimilarity. They introduced a sampling

mechanism to solve the problem denoted MMDCP as a series of smaller MMDCPs in a dynamic fashion, allowing the solution to proven optimality of problems containing up to $600k$ observations. Contardo (2020) present a decremental clustering method to reduce a p-dispersion problem (pDP) to the solution of a series of smaller pDPs. Instances with of up $100k$ nodes could be solved to optimality. Sinnl (2021) proposed an iterative algorithm for the solution of a sequencing problem. Their method iterates throughout all possible values of sequence lengths, in an iterative way that exploits the model outputs from previous iterations. The authors report reductions of up to two orders of magnitude in the computing times as compared to the state-of-the-art method for the same problem.

3 The proposed method

In this section we present our iterative method to solve problem (1) to a given predefined tolerance. To fix ideas, let us assume that we are given, in addition to problem (1), a target tolerance $\epsilon > 0$, an initial tolerance $\epsilon_0 \geq \epsilon$, and a minimum interval size $\delta > 0$. We assume that the tolerances are in relative terms, this is we aim at obtaining a feasible solution $\mathtt{x}$ of (1) whose objective value $z(\mathtt{x}) = \sum_{i=1}^{n} f_i(\mathtt{x}_i)$ is such that $z(\mathtt{x}) - z^* \leq \epsilon |z(\mathtt{x})|$, where z^* is the optimal value of problem (1).

We define a linear piece as a tuple $l = (g, \alpha, \beta, x_0, x_f, r)$ with $x_0 \leq x_f$ representing the linear function $\alpha x + \beta$ that approximates function g within a tolerance of r in the interval $[x_0, x_f]$. The linear piece is said to be a *lower bounding approximation of* g if $\alpha x + \beta \leq g(x)$ for every $x \in [x_0, x_f]$. A piecewise linear approximation of a function g in an interval $[u_0, u_f]$ is a set $\{(g, \alpha^k, \beta^k, x_0^k, x_f^k, r^k) : k = 1 \dots \kappa\}$ of linear pieces such that $x_0^1 = u, x_f^\kappa = v, x_0^i = x_f^{i-1}$ for every $i = 2 \dots \kappa$.

Our method starts by finding an initial piecewise lower bounding approximation for each function f_i covering the domain of variable x_i (that we assume to be bounded) for the initial tolerance ϵ_0. To that end, we use the method described in Codsi et al. (2021) that provides an optimal (in terms of its size) linear bounding approximation for any given tolerance. Please note that their method constructs discontinuous lower bounding linear approximations that do not necessarily interpolate the nonlinear functions (indeed, they almost never do). For a given linear piece l, we denote $\epsilon(l)$ the tolerance associated with that piece, that initially takes the value ϵ_0 uniformly.

We then modify problem (1) by replacing the functions $f_i, i = 1 \dots n$ in the objective by their respective linear approximations. This results in a modified MIP that approximates problem (1) by providing a solution that is feasible w.r.t. the set of constraints $\{Ax = b, x \in \mathcal{X} \subseteq \mathbb{R}_n^+\}$ and whose value (a lower bound) is off the optimal by at most ϵ_0. Note that there is not an unique way of writing the modified MIP. In Vielma et al. (2010) the authors review the literature in piecewise linear approximations. In particular, we use the so-called logarithmic representation that bounds the number of binary variables of the modified MIP to a maximum of $\sum_{i=1}^{n} \lceil \log(\kappa_i) \rceil$. Note however that this is not a condition for our method to work, as it will work as with other MIP representations as well.

The above method sets up a starting point for our iterative mechanism and the solution of the resulting modified MIP will provide a solution $\mathtt{x}^*$ that is feasible w.r.t. the set of constraints of the problem. Please recall that our objective is to approximate problem (1) within a tolerance of $\epsilon < \epsilon_0$. Moreover, because the point found $\mathtt{x}^*$ is feasible w.r.t. the set of constraints, evaluating $\mathtt{x}^*$ on the functions $f_i, i = 1 \dots n$ provides a valid upper bound for the problem. If we are lucky enough, the optimal value of the modified MIP (that provides a lower bound of the problem) will be off by at most ϵ from the upper bound. In that case, the method ends and returns $\mathtt{x}^*$.

In the case where the lower and upper bounds are off by more than ϵ in relative terms, a repair procedure is invoked with the objective of tightening the current linear lower bounding approximation. Let $\kappa \geq 1$ denote the number of times that the modified MIP has been solved. Let $\mathtt{x}_i^*$ be the i-th component of the solution $\mathtt{x}^*$ to the modified MIP. Let $\Delta_i^{\mathtt{x}^*}$ be the set of indices to the linear pieces containing the point $\mathtt{x}_i^*$. Let $\Delta_i \supseteq \Delta_i^{\mathtt{x}^*}$ be a set of indices to contiguous linear pieces and covering an interval $[l(\Delta_i), u(\Delta_i)]$ of size at least δ. If $f_i(\mathtt{x}_i^*) - \alpha_i^k \mathtt{x}_i^* - \beta_i^k \leq \epsilon |f_i(\mathtt{x}_i^*)|$ for every $k \in \Delta_i^{\mathtt{x}^*}$, then the approximation of f_i at the point $\mathtt{x}_i^*$ is sufficiently tight. If not, however, it needs to be tightened. We

propose two methods to tighten the linear pieces in Δ_i of the approximation of f_i. They are referred to as the *conservative* and the *aggressive tightening*, and differ in the speed at which convergence is achieved. In each case, the new tolerance for the pieces in Δ_i is set according to one of the two equations below:

Conservative tightening

$$\epsilon' \leftarrow \frac{1}{2} \min \left\{ \epsilon(l^k) : k \in \Delta_i \right\}. \tag{2}$$

Aggressive tightening

$$\epsilon' \leftarrow \frac{\epsilon_0}{2^\kappa}. \tag{3}$$

By finding a lower bounding linear approximation for f_i using ϵ' as a tolerance within the interval $[l(\Delta_i), u(\Delta_i)]$ we are indeed tightening the approximation of problem (1). This procedure will result in replacing a few linear pieces by potentially multiple other pieces. However, the linear pieces surrounding those indexed by Δ_i will remain unchanged, and the scope of the update procedure will remain local.

Once the repair procedure has been applied to every function f_i, the resulting MIP is solved again and the procedure repeated. The process ends when the lower and upper bounds are within a tolerance of ϵ. The following proposition provides a worst-case guarantee of convergence of the method to the desired precision.

Proposition 1. *If the domain of each variable x_i is bounded within the interval $[l_i, u_i]$ with $l_i \leq u_i$, the algorithm ends in at most*

$$N = \left\lceil \log_2 \left(\frac{\epsilon_0}{\epsilon} \right) \right\rceil \sum_{i=1}^{n} \left\lceil \frac{u_i - l_i}{\delta} \right\rceil \tag{4}$$

iterations and provides a solution $\mathtt{x}^*$ that if far from the optimal by at most ϵ.

Proof. At every iteration, the method either finds a solution $\mathtt{x}^*$ that is at most ϵ off the optimal, or detects one set Δ_i of contiguous linear pieces of size $\geq \delta$ including all pieces containing $\mathtt{x}_i^*$ to apply the refinement. When applying equation (2) or (3), the tolerance associated with the pieces in Δ_i in the new lower linear approximation will be cut of at least half. The number of times that this can happen before a linear piece within a given region is tightened two consecutive times is bounded above by $\lceil (u_i - l_i)/\delta \rceil$, and the number of times that this can happen before reaching the desired tolerance is bounded above by $\lceil \log_2(\epsilon_0/\epsilon) \rceil$. Because at each iteration there is at least one tightening for at least one variable $\mathtt{x}_i^*$, the sum of these quantities along the n dimensions of the domain is an upper bound for the total number of iterations before reaching global convergence. □

It is easy to see that the ϵ' values constructed by applying the aggressive tightening rule are smaller than or equal to those that can be achieved if one applies instead the conservative tightening rule. One can expect that albeit being equally fast in the worst case, the former shall provide quicker convergence in practice. We finally provide a high-level description of our algorithm in the pseudo-code described in Algorithm 1.

