Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Constraint Satisfaction and Fair Multi-Objective Optimization Problems: Foundations, Complexity, and Islands of Tractability

Gianluigi Greco and **Francesco Scarcello**
University of Calabria, Italy
ggreco@mat.unical.it, scarcello@dimes.unical.it

Abstract

An extension of the CSP optimization framework tailored to identify fair solutions to instances involving multiple optimization functions is studied. Two settings are considered, based on the maximization of the minimum value over all the given functions (MAX-MIN approach) and on its lexicographical refinement where, over all solutions maximizing the minimum value, those maximizing the second minimum value are preferred, and so on, until all functions are considered (LEXMAX-MIN approach). For both settings, the complexity of computing an optimal solution is analyzed and the tractability frontier is charted for acyclic instances, w.r.t. the number and the domains of the functions to be optimized. Larger islands of tractability are then identified via a novel structural approach, based on a notion of guard that is designed to deal with the interactions among constraint scopes and optimization functions.

1 Introduction

CSP optimization frameworks extend the basic setting of constraint satisfaction by supporting the definition of an objective function over the space of all possible solutions (see, e.g., [Rossi *et al.*, 2006; Bistarelli *et al.*, 1999]). In some cases, however, more than just one function has to be optimized, so that some form of *multi-objective* optimization comes into play. For instance, in a configuration problem, the customer might have different functions to optimize, including the price and her preferences over the various features of the product.

Two forms of multi-objective optimization have been already considered in the CSP literature, giving rise to *lexicographic* (see, e.g., [Freuder *et al.*, 2010; Greco and Scarcello, 2011]) and *Pareto* (see, e.g., [Torrens and Faltings, 2002]) optimization settings. By abstracting from their peculiarities, these approaches share the idea of looking at the optimization problem from a "global" perspective, by tolerating poor values over some functions provided they are compensated by excellent values over the others. There are several application domains, however, where this perspective is not desirable and a "fair" approach to multi-objective optimization would be more appropriate. A noticeable example comes from *fair allocation* problems (see, e.g., [Bezáková and Dani, 2005]) and, more generally, from applications in multi-agent systems, where different agents have different—possibly contrasting—objective functions to optimize and the goal is to compute a *socially desirable* solution [Brandt *et al.*, 2012]. In fact, there are different possible notions of fairness, which may be more or less appropriate depending on the specific application. In this paper, we focus on the MAX-MIN approach, where fairness means *maximizing* the value associated with the *least* satisfied function/agent/goal.

Example 1.1. Consider the left part of Figure 1, which shows a setting where four indivisible and non-sharable resources $R_1, ..., R_4$ have to be allocated to three agents a_1, a_2, and a_3. Each agent reports not only which resources are of interest to her (represented as the edges incident on the agent), but also a degree of preference over them (represented as edge weights). Agent preferences are additive over the various resources, and we assume that no agent can get more than two resources. Then, the goal is to compute a fair allocation, meaning an allocation where the total weight of the resources assigned to the least satisfied agent is maximized.[1]

For instance, the least satisfied agent in the allocation θ_1 in Figure 1 is a_3. In fact, θ_1 is preferable to θ_2 according to this perspective, even though the overall value achieved with θ_2 (5+9+1) is greater than the one achieved in θ_1 (5+4+3). $\triangleleft$

A few approaches for fair CSP optimization have already been proposed in the literature (see, e.g., [Snow and Freuder, 1990; Dubois and Fortemps, 1999; Bouveret and Lematre, 2009]), where in particular each function can be defined over one constraint or one variable only. In fact, a general model was missing, which is capable to handle the large variety of optimization functions considered in standard CSP optimization [Rossi *et al.*, 2006; Bistarelli *et al.*, 1999]. Moreover, the complexity issues arising with fair optimization and the identification of *islands of tractability*, i.e., of classes of instances that are efficiently solvable, have been unexplored. Our goal with the paper is precisely to fill this gap. In particular,

► We define a MAX-MIN framework with powerful modeling capabilities, where the goal is to maximize the objective function getting the minimum value, and where such

[1]This is known as (a variant of) the Santa Claus problem [Bansal and Sviridenko, 2006], with Santa's goal being to distribute presents in a way that the least lucky kid is as happy as possible.

