Epiphytic Trees: Relational Consistency Applied to Global Optimization Problems

Guilherme Alex Derenievicz$^{\text{orcid: 0000-0002-3970-1766}}$ and Fabiano Silva

Federal University of Paraná, Curitiba - PR, Brazil
{gaderenievicz,fabiano}@inf.ufpr.br

Abstract. Much effort has been spent to identify classes of CSPs in terms of the relationship between network structure and the amount of consistency that guarantees a backtrack-free solution. In this paper, we address Numerical Constrained global Optimization Problems (NCOPs) encoded as ternary networks, characterizing a class of such problems for which a combination of Generalized Arc-Consistency (GAC) and Relational Arc-Consistency (RAC) is sufficient to ensure a backtrack-free solution, called Epiphytic Trees. While GAC is a domain filtering technique, enforcing RAC creates new constraints in the network. Alternatively, we propose a branch and bound method to achieve a relaxed form of RAC, thus finding an approximation of the solution of NCOPs. We empirically show that Epiphytic Trees are relevant in practice. In addition, we extend this class to cover all ternary NCOPs, for which Strong Directional Relational k-Consistency ensures a backtrack-free solution.

1 Introduction

A *Constraint Satisfaction Problem* (CSP) consists of finding an assignment of values to a set of variables that satisfy a constraint network. Local consistency techniques play a central role in solving CSP, pruning values that surely do not constitute a solution of the problem. A wide range of local consistencies have been proposed for finite-domain CSPs, e.g., k-consistency [20], generalized arc-consistency (GAC) [32], hyper-consistency [26] and relational consistency [15].

Many efforts have been spent to identify classes of CSPs by linking the network structure to the level of local consistency that guarantees a backtrack-free solution, i.e., a search that solves the problem without encountering any conflict. Freuder [21] stated that binary networks with *width k* are solved in a backtrack-free manner if achieved strong k-consistency. Jégou [26] extended such results to non-binary networks, showing the relation between hyper-k-consistency and *hypergraph width*. Van Beek and Dechter [3] linked the tightness of a network to the level of relational consistency that guarantees a backtrack-free solution. Although CSP is generally NP-hard, such investigations may identify classes of polynomial problems, like tree-structured CSPs and Horn formulas [14].

In the 1980s, Dechter and Pearl [16] proposed *directional consistency* as a simpler way to enforce local consistency. In short, a variable ordering is used to enforce consistency in a specific direction. Thus, only relations between a variable and its predecessors must be analyzed. The classes of tractable CSPs previously introduced are naturally extended to their directional versions.

Most methods for solving finite-domain CSPs are not efficiently applicable to problems over continuous variables (Numerical CSPs). The well-known arc-consistency [44], for the sake of example, is generally uncomputable on continuous case [10, 13]. Many alternative methods have been proposed over the last three decades by combining classical CSP concepts with interval analysis [34]. In particular, due to the improvement of computer hardware and all the advancement of constraint propagation, interval methods have been applied on a wide range of Numerical Constrained global Optimization Problems (NCOPs) [9, 24, 28, 29, 38, 43]. Despite the great progress of interval techniques, optimizers from the mathematical programming community are generally more efficient than interval solvers [2, 36], although such optimizers are non-rigorous, i.e., they cannot guarantee the global optimality of the solution.

In this paper, we address NCOPs encoded as ternary networks with an objective function (actually a single variable) to be minimized. We characterize an important class of such problems for which *directional relational arc-consistency* is sufficient to ensure a backtrack-free solution (Sec. 3). We call such a class of *Epiphytic Trees*. In the spirit of Freuder [21] and Jégou [26], we define Epiphytic Trees based on structural properties of the constraint network.

Enforcing relational consistency may create new constraints in the network [14, 15], as opposed to domain filtering techniques (e.g. GAC) that only prune variables' domain without changing the network structure. Relational consistency has hardly been used in practice due to the high complexity to be achieved [8, 14]. For this reason, we implement an interval branch and bound method to enforce a relaxed form of this consistency, thus finding an approximation for the global minimum of NCOPs (Sec. 4). The goal of our method is to evaluate the proposed class and not to be compared with state-of-art optimizers, since most of advanced techniques of interval solvers are not considered in our implementation. Nonetheless, we are able to solve about 60% of a set of 130 instances proposed in the COCONUT suite [40] with a close approximation (Sec. 5). To our surprise, all tested instances have shown to be encoded as Epiphytic Trees.

To complete, in order to generalize Epiphytic Trees we extend this class to cover all ternary encoded NCOPs, proposing the natural extension of the *width* parameter [21] to ternary networks, namely *epiphytic width*, and proving that networks with epiphytic width k can be solved in a backtrack-free manner if achieved strong directional relational k-consistency (Sec. 6).

2 Background

A *constraint network* $\mathcal{R}$ is a triple $(X, D, \mathcal{C})$ where $X = \{x_1, \ldots, x_n\}$ is a set of variables with respective domains $D = \{D_1, \ldots, D_n\}$ and $\mathcal{C} = \{R_{C_1}, \ldots, R_{C_m}\}$ is a set of constraints such that $C_i = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$ is the *scope* of the relation $R_{C_i} \subseteq D_{i_1} \times \cdots \times D_{i_k}$, which defines a set of legal assignments to the variables of C_i. The *arity* of a constraint is the cardinality of its scope. A constraint network $\mathcal{R}$ is said to be *k-ary* if all its constraints have arity k or less. If the domains of the variables are continuous sets of real numbers, $\mathcal{R}$ is said a *numerical constraint network*. The problem of finding an assignment of values from domains of variables that satisfy the (numerical) constraint network is abbreviated to (N)CSP. Given a numerical constraint network $\mathcal{R}$ and an objective function $f : \mathbb{R}^n \mapsto \mathbb{R}$, a Numerical Constrained global Optimization Problem (NCOP) consists of finding a solution of $\mathcal{R}$ that minimizes f in $D_1 \times \cdots \times D_n$.

A *hypergraph* is a structure $\mathcal{H} = (V, E)$ where V is a finite set of vertices and $E \subseteq \{S \subseteq V \mid S \neq \emptyset\}$ is a set of edges. A *Berge-cycle* [7] is an alternating sequence of edges and vertices $(C_1, x_1, C_2, x_2, \ldots, C_m, x_m, C_{m+1} = C_1)$ such that $m \geq 2$ and $x_i \in (C_i \cap C_{i+1})$, for all $1 \leq i \leq m$. A hypergraph is said *Berge-acyclic* if it has no Berge-cycles. The *constraint hypergraph* is a structural representation of the network, where vertices represent variables and edges represent scopes of constraints. For abuse of notation, we will often use the terms *variable* and *vertex* as synonyms, as well as the terms *constraint* and *edge*. Furthermore, we represent the constraint network $\mathcal{R} = (X, D, \mathcal{C})$ by the hypergraph $\mathcal{H} = (X, C)$, where $C = \{C_i \mid R_{C_i} \in \mathcal{C}\}$.