We will now illustrate our method by means of a very simple example. Let us consider the univariate function $f(x) = 300 + x^2 - 30x + 25\sin(x)$ depicted in Figure 1a whose minimum in the interval $[3\pi, 7\pi]$ is the point $\mathtt{x} = 17.11$ of value $f(\mathtt{x}) = 54.81$. In Figure 1b we depict a piecewise linear lower bounding approximation of f in the same interval for a relative tolerance of 20%. When optimizing this approximation instead of f, we obtain as a solution the point $\mathtt{x}' = 17.89$. The approximation when evaluated in this point provides a lower bound of value 50.3. Let us assume that this difference of more than 4 units is unacceptable. Thus, the refinement procedure needs to be put in place. Note that the point $\mathtt{x}'$ is located at the edge of two pieces. Therefore, the two pieces containing it need to be considered for the refinement process. The refinement is then executed but

Algorithm 1: Iterative piecewise linear bounding

Input: Problem (1), tolerances $\epsilon, \epsilon_0, \delta$
Output: Feasible solution x^*, dual bound $z^l(x^*)$, primal bound $z(x^*)$ s.t. $z(x^*) - z^l(x^*) \leq \epsilon|z(x^*)|$

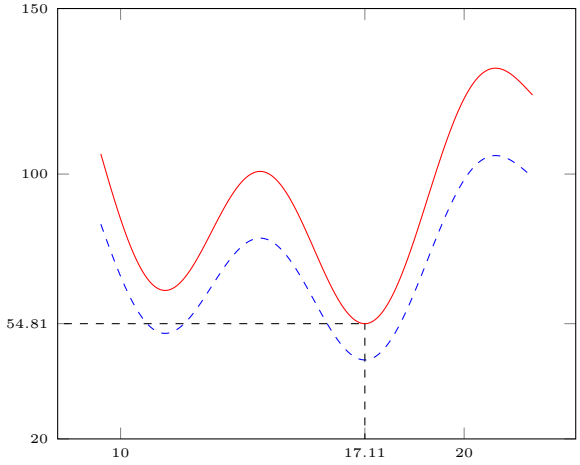
```
{f_i^l : i = 1...n} ← initial piecewise lower bounding approximation with tolerance ε_0
while true do
    Solve min_x{Σ_i f_i^l(x_i) : Ax = b, x ∈ X ⊆ R_n^+}, let x* be the optimal solution
    Let z^l(x*) ← Σ_i f_i^l(x_i^*), and z(x*) ← Σ_i f_i(x_i^*)
    if z(x*) − z^l(x*) < ε|z(x*)| then
        return (x*, z^l(x*), z(x*))
    else
        Refine {f_i^l : i = 1...n}
    end if
end while
```

subject to the interval $[15.63, 18.93]$. Figure 1c depicts the resulting tightening when the tolerance in this restricted interval is decreased to 10%. The optimal solution of the resulting MIP in this new approximation provides as solution $\mathtt{x}'' = 17.3$ and a dual bound of 49.77, lower than for the previous iteration, but associated to a solution now much closer from the actual optimum. If we perform a second tightening of this approximation, we see that the approximation needs to be tightened for the same interval, but now for a tolerance of 5%. This results in the approximation depicted in Figure 1d. The optimal solution associated with the resulting MIP will be the point $\mathtt{x}''' = 16.92$ with a dual bound associated of 52.49. This is now a much better approximation to the actual optimum value. In the process to reaching this dual bound, two tightenings from the original approximation were necessary, and only one additional linear piece was added along the process.

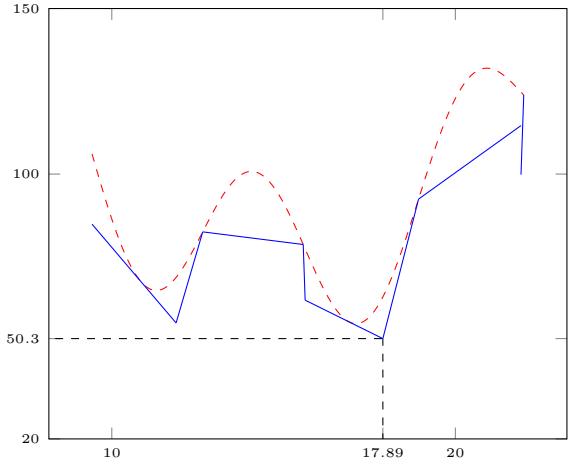
Several remarks are in order. First, as observed, the dual bounds are not necessarily monotone. This is due to the fact that a linear approximation may not be tighter in the whole interval after a refinement. Second, our algorithm works by detecting the promising zones for successive refinements, while ignoring the parts that obviously may never be part of an optimal solution. This is what contributes at keeping the MIPs tractable along the whole solution process. The irrelevant zones of the domain, which can be seen as *noise*, should be approximated as roughly as possible, to focus one's efforts into refining the zones that are most likely to contain an optimal solution of the problem. Third, our method does not require the solution of a NLP oracle to ensure convergence, as it relies purely on the solution of mixed-integer linear programs. Fourth, our method computes lower and upper bounds at each iteration. Fifth, the refinement procedure is based on the computation of the best piecewise linear underestimation on a given interval, instead of arbitrary split of the interval into two. Sixth, our method can handle relative or absolute tolerances. As shown by Ngueveu (2019), the latter may produce smaller MILPs for the same value of ϵ. Seventh, we do not require the piecewise linearizations to be continuous nor to interpolate the nonlinear functions, which can be efficiently exploited for instance by using the method of Codsi et al. (2021) to compute and tighten the approximations.

4 Applications

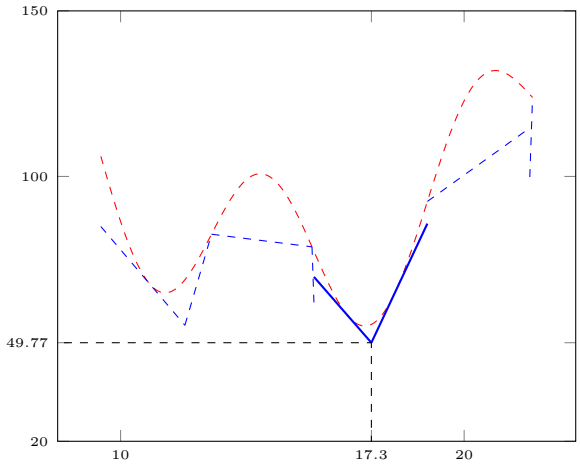
In this section we describe the application of our method to three classes of non-linear optimization problems, namely: the transportation problem (TP), the capacitated facility location problem (CFLP), and the multi-commodity network design problem (MCNDP). First we present examples of three nonlinear functions that we will use to assess the performance of our method. Next, we describe the experimental setup used throughout our campaign. Then, we present nonlinear variants of these optimization problems and provide computational evidence of the performance of our approach to reach near-optimal solutions with very tight numerical guarantees.



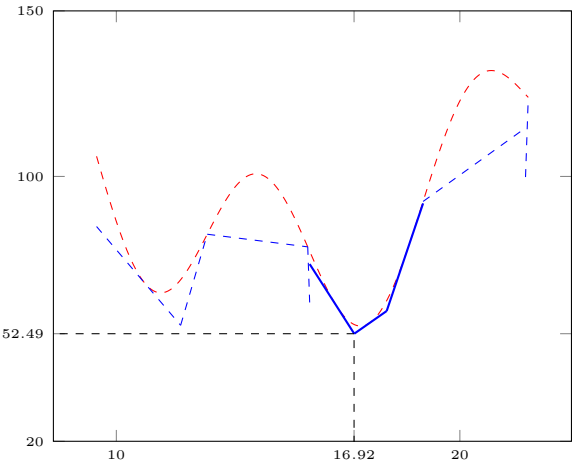
(a) $f(x) = 300 + x^2 - 30x + 25\sin(x)$ in the interval $[3\pi, 7\pi]$



(b) piecewise linear lower bounding approximation of f in the interval $[3\pi, 7\pi]$ for $\epsilon_0 = 20\%$



(c) First tightening of the piecewise linear lower bounding approximation of f



(d) Second tightening of the piecewise linear lower bounding approximation of f

Figure 1: Approximating $\min\{f(x) = 300 + x^2 - 30x + 25sin(x) : 3\pi \leq x \leq 7\pi\}$

