Constraints (2017) 22:338–376
DOI 10.1007/s10601-016-9267-5

SURVEY

Domain reduction techniques for global NLP and MINLP optimization

Yash Puranik[1] · **Nikolaos V. Sahinidis**[1]

Published online: 14 January 2017

Abstract Optimization solvers routinely utilize presolve techniques, including model simplification, reformulation and domain reduction techniques. Domain reduction techniques are especially important in speeding up convergence to the global optimum for challenging nonconvex nonlinear programming (NLP) and mixed-integer nonlinear programming (MINLP) optimization problems. In this work, we survey the various techniques used for domain reduction of NLP and MINLP optimization problems. We also present a computational analysis of the impact of these techniques on the performance of various widely available global solvers on a collection of 1740 test problems.

Keywords Constraint propagation · Feasibility-based bounds tightening · Optimality-based bounds tightening · Domain reduction

1 Introduction

We consider the following mixed-integer nonlinear programming problem (MINLP):

$$\left.\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & x_l \leq x \leq x_u \\ & x \in \mathbb{R}^{n-m} \times \mathbb{Z}^m \end{array}\right\} \tag{1}$$

MINLP is a very general representation for optimization problems and includes linear programming (LP), mixed-integer linear programming (MIP) and nonlinear programming

✉ Nikolaos V. Sahinidis
sahinidis@cmu.edu

Yash Puranik
puranik@cmu.edu

[1] Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213, USA

(NLP) in its subclasses. A variety of applications in diverse fields are routinely formulated using this framework including water network design [62, 94], hydro energy systems management [44], protein folding [143], robust control [18], trim loss [84], heat exchanger network synthesis [69], gas networks [128], transportation [68], chemical process operations [76], chemical process synthesis [77], crystallographic imaging [178], and seizure predictions [148]. Modelling via nonconvex objective functions or constraints is necessitated for many of these practical applications. Aside from the combinatorial complexity introduced by the integer variables, nonconvexities in objective function or the feasible region lead to multiple local minima and provide a challenge to the optimization of such problems.

Branch-and-bound [138] based methods can be exploited to solve Problem (1) to global optimality. Inspired by branch-and-bound for discrete programming problems [109], branch-and-bound was adapted for continuous problems by Falk and Soland [58]. The algorithm proceeds by bounding the global optimum by a valid lower and upper bound throughout the search process. Whenever these bounds are within an acceptable tolerance, the algorithm terminates with the upper bound as a (near-)global optimum. The algorithm exhaustively searches the space by branching on variables to divide the space into subdomains. The lower and upper bounding procedures are recursively applied to the new subdomains in a tree search until the bounds converge. Branch-and-bound algorithms are known to terminate within ϵ-accuracy ($\epsilon > 0$) to a global optimum as long as branching, node selection and lower bounding are designed to satisfy certain conditions [89]. For special cases, the true global optimum can be finitely achieved with branch-and-bound [8, 38, 173]. The success of branch-and-bound methods for global optimization is evident from the numerous software implementations available, including ANTIGONE [135], BARON [185], Couenne [25], LindoGlobal [117] and SCIP [3].

In this work, we survey the various domain reduction techniques that are employed within branch-and-bound algorithms. While these techniques are not necessary to ensure convergence to the global optimum, they typically speed up convergence. These techniques often exploit feasibility analysis to eliminate infeasible parts of the search space. Alternatively, the methods can also utilize optimality arguments to shrink the search space while ensuring that at least one optimal solution is retained. Domain reduction techniques constitute the major component of the solution methods for satisfiability problems through unit propagation [126] and for constraint programming (CP) through various filtering algorithms that achieve differing levels of consistencies [29]. They are also exploited in artificial intelligence (AI) [54] and interval analysis [98, 142]. Some of the other names used in the literature for these methods include bound propagation, bounds tightening, bound strengthening, domain filtering, bound reduction and range reduction.

Mathematical-programming-based methods for solving nonconvex MINLPs often rely on the solution of a relaxation problem for finding a valid lower bound. The strength of the relaxations employed for lower bounding depends on the diameter of the feasible region. Smaller domains lead to tighter relaxations. For example, Fig. 1 shows the convex relaxation for a simple univariate concave function. A convex relaxation for a given function defined on a nonempty convex set is a convex function that underestimates the given function on its domain. The convex relaxation over the reduced domain provides a better approximation for the univariate concave function thereby providing better lower bounds. Domain reduction techniques not only reduce the diameter of the search space, but they also improve the tightness of convex relaxations.

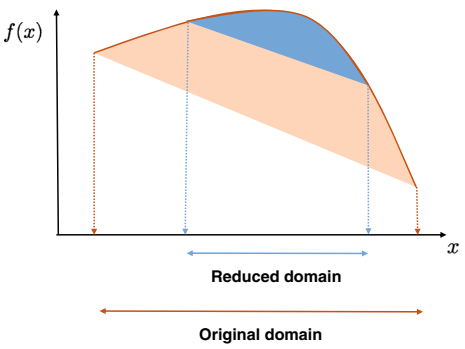

Fig. 1 Reduction in domains leads to tighter relaxations

Domain reduction techniques have been extensively studied in various communities including AI, CP, mathematical programming, interval analysis and computer science where they can be viewed as chaotic iterations [14]. The primary objective of this paper is to review key domain reduction techniques applicable to (nonconvex) NLPs and MINLPs and point to connections between methods from constraint programming and interval arithmetic communities wherever applicable. We also present results on standard test libraries with different global solvers to demonstrate the impact of domain reduction strategies on their performance.

The remainder of the paper is organized as follows. Representation of general MINLPs through factorable reformulations and directed acyclic graphs is described in Section 2. Methods for constraint propagation and bounds tightening in global optimization of MINLPs often rely on interval arithmetic. A brief introduction to this topic is provided in Section 3. We introduce presolving for optimization in Section 4. Domain reduction techniques that rely on eliminating infeasible regions for the problem are described in Section 5. Techniques that utilize optimality arguments for carrying out domain reduction are described in Section 6. Computational tradeoffs in the implementation of some of the advanced techniques are discussed in Section 7. The computational impact of many of these techniques on the performance of widely available solvers is investigated in Section 8. Finally, we conclude in Section 9.

2 Representation

A crucial step in the branch-and-bound algorithm is the construction of relaxations. One of the most widely used methods for this purpose is the idea of factorable reformulations. It involves splitting a problem into basic atomic functions that are utilized for computing the function, an idea exploited by McCormick [131], who developed a technique that constructed non-differentiable relaxations of optimization problems. Ryoo and Sahinidis [159] gain differentiability by introducing new variables and equations for each of the intermediate functional forms. These functional forms are simple in nature like the bilinear term. A similar idea was proposed by Kearfott [97] where new variables and equations are introduced to decompose nonlinearities to allow for more accurate computations of interval Jacobian matrices.

Consider the following example:

$$\left.\begin{array}{ll} \min & 3x + 4y \\ & x - y \leq 4 \\ & xy \leq 3 \\ & x^2 + y^2 \geq 1 \\ & 1 \leq x \leq 5 \\ & 1 \leq y \leq 5 \end{array}\right\} \tag{2}$$

A factorable reformulation can proceed by introducing a new variable for every nonlinearity occurring in the model. We replace z_1 for x^2, z_2 for y^2 and z_3 for xy.

A factorable reformulation of the model is thus given by:

$$\begin{array}{ll} \min & 3x + 4y \\ & x - y \leq 4 \\ & z_3 \leq 3 \\ & z_1 + z_2 \geq 1 \\ & z_1 = x^2 \\ & z_2 = y^2 \\ & z_3 = xy \\ & 1 \leq x \leq 5 \\ & 1 \leq y \leq 5 \end{array} \tag{3}$$

(4)

It is trivial to outer approximate the univariate terms of this model (cf. Fig. 1), while the bilinear term in (3) may be outer approximated by its convex and concave envelopes [131]. The combination of these outer approximators provides a convex relaxation of the original problem. In general, factorable reformulations decompose the problem into simpler functional forms for which convex relaxations are known. Thus, a convex relaxation can be obtained by reformulating a problem into its factorable form and relaxing each of the simple functional forms present in the model. For example, see Algorithm Relax f(x) in Tawarmalani and Sahinidis [185].

Factorable reformulations can be conveniently represented through a directed acyclic graph [167, 179]. In this graph representation, variables (x, y, z) and constants are leaf nodes, vertices are elementary operations $(+, -, *, /, \log, \exp,$ etc.) and the functions to be represented are the root nodes. Variable and constraint bounds are represented through suitable intervals for the root and leaf nodes. Common subexpressions are combined to reduce the size of the graph as doing so is known to tighten the resulting relaxations [185, Theorem 2]. Different mathematical formulations can be generated from the same DAG depending on the needs of the solver. Expressions are evaluated by propagating values from the leaves to the root node through the edges in a forward mode. Backward propagation is utilized for the computation of slopes and derivatives. Slopes can also be utilized to construct linear relaxations for the problem. Figure 2 represents the DAG for Problem (2).

3 Interval arithmetic

Interval arithmetic is a system of arithmetic based on intervals of real numbers [137]. An interval variable is defined using a variable's lower and upper bounds; the variable itself

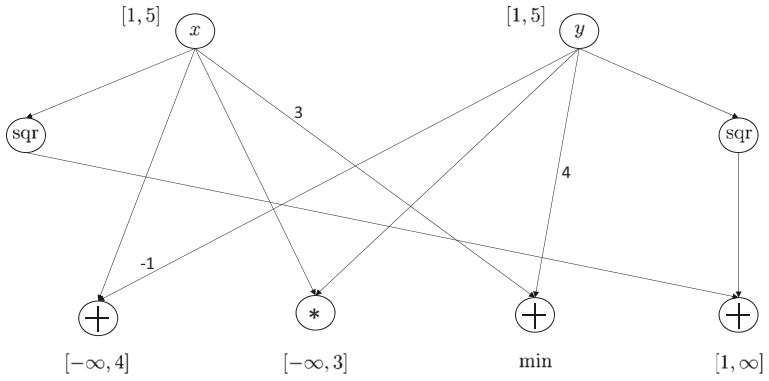

Fig. 2 Directed Acyclic Graph representation for Problem (2)

is restricted to lie between the bounds. Consider the interval variables $x = [x^l, x^u]$ and $y = [y^l, y^u]$. Addition on two intervals can be defined as:

$$x + y = [x^l + y^l, x^u + y^u] \tag{5}$$

The addition operator defined by (5) has the following property:

$$x + y = \{x + y \mid x \in x \text{ and } y \in y\} \tag{6}$$

Extensions to the definitions of other classic operators can be similarly defined:

$$x - y = [x^l - y^u, x^u - y^l] \tag{7}$$

$$x \times y = [\min(x^l \times y^l, x^l \times y^u, x^u \times y^l, x^u \times y^u)] \tag{8}$$

$$\frac{1}{x} = [1/x^u, 1/x^l], \quad 0 \notin [x^l, x^u] \tag{9}$$

$$\frac{x}{y} = x \times 1/y, \quad 0 \notin [y^l, y^u] \tag{10}$$

These operators can be suitably defined to account for infinities within the intervals and for the case when 0 lies in the interval of the denominator in a division operation [107]. Natural extensions of factorable functions can be computed by replacing all the elementary operations involved in the computation of a factorable function with their interval counterparts. A natural extension provides valid lower and upper bounds for the value of a function.

Floating point arithmetic is susceptible to rounding errors, which can lead to solutions being lost even for simple problems. Neumaier and Shcherbina [145] provide an example of a simple MIP problem that leads to incorrect solutions by major commercial solvers due to roundoff errors. Interval arithmetic methods utilize outward rounding to ensure no points are lost due to roundoff errors. These methods are utilized to design rigorous branch-and-bound-based optimization and root finding methods that ensure no solutions are lost due to roundoff errors [15, 193]. The methods bound function values through interval arithmetic and carry out domain reduction through branching and fathoming tests based on monotonicity, convexity, infeasibility as well as interval Newton type methods which provide conditions for existence and uniqueness of solutions within a box [81, 142]. Interval-based methods have been used for practical applications like solvent blend design [177]. Interval-based solvers include ICOS [110], Globsol [99] and Numerica [193]. Interval-based solvers are often slower than their nonrigorous counterparts. More recently, aspects of safe computations are being introduced with different parts of nonrigorous optimization algorithms.

For example, Neumaier and Shcherbina [145] describe methods to guarantee safety in the solution of MIPs through suitable preprocessing of the LP relaxations and postprocessing of their solutions. Their methods lead to valid results even when the LP solver itself does not use rigorous methods for obtaining a solution. Borradaile and van Hentenryck [34] provide ways of constructing numerically safe linear underestimators for univariate and multivariate functions. Numerically safe methods for computation have also been developed by the CP community, and are referenced in the remainder of the paper including in Sections 5 and 6.

4 Presolving optimization models

The idea of analyzing and converting an optimization model into a form more amenable to fast solution is old. Starting with the work of Brearley et al. [35], a number of techniques have been used for analyzing models in the operations research community. We use the term presolve to denote all the techniques used for simplification of optimization models. These techniques have been developed extensively for linear programming models [74, 93, 189, 190] and for mixed-integer linear programming models [87, 121, 141, 157, 164]. Bixby and Rothberg [31] observe that turning off root node presolve at the start of a branch-and-bound search degrades performance of CPLEX 8.0 by a factor of 10.8, while turning off presolve at every node other than the root node degrades the performance by a factor of 1.3 on certain MIP models, demonstrating the importance of presolve. See Achterberg and Wunderling [6] for an extensive computational analysis of the impact of various components of presolve algorithms for MIPs. Some of the ideas for simplification of models include:

- Elimination of redundant constraints
- Identification and elimination of dominated constraints (dominated constraints are constraints with a feasible region that is a superset of the feasible region of other constraints in the model)
- Elimination of redundant variables
- Assimilating singleton rows into bounds on variables
- Tightening bounds on dual variables
- Fixing variables at their bounds
- Increasing sparsity in the model
- Rearrangement of variables and constraints to induce structure

Similar preprocessing operations can be derived for nonlinear problems [10, 63, 75, 119, 133]. Some of the general guidelines include:

- Avoid potentially undefined functions
- Reduce nonlinearity in the model
- Improve scaling of model
- Increase convexity in the model through reformulations

Amarger et al. [10] provide a software implementation REFORM to carry out many of these reformulations. Presolve techniques are usually implemented at the solver level by developers of various software for optimization. The modelling systems AIMMS [7] and AMPL [11] also provide dedicated presolve systems that are applied to all optimization models [67, 91]. The success of presolve in mathematical programming has led to efforts for its extension to general constraint programming such as automated reformulation of CP models [115, 147]. These methods can lead to considerable simplifications in the model,

and can also lead to a reduction in the memory required to solve the problem. Another advantage of these presolve methods is that they are often able to detect infeasibility in optimization models. If a presolved model is infeasible, then the original model is also infeasible. Presolve methods can also be used to detect and correct the causes of infeasibilities for infeasible optimization models. Chinneck [46] provides examples of simple cases where infeasibilities can be correctly diagnosed by analyzing the sequence of reductions obtained with presolve. More recently, Puranik and Sahinidis [152] provide an automated infeasibility diagnosis methodology through the isolation of irreducible inconsistent sets (IISs). An IIS is defined as an infeasible set of constraints that has every subset feasible. Isolating an IIS can help accelerate the process of model correction by allowing the model expert to focus onto a smaller problem area within the model. The authors of [152] propose a deletion presolve procedure that exploits feasibility-based bounds tightening techniques in order to accelerate the isolation of an IIS.

While simplified models are often easier to solve than their original counterparts, once an optimal solution has been obtained, the modelling system must return a solution that can be interpreted by the user with respect to the original model form that was specified by the user. Restoration procedures to obtain primal and dual solutions to the original problem are described by Andersen and Andersen [12] and Fourer and Gay [67]. Such procedures are not discussed here. In the subsequent sections, we review domain reduction techniques that are a major component of all presolve algorithms.