Many local consistencies have been defined for non-binary networks [15, 20, 23, 26, 32]. These techniques prune values that surely do not constitute a solution of the problem. However, as opposed to global consistency, in general they cannot guarantee that a consistent instantiation of a subset of variables can be extended to all remaining variables while satisfying the whole network. In this paper, we focus on two main techniques: generalized arc-consistency [32] and relational arc-consistency [15], both local as only one constraint is considered at once.

Definition 1 (Generalized Arc-Consistency (GAC)).

- *A constraint R_{C_i} is GAC relative to $x_k \in C_i$ iff for every value $a_k \in D_k$ there exists an instantiation I of variables in $C_i \setminus \{x_k\}$ such that $I \cup \langle x_k = a_k \rangle$ satisfies R_{C_i}.*
- *A constraint R_{C_i} is (full) GAC iff it is GAC relative to all $x_k \in C_i$.*
- *A network $\mathcal{R}$ is (full) GAC iff every constraint of $\mathcal{R}$ is GAC.*
- *A network $\mathcal{R}$ is directional GAC, according to an ordering $b = (x_1, \ldots, x_n)$, iff every constraint R_{C_i} is GAC relative to its earliest variable in b.*

Definition 2 (Relational Arc-Consistency (RAC)).

- *A constraint R_{C_i} is RAC relative to $x_k \in C_i$ iff any consistent assignment to all variables of $C_i \setminus \{x_k\}$ has an extension to x_k that satisfies R_{C_i}.*
- *A constraint R_{C_i} is (full) RAC iff it is RAC relative to all $x_k \in C_i$.*
- *A network $\mathcal{R}$ is (full) RAC iff every constraint of $\mathcal{R}$ is RAC.*
- *A network $\mathcal{R}$ is directional RAC, according to an ordering $b = (x_1, \ldots, x_n)$, iff every constraint R_{C_i} is RAC relative to its latest variable in b.*

Dechter and Pear proposed directional consistency motivated by the fact that "full consistency is sometimes unnecessary if a solution is going to be generated by search along a fixed variable ordering" [14, p. 91]. In this work, we address

NCOPs by encoding the objective function $x_1 = f(\mathbf{x})$ and the constraint network $\mathcal{R}$ as a ternary NCSP, and performing a search along a fixed variable ordering *starting by* x_1. The idea behind this strategy is that if a consistent instantiation of the encoded network with initial (partial) assignment $\langle x_1 = \min D_1 \rangle$ is achieved by the search, then such solution minimizes f in $D_1 \times \cdots \times D_n$.

2.1 Interval Arithmetic

A *closed interval* $X = [\underline{x}, \overline{x}]$ is the set of real numbers $X = \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\}$, where $\underline{x}, \overline{x} \in \mathrm{I\!R} \cup \{-\infty, \infty\}$ are the *endpoints* of X. This definition is naturally extended for *open* and *half-open* intervals. We denote by $\mathbb{I}$ the set of all intervals in $\mathrm{I\!R}$. A *box* $B \in \mathbb{I}^n$ is a tuple of intervals. A box $B = (X_1, \ldots, X_n)$ is a refinement of the box $D = (Y_1, \ldots, Y_n)$, denoted by $B \subseteq D$, iff $X_i \subseteq Y_i$, for all $1 \leq i \leq n$.

Given two intervals $X, Y \in \mathbb{I}$, the *interval extension* of any binary operator $\circ$ well defined in $\mathrm{I\!R}$ is defined by (1).

$$X \circ Y = \{x \circ y \mid x \in X,\ y \in Y \text{ and } x \circ y \text{ is defined in } \mathrm{I\!R}\} \ . \tag{1}$$

The set (1) can be an interval, the empty set or a disconnected set of real numbers (union of intervals or a *multi-interval*). It is possible to compute $X \circ Y$, for all algebraic and the most common transcendentals functions, only analyzing the endpoints of X and Y [33], e. g., $[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$, $[\underline{x}, \overline{x}] \cdot [\underline{y}, \overline{y}] = [\min\{\underline{xy}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{xy}\}, \max\{\underline{xy}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{xy}\}]$, etc.

General factorable functions $f : \mathrm{I\!R}^n \mapsto \mathrm{I\!R}$ are also extended to intervals. An interval function $\mathcal{F} : \mathbb{I}^n \mapsto \mathbb{I}$ is an interval extension of $f : \mathrm{I\!R}^n \mapsto \mathrm{I\!R}$ if $\forall B \in \mathbb{I}^n : f[B] \subseteq \mathcal{F}(B)$, where $f[B]$ denotes the image of f under B, i. e. $f[B] = \{f(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n) \in B\}$. Generally, if the interval extension $\mathcal{F}$ is defined by the same expression of f, using the respective interval operators, then $\mathcal{F}(B)$ is a good estimate of $f[B]$. More specifically, if each variable occurs only once in the expression of f, then $\mathcal{F}(B)$ is exactly the image of this function [4]. The reason that an interval extension may overestimate the image of a real functional is that interval operators consider each occurrence of a variable as a different variable.

Example 1. The images of ${x_1}^2 - x_2$ and $x_1 x_1 - x_2$, under box $([-2, 2], [-1, 1])$ are both $[-1, 5]$. However, the interval extensions ${X_1}^2 - X_2$ and $X_1 X_1 - X_2$ computes, respectively, $[-2, 2]^2 - [-1, 1] = [-1, 5]$ and $[-2, 2] \cdot [-2, 2] - [-1, 1] = [-5, 5]$.

2.2 Interval Consistency

Since the late eighties, interval arithmetic has been used to deal with numerical constrained problems and several techniques have been extended to NCSP [18, 24, 25, 30, 35, 38, 39, 43]. For instance, GAC is easily achieved on ternary constraints of the form $x_1 = x_2 \circ_1 x_3$ by computing the intersection of each relevant domain with the respective *projection function*:

$$\texttt{GAC\_contractor}(x_1 = x_2 \circ_1 x_3) := \begin{cases} D_1 \leftarrow D_1 \cap (D_2 \circ_1 D_3) \\ D_2 \leftarrow D_2 \cap (D_1 \circ_2 D_3) \\ D_3 \leftarrow D_3 \cap (D_1 \circ_3 D_2) \end{cases} \tag{2}$$

where $\circ_2$ and $\circ_3$ maintain $(x_1 = x_2 \circ_1 x_3) \Leftrightarrow (x_2 = x_1 \circ_2 x_3) \Leftrightarrow (x_3 = x_1 \circ_3 x_2)$.

Strictly, the box generated by (2) is not complete due to the finite precision of machine numbers. Furthermore, domains may be disconnected sets of real numbers (union of intervals). In general, there is no consensus in literature about the performance of maintaining multi-interval representation [10, 18, 41]. Many relaxed consistencies have been proposed to deal with these problems, e.g., hull consistency [6] and box consistency [5], however, the strong property of GAC (the existence of local instantiation given any assignment of a single variable) is lost. Despite the efficiency of the GAC contractor, such procedure may never terminate when applied to the whole network[1] [10, 13], unless the hypergraph holds specific structural properties [11, 18] or a maximum precision is imposed.