4.1 Nonlinear cost functions

Figure 2 depicts three classes of nonlinear functions that we consider across the three classes of optimization problems studied in this manuscript. Figure 2a represents a situation in which economies of scale are observed from the beginning until a certain capacity is attained, moment at which the cost starts increasing at a much faster pace. Figure 2b, on the other hand, depicts a situation in which an exponential growth in the cost is observed at the beginning, until reaching a point where economies of scale start playing a role to keep the costs bounded. Figure 2c depicts a situation in which congestion is observed since the beginning, but where the activation of some additional resources helps mitigate the congestion effect past a certain threshold. Note the three isolated dots in the cost functions, representing a cost of 0 when the associated variable takes a value of zero, and a strictly positive and increasing cost as soon as this variable starts taking a positive value. For real-valued parameters $a_i, i = 1 \dots 8$ the three cost functions can be written as follows:

$$f^1(x) = \begin{cases} a_1 x^3 + a_2 x^2 + a_3 x + a_4 & x > 0 \\ 0 & x = 0 \end{cases} \qquad \text{(concave then convex)}$$

$$f^2(x) = \begin{cases} a_1 + \dfrac{a_2}{1 + a_3 \mathrm{e}^{a_4(x - a_5)}} & x > 0 \\ 0 & x = 0 \end{cases} \qquad \text{(convex then concave)}$$

$$f^3(x) = \begin{cases} \min\left\{ a_1 + a_2 \left(x + \frac{x^{a_3}}{a_4} \right), a_5 + a_6 \left(x + \frac{x^{a_7}}{a_8} \right) \right\} & x > 0 \\ 0 & x = 0 \end{cases} \qquad \text{(min of two convex)}$$

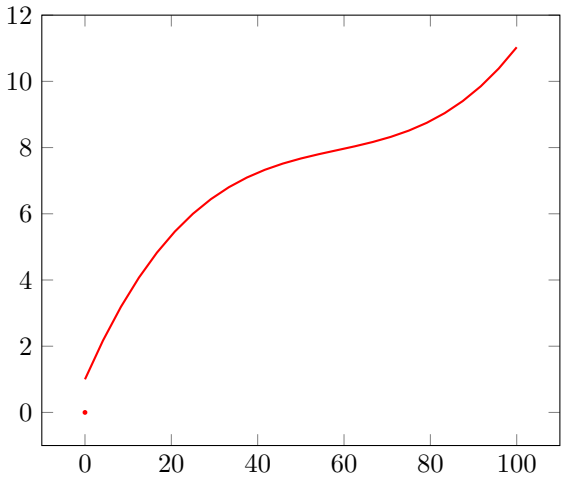
To calibrate the parameters $a_i, i = 1 \dots 8$ for each of the cost functions we proceed as follows. Let y_i^{max} be the maximum nominal value for a variable x_i, and let $[0, x_i^{max}]$ be its domain. For a function of the type f^1, the parameters a_1, a_2, a_3, a_4 are chosen so the function passes through the coordinates (x, y): $(0, \frac{1}{2} y_i^{max}), (\frac{1}{3} x_i^{max}, \frac{3}{4} y_i^{max}), (\frac{2}{3} x_i^{max}, \frac{7}{8} y_i^{max}), (x_i^{max}, 2 y_i^{max})$. For a function of the type f^2, the parameters $a_i, i = 1 \dots 5$ are as follows: $a_1 = \frac{1}{2} y_i^{max}, a_2 = \frac{3}{2} y_i^{max}, a_3 = \frac{1}{2} x_i^{max}, a_4 = \frac{10}{x_i^{max}}, a_5 = -\frac{x_i^{max}}{10}$

4.2 Experimental setup

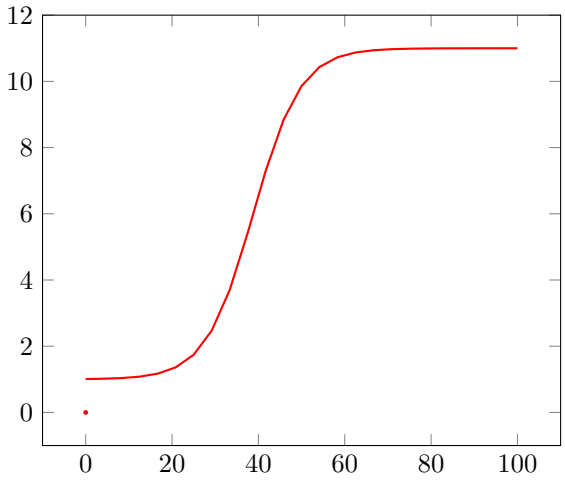
To approximate the cost functions, we consider the piecewise linear lower bounding approximation introduced in Codsi et al. (2021) that, given a relative tolerance ϵ, computes a set of contiguous —not necessarily continuous at the breakpoints— linear pieces approximating each function $f_i(\cdot)$. Their procedure possesses several characteristics that makes it a good choice for our purpose: 1) it relies on a greedy mechanism that runs extremely quick; 2) it can handle absolute and relative tolerances; and 3) it generates a number of linear pieces whose size is provably minimum provided that the linear pieces are not required to meet at the breakpoints. We then use the MIP representation of Vielma et al. (2010) to model the approximated problem using a logarithmic number (in terms of the number of linear pieces of the approximation) of additional binary variables.

We consider varying values of ϵ, namely $\epsilon \in \{10^{-2}, 10^{-3}, 10^{-4}\}$ and a fixed initial tolerance $\epsilon_0 = 10^{-1}$. We consider three variants of our method, namely one using the conservative tightening rule, another using the aggressive tightening rule, and a third one where we set $\epsilon_0 = \epsilon$. We refer to the first two as the *iterative variants* and the latter as the `NAIVE` variant. In addition, we consider five other mixed-integer nonlinear solvers for benchmarking purposes. A summary description of the algorithms used in our benchmarking can be found in Table 1.

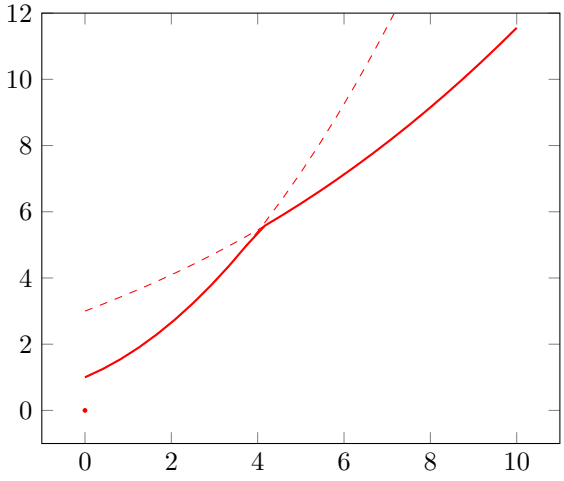
Our method has been coded in Julia v1.5 using the JuMP 0.21 interface, and uses CPLEX v20.1 as general-purpose solver for the three variants of our method (`CN2021-ct`, `CN2021-at` and `NAIVE`). These three variants have been executed on an Intel(R) Xeon(R) CPU E5-2637 v2 @ 3.50GHz with 128 GB of RAM. While the machine is capable of running code in parallel, for reproducibility purposes we limit the number of threads to one on all settings. For the five other solvers, we have submitted



(a) Cost function of the type f^1



(b) Cost function of the type f^2



(c) Cost function of the type f^3

Figure 2: Examples of three types of cost functions

Algorithm	Description	Reference
`CN2021-at`	Our method with the aggressive tightening rule	This paper
`CN2021-ct`	Our method with the conservative tightening rule	This paper
`NAIVE`	Our method with $\epsilon_0 = \epsilon$	This paper
`SCIP`	Direct solution of the MINLP using SCIP	Gamrath et al. (2020)
`BARON`	Direct solution of the MINLP using BARON	Sahinidis (2017)
`ANTIGONE`	Direct solution of the MINLP using ANTIGONE	Misener and Floudas (2014)
`COUENNE`	Direct solution of the MINLP using COUENNE	Belotti et al. (2009)
`LINDOGL`	Direct solution of the MINLP using LINDO Global	Lin and Schrage (2009)

Table 1: Eight solvers for benchmarking purposes

them to the NEOS Server (Czyzyk et al., 1998; Dolan, 2001; Gropp and Moré, 1997), which runs on multiple (different) machines powered with Intel CPUs of frequencies ranging between 2.2 and 2.8 GHz in a multicore, hyper-threading environment. In Table 2 we provide the PassMark scores of each of the machines involved in our experiments as well as their physical locations. We give a maximum time of 3,600 seconds (one hour) before a problem times out and is deemed as unsolved.