5 Reduction of infeasible domains

The methods described in this section eliminate regions from the search space where no feasible points of Problem (1) can exist. Global optimization algorithms maintain continuous and discrete variable domains through upper and lower bounds. Domain reduction is achieved by making these bounds tighter. Therefore, domain reduction is also commonly referred to as bounds tightening.

The tightest possible bounds based on feasibility of the constraints of Problem (1) can be obtained by solving the following problems for each of the n variables ($k = 1, \ldots, n$):

$$\left.\begin{array}{ll} \min & \pm x_k \\ \text{s.t.} & g(x) \leq 0 \\ & x^l \leq x \leq x^u \\ & x \in \mathbb{R}^{n-m} \times \mathbb{Z}^m \end{array}\right\} \tag{11}$$

$\pm x_k$ in Problem (11) denotes two optimization problems, where x_k and $-x_k$ are individually minimized. Solution of these optimization problems returns the tightest possible bounds on the feasible region. If these bounds are tighter than the user-specified bounds in the model, we can achieve domain reduction by using them. However, since the constraints of Problem (11) are potentially nonconvex and due to the presence of integer variables, these are expensive global optimization problems, which might be as hard to solve as Problem (1). A computationally inexpensive way as compared to Problem (11) to obtain valid bounds for each of the variables is by solving the following problems for each of the n variables ($k = 1, \ldots, n$):

$$\left.\begin{array}{ll} \min & \pm x_k \\ \text{s.t.} & g_{\text{conv}}(x) \leq 0 \\ & x^l \leq x \leq x^u \\ & x \in \mathbb{R}^n \end{array}\right\} \tag{12}$$

where $g_{\text{conv}}(x) \leq 0$ refers to a convex relaxation of the constraints. The convex relaxation can be nonlinear. The integrality restrictions on the variables are relaxed as well. While bounds obtained from Problem (12) in general are weaker than the bounds obtained from Problem (11), they require the solution of convex optimization problems and are therefore obtained more efficiently. Convex relaxations are utilized in branch-and-bound methods for obtaining valid lower bounds to the optimum, and are thus already available for bounds tightening. To exploit the efficiency and robustness of LP solvers, these convex relaxations are linearized through outer approximation to obtain linear relaxations [186]. Linear relaxations may provide weaker bounds than nonlinear ones, but can be solved very efficiently. This linearization can also be utilized for obtaining tighter bounds on variables through the solution of the following problems for each of the n variables ($k = 1, \ldots, n$):

$$\left.\begin{array}{ll} \min & \pm x_k \\ \text{s.t.} & g_{\text{lin}}(x) \leq 0 \\ & x^l \leq x \leq x^u \\ & x \in \mathbb{R}^n \end{array}\right\} \tag{13}$$

where $g_{\text{lin}}(x) \leq 0$ refers to a linearized outer approximation of the feasible region. The bounds obtained from Problem (13) are in general weaker than the bounds obtained from Problem (12), but require the solution of linear instead of nonlinear programming problems and are therefore solved more efficiently.

Feasibility-based arguments for reduction can also be utilized for tightening constraints. Consider a linear set of constraints $Ax \leq b$. Tight bounds on constraint i can be obtained with the solution of following optimization problems [164]:

$$\left.\begin{array}{lll} \min & \pm a_i^T x & \\ \text{s.t.} & a_j^T x \leq b_j & j = 1, \ldots, i-1, i+1, \ldots, n \\ & x^l \leq x \leq x^u & \\ & x \in \mathbb{R}^n & \end{array}\right\} \tag{14}$$

Problem (14) can help identify redundancy if the maximum value of $a_i^T x$ is strictly less than b_i. Conversely, if the minimum value of $a_i^T x$ is strictly greater than b_i, infeasibility in the problem is identified.

In general, full solution of LP or NLP problems for bounds tightening can be expensive. To balance the computational effort involved with the reduction in bounds obtained, these techniques are usually carried out only at the root node and/or for a subset of variables. They are utilized only sparingly through the rest of the search or not at all. In some cases, however, solving optimization problems for tightening methods throughout the search has been shown to provide significant computational benefits [43].

Note that reduction methods exploiting Problems (11), (12) or (13) are often referred to in the literature as optimality-based bounds tightening. While these methods utilize the solution of optimization problems, they only carry out reduction of infeasible regions from the search space. For this reason, we prefer to classify them under feasibility-based reduction techniques.

5.1 Bounds propagation techniques

Propagation-based bounds tightening techniques will be simply referred to as propagation in the remainder of the paper. These techniques find their roots in several works in the literatures of mathematical programming [35], constraint logic programming [47], interval arithmetic [50], and AI [54]. These methods are often referred to as bounds propagation

techniques in CP and are a specialization of constraint propagation. Davis [54] refers to a constraint network with nodes (variables) which can take possible labels (domains) and are connected by constraints. He further refers to six different categories of constraint propagation based on the type of information which is updated:

- Constraint inference: New constraints are inferred and added.
- Label inference: Constraints are utilized to restrict the sets of possible values for nodes.
- Value inference: Nodes are partially initialized and constraints are utilized to complete assignments for all nodes.
- Expression inference: Nodes are labelled with values expressed over other nodes.
- Relaxation: Nodes are assigned exact values which may violate certain constraints.
- Relaxation labelling: Nodes are assigned labels using probabilities. Updates in the network involve update of the probabilities.

Propagation is equivalent to label inference in the AI community. Davis utilizes the Waltz algorithm [199] to describe bounds propagation in a constraint network. In the CP community, these methods are often utilized for solving discrete constraint satisfaction problems (CSP) [88] with backtracking-based search methods [192]. However, they are also utilized for continuous CSPs [28, 54].

Hager [79] discusses a reduce operator to eliminate regions outside the solution set for solving systems of constraints. These methods have also been developed extensively in the mathematical programming community, first for LPs, and later for nonlinear programming problems [80, 82, 108, 132, 187]. Mclinden and Mangasarian [122] demonstrate inference of bounds for simple monotonic complementarity problems and convex problems. Lodwick [118] analyzed the relationship between the constraint propagation methods from AI and bound tightening methods from mathematical programming. These methods typically operate by systematically analyzing one constraint at a time to infer valid bounds on variables.

Propagation techniques are computationally inexpensive. For example, consider a set of linear constraints:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i = 1, ..., k$$
$$x^l \leq x \leq x^u$$

The following inequalities are implied by every linear constraint:

$$x_h \leq \frac{1}{a_{ih}} (b_i - \sum_{j \neq h} \min \left(a_{ij} x_j^U, a_{ij} x_j^L \right)), \qquad a_{ih} > 0 \tag{15}$$

$$x_h \geq \frac{1}{a_{ih}} (b_i - \sum_{j \neq h} \min \left(a_{ij} x_j^U, a_{ij} x_j^L \right)), \qquad a_{ih} < 0 \tag{16}$$

If these inequalities imply tighter bounds on x_h than the ones specified by the model, the bounds can be updated. For example, consider the set of inequalities:

$$x_1 + x_2 \geq 4 \tag{17}$$
$$x_2 + x_3 \leq 1$$
$$x_1 \in [-2, 4]$$
$$x_2 \in [0, 4]$$
$$x_3 \in [-1, 1] \tag{18}$$

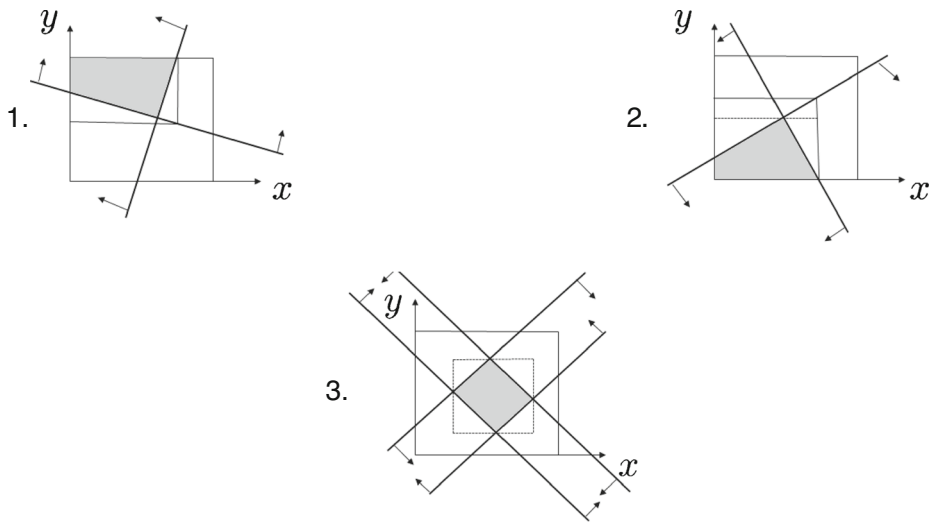

Fig. 3 Domain reductions through linear propagation

From inequality (17), we can infer $x_1 \geq 4 - \max(0, 4) = 0$. Thus, the lower bound of x_1 is updated to 0. From inequality (18), we can infer that $x_2 \leq 1 - \min(-1, 1) = 2$. Thus, the upper bound of x_2 is updated to 2. By analyzing inequality (17) again, we can update the lower bound of x_1 to 2. The domains of the variables after these bounds tightening steps are $x_1 \in [2, 4]$, $x_2 \in [0, 2]$ and $x_3 \in [-1, 1]$. Harvey and Schimpf [85] describe how bounds can be iteratively tightened in sublinear time for long linear constraints with many variables.

Inequalities (15) and (16) indicate that only one of the bounds for a variable can be updated from an inequality constraint based on the sign of its coefficient. Achterberg [2] formalizes this through the concept of variable locks. Thus, inequality (17) down locks variables x_1 and x_2, since the lower bounds for x_1 and x_2 cannot be moved arbitrarily without violating inequality (17). The concept of variable locks allows for efficient duality fixing of variables. Note that there is no up lock for variable x_1 and no down lock for variable x_3 based on the inequalities (15) and (16). Thus, if the coefficient of x_1 in the objective function is negative, x_1 can be set to its upper bound. Similarly, if the coefficient of x_3 in the objective function is positive, x_3 can be set to its lower bound. The concept of variables locks was extended for CP to develop presolve procedures for cumulative constraints by Heinz et al. [86]. Duality fixing is extended by Gamrath et al. [70] to allow for fixing of a singleton column. The authors also define dominance between two variables for a MIP and show how dominance information can be used for fixing variables at their bounds.

Figure 3 considers three cases of propagation. In Case 1, bounds for x and y are successfully tightened through iterative application of propagation, with no further tightening possible via Problem (11). In Case 2, the bound obtained for x by propagation is the tightest, however the bound for y can be tightened further by Problem (11). In Case 3, bounds for neither x nor y can be tightened by propagation, whereas significant reduction is possible via Problem (11).

In general, reduction in domain through propagation is not guaranteed. Consider the following example:

$$\begin{aligned} x_1 + x_2 &\geq 0 \\ x_1 + x_2 &\leq 4 \\ -x_1 + x_2 &\geq -2 \\ -x_1 + x_2 &\leq 2 \\ x_1 &\in [-3, 5] \\ x_2 &\in [-3, 5] \end{aligned}$$

The tightest possible domain for x_1 and x_2 based on feasibility arguments is $[-1, 3]$ and can be obtained by solving Problem (11). However, analyzing constraints one-at-a-time leads to no reduction of bounds for this problem. The reason for this drawback is that propagation only analyzes one constraint at a time, whereas Problem (11) utilizes information from all the constraints in the model simultaneously.

Belotti [22] considers bound reduction by generating a convex combination of two linear inequalities and utilizing this combination for bound reduction. The author proposes a univariate optimization problem to determine the optimal value for the convex multipliers. However, the computational complexity of considering all possible combinations of m constraints in n variables is $O(m^2n^3)$. The author proposes a heuristic scheme which, when implemented only on a subset of the nodes of the branch-and-bound tree, shows computational benefits of using this method. For other presolve-based methods for mixed-integer programs analyzing more than one constraint at a time, see Achterberg et al. [4]. Domes and Neumaier [55] propose the generation of a new constraint by taking the linear combination or aggregation of all constraints for quadratic problems. This new constraint can uncover new relationships between the variables and can lead to domain reduction. The authors propose the use of the dual multipliers of a local solution in the case of a feasible subproblem and the use of a constraint violation measure in the case of an infeasible subproblem to aggregate constraints. Their method shows computational benefits in reducing the cluster effect (Section 6.4). The notion of global constraints in CP [154] is related. A global constraint provides a concise representation for a set of individual constraints. Filtering algorithms on a global constraint effectively carry out domain reduction by considering multiple individual constraints simultaneously. Lebbah et al. [111] present a global constraint for a set of quadratic constraints and describe filtering algorithms. Similar ideas have been extended to polynomial constraints [113].

Bounds can also be inferred for nonlinear constraints via propagation. Consider a bilinear term of the form $x_i = x_j x_k$. Then,

$$x_i \leq \max\{x_j^L x_k^L, x_j^L x_k^U, x_j^U x_k^L, x_j^U x_k^U\} \tag{19}$$

and

$$x_i \geq \min\{x_j^L x_k^L, x_j^L x_k^U, x_j^U x_k^L, x_j^U x_k^U\} \tag{20}$$

represent valid bounds for variable x_i and can be used to potentially tighten any user-specified bounds for this variable.

In the context of factorable reformulations, an optimization problem is already decomposed into simple functional forms, which can be readily utilized for domain propagation. Consider, for instance, the factorable reformulation for Problem (2). From constraint (4), we can infer that $0 \leq z_3 \leq 3$. From constraint (3), we can infer that $1 \leq y \leq 3$.

The iterative application of simple constraints such as (15)–(16) and (19)–(20) is referred to as *poor man's LPs* and *poor man's NLPs* [160, 173] because this iterative application is an inexpensive way to approximate the solution of linear and nonlinear optimization problems that aim to reduce domains of variables. Propagation based on these techniques can be applied not only to the original problem constraints but also to any cutting planes and other valid inequalities that might be derived by the branch-and-bound solution algorithm. Moreover, all these techniques can be implemented efficiently via the DAG representation. If a bound on a variable x has changed, it can be propagated to other variables that depend on x through a simple forward propagation on the DAG based on interval arithmetic. Similarly, the variables on which x depends can be updated through a backward propagation.

5.2 Convergence of propagation

Propagation methods can be carried out iteratively as long as there is an improvement in variable bounds. However, these methods can fail to reach a fixed point finitely. Consider as an example [144]: $x_1 + x_2 = 0, x_1 - qx_2 = 0$ with $q \in (0, 1)$. Propagation will converge to $(0, 0)$ at the limit, i.e., as the number of iterations approaches infinity. On the contrary, any linear programming based method will terminate at $(0, 0)$ quickly as the only feasible point. A necessary condition for nonconvergence of propagation iterations is the presence of cycles in expression graphs [59]. The problem of achieving a fixed point with propagation iterations in the presence of only integer variables has been shown to be NP-complete [32, 33]. The fixed point obtained is independent of the order in which the variables are considered [41]. In practice, the iterations are usually terminated when the improvement in variable domains is insignificant or when an upper limit on the number of iterations is reached.

For linear inequalities in the presence of continuous variables alone, the fixed point can be obtained in polynomial time by the solution of a large linear program [23]. For nonlinear problems and for problems with integer variables, a linear relaxation can be solved to obtain reduced bounds [24].

5.3 Consistency

Constraint programming algorithms for constraint satisfaction problems rely on the formalized notion of consistency to propagate domains along constraint networks and remove values from variable domains that cannot be a part of any solution. Differing levels of consistency exist, including arc consistency and k-consistency, while various algorithms have been proposed to achieve these consistencies [29]. These concepts were originally proposed for discrete problems but were also extended to continuous constraint satisfaction problems [26, 27, 30, 49, 116, 162].