While the GAC contractor shrinks the domain of variables, an RAC contractor must tighten the constraint on $C_i \setminus \{x_k\}$ (if such constraint does not exist, it must be added to the network). For example, the constraint $x_1 = x_2 + x_3$ is not RAC relative to x_1 under the box $([1, 3], [0, 2], [0, 2])$, as for $x_2 = x_3 = 0$ or $x_2 = x_3 = 2$ there is no consistent assignment to x_1. We may turn this constraint RAC by adding the constraints $x_2 + x_3 \geq 1$ and $x_2 + x_3 \leq 3$ to the network.

For binary constraints, GAC and RAC are identical [14].

2.3 Decomposition of Constraint Networks

Generally, in order to apply the GAC contractor on k-ary numerical constraints a procedure of decomposition that transforms the constraint into an equivalent set of ternary constraints is used [19]. Such procedure is efficient and solves the *dependence problem* (multiple occurrences of the same variable) and the *isolation problem* (the projection functions of (2) are hard to obtain if the constraint is a complex combination of many variables). However, the decomposition increases the *locality problem*, as shown in Example 2.

Example 2. Let $x_1(x_2 - x_1) = 0$ such that $D_1 = [0, 2]$ and $D_2 = [1, 2]$. With an auxiliary variable x_3 this constraint is decomposed into a network of two new constraints: $x_2 - x_1 = x_3$ and $x_1 x_3 = 0$. Contractor (2) is then repeatedly applied to both constraints until a fixed box is obtained, which is $([0, 2], [1, 2], [-1, 2])$. Although this box turns each decomposed constraint consistent, the original constraint is not GAC (for $x_1 = 0.5$, $\nexists x_2 \in D_2$ such that $x_1(x_2 - x_1) = 0$).

[1] As occur with the network $x_1 = x_2$ and $x_1 = 0.5 \cdot x_2$, where only the box $B = ([0, 0], [0, 0])$ turns both the constraints GAC but the contractor does an endless bisection on any initial arbitrary box.

Due to the locality problem, interval methods are composed by three main phases that are repeatedly applied until a solution is found: (i) *prune*: where contractors of interval consistency are applied; (ii) *local search*: where cheap search methods are used to detect if the current box contains a solution of the entire network (original problem), e.g., selecting the midpoint of the box or applying the Newton method; and (iii) *branch*: where a variable x is chosen and its domain is bisected to continue the search in a recursive fashion.

Likewise, a NCOP $(f, \mathcal{R})$ can be decomposed into a ternary network by adding a *root* variable $x_1 = f(\mathbf{x})$. For example, the instance (3) can be encoded as (4), where x_1 is the root variable that encodes the objective function.

$$\min \quad y_1^2 - y_2 \qquad \text{s.t.} \quad \{y_1 \leq 2y_2\} \tag{3}$$

$$\min \quad x_1 \qquad \text{s.t.} \quad \{x_1 = x_2 - y_2, x_2 = y_1^2, y_1 \leq 2y_2\} \tag{4}$$

3 Backtrack-Free Solution of Ternary Encoded NCOPs

We consider the problem of finding a solution of decomposed NCOPs by performing a backtrack-free search along a variable ordering starting by the *root* variable. For this, we classify ternary networks in three primary groups, based on the acyclic behavior of the hypergraph representation.

Let $\mathcal{R}_1$ be a network composed by constraints R_{C_1}, R_{C_2} and R_{C_3}, whose constraint hypergraph is represented in Fig. 1 by black edges, with root x_1.

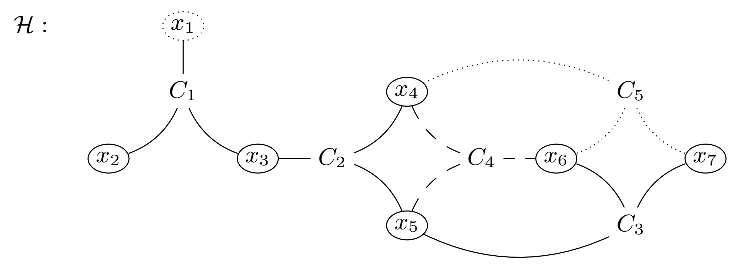

Fig. 1. Constraint hypergraph of networks $\mathcal{R}_1$ (composed by constraints R_{C_1}, R_{C_2} and R_{C_3}), $\mathcal{R}_2$ (composed by constraints $R_{C_1}, \ldots, R_{C_4}$) and $\mathcal{R}_3$ (all constraints).

Considering the initial partial instantiation $\langle x_1 = \min D_1 \rangle$ we want to propagate this value over all the network in a backtrack-free manner, which can be achieved if $\mathcal{R}_1$ is GAC [11, 18]. For instance, by constraint R_{C_1} the values of x_2 and x_3 are settled, as GAC ensures that for all $x_1 \in D_1$ there are $x_2 \in D_2$ and $x_3 \in D_3$ that satisfy R_{C_1} (if multiple values of x_2 and x_3 satisfy the constraint an instantiation can be chosen arbitrarily). This process continues for the entire network, processing constraints with one, and only one, already instantiated variable. A possible ordering of processed constraints for $\mathcal{R}_1$ is $(R_{C_1}, R_{C_2}, R_{C_3})$.

In general, such an ordering must have as its first constraint the one with the scope containing the root variable and ensure property (5).

$$|C_{p_i} \cap \bigcup_{j=1}^{i-1} C_{p_j}| < 2, \quad \text{for all } 1 < i \leq m \text{ in the ordering } (R_{C_{p_1}}, \ldots, R_{C_{p_m}}) \ . \tag{5}$$

Actually, an ordering d with these properties exists iff the constraint hypergraph is Berge-acyclic, because each constraint in d connects with predecessors in only one point (the common variable) [11]. Furthermore, given such a constraint ordering, directional GAC along d is sufficient to ensure this backtrack-free instantiation, instead of full GAC.

Next, let $\mathcal{R}_2$ be the network obtained by adding to $\mathcal{R}_1$ the dashed constraint R_{C_4} in Fig. 1. In this case, there are no guarantees that the previous instantiation is consistent with R_{C_4}. More than that, for this network an ordering ensuring (5) does not exist, which means GAC cannot guarantee the existence of a complete instantiation extending $\langle x_1 = \min D_{x_1} \rangle$.