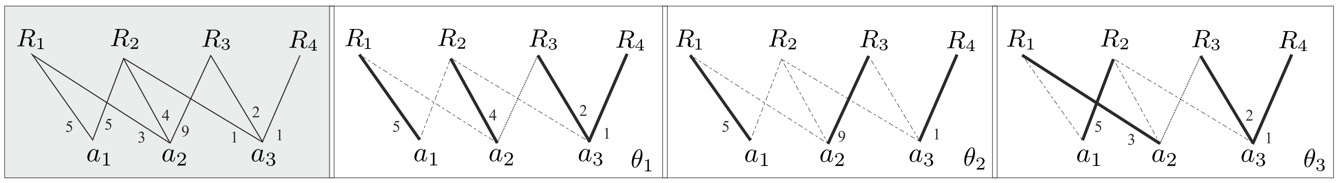

Figure 1: The 'fair" allocation problem in Example 1.1, and three solutions for it (θ_1, θ_2, and θ_3).

functions can involve an arbitrary number of constraints and are defined via suitable algebraic operators (as in standard constraint optimization). In order to get a higher level of fairness, we also consider the LEXMAX-MIN framework where, over all solutions maximizing the minimum value, we look for those maximizing the second minimum value and so on, in a lexicographical MAX-MIN perspective.

► Since (fair) constraint optimization is NP-hard in general, we first focus on the class of acyclic instances, which forms a well-known island of tractability for plain CSPs. It turns out that acyclicity does not suffice alone to guarantee tractability in our settings. Motivated by the bad news, we then look for tractable classes via restrictions on the number of optimization functions involved and/or the size of their domains, and chart the tractability frontier w.r.t. them.

► Finally, we propose a more powerful method to identify larger islands of tractability where the "structure" of the optimization functions is taken into account together with the structure of the underlying CSP instance. In particular, we define the notion of guarded instances and show that, for such nearly-acyclic instances, optimal (LEX)MAX-MIN solutions can be efficiently computed without restrictions on the number of optimization functions or on their domains.

2 Formal Framework

Constraint Satisfaction (see,e.g.,[Dechter, 2003]). Let Var be a set of variables and let $\mathcal{U}$ be a set of constants. We use the logic-based characterization of CSPs [Kolaitis and Vardi, 2000; Scarcello *et al.*, 2008]. Therefore, a CSP instance is viewed as a pair (Φ, DB), where DB is a *constraint database*, i.e., a finite set of ground atoms of the form $r_i(c_1, ..., c_k)$ with $c_j \in \mathcal{U}$, for each $j \in \{1, ..., k\}$, and where Φ is a *constraint formula*, i.e., a conjunction of atoms $r_1(\mathbf{u_1}) \wedge \cdots \wedge r_m(\mathbf{u_m})$ such that $\mathbf{u_1}, ..., \mathbf{u_m}$ are lists of terms, that is, variables in Var or constants in $\mathcal{U}$. A *solution* to the instance (Φ, DB) is a substitution $\theta : \mathcal{X} \mapsto \mathcal{U}$ such that $\mathcal{X} = Var$ and $r_j(\theta(\mathbf{u_j})) \in$ DB, for each atom $r_j(\mathbf{u_j})$ in Φ.[2] The set of all solutions to (Φ, DB) is denoted by Φ^{DB}. A substitution $\theta : \mathcal{X} \mapsto \mathcal{U}$ is *partial* if it is undefined on some variable, i.e., $\mathcal{X} \subset Var$. We often denote $\theta : \mathcal{X} \mapsto \mathcal{U}$ extensively, i.e., as the set of all pairs of the form X/c, where $c = \theta(X)$, for each $X \in \mathcal{X}$.