The domain reduction algorithms typically employed for global optimization of MINLPs are usually not iterated until they reach a certain level of consistency because attempting to establish consistency can lead to a prohibitively large computational effort. Domain reduction techniques are not even necessary to prove convergence of branch-and-bound based methods for global optimization. In contrast to CP methods, the availability of relaxation methods for generating bounds allows the mathematical-programming-based branch-and-bound algorithms to converge without establishing any levels of consistency.

5.4 Techniques for mixed-integer linear programs

Many techniques for presolving LPs can also be directly utilized for MIPs. However, specialized reduction methods can be developed for MIPs. For example, a number of methods have been developed for the analysis of 0-1 binary programs [51, 78, 164]. Probing and shaving techniques are equivalent to the singleton consistency techniques from CP. Singleton consistency techniques [150] proceed by assigning a fixed value to a variable and carrying out propagation. If this assignment leads to infeasibility of the constraint program, the value assigned to the variable cannot be a part of any solution and can be eliminated. Probing techniques, that can be used for binary programs, involve the fixing of a binary variable x_i to say 0. If basic preprocessing and other domain reduction methods are able to prove infeasibility for this subproblem, then the variable x_i can be fixed to 1. If fixing the binary variable x_i to both 0 and 1 lead to infeasibility, the problem can be declared as infeasible. For binary programs, if probing is carried on all binary variables, it leads to singleton consistency. Probing has also been utilized for satisfiability problems [120]. A similar technique for continuous problems is called shaving [129, 144]. The method proceeds by removing a fraction of the domain of a variable $[x_i^l, x_i^u]$ as either $[x_i^l + \epsilon, x_i^u]$ or $[x_i^l, x_i^u - \epsilon]$ and testing whether the reduced domain leads to provable infeasibility of the model. If infeasibility is indeed proved, the domain of the variable can be reduced to the complement of the box used for testing. Neumaier [144] suggests using $\epsilon = 0.1 \times (x_i^u - x_i^l)$ for this method. Shaving is widely used in the sub-field of CP based scheduling [191]. Belotti et al. [25] call this technique aggressive bounds tightening (ABT) and implement it in the solver COUENNE. Once a fraction of the domain of a variable has been eliminated, propagation is invoked in the hopes of proving infeasibility in the subproblem. However, ABT and shaving techniques in general are expensive and reduction in the domain is not guaranteed. Faria and Bagajewicz [62] propose several variants of this shaving strategy for bilinear terms and utilize it in a branch-and-bound algorithm for water management and pooling problems [60, 61]. Nannicini et al. [140] propose a similar strategy which they refer to as aggressive probing. In contrast to ABT technique, the nonconvex restrictions created by restricting a variable domain in their method are solved to global optimality with branch-and-bound in order to carry out the maximum possible domain reduction.

Implication-based reductions utilize relations between the values of different variables that must be satisfied at an optimal solution. For instance, if fixing two binary variables x_i and x_j to 0 leads to infeasibility, then we have an implication requiring that one of the variables must be nonzero, which can be represented by the inequality $x_i + x_j \geq 1$. Such relations can be efficiently represented through the use of conflict graphs [17]. Conflict graphs are utilized to generate valid inequalities that strengthen the MIP formulation. In general, if fixing a binary variable x_i to 0 implies that variable x_j must take value v, then the following inequalities are valid for the problem [121]:

$$x_j \leq v + (x_j^u - v)x_i$$
$$x_j \geq v - (v - x_j^l)x_i$$

Implications can be derived by carrying out probing by fixing more than one variable at a time and/or through the analysis of problem structure. Implications can also be utilized for identifying and eliminating redundant constraints. Inequalities derived from implications lead to automatic disaggregation for some constraints [164]. Disaggregated constraints, while redundant for the MIP formulation, lead to tighter LP relaxations for the problem.

Achterberg et al. [5] show how implications can be derived from conflicts (See Section 5.5) and from knapsack covers.

Tighter formulations can also be obtained for a problem through coefficient reduction. Consider the example of a linear constraint on binary variables from [127]:

$$-230x_{10} - 200x_{16} - 400x_{17} \leq -5$$

The coefficients of this constraint can be reduced to:

$$-x_{10} - x_{16} - x_{17} \leq -1$$

While the set of binary values satisfying both of the above constraints are the same, the reduced constraint has a tighter LP relaxation. Approaches for coefficient reduction are provided by [51, 164]. Andersen and Pochet [13] prove that, if no coefficients for an MIP system can be strengthened, then there does not exist a dominating constraint that can be used to replace an existing constraint in the MIP system to tighten its relaxation. They also describe an optimization formulation and an algorithmic solution to the problem of strengthening a coefficient in a constraint as much as possible.

5.5 Conflict analysis

Branch-and-bound based algorithms often encounter infeasibility in subproblems during the search for global optimum. Solvers for the satisfiability problem (SAT) also utilize a backtracking-based branching scheme for their solution [139]. The SAT problem consists of binary variables constrained by a set of logical conditions. The SAT problem has a solution if there exists an instantiation of the binary variables satisfying all the logical conditions. SAT-based solvers learn and add conflict clauses from infeasible subproblems [126]. These clauses are created by identifying the corresponding instantiations of a subset of variables leading to infeasibility and prohibiting them. This often leads to a reduction in the search tree. The idea of conflict analysis is similar to the idea of no-good learning from the CP community [96, 100, 114, 180]. No-good is a generalization of a conflict clause from SAT to CP.

The ideas of conflict analysis have been extended for MIP [1, 163]. However, generation of conflict clauses is more complicated for MIP due to the presence of both continuous and general integer variables. Infeasibility in SAT problems and CP problems is detected through a chain of logical deductions caused by fixing some variables. However, in MIP, infeasibility can be identified through either such deductions or an infeasible LP relaxation. Achterberg [1] proposes a generalized conflict graph (termed implication graph in [163]) for representation of bound propagations that can be used to identify cause of infeasibility. Note that the conflict graph and the implication graph for conflict analysis are defined differently than in Section 5.4. In the case of an infeasible LP relaxation, Achterberg proposes identifying a minimum bound-cardinality IIS. Representation of conflict causes for MIP requires use of disjunctive constraints which must be reformulated with additional binary variables and inequalities. Limited computational analysis demonstrates a reduction in the number of nodes and time for solution of MIPs with conflict analysis.

6 Reduction of suboptimal domains

In contrast to the methods of the previous section, the methods described here can lead to the elimination of feasible points from the domain under the condition that at least one globally optimal solution remains within the search space.

Consider the following convex relaxation of Problem (1):

$$\left.\begin{array}{ll} \min & f_{\text{conv}}(x) \\ \text{s.t.} & g_{\text{conv}}(x) \leq 0 \\ & x^l \leq x \leq x^u \\ & x \in \mathbb{R}^n \end{array}\right\} \tag{21}$$

As discussed before, convex relaxations are often constructed and linearized in a global branch-and-bound search for obtaining valid lower bounds. Assume that the optimal objective for Problem (21) has value L and, at the optimal solution, a bound $x_j \leq x_j^u$ is active with a Lagrange multiplier $\lambda_j > 0$. Let U be a known valid upper bound for the optimal objective function value of Problem (1). Then, the following constraint does not exclude any optimal solutions better than U (Theorem 2 in [158]):

$$x_j^l \geq x_j^u - \frac{U - L}{\lambda_j} \tag{22}$$

A geometric interpretation of this cut can be observed in Fig. 4, where x_j^* denotes the right-hand-side of (22). The constraint excludes values of x_j for which the convex relaxation is guaranteed to have its value function to be greater than or equal to U. Consequently, the nonconvex problem also has its value function guaranteed to be greater than or equal to U in this domain. A corresponding cut can also be derived if, at the optimal solution L of the convex relaxation, a variable is at its lower bound, i.e., $x_j = x_j^l$, with the corresponding Lagrange multiplier $\lambda^j > 0$. The upper bound can then be potentially tightened without losing optimal solutions by the following cut:

$$x_j^u \leq x_j^l + \frac{U - L}{\lambda_j} \tag{23}$$

For variables that are not at their bounds at the relaxation solution, probing tests can be carried out by temporarily fixing variables at their bounds and solving the restricted problem. For example, Tests 3 and 4 in [158] work as follows. Set $x_j = x_j^u$ and solve

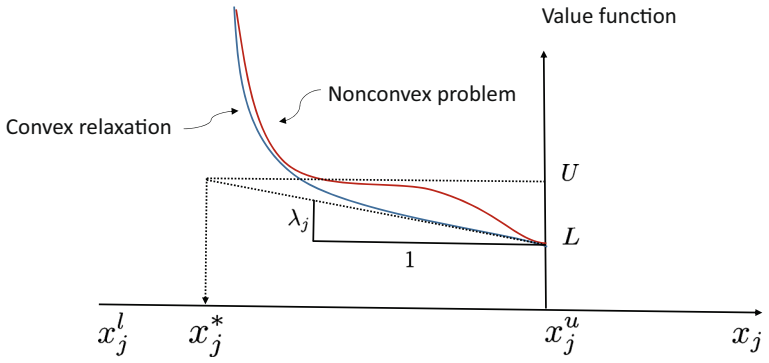

Fig. 4 Multipliers-based reduction geometrically

Problem (21). If the corresponding multiplier λ^j is positive, then the following constraint is valid:

$$x_j^l \geq x_j^u - \frac{U - L}{\lambda_j} \tag{24}$$

A similar probing test can be developed by fixing a variable at its lower bound [158, 159, 173]. Reduction by (22) and (23) only requires the solution of the relaxation problem, and thus can be implemented at every node of the branch-and-bound tree without much computational overhead. On the other hand, probing with fixing variables at their bounds requires the solution of $2n$ convex problems. Probing is usually carried out only at the root node and then only at some nodes of the tree or only for a subset of the variables. The probing tests are a generalization of probing for the integer programming case. However, they differ from the integer programming case in the sense that probing leads to variable fixing in the case of binary variables, whereas in the continuous case it leads to reduction in the variable domains.

Similarly, if at the optimal solution of the relaxation Problem (21), a constraint $g^j_{\text{conv}}(x) \leq 0$ is active with the corresponding multiplier $\mu_i \geq 0$, then the following constraint does not violate any points with objective values better than U [159]:

$$g^j_{\text{conv}}(x) \geq -\frac{U - L}{\mu_j} \tag{25}$$

These inequalities can be appended to the original formulation. However, this process can lead to the accumulation of a large number of constraints. Alternatively, these are often utilized for propagation-based tests locally at the current node and then discarded. Lebbah et al. [112] provide a safe way of implementing duality-based reduction techniques in the context of floating point arithmetic. It must be noted that suboptimal Lagrangian multipliers often provide greater reduction through constraints (22), (23), (24) and (25). A more detailed discussion on the use of other than optimal multipliers is provided in Tawarmalani and Sahinidis [185] and Sellmann [171].

If a valid upper bound U is available, it can also be exploited by reformulating Problem (12) as follows:

$$\left.\begin{array}{ll} \min & \pm x_k \\ \text{s.t.} & g(x) \leq 0 \\ & f_{\text{conv}}(x) \leq U \\ & x^l \leq x \leq x^u \\ & x \in \mathbb{R}^n \end{array}\right\} \tag{26}$$

Problem (26) differs from Problem (12) by the addition of a single constraint. In this constraint, $f_{\text{conv}}(x)$ refers to the convex relaxation of the objective function. It can be replaced by a suitable linearization $f_{\text{lin}}(x)$ for use in formulation (27).

$$\left.\begin{array}{ll} \min & \pm x_k \\ \text{s.t.} & g_{\text{lin}}(x) \leq 0 \\ & f_{\text{lin}}(x) \leq U \\ & x^l \leq x \leq x^u \\ & x \in \mathbb{R}^n \end{array}\right\} \tag{27}$$

Zamora and Grossmann [201] refer to Problem (26) as the contraction subproblem. They define a contraction operation that uses the solution of Problem (26) along with multipliers-based reduction steps described in this section to carry out domain reduction. Their algorithm is therefore referred to as branch-and-contract. Optimality-based arguments can also be used for filtering in CP, see for example [66, 188].

Note that Problem (26) can be solved iteratively to obtain further tightening. However, convergence to a fixed point can be slow. Caprara and Locatelli [41] propose a different approach to carry out bounds tightening called nonlinearities removal domain reduction (NRDR). NRDR relies on the solution of parametric univariate optimization problems to find tight bounds. The method reduces nonlinearities due to a single variable at every iteration. The authors further demonstrate that, under certain assumptions, their domain reduction method is equivalent to carrying out optimization-based bounds tightening iteratively until it reaches a fixed point. Caprara et al. [42] show that the NRDR method leads to bounds consistency for a special case of linear multiplicative programming problems with only two variables in the objective function.

6.1 A unified theory for feasibility- and optimality-based tightening

Tawarmalani and Sahinidis [185] provide a unified theory of feasibility- and optimality-based bounds tightening techniques. This theory evolves around Lagrangian subproblems created by the dualization of constraints. Consider the problem:

$$\left.\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & x_l \leq x \leq x_u \\ & x \in \mathbb{R}^n \end{array}\right\} \tag{28}$$

Define the Lagrangian subproblem as follows:

$$\inf_{x_l \leq x \leq x_u} \{-y_0 f(x) - y g(x)\}$$

The dual variables y_0 and y are nonpositive. Suppose an upper bound U is available for the optimal solution of Problem (28). A domain reduction master problem can be constructed as follows:

$$\left.\begin{array}{ll} \inf & h(\mu_0, \mu) \\ \text{s.t.} & f(x) \leq \mu_0 \leq U \\ & g(x) \leq \mu \leq b \\ & x_l \leq x \leq x_u \end{array}\right\} \tag{29}$$

Using the linear objective function $h(\mu_0, \mu) = a_0\mu_0 + a^T\mu$, Problem (29) can be equivalently stated as:

$$\left.\begin{array}{ll} \inf & a_0\mu_0 + a^T\mu \\ \text{s.t.} & -y_0(f(x) - \mu_0) - y^T(g(x) - \mu) \leq 0 \quad \forall (y_0, y) \leq 0 \\ & (\mu_0, \mu) \leq (U, 0) \\ & x_l \leq x \leq x_u \end{array}\right\} \tag{30}$$

Solving Problem (30) can be as hard as solving Problem (28). Instead, the lower bound for Problem (30) can be obtained from the solution of the following relaxed master problem:

$$\left.\begin{array}{ll} \inf & a_0\mu_0 + a^T\mu \\ \text{s.t.} & y_0\mu_0 + y^T u + \inf_{x_l \leq x \leq x_u}\{-y_0 f(x) - y^T g(x)\} \leq 0 \quad \forall (y_0, y) \leq 0 \\ & (\mu_0, \mu) \leq (U, 0) \end{array}\right\} \tag{31}$$

Many domain reduction operations can be obtained by suitable choice of the coefficients (a_0, a) in the objective function for Problem (31). For example, (25) can be derived by setting (a, a_0) to e_j in Problem (31), where e_j is the j^{th} column of the identity matrix.

Similarly, propagation for linear constraints described by inequalities (15) and (16) can derived by application of duality theory to the relaxed problem formed by relaxing all but the i^{th} linear constraint under consideration:

$$\begin{array}{ll} \min & x_h \\ \text{s.t.} & a_i^T x \leq b_i \\ & x \leq x^u \\ & x \geq x^l \end{array}$$

Assuming $a_{ih} < 0$ and x_h is not at its lower bound, the optimal dual solution for the linear program can be obtained as:

$$\begin{aligned} \mu &= 1/a_{ih} \\ \lambda_j &= -\max\{a_{ij}/a_{ih}, 0\} \text{ for all } j \neq h \\ \sigma_j &= \min\{a_{ij}/a_{ih}, 0\} \text{ for all } j \neq h \\ \lambda_h &= \sigma_h = 0 \end{aligned}$$

Here, μ is the dual multiplier corresponding to $a_i^T x \leq b_i$, λ_j is the dual multiplier corresponding to $x_j \leq x_j^u$, and σ_j is the dual multiplier corresponding to $x_j \geq x_j^l$. The domain reduction master problem can be constructed as:

$$\left.\begin{array}{ll} \min & x_h \\ \text{s.t.} & -x_h + \mu u + \lambda^T v - \sigma^T w \leq 0 \\ & u \leq b_i \\ & v \leq x^u \\ & w \leq x^l \end{array}\right\} \tag{32}$$

Equation (16) follows from Problem (32). Equation (15) can be similarly derived.