We may consider another ordering $d' = (R_{C_1}, R_{C_2}, R_{C_4}, R_{C_3})$. Here, variables x_1, x_2, x_3, x_4 and x_5 can be safety instantiate under directional GAC. However, the next constraint R_{C_4} has two variables x_4 and x_5 already valued; in this way there is no guarantee that there exists $a_6 \in D_6$ satisfying the constraint. On the other hand, if R_{C_4} is RAC relative to x_6, instead of GAC, then the existence of $a_6 \in D_6$ satisfying R_{C_4} is guaranteed and the instantiation can be extended. Similarly, this strategy can be applied to R_{C_3}, instantiating the remaining variable x_7. The ordering d' ensures property (6).

$$|C_{p_i} \cap \bigcup_{j=1}^{i-1} C_{p_j}| < 3, \quad \text{for all } 1 < i \leq m \text{ in the ordering } (R_{C_{p_1}}, \ldots, R_{C_{p_m}}) \ . \tag{6}$$

In general, a network can be ordered according to (6) if its constraint hypergraph has a "partial acyclic" structure we call *Epiphytic Tree* (Sec. 3.1). Differently from the first example, where a Berge-acyclic hypergraph rarely occurs in practice, a ternary constraint network[2] seems to be usually represented by an Epiphytic Tree. In such an order, for a backtrack-free instantiation, constraints R_{C_i} sharing only one variable with predecessor constraints must satisfy directional GAC while those sharing two variables must satisfy directional RAC.

Finally, all possible orderings of constraints starting by R_{C_1} of the network $\mathcal{R}_3$ obtained by adding to $\mathcal{R}_2$ the constraint R_{C_5} (Fig. 1) do not ensure (5) nor (6), which means neither GAC nor RAC is sufficient to ensure a backtrack-free instantiation of this network. In other words, for all ordering $(R_{C_{p_1}}, \ldots, R_{C_{p_m}})$ such that $x_1 \in C_{p_1}$ there will be some C_{p_i} such that $|C_{p_i} \cap \bigcup_{j=1}^{i-1} C_{p_j}| = 3$.

3.1 Epiphytic Trees

Informally, an Epiphytic Tree (ET) is a hypergraph composed by an ordered set of trees (w.r.t. Berge-cycles), where each tree "sprouts" from its predecessor

[2] Obtained by encoding NCOPs.

through a single edge or from the "ground". The root of an ET is the root x_1 of the first tree. In this case, the ET is said to be *according to* x_1. For instance, on Fig. 2 the constraint hypergraph of the network $\mathcal{R}_2$ (Fig. 1) is properly drawn to show that this network is represented by an ET according to x_1 (directed edges indicate the ordering of the trees).

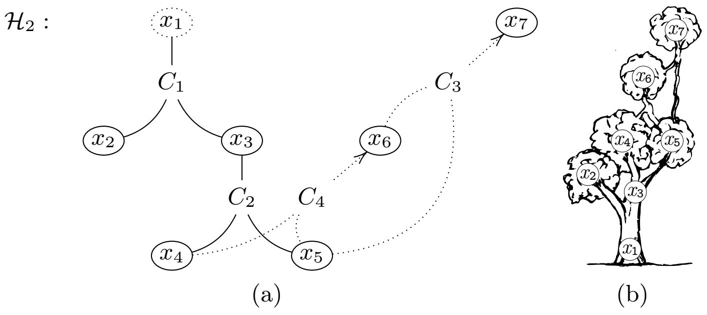

Fig. 2. (a) Epiphytic Tree according to x_1 composed by three trees and (b) an illustration of the botanical inspired nomenclature.

Definition 3 (Epiphytic Tree). *An Epiphytic Tree (ET) according to* x_1 *is a triple* $(\mathcal{A}, \Omega, t)$*, where* $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ *is an ordered set of disjointed hypergraphs* $\mathcal{A}_i = (V_i, E_i)$*,* Ω *is a set of edges not in* $\bigcup_{i=1}^{n} E_i$*, over the vertices of* $\bigcup_{i=1}^{n} V_i$*, and* $t : \Omega \mapsto V_\Omega$ *is a function that associates each edge* $C_i \in \Omega$ *with one vertex* $t(C_i) \in C_i$ *such that:*

1. $\mathcal{A}_1$ *is a rooted tree in* x_1 *(i.e., connected and Berge-acyclic).*
2. $\forall \mathcal{A}_{i>1} \in \mathcal{A}$ *: there is at most one* $C_i \in \Omega$ *such that* $t(C_i) \in V_i$ *and* $\mathcal{A}_i$ *is a rooted tree in* $t(C_i)$ *(or in any other vertex if such edge* C_i *does not exist);*
3. *if* $t(C_i) \in V_i$*, then* $C_i \setminus \{t(C_i)\} \subseteq \bigcup_{j=1}^{i-1} V_j$*.*

The epiphytic height of an ET is the cardinality of its Ω *set.*

Example 3. The network of Fig. 2 is represented by an ET according to x_1 given by $((\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3), \Omega, t)$, where $\mathcal{A}_1 = (\{x_1, \ldots, x_5\}, \{C_1, C_2\})$, $\mathcal{A}_2 = (\{x_6\}, \emptyset)$ and $\mathcal{A}_3 = (\{x_7\}, \emptyset)$ are disjointed, connected and Berge-acyclic hypergraphs, $\Omega = \{C_3, C_4\}$ and t is a function ensuring the conditions of Definition 3 such that $t(C_4) = x_6$ and $t(C_3) = x_7$. The epiphytic height of this ET is 2.

Given a network $\mathcal{R}$ and its ET $(\mathcal{A}, \Omega, t)$, we can efficiently obtain a constraint ordering d of $\mathcal{R}$ that ensures (6), constructing it from first to last element following the order $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$: for each hypergraph $\mathcal{A}_i$, put in d the edges (constraints) following the topological order of $\mathcal{A}_i$ from its root, and then the edge $C_{i+1} \in \Omega$ (if it exists). On the other hand, given a constraint ordering $(R_{C_1}, \ldots, R_{C_m})$ ensuring (6), we can construct an ET by putting in Ω the edges

C_i such that $|C_i \cap \bigcup_{j=1}^{i-1} C_j| = 2$ and defining each hypergraph $\mathcal{A}_i$ by the maximal connected component of the hypergraph induced by the remaining edges.

For any ET, the edges of Ω represent the constraints that must be RAC to ensure a backtrack-free solution. In this way, the smaller the cardinality of Ω, the "easier" to solve is the problem. A solution is found in polynomial time when the constraint hypergraph is a single tree, i.e., $\Omega = \emptyset$.

There are problems that cannot be represented by ETs. Given a network $\mathcal{R}$, if there exists a subset of constraints S such that all variables appearing in this subset occur in more than one constraint, then $\mathcal{R}$ does not have such representation. Theorem 1 formalizes this result by the equivalence between ETs and constraint orderings ensuring (6). For this, $\mathcal{H}' = (V', E')$ is said a *partial hypergraph* of $\mathcal{H} = (V, E)$ if $E' \subseteq E$ and $V' = \bigcup_{C_i \in E'} C_i$. We denote by $\mathcal{H} - C_i$ the partial hypergraph with edges set $E \setminus \{C_i\}$.