Example 2.1. The setting of Example 1.1 can be formalized as a CSP instance (Φ, DB) over a domain $\mathcal{U} = \{a_1, a_2, a_3\}$ of constants corresponding to the agents plus the fresh constant '-', and over a set $Var = \{R_1, R_2, R_3, R_4\}$ of variables corresponding to the resources. In particular, let $U_j \subseteq \mathcal{U}$ be the set of all agents requiring R_j plus the constant '-' (no agent), for each $j \in \{1, ..., 4\}$. Then, (Φ, DB) is such that:

[2]As usual, $\theta(\mathbf{u_j})$ denotes the list $(\theta(\mathbf{u_1}), \ldots, \theta(\mathbf{u_m}))$.

$\Phi = \ req_1(R_1, R_2) \wedge req_2(R_1, R_2, R_3) \wedge req_3(R_2, R_3, R_4)$; and

$\mathrm{DB} = \{req_1(a_i, a_j) | \{a_i, a_j\} \in U_1 \times U_2\} \cup$
$\{req_2(a_i, a_j, a_k) | \{a_i, a_j, a_k\} \in U_1 \times U_2 \times U_3 \setminus \{a_2, a_2, a_2\}\} \cup$
$\{req_3(a_i, a_j, a_k) | \{a_i, a_j, a_k\} \in U_2 \times U_3 \times U_4 \setminus \{a_3, a_3, a_3\}\}.$

where req_i in Φ is defined over the variables/resources that are required by agent a_i, for each $i \in \{1, 2, 3\}$.

Note that in any solution θ, each resource can be mapped via θ to one agent chosen among those that are actually requiring it, and no agent can get more than two resources, with the constant '-' meaning that the resource is not allocated.

Consider now the four substitutions $\theta_1 = \{R_1/a_1, R_2/a_2, R_3/a_3, R_4/a_3\}$, $\theta_2 = \{R_1/a_1, R_2/\text{-}, R_3/a_2, R_4/a_3\}$, $\theta_3 = \{R_1/a_2, R_2/a_1, R_3/a_3, R_4/a_3\}$, and $\theta_4 = \{R_1/a_1, R_2/a_3, R_3/a_3, R_4/a_3\}$. Note that θ_1, θ_2, and θ_3 are three solutions (corresponding to the allocations in Figure 1), while θ_4 is not a solution because agent a_3 gets three resources. $\lhd$

MAX-MIN and LEXMAX-MIN Optimization. The basic setting for constraint satisfaction is now extended by considering a MAX-MIN optimization framework defined on top of a set of valuation functions over the possible substitutions.

A *valuation function* $\mathcal{F}$ is a tuple $\langle w, \oplus \rangle$ where $w : \bar{X} \times \mathcal{U} \mapsto \mathbb{R}$, with $\bar{X} \subseteq Var$, is a real valued function over variable assignments and where $\oplus$ is a commutative, associative, and closed binary operator over a set $\mathcal{D} \subseteq \mathbb{R}$ of real numbers that includes the identity element $\perp$ w.r.t. $\oplus$ (e.g., 0 w.r.t. $+$, 1 w.r.t. $\times$, etc.). We assume also that $\oplus$ is distributive over the standard 'max' operator for real numbers. The set $\bar{X}$ is called the domain of $\mathcal{F}$, and it will be denoted by $\mathrm{dom}(\mathcal{F})$. For a (possibly partial) substitution $\theta : \mathcal{X} \mapsto \mathcal{U}$, define

$\mathcal{F}(\theta) = \bigoplus_{\{X/u \in \theta | X \in \bar{X}\}} w(X, u)$, if $\mathcal{X} \cap \bar{X} \neq \emptyset$; and

$\mathcal{F}(\theta) = \perp$, otherwise (i.e., if $\mathcal{X} \cap \bar{X} = \emptyset$).