Tawarmalani and Sahinidis [185] also present a duality-based reduction scheme that utilizes dual feasible solutions and a learning reduction heuristic. In branch-and-bound algorithms, branching is typically carried out by various heuristics for the selection of the branching variable and the branching point. If one of the nodes created due to a branching decision is proven to be inferior, the learning reduction heuristic attempts to expand on the region defined by this node that is proven to be inferior by the construction of dual solutions for the other node. The authors remark that all dual feasible solutions can be utilized to carry out reductions. This idea is instantiated in the Lagrangian variable bounds propagation by Gleixner and coworkers [71, 72]. To avoid solving $2n$ optimization problems at every node with Problem (27), valid Lagrangian variable inequalities can be generated by aggregating the linear relaxation constraints with the dual solution of Problem (27). While redundant for the linear relaxation, these inequalities approximate the effect of solving Problem (27) locally at every node and can be used to infer stronger bounds on variables when variable bounds are updated through branching or otherwise if a better upper bound is obtained. Gleixner et al. [71] provides heuristics for ordering the generated Lagrangian variable inequalities to achieve maximum tightening. An aggressive filtering strategy is proposed that involves the solution of LPs to determine variables for which bounds cannot be tightened. The optimization steps for these variables in Problem (27) can therefore be skipped, leading to computational savings. The authors also provide heuristics for ordering the LP solves in Problem (27) to allow for more efficient warm starting and reduce the number of simplex iterations.

6.2 Pruning

If the lower bound obtained at a node of the branch-and-bound search is worse than the best known upper bound U, the node can be fathomed. We are guaranteed that the globally optimal solution cannot lie at this node, since all feasible solutions at this node have an objective value greater than U. This process is also referred to as pruning and is a special case of a general concept called dominance; node n_1 dominates node n_2 if, for every feasible solution s in n_2, there is a complete solution in n_1 that is as good or better than s. The idea of dominance is old, first introduced by Kohler and Steiglitz [105] and developed in more detail by Ibaraki [92]. Dominance relations can be utilized to speed up the search. For MIP problems, Fischetti and Toth [65] propose solution of an auxiliary MIP involving only fixed variables at a node to determine whether the current node is dominated by another node which may or may not have been explored yet. However, the overhead of solving the auxiliary MIP is fairly large. Fischetti and Salvagnin [64] propose improvements to this scheme and demonstrate computational benefits for network loading problems arising in telecommunication. Sewell et al. [172] propose a memory-based dominance rule for a scheduling application. Memory-based dominance rules require storage of the entire search tree, and their performance is dependent on the memory available. However, since information from the entire tree is available, considerably stronger pruning rules can be determined. Memory-based search algorithms are related to the heuristic search algorithms from AI [57, 182].

Problems involving integer variables often have a high degree of symmetry. Symmetry is detrimental for branch-and-bound algorithms as it can lead to repetitive work for the solver. Most symmetry breaking approaches rely on exploiting information about the specific problem being considered. A more general technique called isomorphic pruning has been proposed by Margot [123, 124]. Isomorphic pruning relies on lexicographic tests to determine if a node can be pruned. Orbital branching [146] is another method that can be utilized to tackle symmetry. The branching method identifies orbits of equivalent variables. Orbital branching proceeds by fixing a variable in the orbit to one at a node, and fixes all the variables in the orbit to zero in another node. Thus, orbital branching implicitly prunes all the other nodes which involve each of the other variables in the orbit fixed to one.

6.3 Exploiting optimality conditions

The Karush-Kuhn-Tucker [95, 106] conditions are necessary for a point to be locally optimal for a nonlinear programming problem under certain constraint qualifications. See Schichl and Neumaier [168] for a derivation and a general discussion of these conditions. The conditions can be used to reduce the search space for a nonconvex NLP, since globally optimal points also satisfy them. Vandenbussche and Nemhauser generate valid inequalities for quadratic programs with box constraints through the analysis of optimality conditions [196] and also utilize them in a branch-and-cut scheme [195]. Optimality conditions have been extensively utilized in branch-and-bound algorithms for quadratic programs [37–39, 45, 83, 90]. Optimality conditions can also be utilized for pruning of nodes. For example, for an unconstrained optimization problem, if 0 is not contained within the interval inclusion function for the partial derivatives of the objective function at a node, the corresponding node can be pruned [125]. This test is often referred to as the monotonicity test.

Sahinidis and Tawarmalani [161] have added a modelling language construct for BARON which allows for the specification of certain constraints as relaxation-only. Relaxation-only

constraints are utilized for the construction of convex relaxations and for domain reduction, but are not utilized in local search for obtaining upper bounds. The authors use first-order optimality conditions explicitly as relaxation-only constraints and observe improved convergence for some univariate optimization problems. Amaran and Sahinidis [9] use a similar strategy and show significant computational benefits for parameter estimation problems. They analyze their results and demonstrate that explicit use of optimality conditions aids in domain reduction steps of BARON leading to computational speedups. Puranik and Sahinidis [151] propose a strategy for carrying out implicit bounds tightening on optimality conditions for bound-constrained optimization problems. Their method does not require the generation of optimality conditions which can be time consuming and lead to increase in memory requirements. For a large collection of test problems, this strategy leads to computational speedups and reduction in the number of nodes.

6.4 Cluster effect

Branch-and-bounds methods often undergo repeated branching in the neighbourhood of the global solution before converging. This problem is referred to as the cluster effect and was first studied by Du and Kearfott [56] in the context of interval-based branch-and-bound methods. Subsequent analysis [144, 169, 200] also demonstrates the importance of convergence order of the bounding operation (see Definition 1 in [200]). These results indicate that at least second-order convergence is required to overcome the cluster effect. Thus, tighter relaxations can indeed help mitigate the cluster effect and their development is the subject of extensive research in the area [19, 20, 101–103, 134, 155, 174–176, 183, 184, 202]. Neumaier and coworkers provide methods to construct exclusion regions for the solution of systems of equations [166] and for the solution of optimization problems [165]. These exclusion regions guarantee that no other solution can lie within the exclusion box around a local minimizer or a solution for systems of equations. These boxes can then be eliminated from the search space. These boxes are constructed through existence and uniqueness tests based on Krawczyk operator or the Kantorovich Theorem (see Chapter 1, [98]). Other methods to construct exclusion regions include back boxing [194] and ϵ-inflation [130].

7 Implementation of domain reduction techniques

Domain reduction strategies, if successful, typically lead to a reduction in the number of nodes searched in a branch-and-bound tree. Techniques like propagation are computationally inexpensive and can be applied at every node without much overhead. However, they are not as efficient in carrying out domain reduction as the more computationally intensive strategies like Problem (11) or probing. Thus, there exists a need to balance the effort involved in domain reduction in order to reduce the average effort per node. Multiple heuristic or learning strategies are employed for this purpose. These ideas are important in constraint satisfaction problems where multiple filtering algorithms are available achieving varying degrees of consistency. Stergiou [181] experimentally analysed the domain reduction events through filtering techniques. Drawing insights from the clustering of this experimental data, various heuristics are proposed for choosing the propagation algorithm on the fly. Based on insights from Stergiou [181] that propagation events often occur in close clusters, Araya et al. [16] propose an adaptive strategy. Thus, if a constraint propagation mechanism succeeds in carrying out domain reduction at a given node, it should be exploited repetitively in other nodes that are geometrically close to the node until the method fails.

Similar heuristics are also utilized for solution of MINLP problems. For example, the aggressive probing strategy of Nannicini et al. [140] solves probing problems to global optimality and thus has a huge computational overhead. To avoid excessive work, Nannicini et al. propose a strategy based on support vector machines [170] to predict when this aggressive probing strategy is likely to succeed based on the success of propagation operations. Aggressive probing is only carried out when its chances of success are high. Couenne solves linear versions of Problem (11) in the branch-and-bound tree in all nodes up to a depth specified by a parameter L. For nodes at a depth $d > L$, the strategy is applied with a probability 2^{L-d}. A similar strategy is employed by ANTIGONE. The idea of propagation of Lagrangian variable bounds [71, 72] can also be thought of as a means of balancing the computational effort for tightening based on solving variants of Problem (11). Vu et al. [198] show significant computational benefits by carrying out propagation on a subset of the nodes and a partial subgraph of the DAG rather than the entire graph. Vu et al. [197] present multiple strategies for combining the various reduction schemes from constraint programming and mathematical programming for constraint satisfaction problems.

8 Computational impact of domain reduction

We demonstrate the computational impact of domain reduction techniques on three widely available solvers: BARON [185], Couenne [25] and SCIP [3]. BARON is commercial [21] and also free through the NEOS server [52]. SCIP is free for academics and commercial for all others. Couenne is an open-source and free software. All three solvers are available under the GAMS modeling system [36] and provide options that allow us to turn off their domain reduction algorithms. The global MINLP solvers Antigone [135] and LindoGlobal [117] are also available under GAMS but were not used in these experiments since they do not offer facilities that turn off their presolve routines.

The test libraries used in our tests are the Global library [73], Princeton library [149], MINLP library [40] and the CMU-IBM library [48]. Global and Princeton libraries consist of NLP problems, while the MINLP and CMU-IBM libraries consist of MINLP problems. While over 25 % of the Princeton library are convex and the NLP relaxations of the CMU-IBM library problems are all convex, the Global and MINLP libraries contain mostly nonconvex problems. In the sequel we present results for each library separately so as to demonstrate that the observed trends are not dominated by any particular library, convexity, or integrality properties. Key statistics on the test sets are summarized in Table 1.

All computational tests were run on a 64-bit Intel Xeon X5650 2.66 Ghz processor running CentOS release 7. The tests were carried out with a time limit of 500 seconds and absolute and relative optimality tolerances set to 10^{-6}. All solvers were run under two

Table 1 Test set statistics

Test Library	Global	Princeton	CMU-IBM	MINLP
Number of problems	369	980	142	249
Avg. no. of continuous variables	1092	1355	369	346
Avg. no. of binary variables	0	0	139	235
Avg. no. of integer variables	0	0	0	24
Avg. no. of constraints	785	836	956	534

Table 2 BARON options used for tests

Option	Value
LBTTDo	0
MDo	0
OBTTDo	0
PDo	0
TDo	0

different settings: (1) default options and (2) default options with domain reduction turned off. Our main objective is to compare these two settings. Comparing the relative impact of the individual reduction techniques on the performance of global solvers is beyond the scope of this work. Readers can refer to Kılınç and Sahinidis [104] for experiments showing the effect of different bounds reduction strategies in BARON. In the results presented below, the suffix "nr" is used to denote a solver applied with domain reduction techniques turned off. Comparisons are presented in terms of performance profiles generated through PAVER [136]. In these profiles, a solver is considered to have solved a problem if it obtains the best solution amongst the solvers compared in the profile within a given multiple of the

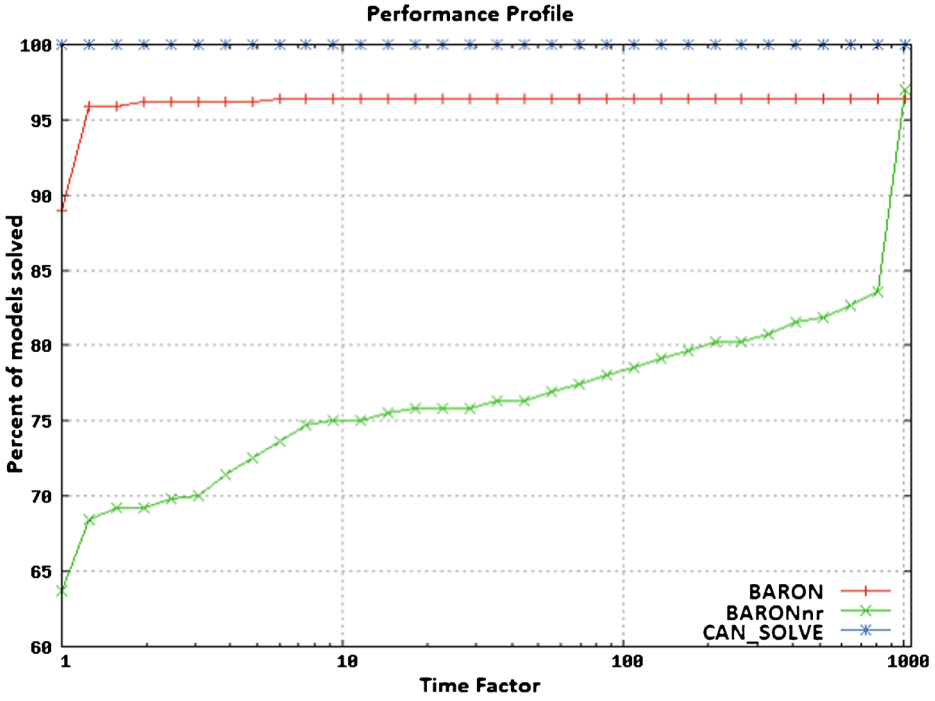

Fig. 5 Performance profiles for BARON on Global library

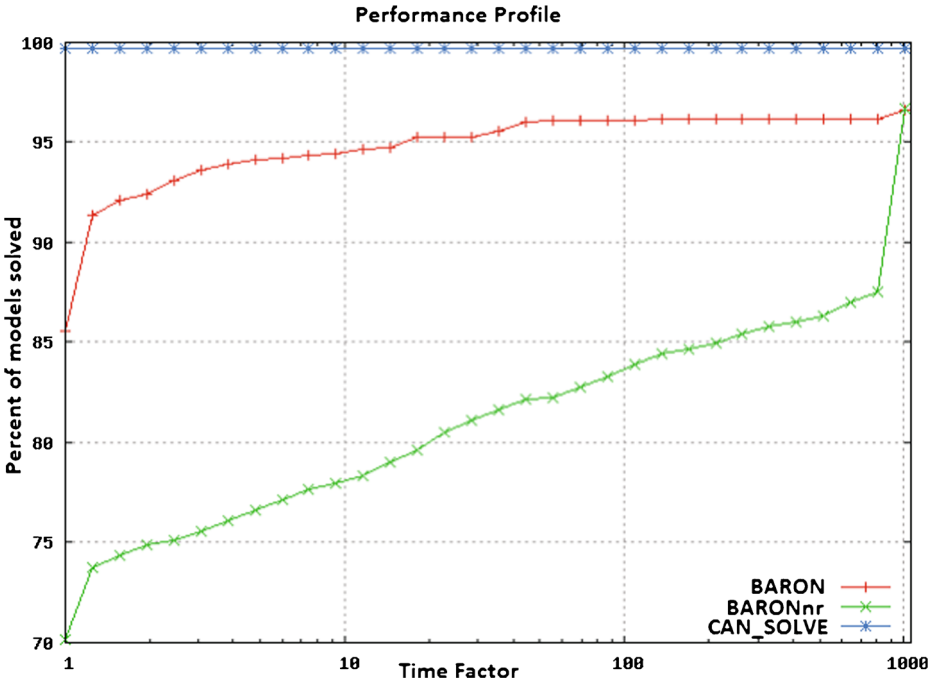

Fig. 6 Performance profiles for BARON on Princeton library

time taken by the fastest solver that solves a problem. CAN_SOLVE denotes the number of problems in the library that can be solved with all the solvers compared in the profile.

8.1 BARON

The solver options and their values utilized for tests with BARON are summarized in Table 2. These turn off the various domain reduction techniques utilized in BARON. The remaining options are utilized at their default settings.