CPU	PassMark score	Location
Intel Xeon E5-2637 v2 @ 3.50G	6666	GERAD
Intel Xeon E5-2430 @ 2.20G	5867	NEOS Server
Intel Xeon X5660 @ 2.80GHz	6254	NEOS Server
Intel Xeon E5-2698 v3 @ 2.30GHz	19552	NEOS Server

Table 2: Description of the machines

4.3 Transportation problem

In the transportation problem (TP, Ford Jr and Fulkerson (1956)), we are given a set U of n origins, and a set V of m destinations. With each origin $u \in U$ we associate an offer $o_u > 0$ of a given commodity, and with each destination $v \in V$ a demand $d_v > 0$ of the same commodity and such that $\sum_{u\in U} o_u = \sum_{v\in V} d_v$. For the sake of simplicity, we assume that the offers and demands are all integer-valued. For each pair $(u, v) \in U \times V$ we are given a cost function $f_{uv} : \mathbb{R} \to \mathbb{R}$ such that $f_{uv}(x)$ represents the cost of transporting x units of flow from u to v. The objective is to select the amounts $(x_{uv})_{u\in U,v\in V}$ to transport along the arcs $(u, v) \in U \times V$ such that: 1) each origin $u \in U$ sends exactly o_u units of flow; 2) each destination $v \in V$ receives exactly d_v units of flow; and 3) the total cost $z = \sum_{(u,v)\in U\times V} f_{uv}(x_{uv})$ is minimized.

If the cost functions are all linear, the TP described above resorts to a classical linear TP, which can be solved in polynomial time using a network flow algorithm (Kleinschmidt and Schannath, 1995). In this article, we are interested in the scenario where the cost functions are separable but otherwise might take arbitrary forms (for instance non-convex). Assuming that the cost functions are non-decreasing, the problem can be modeled as a non-linear optimization problem using the notation already introduced as follows:

$$\min_{x} \qquad z = \sum_{u\in U, v\in V} f_{uv}(x_{uv}) \tag{5}$$

subject to

$$\sum_{v\in V} x_{uv} \le o_u \qquad u \in U \tag{6}$$

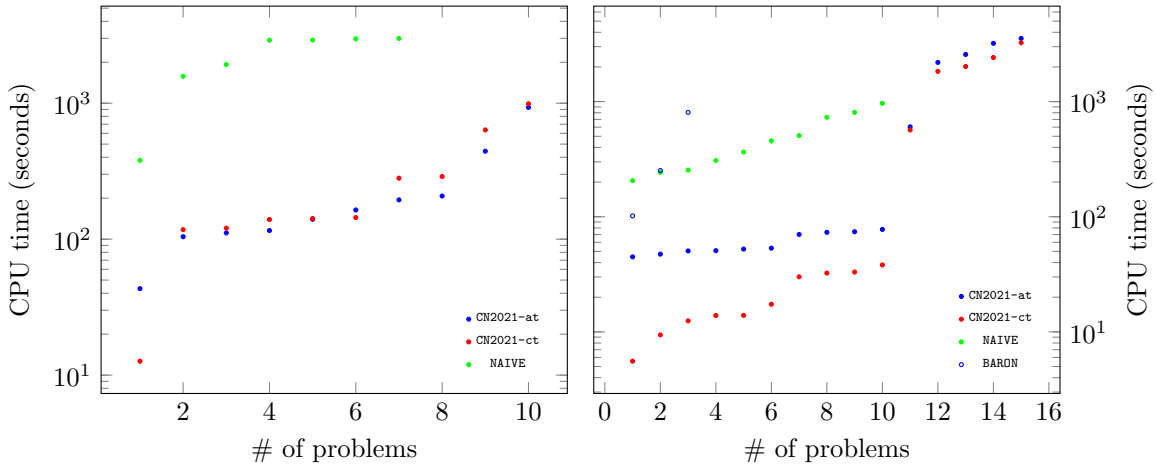
$$\sum_{u\in U} x_{uv} \ge d_v \qquad v \in V \tag{7}$$

$$x_{uv} \ge 0 \qquad u \in U, v \in V. \tag{8}$$

To assess the effectiveness of our approach to approximate the nonlinear TP, we have generated a set of random instances, as follows. We consider squared problems with $n = m \in \{5, 10\}$ origins and destinations in an Euclidean space on a square of dimensions 100×100. A nominal unit cost c_{uv} for an arc (u, v) is computed as the Euclidean distance between the points plus a random noise added from an uniform distribution in the interval $[-3, 3]$. The domain of variable x_{uv} is the interval $[0, \min\{o_u, d_v\}]$. We consider the non-convex cost functions f^1, f^2 as described in Section 4.1. Ten instances are generated for each value of $n = m \in \{5, 10\}$ for a total of 20 problems.

We present summarized results in Figure 3. In this figure, we plot CPU time profiles for each of the eight solvers considered in our study, restricted to the finest value of ϵ, namely for $\epsilon = 10^{-4}$.

The plots represent the number of problems solved to optimality (x-axis) within a certain time in seconds (y-axis). As these results show, the iterative variants of our method outperform all other solvers considered in this study. As a matter of fact, of the five general-purpose MINLP solvers considered in this study, only BARON was successful at solving a few problems. All other solvers timed out systematically. We can also note that the shape of the objective seems to play a role in the difficulty of a problem. In the specific case of the TP, only half of the problems were solved within the desired precision for functions of the type f^1, but 80% of them were solved for cost functions of the type f^2. When comparing CN2021-at and CN2021-ct, we see that the former performs slightly better for the cost functions of type f^1, and the opposite happens for the cost functions of the type f^2.



(a) functions of type f^1

(b) functions of type f^2

Figure 3: CPU time profiles (in seconds) for the TP

4.4 Capacitated facility location

The capacitated facility location problem (CFLP, Sridharan (1995)) deals with the problem of deciding the location of one or more facilities among an universe U of n total facilities, and to assign m customers in a set V to the selected facilities. We associate an opening cost $f_u > 0$ and a capacity κ_u to each facility $u \in U$, and an assignment cost $c_{uv} > 0$ to each assignment of a customer v to a facility u. Each customer $v \in V$ has a demand of $d_v > 0$ units. We seek to determine: 1) what facilities to open; 2) what fraction of the demand of a customer must be assigned to every open facility; 3) while respecting the facilities' capacities; 4) at minimum total cost.

The non-linearities in this problem may come from two sources:

Nonlinear warehousing costs by replacing the opening cost f_u of the facilities by a nonlinear term $g_u(\cdot)$ representing the warehousing cost associated with the service of the demand fulfilled by facility u.

Nonlinear assignment costs by replacing each assignment cost c_{uv} by a nonlinear assignment function $h_{uv}(\cdot)$ such that $h_{uv}(s)$ represents the cost associated with servicing a fraction $s \in [0, 1]$ of the demand of customer v by facility u.

Let us introduce a model that includes both settings at once. Let us consider binary variables y_u for every $u \in U$, to model the choice of opening or not a given facility. Let s_u be a continuous variable

representing the amount of demand serviced by facility u. For each possible assignment (u, v) of a customer v to a facility u, let x_{uv} be the fraction of the demand of v that is serviced by facility u. Using the notation already introduced —and by assuming that the cost functions g, h are non-decreasing— the following model solves the CFLP while minimizing the total nonlinear costs:

$$\min_{x,y} \quad z = \sum_{u \in U} g_u(s_u) + \sum_{u \in U, v \in V} h_{uv}(x_{uv}) \tag{9}$$