Theorem 1. *A constraint network $\mathcal{R}$ represented by a hypergraph $\mathcal{H}$ has a constraint ordering starting by C_1 ensuring (6) iff all partial hypergraph $\mathcal{H}'$ of $\mathcal{H}$ with more than one edge have at least one vertex $x' \notin C_1$ with degree 1.*

Proof. ($\Rightarrow$) The proof is by contradiction. Let d be a constraint ordering of $\mathcal{R}$ ensuring (6) and suppose that there is a partial hypergraph $\mathcal{H}'$ of $\mathcal{H}$ with more than one edge such that all vertices $x' \notin C_1$ have degree greater than 1. Let C_k be the latest edge of $\mathcal{H}'$ in d ($k > 1$). Since all vertices $x' \notin C_1$ have degree greater than 1, then all $x' \in C_k$ also belong to another edge of $\mathcal{H}'$. However, all other edges of $\mathcal{H}'$ have index less than k in d and, therefore, $|C_k \cap \bigcup_{j=1}^{k-1} C_j| = 3$. Hence d is not an ordering ensuring (6). ($\Leftarrow$) Let $\mathcal{R}$ be a network such that all partial hypergraphs $\mathcal{H}'$ of $\mathcal{H}$ with more than one edge have at least one vertex $x' \notin C_1$ with degree 1. Let $(\mathcal{H}_1, \dots, \mathcal{H}_m = \mathcal{H})$ be a sequence of partial hypergraphs such that $\mathcal{H}_1$ has only the edge C_1 and, for all $1 \leq i < m$, the hypergraph $\mathcal{H}_i$ is obtained by removing from $\mathcal{H}_{i+1}$ the edge C_{i+1} that contains a vertex $x \notin C_1$ with degree 1. The sequence of removed edges $(C_2, \dots, C_m)$ is such that all C_i have at least one vertex x that does not belong to any $C_{j<i}$ (x has degree 1 in $\mathcal{H}_i$). That is, $|C_i \cap \bigcup_{j=1}^{i-1} C_j| < 3$, for all $1 < i \leq m$. Hence the ordering $(R_{C_1}, \dots, R_{C_m})$ ensures (6). □

The proof of Theorem 1 motivates a method for finding an ordering according to x_1 of a network $\mathcal{R}$, building a sequence d from last to first element, choosing in each step an edge with a vertex of degree 1 in the partial constraint hypergraph obtained by removing processed constraints. Actually, this algorithm is the natural hypergraph extension of the `min-width` method presented in [14] (originally proposed by Freuder [21]), but restricted to always choose a vertex with degree 1. For ternary networks, this method is linear in the number of edges, since we can update in time $O(1)$ the degree of all vertices at each edge removal. Different choices of the picked vertex with degree 1 may result in distinct ETs. Considering the epiphytic height and the "meaning" of the edges in Ω are important tasks to obtain an accurate representation of the original system. In our implementation (Sec. 4), we choose edges in order to minimize the epiphytic height.

4 Achieving Relational Consistency

Enforcing relational consistency generally requires exponential time and space. Indeed, relational consistency can solve NP-Complete problems [14]. One of the first algorithms to achieve such consistency, as observed by Dechter and Rish [17], is the well-known Davis-Putnam procedure for CNF satisfiability. A first practical algorithm for high levels of relational consistency is introduced in [27]. However, "local consistency techniques that only filter domains like GAC tend to be more practical than those that alter the structure of the constraint hypergraph or the constraints' relations" [8, p. 1]. Enforcing directional RAC on ternary constraints adds new binary constraints to the network that changes the structure of the hypergraph[3]. Furthermore, adding new constraints is useful only if these constraints would be processed first, altering the given constraint ordering of the network. Thus, we avoid adding new constraints by applying a domain filtering algorithm that achieves a relaxed form of directional RAC.

4.1 Approximating Directional RAC

A constraint $R_{C_i} \equiv (x_1 = x_2 \circ x_3)$ is RAC relative to x_1 iff $D_1 \supseteq D_2 \circ D_3$, i.e., $\forall a_2 \in D_2, a_3 \in D_3, \exists a_1 \in D_1 \mid a_1 = a_2 \circ a_3$. Without loss of generalization, we can assume that R_{C_i} is GAC relative to x_1 ($D_1 \subseteq D_2 \circ D_3$). Thus, the required RAC condition is reduced to the equality $D_1 = D_2 \circ D_3$. If the constraint is not RAC (i.e., $D_1 \subset D_2 \circ D_3$) an attempt to achieve such consistency, alternatively to adding new constraints to the network, is narrowing the result set of $D_2 \circ D_3$. We can do this by narrowing D_2 or D_3, assured by the *inclusion isotonic*[4] of interval arithmetic. However, any pruning on such domains may exclude feasible instantiations, including the optimal one. We address this problem by using a branch and bound schema with an interval bisection procedure, since there exist infinite sub-boxes of D_2 and D_3 that are RAC under R_{C_i}. However, a tolerance ε must be considered in the equation $D_1 = D_2 \circ D_3$, as there is no guarantee that a bisection procedure can find RAC domains in tractable CPU time. One implication of this relaxation is that an instantiation of these variables may be infeasible in R_{C_i} by a factor of ε, what we call *ε-feasible*.

When applied to all constraints that must be RAC, this procedure is actually a variant of the usual interval branch and bound for NCOP. By combining this method with the natural evaluation of the network described in Sec. 3 we obtain an approximation of the global minimum of encoded optimization problems. Algorithm 1 describes such procedure. It follows a depth-first search applying the GAC contractor as a local procedure for pruning inconsistent values. If no empty domain is generated, the instantiation of the network starting by $\langle x_1 = \min D_{x_1} \rangle$ is attempt, constructing an ordered set I_Ω of non ε-feasible

[3] The same problem occurs with the results of Freuder [21] and Jégou [26]: achieving (hyper-)k-consistency creates new constraints of arity $k-1$ thus changing the width of the (hyper)graph.

[4] Given $B, B' \in \mathbb{I}^n$ and an interval function $\mathcal{F}$, if $B' \subseteq B$, then $\mathcal{F}(B') \subseteq \mathcal{F}(B)$.

constraints. These constraints must be narrowed to achieve RAC, what is done by the procedure `select_and_bisect`. Otherwise, the instantiation is a *quasi-solution* of the problem and the search of this branch terminates. Branches that surely have no solution better than the current optimal candidate are not processed by the search, because we added a dynamic constraint $x_1 < z^*$ to the network. A parameter of complexity of this method is the epiphytic height, as it bounds the size of I_Ω and only variables of constraints in this set are bisected.