Figures 5, 6, 7 and 8 demonstrate the performance of BARON with and without domain reduction techniques employed on the Global, Princeton, CMU-IBM and MINLP libraries. The profiles indicate a huge deterioration in performance across all test libraries when reduction is turned off. Interestingly, for the continuous test libraries, the performance profile of the no-reduction version of BARON "catches up" with that of the reduction-based version at the end of the profiles. This simply means that, without reduction, BARON's heuristics are still able to come up with (near-)global solutions but the branch-and-bound algorithm is not able to provide sufficiently strong lower bounds in order to prove global optimality. For the MINLP case, the no-reduction-based algorithm is not even able to find good feasible solutions.

Results in Table 3 show that turning off domain reduction techniques leads to a huge increase in the number of nodes required by BARON. The increase in nodes is also accompanied by a significant increase in computational time across all test libraries.

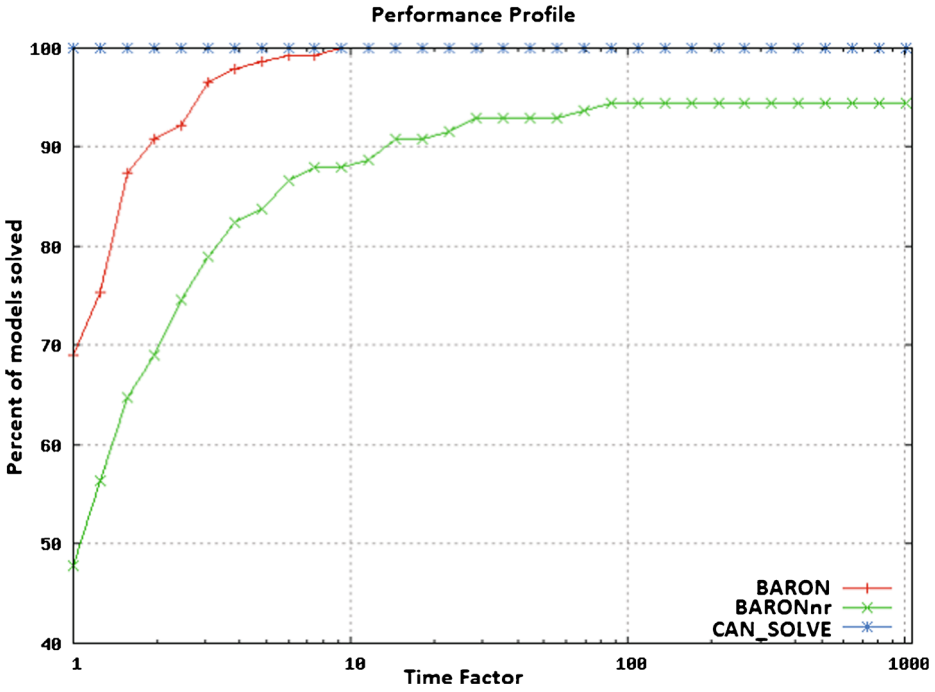

Fig. 7 Performance profiles for BARON on CMU-IBM library

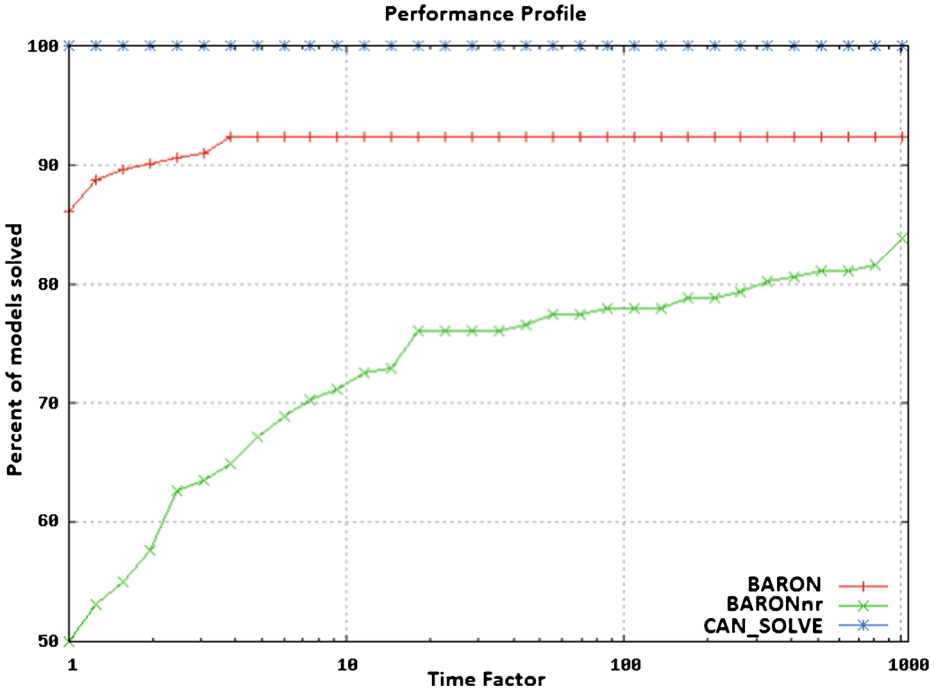

Fig. 8 Performance profiles for BARON on MINLP library

Table 3 Performance deterioration for BARON when domain reduction is turned off

Test Library	Global	Princeton	CMU-IBM	MINLP
% increase in number of nodes	1180	261	546	802
% increase in computational time	67	70	75	47

Table 4 Couenne options used for tests

Option	Value
aggressive_fbbt	no
branch_fbbt	no
feasibility_bt	no
max_fbbt_iter	0
optimality_bt	no
redcost_bt	no
fixpoint_bt	no
two_implied_bt	no

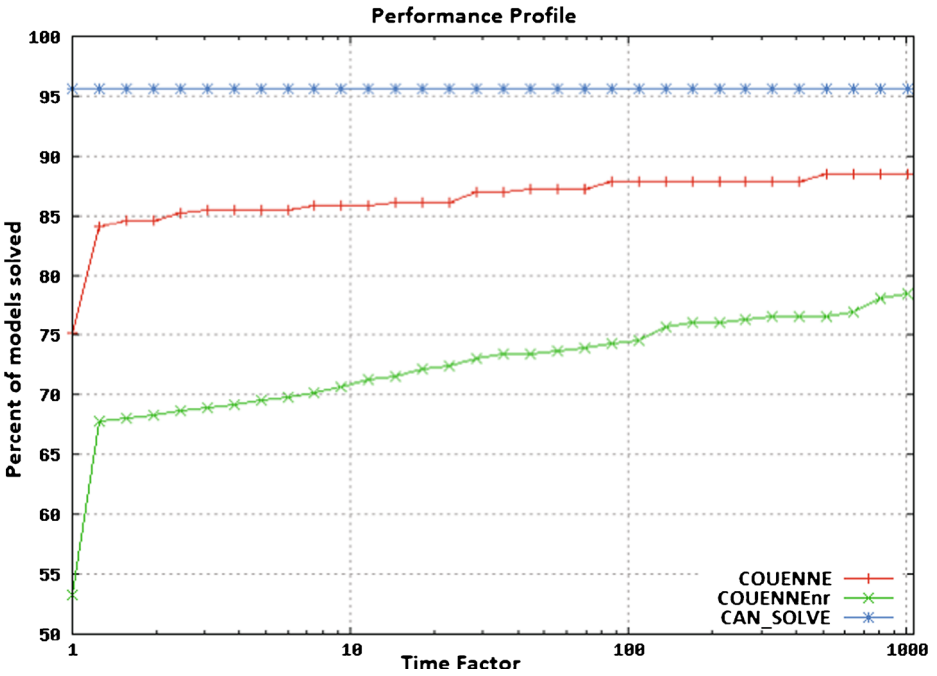

Fig. 9 Performance profiles for Couenne on Global library

8.2 Couenne

The solver options and their values utilized for tests with Couenne are summarized in Table 4. These turn off the various domain reduction techniques utilized in Couenne. The remaining options are utilized at their default settings.

Figures 9, 10, 11 and 12 demonstrate the performance of Couenne with and without domain reduction techniques employed on the Global, Princeton, CMU-IBM and MINLP libraries. Similar to BARON, turning off domain reduction techniques has a huge impact on the performance of Couenne. Contrary to BARON, without reduction, this solver is unable to find good solutions for the NLP test libraries; as a result, the performance profiles for the no-reduction-based algorithm do not catch up with the reduction-based algorithm. Table 5 indicates a significant increase in computational time and the number of nodes explored in the branch-and-bound search when reduction is turned off. It should be pointed out that no comparisons are possible between BARON and Couenne by looking at their respective performance profiles since these profiles depend solely on the solvers included in each figure.

8.3 SCIP

The solver options and their values utilized for tests with SCIP are summarized in Table 6. As before, the remaining options are used at their default settings.

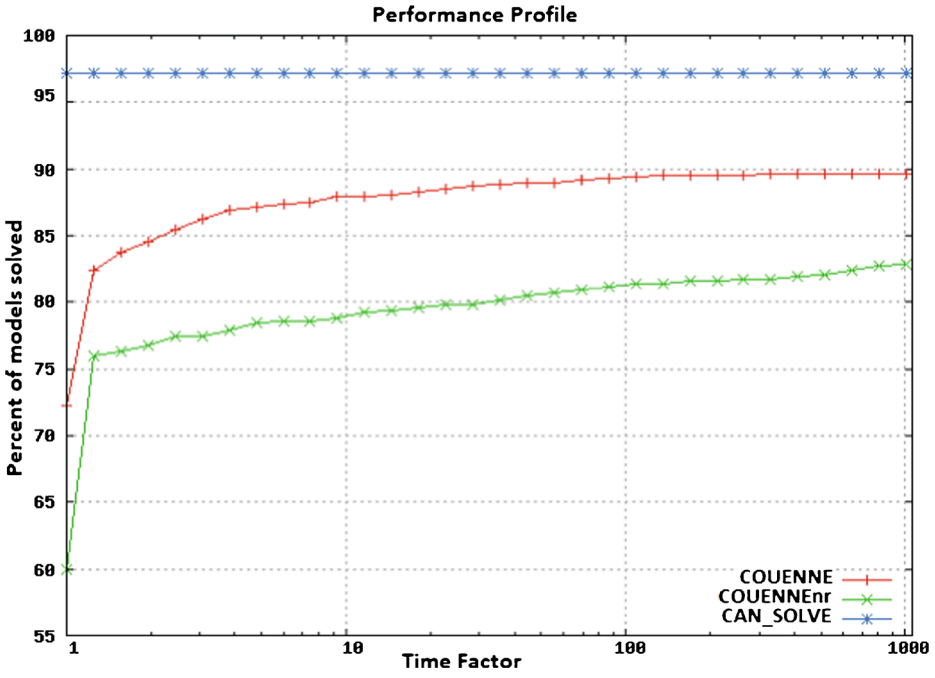

Fig. 10 Performance profiles for Couenne on Princeton library

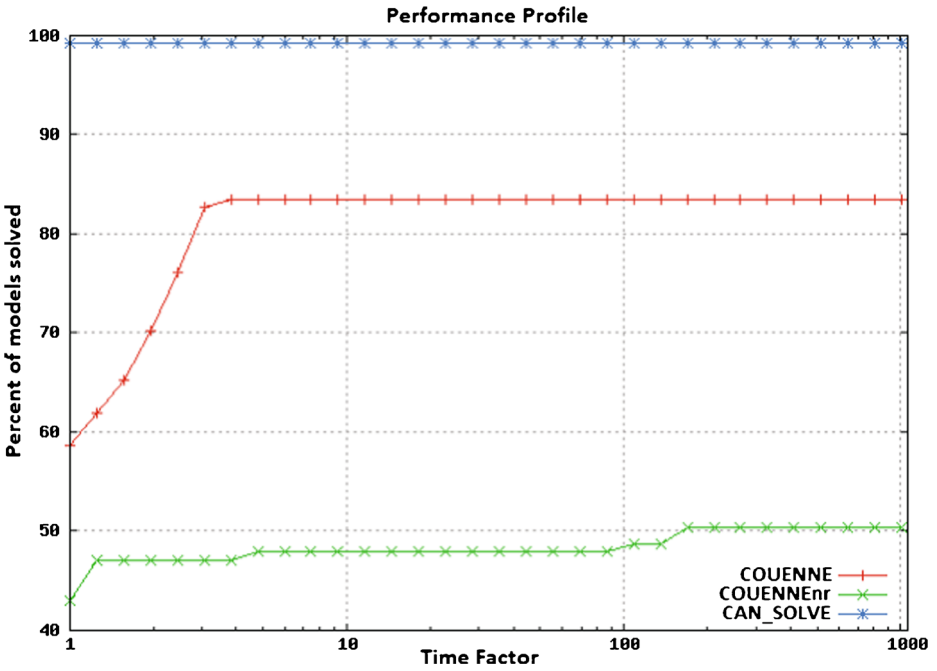

Fig. 11 Performance profiles for Couenne on CMU-IBM library

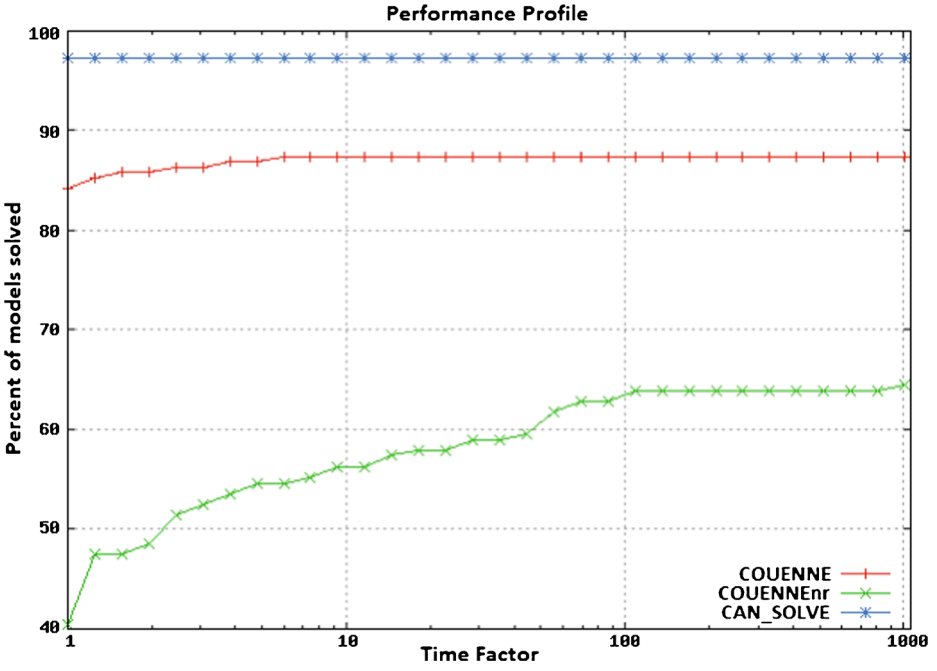

Fig. 12 Performance profiles for Couenne on MINLP library

Table 5 Performance deterioration for Couenne when domain reduction is turned off

Test Library	Global	Princeton	CMU-IBM	MINLP
% increase in number of nodes	129	21	186	171
% increase in computational time	26	19	12	16

Table 6 SCIP options used for tests

Option	Value
presolving/maxrounds	0
presolving/components/maxrounds	0
presolving/convertinttobin/maxrounds	0
presolving/domcol/maxrounds	0
presolving/dualagg/maxrounds	0
presolving/dualinfer/maxrounds	0
presolving/gateextraction/maxrounds	0
presolving/implfree/maxrounds	0
presolving/implics/maxrounds	0
presolving/inttobinary/maxrounds	0
presolving/redvub/maxrounds	0
presolving/stuffing/maxrounds	0
presolving/trivial/maxrounds	0
presolving/tworowbnd/maxrounds	0
propagating/maxrounds	0
propagating/dualfix/maxprerounds	0
propagating/genvbounds/maxprerounds	0
propagating/obbt/freq	-1
propagating/obbt/maxprerounds	0
propagating/obbt/tightintboundsprobing	False
propagating/probing/maxprerounds	0
propagating/pseudoobj/maxprerounds	0
propagating/redcost/maxprerounds	0
propagating/rootredcost/maxprerounds	0
propagating/vbounds/maxprerounds	0
conflict/useprop	False
conflict/useinflp	False
conflict/usepseudo	False
heuristics/bound/maxproprounds	0
heuristics/clique/maxproprounds	0
heuristics/randrounding/maxproprounds	0
heuristics/shiftandpropagate/freq	-1
heuristics/shifting/freq	-1
heuristics/vbounds/maxproprounds	0
misc/allowdualreds	False
misc/allowobjprop	False