Example 2.2. In our running example, each agent a_i can be equipped with a valuation function $\mathcal{F}_i = \langle w_i, + \rangle$ whose domain $\mathrm{dom}(\mathcal{F}_i) = \{R_j \mid a_i \in U_j\}$ consists of all the resources/variables that are desired by agent a_i. Moreover, w_i is such that $w_i(R_j/c) = 0$, for each $c \neq a_i$, while $w_i(R_j/a_i)$ is the worth for agent a_i getting the resource R_j, i.e., the weight associated with the edge between a_i and R_j in Figure 1. For instance, we have $\mathcal{F}_1(\theta_1) = w_1(R_1/a_1) = 5$, while $\mathcal{F}_3(\theta_1) = w_3(R_3/a_3) + w_3(R_4/a_3) = 2 + 1 = 3$. $\lhd$

Valuation functions are now combined to look for solutions maximizing the minimum value achieved over them.

Definition 2.3. Let (Φ, DB) be a CSP instance, let L be a set of valuation functions, and consider the following value:

$$\textsc{Max-Min-Val}(\Phi, \mathrm{DB}, L) = \max_{\theta' \in \Phi^{\mathrm{DB}}} \min_{\mathcal{F} \in L} \mathcal{F}(\theta').$$

A substitution $\theta \in \Phi^{\mathrm{DB}}$ is a MAX-MIN solution to (Φ, DB) w.r.t. L if $\min_{\mathcal{F} \in L} \mathcal{F}(\theta) = \textsc{Max-Min-Val}(\Phi, \mathrm{DB}, L)$. $\square$

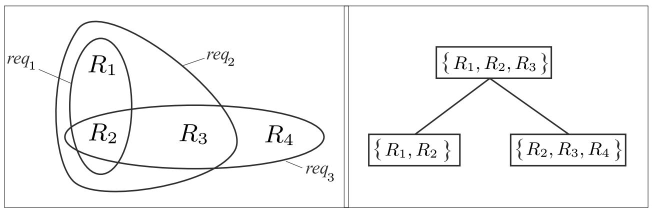

Figure 2: The hypergraph in Example 3.1, and a join tree for it.

Example 2.4. By looking again at Figure 1, we can check that $\min_{i\in\{1,2,3\}} \mathcal{F}_i(\theta_1) = \min_{i\in\{1,2,3\}} \mathcal{F}_i(\theta_3) = 3$, while $\min_{i\in\{1,2,3\}} \mathcal{F}_i(\theta_2) = 1$. In fact, it can be seen that θ_1 and θ_3 are both MAX-MIN solutions. $\lhd$

By definition, MAX-MIN solutions are sensible to the minimum satisfaction level only. A lexicographic refinement is hence considered next to achieve a higher level of fairness.

Let L be a set of valuation functions. For any $\theta \in \Phi^{\mathrm{DB}}$, we denote by $L(\theta)$ the non-decreasingly sorted vector containing the value $\mathcal{F}(\theta)$, for each $\mathcal{F} \in L$. Such vectors will be compared via the binary operator '$\succ$': For two vectors $(v_1, ..., v_m)$ and $(v'_1, ..., v'_m)$, $(v_1, ..., v_m) \succ (v'_1, ..., v'_m)$ holds if there is an integer $j \in \{0, ..., m-1\}$ such that $v_i = v'_i$, for each $1 \leq i \leq j$, and $v_{j+1} > v'_{j+1}$. Moreover, for a set of vectors $\mathcal{S}$, we define $\max \mathcal{S}$ as the vector $\bar{v} \in \mathcal{S}$ such that $\bar{v} \succ \bar{v}'$, for each $\bar{v}' \in \mathcal{S}$.