Algorithm 1 Relaxed Directional RAC **In:** $(\mathcal{R} = (X, D, \mathcal{C}), x_1, \varepsilon)$ **Out:** z^*

```
push($D, queue$)
let $d$ be a constraint ordering of $\mathcal{R}$ starting by $x_1$ ensuring (6)
while $queue \neq \emptyset$ do
  $D \leftarrow$ pop($queue$)
  $D \leftarrow$ directional_GAC($X, D, \mathcal{C} \cup \{x_1 < z^*\}, d$)
  if $\forall D_i \in D : D_i \neq \emptyset$ then
    $(z, I_\Omega) \leftarrow$ attempt_instantiation($X, D, \mathcal{C}, d, \varepsilon$)
    if $I_\Omega = \emptyset$ then $z^* \leftarrow z$
    else
      $(D', D'') \leftarrow$ select_and_bisect($X, D, \mathcal{C}, d, I_\Omega$)
      push($D', queue$)
      push($D'', queue$)
    endif
  end if
end while
return $z^*$
```

The procedure `attempt_instantiation`$(X, D, \mathcal{C}, d, \varepsilon)$ tries to evaluate the network along the ordering d. Constraints R_{C_i} such that $C_i \notin \Omega$ are instantiated using the GAC contractor (2), as the network is directionally GAC along d. Otherwise, let $R'_{C_i} \equiv (x_1 = x_2 \circ x_3)$ be the constraint equivalent to R_{C_i}, $C_i \in \Omega$, such that $t(C_i) = x_1$. Since x_2 and x_3 are already instantiated along d, the procedure evaluates x_1 with the value of D_1 *closest to* $x_2 \circ x_3$. If the absolute error $|x_1 - (x_2 \circ x_3)|$ is greater than ε, then this instantiation is not ε-feasible, and R_{C_i} is put in I_Ω. The search continues for the next constraint of d whenever the partial instantiation is ε-feasible or not.

The procedure `select_and_bisect`$(X, D, \mathcal{C}, d, I_\Omega)$ selects a constraint $R_{C_i} \in I_\Omega$ and split the domain of its variables at the midpoint of D, returning two sub-boxes D' and D''. We provide a heuristic for selecting the constraint R_{C_i} by choosing the one with the highest index in I_Ω. Then, variables $x \in C_i \setminus \{t(C_i)\}$ are bisected and the sub-box that *minimizes* the required ε to turn R_{C_i} RAC is defined as the new box D' (and its complement as D'') to continue the search.

Although directional GAC would be enough if the constraints in Ω were RAC, as we are approximating this consistency by a factor of ε then high levels of consistency may be used to improve the branch and bound method. For instance, the procedure `directional_GAC` can be replaced by a full GAC contractor, narrowing the current box in a more effective way than using only the directional approach. In our experiments (Sec. 5), we implemented both the methods.

4.2 Comparison with the Usual Interval Branch and Bound

We compare our method with the usual interval branch and bound for NCOP. For this, we consider the up-to-date survey given on [1]. First, the general form of our procedure `attempt_instantiation` is the called `upper_bounding`, and provides a feasible instantiation of each branched sub-box. Local search methods are generally used to find such instantiations, e.g. applying the Newton method. Our strategy simply tries the natural evaluation of the network, as in [37] and [2], however, it minimizes the objective function by the initial instantiation $\langle x_1 = \min D_1 \rangle$. The main difference is that our approach may not find a solution in this branch, but if it does, then the solution minimizes the encoded objective function in the current box and no further searches in sub-boxes *of this branch* are needed. It is worth noting that others interval solvers [37, 42] also relax constraints on the evaluation phase.

Furthermore, both general interval branch and bound and our approach present the `select_and_bisect` procedure. The choice of which variable and slice of its domain will be the next branch of the search tree is crucial to the effectiveness of backtrack searches. Several heuristic have been proposed to date, e.g., choose the variable with the largest domain or use Smear-based schemas [12]. In this sense, our method provides a non-problem-specific heuristic for choice of variables on branch phase: right-sided variable of non ε-feasible constraints in Ω that maximizes the RAC approximation of the network. Such heuristic is an important contribution of this work, as finding local minima is a great strategy to reduce the search space in branch and bound algorithms.

To complete, on interval branch and bound methods an interval box may or may not contain a real solution of the constraint network. There are some techniques to assure the existence of such solution but, in general, the networks must satisfy some conditions. In our approach, the solution is always an instantiation of real numbers ε-feasible with the problem.

5 Experimental Results

In order to verify the codification of NCOPs as ETs, we executed algorithm 1 over an experimental set of 130 instances from COCONUT benchmark [40], which consists of academic and real life global optimization problems with industrial relevance. We considered only instances with numerical variables and operators $+$, $-$, $*$, $/$ and *pow*.

First, the experiments showed that Epiphytic Tree is a class of problems relevant in practice, since all tested instances are represented by this structure. Then, we considered a set of parameterized executions with ε varying from 10^{-4} to 10^{1} and a timeout of $7200s$. The method found ε-feasible solutions for 103 instances (about 79%). Using a full GAC contractor[5] instead of the directional one (see Sec. 4.1) this number increased to 108 instances (83%). Of those, 82 instances were solved with a global minimum closest to the best known with an absolute error less than 10 (74 instances using directional GAC). Figure 3 shows these results. Some instances with absolute error above 40 presents a relative error (the ratio of the absolute error to the global minimum) below 60%. The cluster of points in the 100% line are instances for which our method found a solution with value zero, while their global minimum are non-zero values.

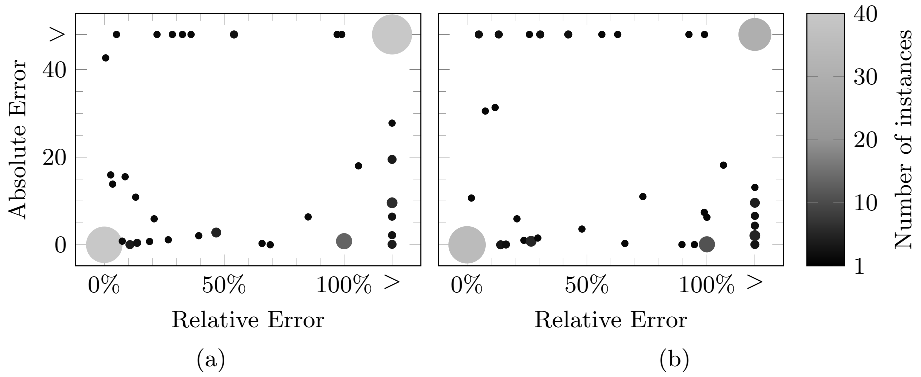

Fig. 3. Absolute error vs. relative error between the solution found and the best known of each instance, using (a) directional GAC and (b) full GAC. The circumference of each point increases according to the number of instances with similar error values.

It is worth remarking that we have not implemented many well-known techniques of current optimization solvers, such as *look-ahead* and *look-back* schemas, high levels of local consistency and more sophisticated local search. Hence, we cannot compare the CPU time of our method to that of state-of-art solvers. Nevertheless, in practice, the proposed method is able to find an approximation of the global minimum of encoded problems; thus, it can be used to improve optimization solvers in means of pre-processing, local search or upper bounding.