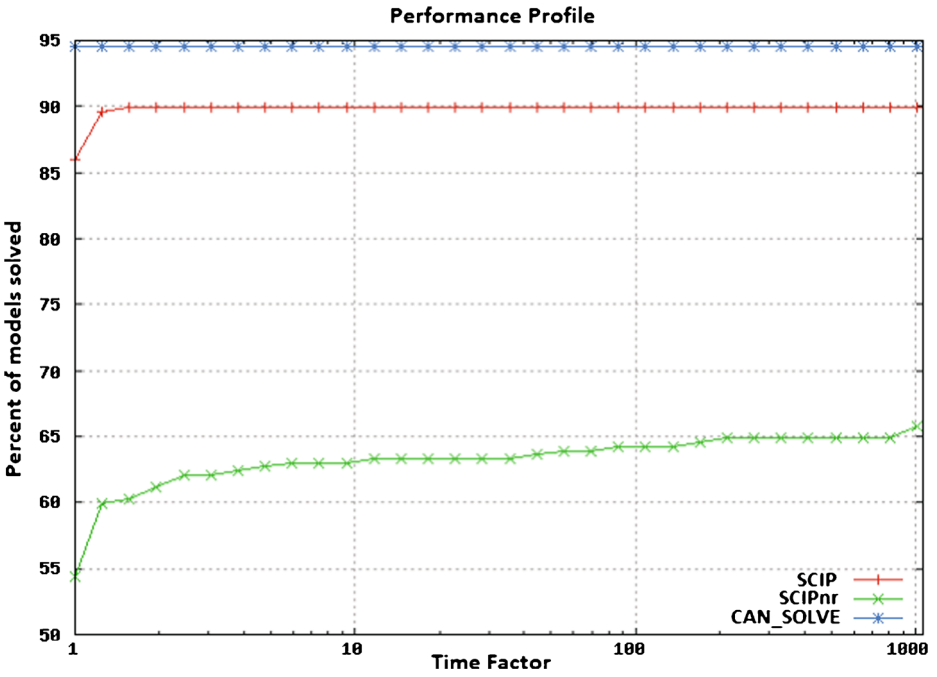

Fig. 13 Performance profiles for SCIP on Global library

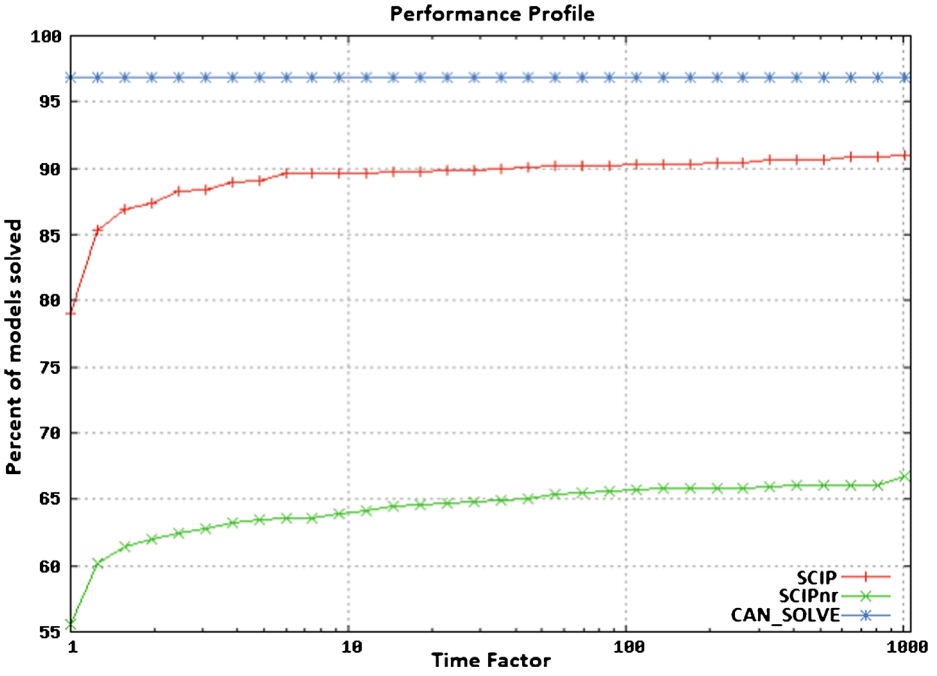

Fig. 14 Performance profiles for SCIP on Princeton library

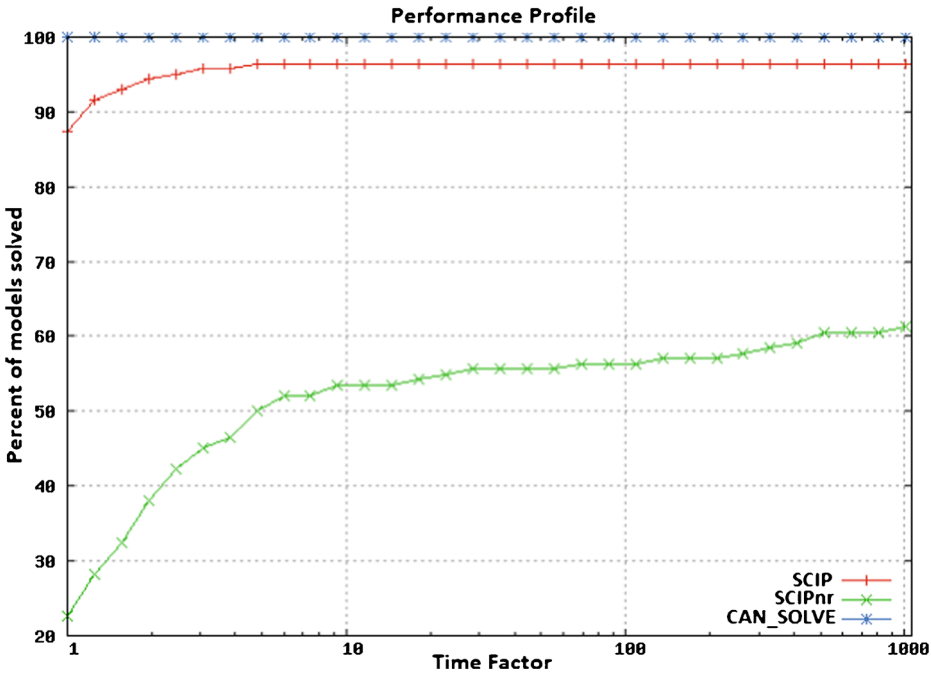

Fig. 15 Performance profiles for SCIP on CMU-IBM library

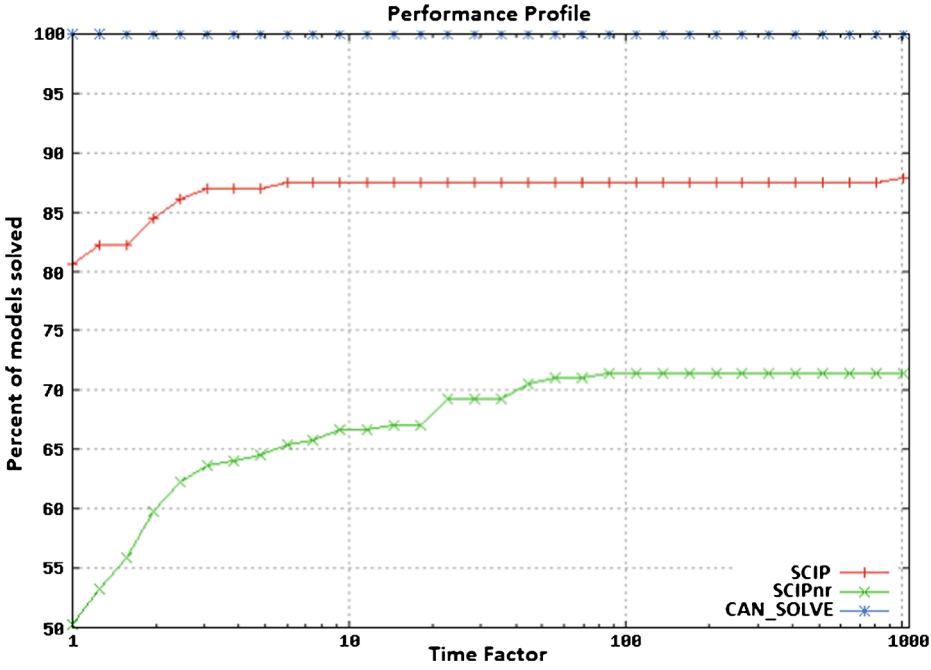

Fig. 16 Performance profiles for SCIP on MINLP library

Table 7 Performance deterioration for SCIP when domain reduction is turned off

Test Library	Global	Princeton	CMU-IBM	MINLP
% increase in number of nodes	174	152	417	56
% increase in computational time	24	33	141	18

Figures 13, 14, 15 and 16 demonstrate the huge impact of turning off domain reductions with SCIP on the Global, Princeton, CMU-IBM and MINLP libraries. Similar to Couenne, this solver is also not able of finding good solutions for continuous problems when reduction is turned off. Results in Table 7 indicate a significant deterioration in performance for SCIP as other solvers.

8.4 Relative solver performance

Figure 17 describes the performance of BARON, Couenne and SCIP on all test libraries aggregated together. Even without reduction, BARON dominates over the other two solvers, suggesting that this solver has an edge over the other two solvers in terms of its technology for relaxation construction, branching schemes, and primal feasibility heuristics. However, the impact of domain reduction techniques is clear and rather substantial on all three solvers. Interestingly, the relative order of SCIP and Couenne changes when reduction is turned off,

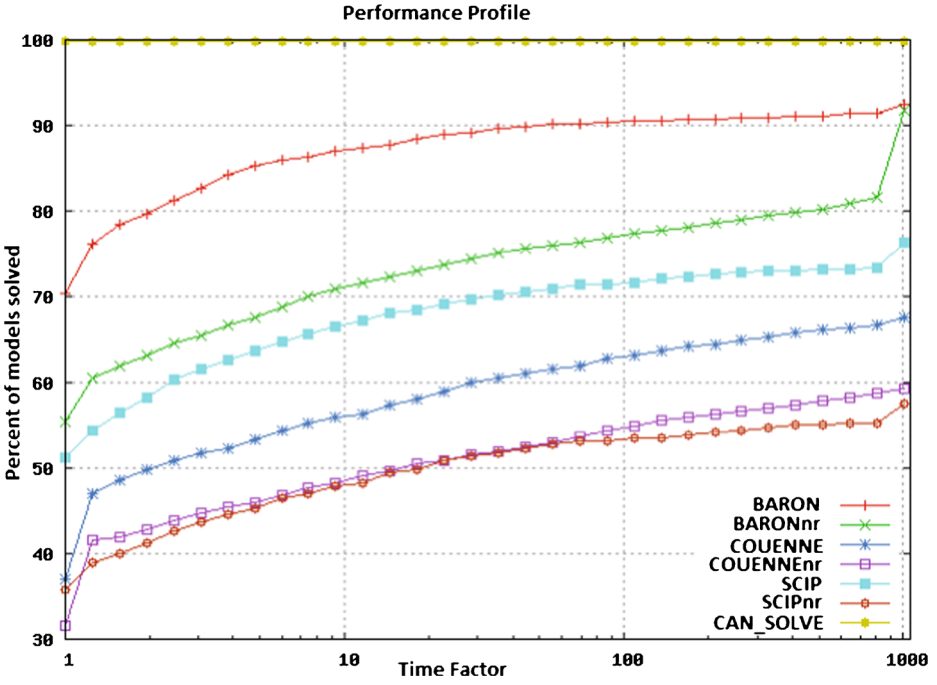

Fig. 17 Performance profiles for all solvers on all libraries combined

suggesting that SCIP is relying much more on domain reduction than Couenne does. Equivalently, SCIP enjoys a substantial advantage over Couenne thanks to its implementation of a more extensive set of reduction techniques.

9 Conclusions

We have presented a review of the various domain reduction techniques proposed in literature for the purpose of global NLP and MINLP optimization. These techniques vary in complexity including simple ones like propagation to more computationally intensive ones that involve full solution of optimization subproblems. Application of some of the more complex techniques requires the use of smart heuristics to ensure that they are utilized only when domain reduction is likely. We have also presented computational results with BARON, SCIP and Couenne on publicly available test libraries. The results show that domain reduction techniques have a significant impact on the performance of these solvers. Incorporation of domain reduction within branch-and-bound leads to huge reductions in computational time and number of nodes required for solution. Future research in this area should focus on the development of domain reduction techniques based on sets of constraints (i.e., global filtering methods) for broad classes of structured problems but also for important science and engineering problems, such as pooling problems [153], protein folding [143], and network design problems [53, 156].

References

1. Achterberg, T. (2007). Conflict analysis in mixed integer programming. *Discrete Optimization*, *4*, 4–20.
2. Achterberg, T. (2009). *Constraint Integer Programming*. Berlin: Ph.D. thesis, Technische Universität.
3. Achterberg, T. (2009). SCIP: solving constraint integer programs. *Mathematical Programming Computation*, *1*, 1–41.
4. Achterberg, T., Bixby, R.E., Gu, Z., Rothberg, E., & Weninger, D. (2014). Multi-row presolve reductions in mixed integer programming. In *Proceedings of the Twenty-Sixth RAMP Symposium. Hosei University, Tokyo*.
5. Achterberg, T., Sabharwal, A., & Samulowitz, H. (2013). Stronger inference through implied literals from conflicts and knapsack covers, In Gomes, C., & Sellmann, M. (Eds.) *Proceedings of 10th International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (pp. 1–11). Berlin: Springer.
6. Achterberg, T., & Wunderling, R. (2013). Mixed integer programming: Analyzing 12 years of progress, In Jünger, M., & Reinelt, G. (Eds.) *Facets of Combinatorial Optimization* (pp. 449–481). Berlin: Springer.
7. AIMMS: AIMMS Modeling Language (2015). http://www.aimms.com/.
8. Al-Khayyal, F.A., & Sherali, H.D. (2000). On finitely terminating branch-and-bound algorithms for some global optimization problems. *SIAM Journal on Optimization*, *10*, 1049–1057.
9. Amaran, S., & Sahinidis, N.V. (2012). Global optimization of nonlinear least-squares problems by branch-and-bound and optiMality constraints. *TOP*, *20*, 154–172.
10. Amarger, R.J., Biegler, L.T., & Grossmann, I.E. (1992). An automated modelling and reformulation system for design optimization. *Computers & Chemical Engineering*, *16*, 623–636.
11. AMPL: AMPL Modeling Language. http://www.ampl.com/.
12. Andersen, D.E., & Andersen, K.D. (1995). Presolving in linear programming. *Mathematical Programming*, *71*, 221–245.
13. Andersen, K., & Pochet, Y. (2010). Coefficient strengthening: A tool for reformulating mixed-integer programs. *Mathematical programming*, *122*, 121–154.
14. Apt, K.R. (1999). The essence of constraint propagation. *Theoretical computer science*, *221*, 179–210.
15. Araya, I., & Reyes, V. (2015). Interval Branch-and-Bound algorithms for optimization and constraint satisfaction: A survey and prospects. *Journal of Global Optimization*, 1–30.