subject to

$$\sum_{u \in U} x_{uv} \geq 1 \qquad v \in V \tag{10}$$

$$\sum_{v \in V} d_v x_{uv} \leq s_u \qquad u \in U \tag{11}$$

$$s_u \leq \kappa_u y_u \qquad u \in U \tag{12}$$

$$0 \leq x_{uv} \leq y_u \qquad u \in U, v \in V \tag{13}$$

$$y_u \in \{0, 1\} \qquad u \in U. \tag{14}$$

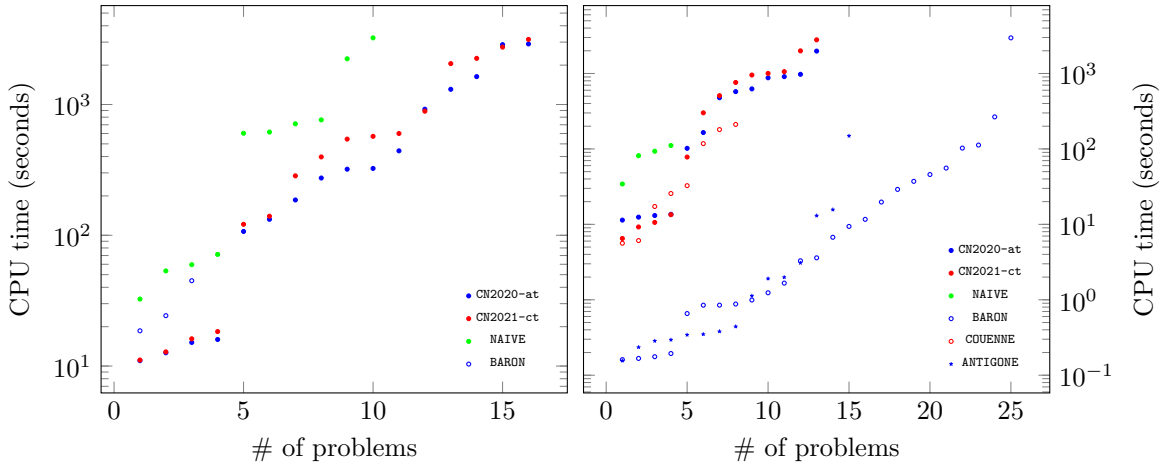
We consider the two possible scenarios for the source of the non-linearities separately: a first scenario with nonlinear warehousing costs, but with linear assignment costs; and a second scenario with nonlinear assignment costs, but with constant fixed costs. In both cases, we consider two classes of nonlinear cost functions: functions of the type f^1, and functions of the type f^2 as described in Section 4.1.

We consider the ORLib instances from the CFLP literature introduced by Beasley (1990). This benchmark dataset contains 40 problems with a number of faclities and customers ranging between $[16, 100]$ and $[50, 1000]$, respectively. We restrict our analysis to the 37 smaller instances, thus we ignore the three largest problems that are too difficult for all the algorithms considered in this study and from which no relevant conclusions can be derived. Note also that we have slightly modified the instances, as follows. The nominal fixed costs of some of the facilities are equal to zero. Our implementation assumes that the nonlinear cost functions only take the value zero at the origin, and a value $\geq \Delta > 0$ in the rest of the domain, with Δ being a certain real parameter. Therefore we set the nominal fixed costs to be equal to 1 for those facilities. We then construct nonlinear variants of the cost functions on the modified costs following the recipes described in Section 4.1. Although this is only relevant in the context of nonlinear warehousing costs, we perform the same modification for the problem variant with nonlinear assignment costs.

4.4.1 Nonlinear warehousing costs

In this section we present summary results of the performance of the different solvers for approximating the CFLP with nonlinear warehousing costs, this is when every fixed cot f_u is replaced by a nonlinear cost function $g_u(\cdot)$.

In Figure 4 we plot the time profiles for the solvers that succeeded at solving at least one problem before timing up. The graphics plot the number of instances successfully approximated to the desired tolerance (`x`-axis) within a certain time (`y`-axis). We restrict our analysis here to a precision of $\epsilon = 10^{-4}$. The results show a high sensitivity to the shape of the cost function. For the cost functions of the form f^1, the iterative variants outperform all other solvers. In fact, from the five general-purpose MINLP solvers considered in this study, only `BARON` provided meaningful results, but still far from being competitive against the iterative variants of our method, which scaled to solve more problems, and faster. The opposite behavior can be observed when one looks at the results for the cost functions of the form f^2. In this case, `BARON` takes the lead by a significant margin, with `ANTIGONE` being the only other solver that seems competitive against it, but only on the easy problems.



(a) functions of type f^1

(b) functions of type f^2

Figure 4: CPU time profiles (in seconds) for the CFLP with nonlinear warehousing costs

4.4.2 Nonlinear assignment costs

We now consider the problem of replacing every assignment cost c_{uv} by a nonlinear function $h_{uv}(\cdot)$ using functions of the type f^2 as described in Section 4.1. Please note that given the much larger number of linear pieces required to achieve reasonable approximations, none of the solvers were successful at handling this problem for the functions of the type f^1. All of them timed out. For that reason, we restrict our analysis for the cost functions of the type f^2.

In Figure 5 we plot the time profiles of all meaningful solvers (meaning that we omit those for which all problems timed out). As before, we plot the number of instances successfully approximated to the desired tolerance (`x`-axis) within a certain time (`y`-axis). We restrict our analysis here to a tolerance of $\epsilon = 10^{-4}$. Of the five general purpose MINLP solvers, four of them time out systematically on all problems. The other, `BARON`, is competitive against the iterative variants of our solver on the easier problems but scales worse on the harder problems. We also see that the iterative variants of our method solve the same number of problems as the `NAIVE` variant, only much faster. No significant differences are perceived between `CN2021-at` and `CN2021-ct`.

4.5 Multi-commodity network design

The congested multicommodity network design problem (cMCNDP, Parakevopoulos et al (2016)) deals with the problem of dispatching multiple commodities throughout a configurable network taking into account explicitly the congestion occurring at transportation nodes. We are given a set of nodes N, a set of arcs A and a set of commodities P. Each node $i \in N$ has a maximal capacity Q_i, a demand of commodity p denoted D_i^p, and a nonlinear cost function f_i. Each arc $(i,j) \in A$ has a fixed opening cost O_{ij} and a capacity U_{ij}. Each commodity $p \in P$ has a quantity to be shipped W_p and a unit routing cost over arc (i,j) denoted R_{ij}^p. Our method is applied on the formulation proposed by Codsi et al. (2021). Let us consider a binary variable y_{ij} for every $(i,j) \in A$ to represent the choice of using the arc or not. Let continuous variables $x_{ij}^p \geq 0$ for every $p \in P$ and $(i,j) \in A$ be the flow of every commodity on every arc. Let us consider a variable v_i defined as follows: $v_i = \sum_{j \in N^-} \sum_{p \in P} x_{ij}^p, \forall i \in N$. Using these notations, the multicommodity network design with congestion problem can be formulated as follows, minimizing the total design cost, usage costs and a nonlinear function modeling the sum of

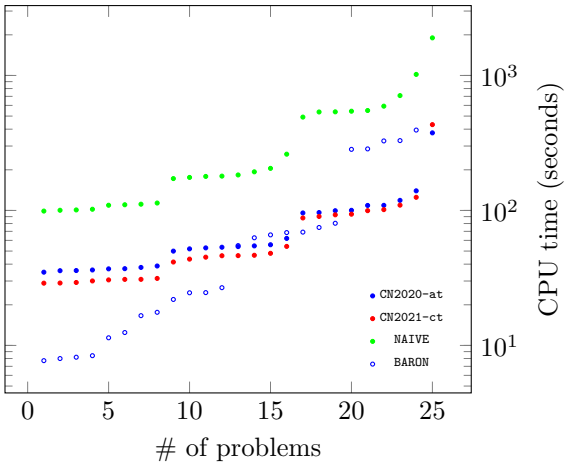


Figure 5: CPU time profiles (in seconds) for the CFLP with nonlinear assignment costs

the congestion cost and capacity upgrade cost.

$$\min \sum_{(i,j)\in A} O_{ij} y_{ij} + \sum_{(i,j)\in A} \sum_{p\in P} R_{ij}^p x_{ij}^p + \sum_{i\in N} f_i(v_i) \tag{15}$$

subject to

$$\sum_{j\in N_i^+} x_{ij}^p - \sum_{j\in N_i^-} x_{ji}^p = D_i^p, \qquad i \in N, p \in P \tag{16}$$