Definition 2.5. Let (Φ, DB) be a CSP instance, and let L be a set of valuation functions. A substitution $\theta \in \Phi^{\mathrm{DB}}$ is a LEXMAX-MIN solution to (Φ, DB) w.r.t. L if $L(\theta) = \max\{L(\theta') \mid \theta' \in \Phi^{\mathrm{DB}}\}$. $\square$

Example 2.6. Let $L = \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$ be the set of the valuation functions in our running example, and note that $L(\theta_1) = (3, 4, 5)$ and $L(\theta_3) = (3, 3, 5)$. Therefore, $L(\theta_1) \succ L(\theta_3)$ holds. In fact, θ_1 is a LEXMAX-MIN solution. $\lhd$

Problems Definition and Complexity Setting. In the paper, we analyze the complexity of the problem of computing MAX-MIN (resp., LEXMAX-MIN) solutions to CSP instances, hereinafter referred to as problem MAX-MIN-CSP (resp., LEXMAX-MIN-CSP), for short. The problem receives as input a triple (Φ, DB, L), where (Φ, DB) is a CSP instance and L is a set of valuation functions. In the analysis, we assume that functions are listed in the input, and that the underlying binary operators are polynomial-time computable.

Towards a finer grained analysis, different classes of sets of valuation functions are considered. Indeed, depending on the symbols $\mathrm{D} = \{1, \mathrm{k}, \infty\}$ and $\mathrm{F} = \{1, \mathrm{h}, \infty\}$, nine scenarios MAX-MIN-CSP$[\mathrm{D}, \mathrm{F}]$ (resp., LEXMAX-MIN-CSP$[\mathrm{D}, \mathrm{F}]$) emerge, where the problem is restricted as follows:

- $\mathrm{D} \in \{1, \mathrm{k}\}$, with $\mathrm{k} > 1$ being a fixed natural number, stands for the fact that the problem is restricted to sets of formulas L such that $\max_{\mathcal{F}\in L} |\mathrm{dom}(\mathcal{F})| \leq \mathrm{D}$. Moreover, $\mathrm{D} = \infty$ means that no restriction is considered on the size of the domains.
- $\mathrm{F} \in \{1, \mathrm{h}\}$, with $\mathrm{h} > 1$ being a fixed natural number, stands for the fact that the problem is restricted to sets L such that $|L| \leq \mathrm{h}$. Moreover, $\mathrm{F} = \infty$ means that no restriction is considered on the number of functions.

3 Acyclic CSP Instances

Since solving CSPs—and hence the above extensions—is an NP-hard problem in general, much research has been spent to identify classes of instances over which solutions can efficiently be computed. An approach that is often adopted is to focus on instances (Φ, DB) whose formulas enjoy some desirable "structural" properties. In particular, constraint interactions in Φ can be formalized via the hypergraph $\mathcal{H}(\Phi) = (V, H)$, where $V = \mathit{Var}$ and where, for each atom occurring in the constraint formula Φ, the set H of hyperedges contains a hyperedge including all its variables. Then, a structural property often considered for such hypergraphs is acyclicity.

A (hyper)graph $\mathcal{H}$ is *acyclic*[3] iff it has a join tree [Bernstein and Goodman, 1981]. A *join tree* $\mathcal{JT}(\mathcal{H})$ for a hypergraph $\mathcal{H}$ is a tree whose vertices are the (hyper)edges of $\mathcal{H}$ such that, whenever the same node X occurs in two (hyper)edges h_1 and h_2 of $\mathcal{H}$, then X occurs in each vertex on the unique path linking h_1 and h_2 in $\mathcal{JT}(\mathcal{H})$.

Example 3.1. The hypergraph $\mathcal{H}(\Phi)$ for the formula Φ in Example 2.1 is reported in Figure 2. Note that $\mathcal{H}(\Phi)$ is acyclic, as it is witnessed by the join tree reported in the figure. $\lhd$

3.1 Bad News from Acyclic Instances

It is well-known that *acyclic* CSP instances, i.e., instances whose associated hypergraphs are acyclic, can be solved in polynomial time (see, e.g., [Gottlob *et al.*, 2000]). Hence, the natural question comes into play about whether this result still holds for MAX-MIN-CSP and LEXMAX-MIN-CSP.