16. Araya, I., Soto, R., & Crawford, B. (2015). Adaptive filtering strategy for numerical constraint satisfaction problems. *Expert Systems with Applications*, *42*, 8086–8094.
17. Atamtürk, A., Nemhauser, G.L., & Savelsbergh, M.W.P. (2000). Conflict graphs in solving integer programming problems. *European Journal of Operational Research*, *121*, 40 –55.
18. Balakrishnan, V., & Boyd, S. (1992). Global optimization in control system analysis and design, In Leondes, C.T. (Ed.) *Control and Dynamic Systems, Advances in Theory and Applications*. New York: Academic Press.
19. Bao, X., Khajavirad, A., Sahinidis, N.V., & Tawarmalani, M. (2015). Global optimization of nonconvex problems with multilinear intermediates. *Mathematical Programming Computation*, *7*, 1–37.
20. Bao, X., Sahinidis, N.V., & Tawarmalani, M. (2009). Multiterm polyhedral relaxations for nonconvex, quadratically-constrained quadratic programs. *Optimization Methods and Software*, *24*, 485–504.
21. BARON. http://minlp.com/baron.
22. Belotti, P. (2013). Bound reduction using pairs of linear inequalities. *Journal of Global Optimization*, *56*, 787–819.
23. Belotti, P., Cafieri, S., Lee, J., & Liberti, L. (2010). Feasibility-based bounds tightening via fixed points, In Wu, W., & Daescu, O. (Eds.) *Combinatorial Optimization and Applications* (pp. 65–76). Berlin: Springer.
24. Belotti, P., Cafieri, S., Lee, J., & Liberti, L. (2012). On feasibility based bounds tightening. http://www.optimization-online.org/DB_HTML/2012/01/3325.html.
25. Belotti, P., Lee, J., Liberti, L., Margot, F., & Wächter, A. (2009). Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, *24*, 597–634.
26. Benhamou, F., Goualard, F., Granvilliers, L., & Puget, J. (1999). Revising hull and box consistency. In *Proceedings of the 1999 International Conference on Logic Programming, pp. 230–244. Massachusetts Institute of Technology*.
27. Benhamou, F., McAllester, D., & Hentenryck, P.V. (1994). CLP (intervals) revisited, In Bruynooghe, M. (Ed.) *Proceedings of the 1994 International Symposium on Logic programming* (pp. 124-138). Cambridge: MIT Press.
28. Benhamou, F., & Older, W.J. (1997). Applying interval arithmetic to real, integer, and boolean constraints. *The Journal of Logic Programming*, *32*, 1–24.
29. Bessiere, C. (2006). Walsh Constraint propagation, In Rossi, F., van, P., & Beek, T. (Eds.) *Handbook of Constraint Programming, chap. 2* (pp. 29–83). Amsterdam: Elsevier.
30. Bessiere, C., Stergiou, K., & Walsh, T. (2008). Domain filtering consistencies for non-binary constraints. *Artificial Intelligence*, *172*, 800–822.
31. Bixby, R., & Rothberg, E. (2007). Progress in computational mixed integer programming–a look back from the other side of the tipping point. *Annals of Operations Research*, *149*, 37–41.
32. Bordeaux, L., Hamadi, Y., & Vardi, M.Y. (2007). An analysis of slow convergence in interval propagation, In Bessiere, C. (Ed.) *Principles and Practice of Constraint Programming–CP 2007* (pp. 790–797). Berlin: Springer.
33. Bordeaux, L., Katsirelos, G., Narodytska, N., & Vardi, M.Y. (2011). The complexity of integer bound propagation. *Journal of Artificial Intelligence Research*, *40*, 657–676.
34. Borradaile, G., & van Hentenryck, P. (2005). Safe and tight linear estimators for global optimization. *Mathematical Programming*, *102*, 495–517.
35. Brearley, A.L., Mitra, G., & Williams, H.P. (1975). Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical programming*, *8*, 54–83.
36. Brooke, A., Kendrick, D., & Meeraus, A. (1988). GAMS–A User's Guide. The Scientific Press, Redwood City CA.
37. Burer, S., & Chen, J. (2011). Relaxing the optiMality conditions of box QP. *Computational Optimization and Applications*, *48*, 653–673.
38. Burer, S., & Vandenbussche, D. (2008). A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. *Mathematical Programming*, *113*, 259–282.
39. Burer, S., & Vandenbussche, D. (2009). Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound. *Computational Optimization and Applications*, *43*, 181–195.
40. Bussieck, M.R., Drud, A.S., & Meeraus, A. (2003). MINLPLib–A collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing*, *15*, 114–119.
41. Caprara, A., & Locatelli, M. (2010). Global optimization problems and domain reduction strategies. *Mathematical Programming*, *125*, 123–137.
42. Caprara, A., Locatelli, M., & Monaci, M. (2016). Theoretical and computational results about optiMality-based domain reductions. *Computational Optimization and Applications*, 1–21.

43. Castro, P.M., & Grossmann, I.E. (2014). OptiMality-based bound contraction with multiparametric disaggregation for the global optimization of mixed-integer bilinear problems. *Journal of Global Optimization*, *59*, 277–306.
44. Catalão, J.P.S., Pousinho, H.M.I., & Mendes, V.M.F. (2011). Hydro energy systems management in Portugal: Profit-based evaluation of a mixed-integer nonlinear approach. *Energy*, *36*, 500–507.
45. Chen, J., & Burer, S. (2012). Globally solving nonconvex quadratic programming problems via completely positive programming. *Mathematical Programming Computation*, *4*, 33–52.
46. Chinneck, J.W. (2008). *Feasibility and infeasibility in optimization*. New York: Springer.
47. Cleary, J.G. (1987). Logical arithmetic. *Future computing systems*, *2*, 125–149.
48. CMU-IBM open source MINLP project test set. http://egon.cheme.cmu.edu/ibm/page.htm.
49. Collavizza, H., Delobel, F., & Rueher, M. (1999). Comparing partial consistencies. *Reliable computing*, *5*, 213–228.
50. Cornelius, H., & Lohner, R. (1984). Computing the range of values of real functions with accuracy higher than second order. *Computing*, *33*, 331–347.
51. Crowder, H., Johnson, E.L., & Padberg, M. (1983). Solving large-scale zero-one linear programming problems. *Operations Research*, *31*, 803–834.
52. Czyzyk, J., Mesnier, M., & Moré, J. (1998). The NEOS server. *IEEE Computational Science andamp; Engineering*, *5*, 68–75.
53. D'Ambrosio, C., Lodi, A., Wiese, S., & Bragalli, C. (2015). Mathematical programming techniques in water network optimization. *European Journal of Operational Research*, *243*, 774–788.
54. Davis, E. (1987). Constraint propagation with interval labels. *Artificial intelligence*, *32*, 281–331.
55. Domes, F., & Neumaier, A. (2016). Constraint aggregation for rigorous global optimization. *Mathematical Programming*, *155*, 375–401.
56. Du, K., & Kearfott, R.B. (1994). The cluster problem in multivariate global optimization. *Journal of Global Optimization*, *5*, 253–265.
57. Edelkamp, S., & Schroedl, S. (2011). Heuristic search: Theory and applications Elsevier.
58. Falk, J.E., & Soland, R.M. (1969). An algorithm for separable nonconvex programming problems. *Management Science*, *15*, 550–569.
59. Faltings, B. (1994). Arc-consistency for continuous variables. *Artificial intelligence*, *65*, 363–376.
60. Faria, D.C., & Bagajewicz, M.J. (2011). Global optimization of water management problems using linear relaxation and bound contraction methods. *Industrial & Engineering Chemistry Research*, *50*, 3738–3753.
61. Faria, D.C., & Bagajewicz, M.J. (2011). Novel bound contraction procedure for global optimization of bilinear MINLP problems with applications to water management problems. *Computers & Chemical Engineering*, *35*, 446–455.
62. Faria, D.C., & Bagajewicz, M.J. (2012). A new approach for global optimization of a class of MINLP problems with applications to water management and pooling problems. *AIChE Journal*, *58*, 2320–2335.
63. Ferris, M.C., & Munson, T.S. (2001). Preprocessing complementarity problems, In Ferris, M.C., Mangasarian, O.L., & Pang, J. (Eds.) *Complementarity: Applications, Algorithms and Extensions* (pp. 143-164). Dordrecht: Springer.
64. Fischetti, M., & Salvagnin, D. (2010). Pruning moves. *INFORMS Journal on Computing*, *22*, 108–119.
65. Fischetti, M., & Toth, P. (1988). A new dominance procedure for combinatorial optimization problems. *Operations Research Letters*, *7*, 181–187.
66. Focacci, F., Lodi, A., & Milano, M. (1999). Cost-based domain filtering, In Jaffar, J. (Ed.) *Proceedings of Principles and Practice of Constraint Programming: 5th International Conference* (pp. 189-203). Berlin: Springer.
67. Fourer, R., & Gay, D.M. (1994). Experience with a primal presolve algorithm, In Hager, W., Hearn, D., & Pardalos, P. (Eds.) *Large Scale Optimization: State of the Art* (pp. 135–154). Boston: Springer.
68. Fügenschuh, A., Homfeld, H., Schülldorf, H., & Vigerske, S. (2010). Mixed-integer nonlinear problems in transportation applications. In *Proceedings of the 2nd International Conference on Engineering Optimization (CD-ROM)* (p. 14).
69. Furman, K.C., & Sahinidis, N.V. (2002). A critical review and annotated bibliography for heat exchanger network synthesis in the 20th century. *Industrial & Engineering Chemistry Research*, *41*, 2335–2370.
70. Gamrath, G., Koch, T., Martin, A., Miltenberger, M., & Weninger, D. (2015). Progress in presolving for mixed integer programming. *Mathematical Programming Computation*, *7*, 367–398.
71. Gleixner, A.M., Berthold, T., Müller, B., & Weltge, S. (2016). Three enhancements for optimization-based bound tightening. *ZIB Report*, 15–16.

72. Gleixner, A.M., & Weltge, S. (2013). Learning and propagating Lagrangian variable bounds for mixed-integer nonlinear programming. In *Proceedings of 10th International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (pp. 355–361). Berlin: Springer.
73. GLOBAL Library. http://www.gamsworld.org/global/globallib.htm.
74. Gondzio, J. (1997). Presolve analysis of linear programs prior to applying an interior point method. *INFORMS Journal on Computing*, *9*, 73–91.
75. Gould, N., & Toint, P.L. (2004). Preprocessing for quadratic programming. *Mathematical Programming*, *100*, 95–132.
76. Grossmann, I. (2005). Enterprise-wide optimization: A new frontier in process systems engineering. *AIChE Journal*, *51*, 1846–1857.
77. Grossmann, I.E., Caballero, J.A., & Yeomans, H. (2000). Advances in mathematical programming for the synthesis of process systems. *Latin American Applied Research*, *30*, 263–284.
78. Guignard, M., & Spielberg, K. (1981). Logical reduction methods in zero-one programming-minimal preferred variables. *Operations Research*, *29*, 49–74.
79. Hager, G.D. (1993). Solving large systems of nonlinear constraints with application to data modeling. *Interval Computations*, *3*, 169–200.
80. Hamed, A.S.E., & McCormick, G.P. (1993). Calculation of bounds on variables satisfying nonlinear inequality constraints. *Journal of Global Optimization*, *3*, 25–47.
81. Hansen, E.R. (1992). Global optimization using interval analysis. Pure and Applied Mathematics.
82. Hansen, P., Jaumard, B., & Lu, S.H. (1991). An analytic approach to global optimization. *Mathematical Programming*, *52*, 227–254.
83. Hansen, P., Jaumard, B., Ruiz, M., & Xiong, J. (1993). Global minimization of indefinite quadratic functions subject to box constraints. *Naval Research Logistics (NRL)*, *40*, 373–392.
84. Harjunkoski, I., Westerlund, T., Pörn, R., & Skrifvars, H. (1998). Different transformations for solving non-convex trim-loss problems by MINLP. *European Journal of Operational Research*, *105*, 594–603.
85. Harvey, W., & Schimpf, J. (2002). Bounds consistency techniques for long linear constraints. In *Proceedings of TRICS: Techniques foR Implementing Constraint programming Systems* (pp. 39–46).
86. Heinz, S., Schulz, J., & Beck, J.C. (2013). Using dual presolving reductions to reformulate cumulative constraints. *Constraints*, *18*, 166–201.
87. Hoffman, K.L., & Padberg, M. (1991). Improving LP-representations of zero-one linear programs for branch-and-cut. *ORSA Journal on Computing*, *3*, 121–134.
88. Hooker, J.N. (2007). *Integrated methods for optimization*. New York: Springer Science & Business Media.
89. Horst, R., & Tuy, H. (1996). *Global Optimization: Deterministic Approaches*, 3rd edn. Berlin: Springer Verlag.
90. Hu, J., Mitchell, J.E., & Pang, J. (2012). An LPCC approach to nonconvex quadratic programs. *Mathematical programming*, *133*, 243–277.
91. Hunting, M. (2011). A nonlinear presolve algorithm in AIMMS, An AIMMS white paper, Paragon Decision Technology, BV.
92. Ibaraki, T. (1977). The power of dominance relations in branch-and-bound algorithms. *Journal of the ACM*, *24*, 264–279.
93. Imbert, J., & Hentenryck, P.V. (1996). Redundancy elimination with a lexicographic solved form. *Annals of Mathematics and Artificial Intelligence*, *17*, 85–106.
94. Jezowski, J. (2010). Review of water network design methods with literature annotations. *Industrial & Engineering Chemistry Research*, *49*, 4475–4516.
95. Karush, W. (1939). *Minima of functions of several variables with inequalities as side constraints*. Chicago, IL: Master's thesis, Department of Mathematics, University of Chicago.
96. Katsirelos, G. (2008). *Nogood processing in CSPs*. Ph.D. thesis: University of Toronto.
97. Kearfott, R.B. (1991). Decomposition of arithmetic expressions to improve the behavior of interval iteration for nonlinear systems. *Computing*, *47*, 169–191.
98. Kearfott, R.B. (1996). *Rigorous Global Search: Continuous Problems, Nonconvex Optimization and Its Applications* Vol. 13. Dordrecht: Kluwer Academic Publishers.
99. Kearfott R.B. (2009). GlobSol user guide. *Optimization Methods and Software*, *24*, 687–708.
100. Kell, B., Sabharwal, A., & van Hoeve, W. (2015). BDD-guided clause generation, In Michel, L. (Ed.) *Integration of AI and OR Techniques in Constraint Programming* (pp. 215–230): Springer.
101. Khajavirad, A., Michalek, J.J., & Sahinidis, N.V. (2014). Relaxations of factorable functions with convex-transformable intermediates. *Mathematical Programming*, *144*, 107–140.