6 Extending the Epiphytic Tree Class

We showed that directional RAC is sufficient to solve structures with property (6). Now, we extend this study to hypergraphs that cannot be ordered ensuring such property. For this, we introduce a structural parameter of hypergraphs called *epiphytic width*, considering vertex orderings instead of edge orderings. Many definitions of hypergraph width have been proposed to characterize classes of CSPs. In particular, Gottlob [22] introduced the notion of *hypertree width*

[5] Using an AC-3 [31] style algorithm with an iteration counter to avoid looping.

defined over hypertree decompositions; such procedure is generally NP-hard. Alternatively, epiphytic width is a simpler definition being the natural extension of the graph width proposed by Freuder to binary CSPs [20].

Definition 4 (Epiphytic Width). *The epiphytic width of the vertex x_i in the ordering $b = (x_1, \ldots, x_n)$ is the number of edges $C_1, \ldots, C_k$ such that x_i is the latest variable of any $C_1, \ldots, C_k$ (w.r.t. b). The epiphytic width of the ordering b is the maximum epiphytic width over all x_i. The epiphytic width of $\mathcal{H}$ is the minimum epiphytic width over all vertex ordering starting by x_1.*

Definition 5 (k-Epiphytic Trees). *A hypergraph with epiphytic width k is said a k-Epiphytic Tree.*

Theorem 2 shows that k-Epiphytic Tree is a generalization of Epiphytic Tree.

Theorem 2. *Epiphytic Tree $\equiv$ 1-Epiphytic Tree.*

Proof. We proof this equivalence showing that it is possible to convert an edge ordering $d = (C_1, \ldots, C_m)$ ensuring (6) into a vertex ordering $b = (x_1, \ldots, x_n)$ with epiphytic width 1 such that $x_1 \in C_1$, and vice versa. ($\Rightarrow$) for $k = 1$ to m put in b the vertices of C_k not already in b, such that the last vertex placed is the one with degree 1 in the partial hypergraph with edges $\{C_1, \ldots, C_k\}$ (such vertex exists as d satisfies (6)). The first two vertices, if placed, have epiphytic width 0, because they are not the latest vertex of C_k; by the same reason, if they are already in b, then their epiphytic width will not change. On the other hand, the third vertex is being placed in b for the first time, because it has degree 1 in this sub-order. Since it is the latest vertex of C_k in b, then its epiphytic width is 1. ($\Leftarrow$) for $k = 1$ to n, mark x_k. If this turns out that all vertices of an edge C_i not in d are marked, put C_i in d. Each edge C_i placed in d have at least the vertex x_k of degree 1 in this sub-order, otherwise it should be already marked. Hence $|C_i \cap \bigcup_{j=1}^{i-1} C_j| < 3$. □

Definition 6 (Directional Relational k-Consistency). *Given a variable ordering $b = (x_1, \ldots, x_n)$, a network $\mathcal{R}$ is k-directionally relationally consistent iff for every subset of constraints $\{R_{C_1}, \ldots, R_{C_k}\}$ where the latest variable in any C_i is x_l, and for every $A \subseteq \{x_1, \ldots, x_{l-1}\}$, every consistent assignment to A can be extended to x_l while simultaneously satisfying all the relevant constraints in $\{R_{C_1}, \ldots, R_{C_k}\}$. A network is strongly directional relational $k-$consistent iff it is directional relational j-consistent for every $j \leq k$.*

Theorem 3. *A ternary constraint network $\mathcal{R}$ represented by a k-Epiphytic Tree can be solved in a backtrack-free manner if $\mathcal{R}$ is strongly directional relational k-consistent.*

Proof. Let $b = (x_1, \ldots, x_n)$ be a variable ordering of $\mathcal{R}$ with epiphytic width k in the constraint hypergraph. Thus, every variable x_l is the latest variable of at most k constraints $C_1, \ldots, C_{k'}$, $k' \leq k$. A consistent instantiation of $(x_1, \ldots, x_{l-1})$ can be extended to x_l iff every consistent instantiation of $\bigcup_{j=1}^{k'} C_j \setminus \{x_l\} \subseteq \{x_1, \ldots, x_{l-1}\}$ can be extended to x_l, which is achieved by strong directional relational k-consistency. □

Algorithm 1 can be naturally extended to deal with k-Epiphytic Trees. In general, instead of a single constraint in Ω for each tree of the structure there will be k constraints and all of them must be satisfied under ε.

7 Conclusion

We characterized a class of ternary networks called k-Epiphytic Trees, for which strong directional relational k-consistency is sufficient to ensure a backtrack-free solution. We empirically showed that NCOPs seems to be usually represented by 1-Epiphytic Trees. Despite the importance of these theoretical results, enforcing relational consistency is generally impractical. We proposed a branch and bound algorithm to achieving a relaxed form of directional RAC and evaluated such algorithm by executing it on a set of 130 instances from the COCONUT benchmark, of which about 60% were solved with a close approximation.

Many works have linked the network structure to the level of local consistency that guarantees a backtrack-free solution [21, 26]. However, to the best of our knowledge this is the first time that the relationship between optimization problems and relational consistency is addressed. Previously, Sam-Haroud and Faltings [39] found that $(3, 2)$-relational consistency[6] on *convex* ternary networks is a sufficient condition for global consistency. Besides the convexity restriction (which our result does not impose), the main goal of their work was construct a compact description of the complete solution space, instead of finding the optimum solution. A structure similar to Epiphytic Trees is the *cycle-cutset decomposition* used on finite-domain CSPs (see [14]), in which a consistent instantiation of the variables that constitute a cycle (our Ω set) is first addressed by some enumeration method to then instantiate the remaining variables in a backtrack-free manner. However, as the variable ordering initiates by the cycle-cutset variables, instead of the one representing the objective function, there are no guarantees that a solution found is optimal.

In future works, we want to deepen the relation between Epiphytic Trees and other network structures, such as hypergraph's width and the cycle-cutset schema. Also, we wish to study which Epiphytic Tree representing the network is better for the performance of the proposed algorithm. Furthermore, our method provides a non-problem-specific heuristic for interval branch and bound methods; thus, it can be used to improve optimization solvers in means of pre-processing, local search or upper bounding.

Acknowledgments. This work was supported by CAPES – Brazilian Federal Agency for Support and Evaluation of Graduate Education within the Ministry of Education of Brazil.

[6] A variant of 3-relational consistency where the consistency is considered with relation to two variables instead of one (see [15]).