$$x_{ij}^p \leq W^p y_{ij}, \qquad (i,j) \in A, p \in P \tag{17}$$

$$\sum_{p\in P} x_{ij}^p \leq U_{ij} y_{ij}, \qquad (i,j) \in A \tag{18}$$

$$\sum_{j\in N^-} \sum_{p\in P} x_{ji}^p = v_i, \qquad i \in N \tag{19}$$

$$x_{ij}^p \geq 0, \qquad (i,j) \in A, p \in P \tag{20}$$

$$y_{ij} \in \{0,1\}, \qquad (i,j) \in A \tag{21}$$

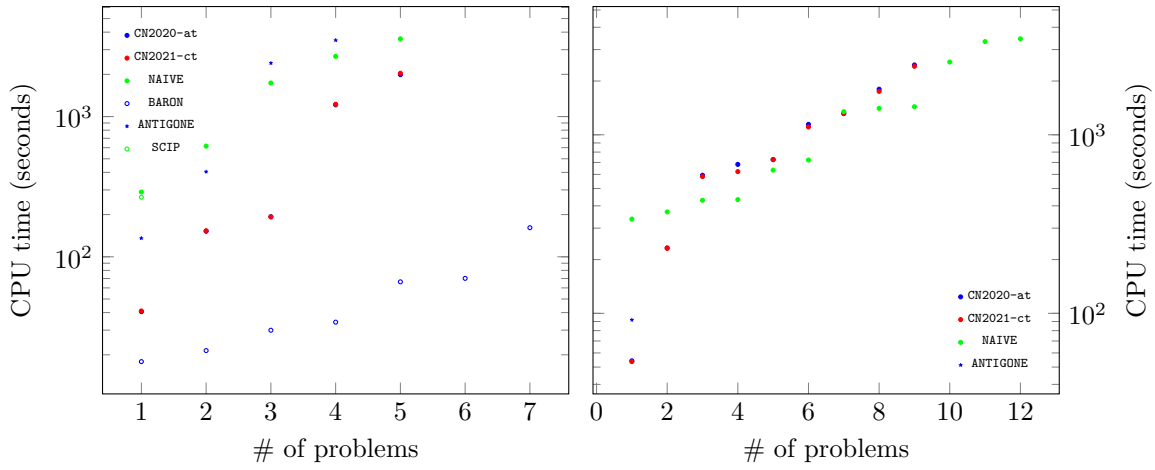
$$v_i \in [0, Q_i], \qquad i \in N. \tag{22}$$

We consider the three types of possible cost functions as described in Section 4.1. Our experimental campaign considers some classical datasets from the MCND literature, namely the so-called C and C+ instances Crainic et al. (2001). These instances contain between 10 and 30 nodes and between 10 and 200 commodities. A preliminary computational campaign has revealed that the number of commodities plays a key role in the performance of our method. To highlight this behavior, we also consider modified versions of some instances. Namely, for every instance containing strictly more than 10 commodites, we consider an additional modified version of it restricted to the first 10 commodities, hence discarding the others.

4.5.1 Results on the original instances

We first present our results for the original —unmodified— instances, but separately for the problems with 10, and $\geq$ 30 commodities, to help visualize the effect of the number of commodities in the performance of our method. We restrict our analysis to a tolerance of $\epsilon = 10^{-4}$. We report aggregate results in the plots of Figures 6-8 separately for each type of cost function.

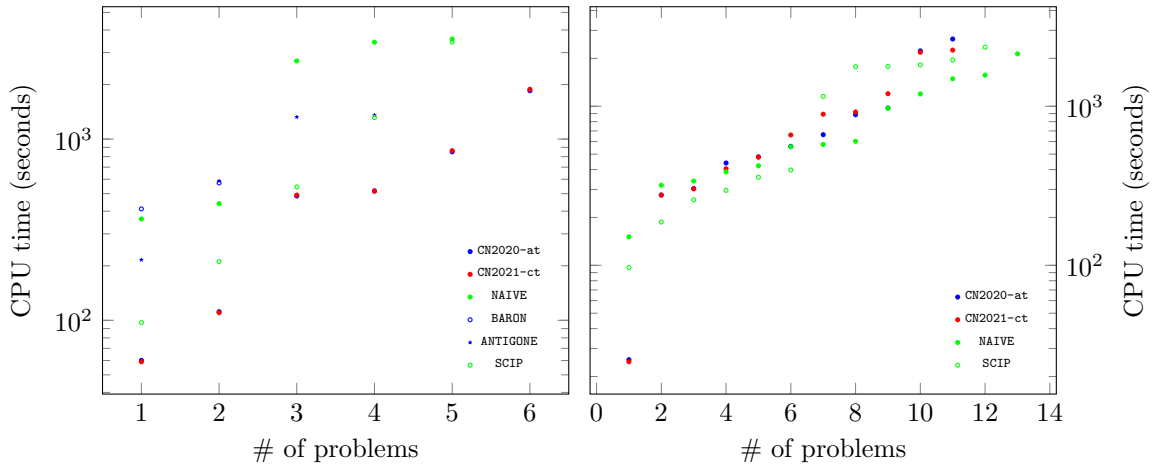
Results for functions of the type f^1 We observe that for problems with a low number of commodities (10), only `BARON` is competitive against our method, being often faster and solves a larger number of problems. The iterative variants of our method seem to perform slightly better than the `NAIVE` variant in this case. For the problems with ≥ 30 commodities, `BARON` stops being competitive and in this case it is the `NAIVE` solver that takes the lead.



(a) 10 commodities (b) ≥ 30 commodities

Figure 6: CPU time profiles (in seconds) for the MCND for functions of te type f^1

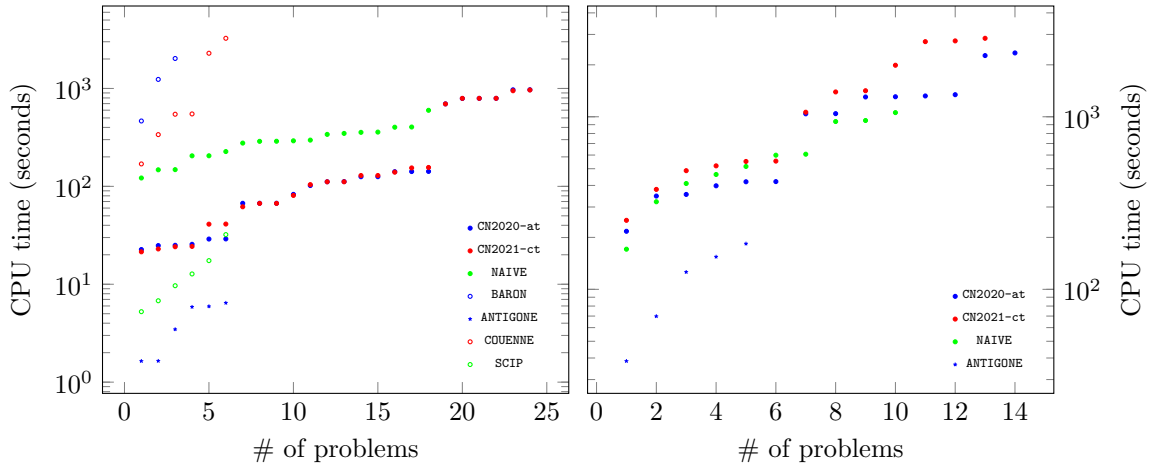
Results for functions of the type f^2 We observe now that for the instances containing 10 commodities, the iterative variants of our method perform best. For the problems with 30 or more commodities, we observe no significant differences between `SCIP` and the three variants of our method.



(a) 10 commodities (b) ≥ 30 commodities

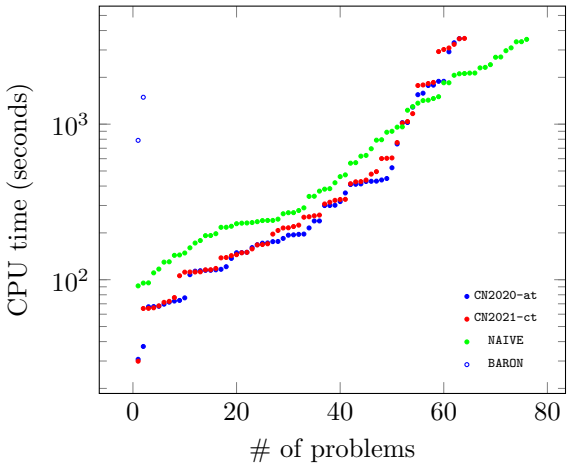
Figure 7: CPU time profiles (in seconds) for the MCND for functions of te type f^2

Results for functions of the type f^3 Given the larger number of problems solved for this cost function, we disaggregate our analysis for the problems with: a) 10 commodities; b) 30 commodities; and c) ≥ 40 commodities. We observe that for the problems with 10 commodities, it is the iterative variants of our method that succeed at solving a larger number of problems, being always faster for the problems that require 30 seconds to solve or more. For the problems with 30 commodities, it is `ANTIGONE` that seems fastest on the easier problems, but is dominated by the iterative variants of our method on the more difficult problems. When looking at the problems with 40 or more commodities, it is the `NAIVE` variant that takes the lead, being faster and more scalable than all other seven solvers. We remark, however, that none of the five general-purpose MINLP solvers seem useful at all at handling these problems, contrarily to the behavior observed for the iterative variants of our method that succeed at solving a non trivial number of problems.