102. Khajavirad, A., & Sahinidis, N.V. (2011). Convex envelopes of products of convex and component-wise concave functions. *Journal of Global Optimization*, *52*, 391–409.
103. Khajavirad, A., & Sahinidis, N.V. (2013). Convex envelopes generated from finitely many compact convex sets. *Mathematical Programming*, *137*, 371–408.
104. Kılınç, M., & Sahinidis, N.V. (2014). Solving MINLPs with BARON. Presented at MINLP Workshop, Pittsburgh http://http://minlp.cheme.cmu.edu/2014/papers/kilinc.pdf.
105. Kohler, W.H., & Steiglitz, K. (1974). Characterization and theoretical comparison of branch-and-bound algorithms for permutation problems. *Journal of the ACM*, *21*, 140–156.
106. Kuhn, H.W., & Tucker, A.W. (1951). Nonlinear programming, In Neyman, J. (Ed.) *Proceedings ofthe Second Berkeley Symposium on Mathematical Statistics and Probability* (pp. 481–492). Berkeley: University of California Press.
107. Kulisch, U.W. (2009). Complete interval arithmetic and its implementation on the computer, In Cuyt, A., Krämer, W., Luther, W., & Markstein, P. (Eds.) *Numerical Validation in Current Hardware Architectures*. Berlin: Springer.
108. Lamar, B.W. (1993). An improved branch and bound algorithm for minimum concave cost network flow problems. *Journal of Global Optimization*, *3*, 261–287.
109. Land, A.H., & Doig, A.G. (1960). An automatic method for solving discrete programming problems. *Econometrica*, *28*, 497–520.
110. Lebbah, Y. (2009). ICOS: A branch and bound based solver for rigorous global optimization. *Optimization Methods and Software*, *24*, 709–726.
111. Lebbah, Y., Michel, C., & Rueher, M. (2005). A rigorous global filtering algorithm for quadratic constraints. *Constraints*, *10*, 47–65.
112. Lebbah, Y., Michel, C., & Rueher, M. (2007). Using constraint techniques for a safe and fast implementation of optiMality-based reduction. In *Proceedings of the 2007 ACM symposium on Applied Computing* (pp. 326–331).
113. Lebbah, Y., Michel, C., Rueher, M., Daney, D., & Merlet, J.P. (2005). Efficient and safe global constraints for handling numerical constraint systems. *SIAM Journal on Numerical Analysis*, *42*, 2076–2097.
114. Lecoutre, C., Sais, L., Tabary, S., & Vidal, V. (2007). Recording and minimizing nogoods from restarts. *Journal on Satisfiability Boolean Modeling and Computation*, *1*, 147–167.
115. Leo, K., & Tack, G. (2015). Multi-Pass High-Level Presolving. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.
116. Lhomme, O. (1993). Consistency techniques for numeric CSPs. In *International Joint Conference on Artificial Intelligence*, (Vol. 93 pp. 232–238).
117. Lin, Y., & Schrage, L. (2009). The global solver in the LINDO API. *Optimization Methods and Software*, *24*, 657–668.
118. Lodwick, W.A. (1989). Constraint propagation, relational arithmetic in AI systems and mathematical programs. *Annals of Operations Research*, *21*, 143–148.
119. Lodwick, W.A. (1992). Preprocessing nonlinear constraints with applications to the pooling problem. *ORSA Journal on Computing*, *4*, 119–131.
120. Lynce, I., & Marques-Silva, J. (2003). Probing-based preprocessing techniques for propositional satisfiability. In *Proceedings of 15th ICTAI* (pp. 105–110).
121. Mahajan, A. (2010). Presolving Mixed-Integer Linear Programs, In Cochran, J.J., Cox, L.A., Keskinocak, P., Kharoufeh, J.P., & Smith, J.C. (Eds.) *Wiley Encyclopedia of Operations Research and Management Science*. New York: Wiley.
122. Mangasarian, O.L., & McLinden, L. (1985). Simple bounds for solutions of monotone complementarity problems and convex programs. *Mathematical Programming*, *32*, 32–40.
123. Margot, F. (2002). Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, *94*, 71–90.
124. Margot, F. (2003). Exploiting orbits in symmetric ILP. *Mathematical Programming*, *98*, 3–21.
125. Markót, M.C., & Schichl, H. (2014). Bound constrained interval global optimization in the COCONUT Environment. *Journal of Global Optimization*, *60*, 751–776.
126. Marques-Silva, J.P., & Sakallah, K.A. (1999). GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, *48*, 506–521.
127. Martin, A. (2001). General Mixed Integer Programming: Computational Issues for Branch-and-Cut Algorithms, In Jünger, M., & Naddef, D.s. (Eds.) *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions* (pp. 1–25). Berlin: Springer.
128. Martin, A., Möller, M., & Moritz, S. (2006). Mixed integer models for the stationary case of gas network optimization. *Mathematical programming*, *105*, 563–582.

129. Martin, P., & Shmoys, D.B. (1996). A new approach to computing optimal schedules for the job-shop scheduling problem, In Cunningham, H.W., McCormick, T.S., & Queyranne, M. (Eds.) *International Conference on Integer Programming and Combinatorial Optimization* (pp. 389–403). Berlin: Springer.
130. Mayer, G. (1995). Epsilon-inflation in verification algorithms. *Journal of Computational and Applied Mathematics*, *60*, 147–169.
131. McCormick, G.P. (1976). Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical Programming*, *10*, 147–175.
132. Messine, F. (2004). Deterministic global optimization using interval constraint propagation techniques. *RAIRO-Operations Research*, *38*, 277–293.
133. Mészáros, C.S., & Suhl, U.H. (2003). Advanced preprocessing techniques for linear and quadratic programming. *OR Spectrum*, *25*, 575–595.
134. Meyer, C.A., & Floudas, C.A. (2005). Convex envelopes for edge-concave functions. *Mathematical programming*, *103*, 207–224.
135. Misener, R., & Floudas, C.A. (2014). ANTIGONE: Algorithms for coNTinuous/Integer Global Optimization of Nonlinear Equations. *Journal of Global Optimization*, *59*, 503–526.
136. Mittelmann, H.D., & Pruessner, A. (2006). A server for automated performance analysis of benchmarking data. *Optimization Methods and Software*, *21*, 105–120.
137. Moore, R.E., Kearfott, R.B., & Cloud, M.J. (2009). Introduction to interval analysis. Siam Philadelphia.
138. Morrison, D.R., Jacobson, S.H., Sauppe, J.J., & Sewell, E.C. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, *19*, 79–102.
139. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., & Malik, S. (2001). Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th annual Design Automation Conference, pp. 530–535*.
140. Nannicini, G., Belotti, P., Lee, J., Linderoth, J., Margot, F., & Wächter, A. (2011). A probing algorithm for MINLP with failure prediction by SVM, In Achterberg, T., & Beck, J.C. (Eds.) *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (pp. 154–169). Berlin: Springer.
141. Nemhauser, G.L., Savelsbergh, M.P., & Sigismondi, G.C. (1994). MINTO, a mixed INTeger optimizer. *Operations Research Letters*, *15*, 47–58.
142. Neumaier, A. (1990). Interval methods for systems of equations Cambridge university press.
143. Neumaier, A. (1997). Molecular modeling of proteins and mathematical prediction of protein structure. *SIAM review*, *39*, 407–460.
144. Neumaier, A. (2004). Complete search in continuous global optimization and constraint satisfaction. *Acta numerica*, *13*, 271–369.
145. Neumaier, A., & Shcherbina, O. (2004). Safe bounds in linear and mixed-integer linear programming. *Mathematical Programming*, *99*, 283–296.
146. Ostrowski, J., Linderoth, J., Rossi, F., & Smriglio, S. (2011). Orbital branching. *Mathematical Programming*, *126*, 147–178.
147. O'Sullivan, B. (2010). Automated Modelling and Solving in Constraint Programming. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.
148. Pardalos, P.M., Chaovalitwongse, W., Iasemidis, L.D., Sackellares, J.C., Shiau, D., Carney, P.R., Prokopyev, O.A., & Yatsenko, V.A. (2004). Seizure warning algorithm based on optimization and nonlinear dynamics. *Mathematical Programming*, *101*, 365–385.
149. Princeton Library. http://www.gamsworld.org/performance/princetonlib/princetonlib.htm.
150. Prosser, P., Stergiou, K., & Walsh, T. (2000). Singleton consistencies, In Dechter, R. (Ed.) *Proceedings of 6th International Conference, CP 2000 Singapore* (pp. 353–368). Berlin: Springer.
151. Puranik, Y., & Sahinidis, N.V. Bounds tightening on optimality conditions for nonconvex box-constrained optimization. Journal of Global Optimization. doi:10.1007/s10898-016-0491-8.
152. Puranik, Y., & Sahinidis, N.V. Deletion presolve for accelerating infeasibility diagnosis in optimization models. INFORMS Journal on Computing (in review).
153. Rajagopalan, S., & Sahinidis, N.V. (2017). The pooling problem, In Terlaky, T., Anjos, M., & Ahmed, S. (Eds.) *Advances and Trends in Optimization with Engineering Applications, MOS-SIAM Book Series on Optimization*. Philadelphia: SIAM.
154. Régin, J.C. (2011). Milano Global constraints: A survey, In Van, P., & Hentenryck, M. (Eds.) *Hybrid optimization: The Ten Years of CPAIOR* (pp. 63–134). New York: Springer.
155. Rikun, A.D. (1997). A convex envelope formula for multilinear functions. *Journal of Global Optimization*, *10*, 425–437.
156. Ríos-Mercado, R.Z., & Borraz-Sánchez, C. (2015). Optimization problems in natural gas transportation systems: A state-of-the-art review. *Applied Energy*, *147*, 536–555.

157. Roy, T.J.V., & Wolsey, L.A. (1987). Solving mixed integer programming problems using automatic reformulation. *Operations Research*, *35*, 45–57.
158. Ryoo, H.S., & Sahinidis, N.V. (1995). Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, *19*, 551–566.
159. Ryoo, H.S., & Sahinidis, N.V. (1996). A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, *8*, 107–139.
160. Sahinidis, N.V. (2003). Global optimization and constraint satisfaction: The branch-and-reduce approach, In Bliek, C., Jermann, C., & Neumaier, A. (Eds.) *Global Optimization and Constraint Satisfaction, Lecture Notes in Computer Science*, (Vol. 2861 pp. 1–16). Berlin: Springer.
161. Sahinidis, N.V., & Tawarmalani, M. (2005). Accelerating branch-and-bound through a modeling language construct for relaxation-specific constraints. *Journal of Global Optimization*, *32*, 259–280.
162. Sam-Haroud, D., & Faltings, B. (1996). Consistency techniques for continuous constraints. *Constraints*, *1*, 85–118.
163. Sandholm, T., & Shields, R. (2006). Nogood learning for mixed integer programming. In *Workshop on Hybrid Methods and Branching Rules in Combinatorial Optimization, Montréal* (p. 138).
164. Savelsbergh, M.W.P. (1994). Preprocessing and probing for mixed integer programming problems. *ORSA Journal on Computing*, *6*, 445–454.
165. Schichl, H., Markót, M.C., & Neumaier, A. (2014). Exclusion regions for optimization problems. *Journal of Global Optimization*, *59*, 569–595.
166. Schichl, H., & Neumaier, A. (2004). Exclusion regions for systems of equations. *SIAM Journal on numerical analysis*, *42*, 383–408.
167. Schichl, H., & Neumaier, A. (2005). Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, *33*, 541–562.
168. Schichl, H., & Neumaier, A. (2006). Transposition theorems and qualification-free optiMality conditions. *SIAM Journal on Optimization*, *17*, 1035–1055.
169. Schöbel, A., & Scholz, D. (2010). The theoretical and empirical rate of convergence for geometric branch-and-bound methods. *Journal of Global Optimization*, *48*, 473–495.
170. Scholkopf, B., & Smola, A.J. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge: MIT press.
171. Sellmann, M. (2004). Theoretical foundations of CP-based Lagrangian relaxation, In Wallace, M. (Ed.) *Proceedings of 10th International Conference, CP 2004, Toronto, Canada* (pp. 634–647). Berlin: Springer.
172. Sewell, E.C., Sauppe, J.J., Morrison, D.R., Jacobson, S.H., & Kao, G.K. (2012). A BB&R algorithm for minimizing total tardiness on a single machine with sequence dependent setup times. *Journal of Global Optimization*, *54*, 791–812.
173. Shectman, J.P., & Sahinidis, N.V. (1998). A finite algorithm for global minimization of separable concave programs. *Journal of Global Optimization*, *12*, 1–36.
174. Sherali, H.D., & Adams, W.P. (1990). A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal of Discrete Mathematics*, *3*, 411–430.
175. Sherali, H.D., & Adams, W.P. (1994). A hierarchy of relaxations and convex hull characterizations for mixed- integer zero-one programming problems. *Discrete Applied Mathematics*, *52*(1), 83–106.
176. Sherali, H.D., & Adams, W.P. (1999). *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems, Nonconvex Optimization and its Applications* Vol. 31. Dordrecht: Kluwer Academic Publishers.
177. Sinha, M., Achenie, L.E.K., & Gani, R. (2003). Blanket wash solvent blend design using interval analysis. *Industrial & Engineering Chemistry Research*, *42*, 516–527.
178. Smith, A.B., & Sahinidis, N.V. (2009). Optimization techniques for phase retrieval based on single-crystal X-ray diffraction data, In Floudas, C.A., & Pardalos, P.M. (Eds.) *Encyclopedia of Optimization* (pp. 2858–2863). Boston: Springer.
179. Smith, E.M.B., & Pantelides, C.C. (1996). Global optimisation of general process models, In Grossmann, I.E. (Ed.) *Global Optimization in Engineering Design* (pp. 355–386). Boston: Kluwer Academic Publishers.
180. Stallman, R.M., & Sussman, G.J. (1977). Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial intelligence*, *9*, 135–196.
181. Stergiou, K. (2009). Heuristics for dynamically adapting propagation in constraint satisfaction problems. *AI Communications*, *22*, 125–141.
182. Sturtevant, N.R., Felner, A., Likhachev, M., & Ruml, W. (2012). Heuristic search comes of age. In *AAAI12: Proceedings of the 26th AAAI Conference on Artifical Intelligence*.

183. Tawarmalani, M., Richard, J.P., & Xiong, C. (2013). Explicit convex and concave envelopes through polyhedral subdivisions. *Mathematical Programming*, *138*, 531–577.
184. Tawarmalani, M., & Sahinidis, N.V. (2002). Convex extensions and convex envelopes of l.s.c. functions. *Mathematical Programming*, *93*, 247–263.
185. Tawarmalani, M., & Sahinidis, N.V. (2004). Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, *99*, 563–591.
186. Tawarmalani, M., & Sahinidis, N.V. (2005). A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, *103*, 225–249.
187. Thakur, L.S. (1990). Domain contraction in nonlinear programming: Minimizing a quadratic concave function over a polyhedron. *Mathematics of Operations Research*, *16*, 390–407.
188. Thorsteinsson, E.S., & Ottosson, G. (2002). Linear relaxations and reduced-cost based propagation of continuous variable subscripts. *Annals of operations research*, *115*, 15–29.
189. Tomlin, J.A., & Welch, J.S. (1983). Formal optimization of some reduced linear programming problems. *Mathematical programming*, *27*, 232–240.
190. Tomlin, L.A., & Welch, J.S. (1986). Finding duplicate rows in a linear programming model. *Operations Research Letters*, *5*, 7–11.
191. Torres, P., & Lopez, P. (2000). Overview and possible extensions of shaving techniques for job-shop problems. In *2nd International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems* (pp. 181–186). Paderborn: Springer.
192. van Beek, P., & Beek, T. (2006). Walsh Backtracking search algorithms, In Rossi, F., & van, P. (Eds.) *Handbook of Constraint Programming, chap. 4* (pp. 85–134). Amsterdam: Elsevier.
193. Van Hentenryck, P., Michel, L., & Deville, Y. (1997). *Numerica: A Modeling Language for Global Optimization*. Cambridge, MA: The MIT Press.
194. van Iwaarden, R.J. (1996). *An improved unconstrained global optimization algorithm*. Ph.D. thesis: University of Colorado at Denver.
195. Vandenbussche, D., & Nemhauser, G.L. (2005). A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Mathematical Programming*, *102*, 559–575.
196. Vandenbussche, D., & Nemhauser, G.L. (2005). A polyhedral study of nonconvex quadratic programs with box constraints. *Mathematical Programming*, *102*, 531–557.
197. Vu, X., Sam-Haroud, D., & Faltings, B. (2009). Enhancing numerical constraint propagation using multiple inclusion representations. *Annals of Mathematics and Artificial Intelligence*, *55*, 295–354.
198. Vu, X., Schichl, H., & Sam-Haroud, D. (2009). Interval propagation and search on directed acyclic graphs for numerical constraint solving. *Journal of Global Optimization*, *45*, 499–531.
199. Waltz, D. (1975). Understanding line drawings of scenes with shadows, In Winston, P.H. (Ed.) *The Pyschology of Computer Vision*. New York: McGraw-Hill.
200. Wechsung, A., Schaber, S.D., & Barton, P.I. (2014). The cluster problem revisited. *Journal of Global Optimization*, *58*, 429–438.
201. Zamora, J.M., & Grossmann, I.E. (1999). A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms. *Journal of Global Optimization*, *14*, 217–249.
202. Zorn, K., & Sahinidis, N.V. (2013). Global optimization of general nonconvex problems with intermediate bilinear substructures. *Optimization Methods and Software*, *29*, 442–462.