References

1. Araya, I., Reyes, V.: Interval branch-and-bound algorithms for optimization and constraint satisfaction: a survey and prospects. Journal of Global Optimization 65(4), 837–866 (2016)
2. Araya, I., Trombettoni, G., Neveu, B., Chabert, G.: Upper bounding in inner regions for global optimization under inequality constraints. Journal of Global Optimization 60(2), 145–164 (2014)
3. van Beek, P., Dechter, R.: Constraint tightness and looseness versus local and global consistency. Journal of the ACM 44(4), 549–566 (1997)
4. Benhamou, F., Granvilliers, L.: Continuos and interval constraints. In: Rossi, P.F., van Beek, Walsh, T. (eds.) Handbook of Constraint Programming (Foundations of Artificial Intelligence), pp. 569–601. Elsevier, New York (2006)
5. Benhamou, F., McAllester, D., van Hentenryck, P.: Clp(intervals) revisited. In: International Symposium on Logic Programming. pp. 124–138. MIT Press, Cambridge (1994)
6. Benhamou, F., Older, W.J.: Applying interval arithmetic to real, integer, and boolean constraints. The Journal of Logic Programming 32(1), 1–24 (1997)
7. Berge, C.: Graphs and Hypergraphs. Elsevier Science, Oxford (1985)
8. Bessiere, C., Stergiou, K., Walsh, T.: Domain filtering consistencies for non-binary constraints. Artificial Intelligence 172(6), 800–822 (2008)
9. Bhurjee, A.K., Panda, G.: Efficient solution of interval optimization problem. Mathematical Methods of Operations Research 76(3), 273–288 (2012)
10. Chabert, G., Trombettoni, G., Neveu, B.: New Light on Arc Consistency over Continuous Domains. Tech. Rep. RR-5365, INRIA (2004)
11. Cohen, D.A., Jeavons, P.G.: The power of propagation: when gac is enough. Constraints 22(1), 3–23 (2017)
12. Csendes, T., Ratz, D.: Subdivision direction selection in interval methods for global optimization. SIAM Journal on Numerical Analysis 34(3), 922–938 (1997)
13. Davis, E.: Constraint propagation with interval labels. Artificial Intelligence 32(3), 281–331 (1987)
14. Dechter, R.: Constraint Processing. Morgan Kaufmann, San Francisco (2003)
15. Dechter, R., van Beek, P.: Local and global relational consistency. Theoretical Computer Science 173(1), 283–308 (1997)
16. Dechter, R., Pearl, J.: Network-based heuristics for constraint-satisfaction problems. Artificial Intelligence 34(1), 1–38 (1987)
17. Dechter, R., Rish, I.: Directional resolution: The davis-putnam procedure, revisited. In: 4th International Conference on Principles of Knowledge Representation and Reasoning. pp. 134–145. Morgan Kaufmann, San Francisco (1994)
18. Faltings, B.: Arc-consistency for continuous variables. Artificial Intelligence 65(2), 363–376 (1994)
19. Faltings, B., Gelle, E.M.: Local consistency for ternary numeric constraints. In: 15th International Joint Conference on Artificial Intelligence. pp. 392–397. Morgan Kaufmann, San Francisco (1997)
20. Freuder, E.C.: Synthesizing constraint expressions. Communications of the ACM 21(11), 958–966 (1978)
21. Freuder, E.C.: A sufficient condition for backtrack-free search. Journal of the ACM 29(1), 24–32 (1982)
22. Gottlob, G., Leone, N., Scarcello, F.: Hypertree decompositions and tractable queries. Journal of Computer and System Sciences 64(3), 579–627 (2002)

23. Gyssens, M.: On the complexity of join dependencies. ACM Transactions on Database Systems 11(1), 81–108 (1986)
24. Hansen, E., Walster, G.W.: Global optimization using interval analysis. Monographs and textbooks in pure and applied mathematics, Marcel Dekker, New York (2004)
25. van Hentenryck, P., McAllester, D., Kapur, D.: Solving polynomial systems using a branch and prune approach. SIAM Journal on Numerical Analysis 34(2), 797–827 (1997)
26. Jégou, P.: On the consistency of general constraint-satisfaction problems. In: 11th National Conference on Artificial Intelligence. pp. 114–119. AAAI Press, Washington (1993)
27. Karakashian, S., Woodward, R.J., Reeson, C., Choueiry, B.Y., Bessiere, C.: A first practical algorithm for high levels of relational consistency. In: 24th AAAI Conference on Artificial Intelligence. pp. 101–107. AAAI Press, California (2010)
28. Kearfott, R.B.: An interval branch and bound algorithm for bound constrained optimization problems. Journal of Global Optimization 2(3), 259–280 (1992)
29. Lebbah, Y., Michel, C., Rueher, M.: An efficient and safe framework for solving optimization problems. Journal of Computational and Applied Mathematics 199(2), 372–377 (2007)
30. Lhomme, O.: Consistency techniques for numeric csps. In: 13th International Joint Conference on Artificial Intelligence. pp. 232–238. Morgan Kaufmann, San Francisco (1993)
31. Mackworth, A.K.: Consistency in networks of relations. Artificial Intelligence 8(1), 99–118 (1977)
32. Mackworth, A.K.: On reading sketch maps. In: 5th International Joint Conference on Artificial Intelligence. pp. 598–606. Morgan Kaufmann, San Francisco (1977)
33. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to interval analysis. Society for Industrial and Applied Mathematics, Philadelphia (2009)
34. Moore, R.E.: Interval analysis. Prentice-Hall, New Jersey (1966)
35. Neumaier, A.: Interval Methods for Systems of Equations. Encyclopedia of Mathematics and its Applications, Cambridge University Press, Cambridge (1991)
36. Neumaier, A., Shcherbina, O., Huyer, W., Vinkó, T.: A comparison of complete global optimization solvers. Mathematical Programming 103(2), 335–356 (2005)
37. Ninin, J., Messine, F., Hansen, P.: A reliable affine relaxation method for global optimization. 4OR 13(3), 247–277 (2015)
38. Ratschek, H., Rokne, J.: New Computer Methods for Global Optimization. Halsted Press, New York (1988)
39. Sam-Haroud, D., Faltings, B.: Consistency techniques for continuous constraints. Constraints 1(1), 85–118 (1996)
40. Shcherbina, O., Neumaier, A., Sam-Haroud, D., Vu, X.H., Nguyen, T.V.: Benchmarking global optimization and constraint satisfaction codes. In: Bliek, C., Jermann, C., Neumaier, A. (eds.) 1st International Workshop on Global Constraint Optimization and Constraint Satisfaction. pp. 211–222. Springer, Berlin (2003)
41. Sidebottom, G., Havens, W.S.: Hierarchical arc consistency for disjoint real intervals in constraint logic programming. Computational Intelligence 8(4), 601–623 (1992)
42. Trombettoni, G., Araya, I., Neveu, B., Chabert, G.: Inner regions and interval linearizations for global optimization. In: 25th AAAI Conference on Artificial Intelligence. pp. 99–104. AAAI Press, California (2011)

43. Van Hentenryck, P.: Numerica: A modeling language for global optimization. In: 15th International Joint Conference on Artificial Intelligence. pp. 1642–1647. Morgan Kaufmann, San Francisco (1997)
44. Waltz, D.: Understanding line drawings of scenes with shadows. In: Winston, P.H. (ed.) The Psychology of Computer Vision, pp. 19–91. McGraw-Hill, New York (1975)