(a) 10 commodities

(b) 30 commodities

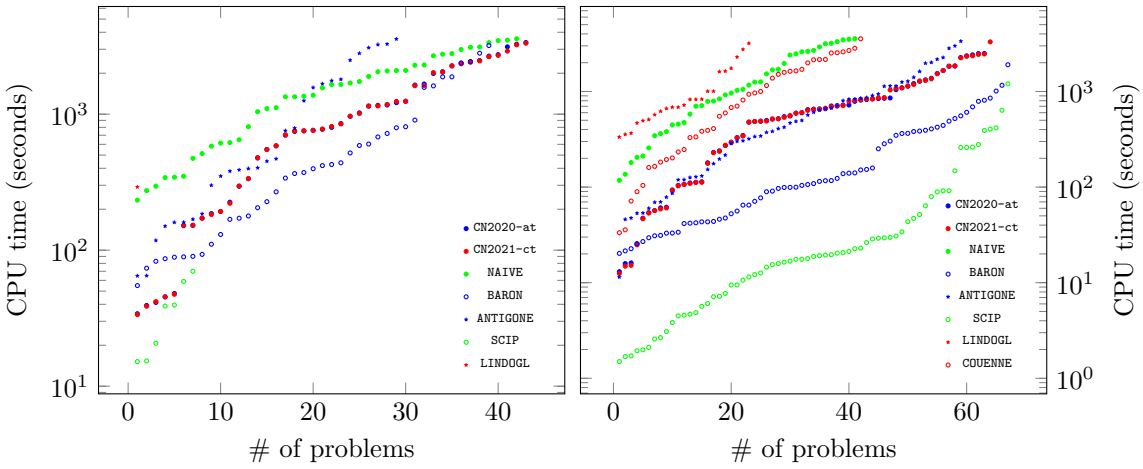


(c) ≥ 40 commodities

Figure 8: CPU time profiles (in seconds) for the MCND for functions of the type f^3

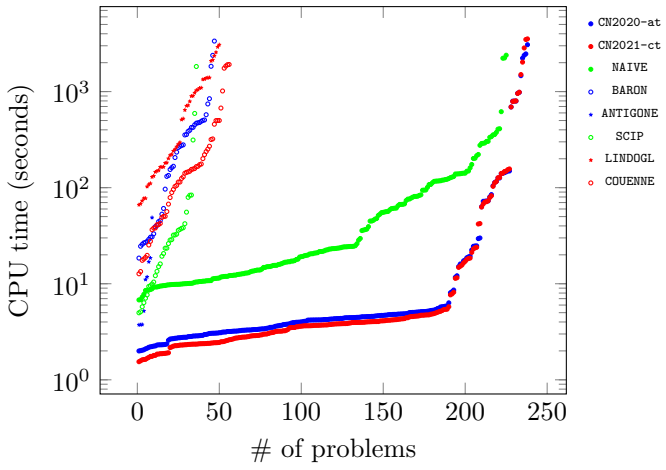
4.5.2 Results on the modified instances with ≤ 10 commodities

In Figure 9 we plot CPU profiles for the three variants of our method, restricted to a tolerance of $\epsilon = 10^{-4}$ for the modified instances with ≤ 10 commodities, and for all three classes of cost functions. It appears that the shape of the cost functions plays an important role in the performance of our method. We see that, for cost functions of the form f^1, `BARON` seems the only one among the general-purpose MINLP solvers to be competitive against the three variants of our method. For the cost functions of the type f^2, it is now `SCIP` that takes the lead, with `BARON` and the two iterative variants of our method completing the podium. When one looks at the problems with cost functions of the type f^3, we observe that the iterative variants of our method take a significant edge over all other six solvers, being able to solve four times as many problems in the allocated time when compared to the general-purpose solvers (surprisingly, not as many more problems when compared to the `NAIVE` variant of the method, albeit generally much faster), and being orders of magnitude faster on the easier problems.



(a) Functions of the type f^1

(b) Functions of the type f^2

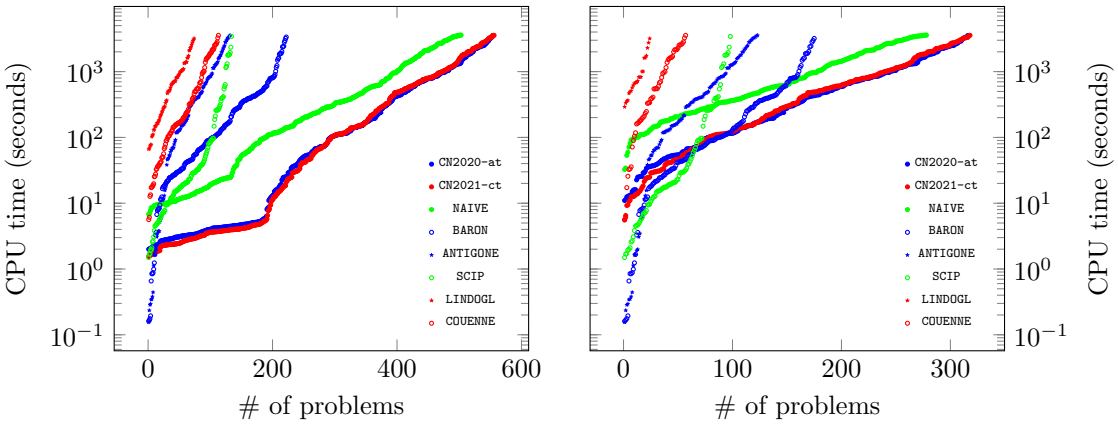


(c) Functions of the type f^3

Figure 9: CPU time profiles (in seconds) for the MCND problems with 10 commodities

5 Overall performance and limitations of the method

The detailed analysis that we provide in the previous section helps at assessing the performance of our method under a variety of different scenarios. We observe scenarios where the iterative variants of our method outperform al other six solvers (*e.g.* MCNDP with 10 commodities and functions of the type f^3), and cases where their performance is rather poor (*e.g.* CFLP with nonlinear warehousing costs and functions of type f^2). But what about the overall performance of our method?. How do `CN2021-at`, `CN2021-ct` compare against all other six solvers *overall*? To answer this question, we present two additional plots in Figure 10 with aggregate data for each of the eight solvers. In the left-most figure, we consider all classes of problems, with all cost functions, combined. In the right-most subplot we omit the results for the MCNDP with functions of the type f^3 on the modified instances with 10 commodities (these are the problems in which our method performs best). It is now clear that the three variants of our method outperform all other five solvers, as they succeed at solving about 3x more problems, about 100x faster overall, and this is still the case even when the problems on which our method performs best are ignored. Moreover, the two iterative variants of our method are about 2x faster on average than the `NAIVE` variant and succeed at solving more problems.



(a) All problems combined

(b) All problems except MCNDP modified with 10 commodities and cost functions f^3

Figure 10: Overall performance of the eight solvers

In Figure 11 we plot profile curves of the optimality gaps achieved by the different solvers after timing out. We ignore the results for the `NAIVE` variant as our logs do not report meaningful results when the method times out at the first iteration. We also ignore all problems for which a solver succeeded within the allocated time limit. The `y`-axis denotes the gap, and the `x`-axis the number of problems x for which the solver —after timing out— achieved a final gap of y or less. We observe that the solvers `CN2021-at` and `CN2021-ct` are typically capable of achieving much tighter gaps (about 100x lower) than the general-purpose MINLP solvers, which is also a nice feature of our method, as it provides tighter optimality guarantees even when unable to achieve the desired precision within the allocated time limit.

In Figure 12 we plot profile curves of the number of linear pieces required for each solver to achieve the desired precision. We restrict our analysis to the finest tolerance considered in this study, notably $\epsilon = 10^{-4}$. The `y`-axis denotes the total number of linear pieces required for the last MIP solved, and the `x`-axis the number of problems x for which the solver required at most y linear pieces to achieve the desired precision. We observe that the solvers `CN2021-at` and `CN2021-ct` behave almost identical, at such an extent that it becomes difficult to distinguish them in this plot. They rely on MIPs with between 100x and 1000x less linear pieces than the `NAIVE` variant of the method.

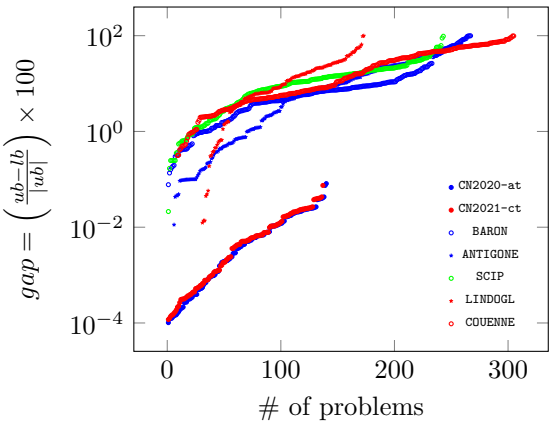


Figure 11: Profile curves of the optimality gaps on the unsolved problems

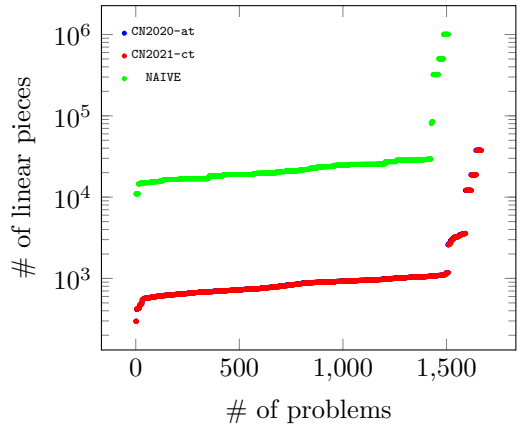


Figure 12: Profile curves of the number of linear pieces required to achieve a tolerance of $\epsilon = 10^{-4}$

Preliminary experiments have also revealed a key limitation of our method, namely its ineffectiveness to handle symmetries. When faced to problems with multiple optima in different regions of the solution space, the method will tend to oscillate between these regions before reaching convergence. In the presence of a combinatorial number of optima, our method will likely refine the neighboring region of each such solution in an alternate way. The conservative tightening rule makes things worse as not only a large number of iterations will be required before landing twice in the same region, but the piecewise linear approximations will only be tightened by a constant factor each time. Mitigating this undesired behavior was indeed the main motivation for introducing the aggressive tightening rule.

6 Concluding remarks

We have introduced an iterative method with optimality guarantees for a general class of separable mixed-integer nonlinear problems. Our method iterates between the solution of a mixed-integer linear problem to compute primal and dual bounds, and of a repair procedure to tighten the bounds if deemed necessary. Our method does not rely on NLP nor MINLP oracles to compute those bounds, hence relying exclusively on the efficiency to model and solve the resulting MILPs. We have proved that our method converges in a finite number of iterations under some very mild assumptions. We have assessed the effectiveness of our method on three optimization problems relevant in practice: the transportation problem, the capacitated facility location problem, and the multicommodity network design problem.

Our results show that our method is efficient at handling these problems, often outperforming state-of-the-art solvers by orders of magnitude. When unable to achieve the desired precision, our solver still shows a robust behavior as it achieves much tighter optimality gaps as when compared to state-of-the-art solvers. The experiments have also revealed some limitations of our method, that we properly report and discuss.

Future research shall focus on mitigating the nocive effects of symmetries as discussed in the previous section. Also, extending our framework to handle objectives separable in functions of two or more variables (as opposed to objectives separable in univariate functions) would provide a significant contribution to the scientific literature.

Acknowledgments

C. Contardo thanks the Natural Sciences and Engineering Research Council (NSERC) of Canada for its financial support, under Grant no 2020-06311. S. U. Ngueveu thanks the FMJH Program PGMO for the financial support also provided by EDF-Thales-Orange. The authors thank the University of Wisconsin, the Arizona State University, the University of Klagenfurt and the University of Minho for facilitating the infrastructure of the NEOS server.

References

D. Aloise and C. Contardo. A sampling-based exact algorithm for the solution of the minimax diameter clustering problem. *Journal of Global Optimization*, 71:613–630, 2018.

J. E. Beasley. Or-library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4–5):597–634, 2009.

R. Burlacu. On refinement strategies for solving MINLPs by piecewise linear relaxations: a generalized red refinement. *Optimization Letters*, 2021. doi: 10.1007/s11590-021-01740-1. URL https://doi.org/10.1007/s11590-021-01740-1.

R. Burlacu, B. Geißler, and L. Schewe. Solving mixed-integer nonlinear programmes using adaptively refined mixed-integer linear programmes. *Optimization Methods and Software*, 35(1):37–64, 2020. doi: 10.1080/10556788.2018.1556661. URL https://doi.org/10.1080/10556788.2018.1556661.

D. Chen and R. Chen. New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. *Computers and Operations Research*, 36(5):1646–1655, 2009.

J. Codsi, B. Gendron, and S. U. Ngueveu. Lina: A faster approach to piecewise linear approximations using corridors and it's application to mixed integer optimization. Technical report, Working paper, 2021.

C. Contardo. Decremental clustering for the solution of p-dispersion problems to proven optimality. *INFORMS Journal on Optimization*, 2:134–144, 2020.

C. Contardo, M. Lori, and R. Kramer. A scalable exact algorithm for the vertex p-center problem. *Computers and Operations Research*, 103:211–220, 2019.

T. G. Crainic, A. Frangioni, and B. Gendron. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112(1-3):73–99, 2001.

J. Czyzyk, M. P. Mesnier, and J. J. Moré. The neos server. *IEEE Journal on Computational Science and Engineering*, 5(3):68–75, 1998.

E. D. Dolan. The neos server 4.0 administrative guide. Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory, 2001.

J. E. Ertel and E. B. Fowlkes. Some algorithms for linear spline and piecewise multiple linear regression. *Journal of the American Statistical Association*, 71(355):640–648, 1976.

L. R. Ford Jr and D. R. Fulkerson. Solving the transportation problem. *Management Science*, 3(1): 24–32, 1956.

H. Freudenthal. Simplizialzerlegungen von beschrankter flachheit. *Annals of Mathematics*, 43(3): 580–582, 1942. ISSN 0003486X. URL http://www.jstor.org/stable/1968813.

G. Gamrath, D. Anderson, K. Bestuzheva, W.-K. Chen, L. Eifler, M. Gasse, P. Gemander, A. Gleixner, L. Gottwald, K. Halbig, G. Hendel, C. Hojny, T. Koch, P. Le Bodic, S. J. Maher, F. Matter, M. Miltenberger, E. Mühmer, B. Müller, M. E. Pfetsch, F. Schlösser, F. Serrano, Y. Shinano, C. Tawfik, S. Vigerske, F. Wegscheider, D. Weninger, and J. Witzig. The SCIP Optimization Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin, March 2020. URL http://nbn-resolving.de/urn:nbn:de:0297-zib-78023.

B. Geißler, A. Martin, A. Morsi, and L. Schewe. Using piecewise linear functions for solving MINLPs. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, pages 287–314, New York, NY, 2012. Springer New York. ISBN 978-1-4614-1927-3.

W. Gropp and J. J. Moré. Optimization environments and the neos server. In M. D. Buhman and A. Iserles, editors, *Approximation Theory and Optimization*, pages 167–182. Cambridge University Press, 1997.

IBM. *CPLEX Optimization Studio 20.1*, 2020.

P. Kleinschmidt and H. Schannath. A strongly polynomial algorithm for the transportation problem. *Mathematical Programming*, 68(1-3):1–13, 1995.

Y. Lin and L. Schrage. The global solver in the lindo api. *Optimization Methods and Software*, 24(4-5):657–668, 2009. doi: 10.1080/10556780902753221. URL `https://doi.org/10.1080/10556780902753221`.

R. Misener and C. A. Floudas. Antigone: algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization*, 59(2-3):503–526, 2014.

S. U. Ngueveu. Piecewise linear bounding of univariate nonlinear functions and resulting mixed integer linear programming-based solution methods. *Euopean Journal of Operational Research*, 275:1058–1071, 2019.

S. Rebennack and J. Kallrath. Continuous piecewise linear delta-approximations for univariate functions: Computing minimal breakpoint systems. *Journal of Optimization Theory and Applications*, 167(2):617–643, 2015.

S. Rebennack and V. Krasko. Piecewise linear function fitting via mixed-integer linear programming. *INFORMS Journal on Computing*, to appear, 2019.

N. V. Sahinidis. *BARON 21.1.13: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2017.

M. Sinnl. An iterative exact algorithm for the weighted fair sequences problem, 2021.

R. Sridharan. The capacitated plant location problem. *European Journal of Operational Research*, 87:203–213, 1995.

B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020. doi: 10.1007/s12532-020-00179-2. URL `https://doi.org/10.1007/s12532-020-00179-2`.

J. P. Vielma, S. Ahmed, and G. Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: unifying framework and extensions. *Operations Research*, 58(2):303–315, 2010.