The answer for the case where there is just one valuation function immediately derives from results in the literature. Indeed, in this case, MAX-MIN optimization is immaterial (even in the lexicographic variant) and the setting reduces to standard constraint optimization over acyclic instances.

Theorem 3.2 (see [Gottlob *et al.*, 2009]). *On classes of acyclic instances,* LEXMAX-MIN-CSP$[\infty, 1]$ *is in* P.

The above tractability result is however tight. Indeed, we next show that intractability emerges in the problem MAX-MIN-CSP$[\infty, 2]$, which turns out to be *weakly*-NP-hard.

Theorem 3.3. MAX-MIN-CSP$[\infty, 2]$ *is weakly-*NP*-hard, even when restricted to classes of acyclic instances.*

Proof Sketch. Given a multi-set $S = \{s_1, s_2, ..., s_n\}$ of integers, let us build a CSP instance (Φ, DB) over $\mathit{Var} = \{X_1, ..., X_n\} \cup \{Y_1, ..., Y_n\}$ and $\mathcal{U} = \{\mathit{in}, \mathit{out}\}$ as follows. The formula Φ is $r(X_1, Y_1) \wedge ... \wedge r(X_n, Y_n)$, while the database DB precisely contains the two ground atoms $r(\mathit{in}, \mathit{out})$ and $r(\mathit{out}, \mathit{in})$. Moreover, consider the valuation functions $\mathcal{F}_1 = \langle w_1, + \rangle$ and $\mathcal{F}_2 = \langle w_2, + \rangle$ such that $\mathrm{dom}(\mathcal{F}_1) = \{X_1, ..., X_n\}$ and $\mathrm{dom}(\mathcal{F}_2) = \{Y_1, ..., Y_n\}$. In particular, for each $s_i \in S$, we have $w_1(X_i/\mathit{in}) = s_i$, $w_1(X_i/\mathit{out}) = 0$, $w_2(Y_i/\mathit{in}) = s_i$, and $w_2(Y_i/\mathit{out}) = 0$.

Now, note that MAX-MIN-VAL$(\Phi, \mathrm{DB}, \{\mathcal{F}_1, \mathcal{F}_2\}) = \frac{1}{2}\sum_{i=1}^{n} s_i$ holds if and only if we can partition S in two multi-sets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of those in S_2. The result follows as this latter problem is weakly-NP-hard [Garey and Johnson, 1990]. $\square$

[3]Actually, there are different notions of hypergraph acyclicity. We use the most liberal one, known as α-acyclicity [Fagin, 1983].

Note that the above result does not preclude the existence of *pseudo-polynomial* algorithms for MAX-MIN-CSP$[\infty, 2]$. In fact, we next describe an algorithm for (LEX)MAX-MIN-CSP whose running time is polynomial when input numbers are taken in unary notation.The result applies to the more general case of instances with any bounded number of functions.

Theorem 3.4. *On classes of acyclic instances,* LEXMAX-MIN-CSP$[\infty, \mathrm{h}]$ *is feasible in pseudo-polynomial time.*

Proof Idea. Let (Φ, DB, L) be an acyclic instance, with $L = \{\mathcal{F}_1, ..., \mathcal{F}_h\}$. We compute an optimal answer (if any) in pseudo-polynomial time, by using as an oracle the algorithm in [Greco and Scarcello, 2011] that, given an additional vector $(v_1, .., v_h)$ of real numbers, decides in pseudo-polynomial time whether there is a solution $\theta \in \Phi^{\mathrm{DB}}$ with $\mathcal{F}_i(\theta) = v_i$, for each $i \in \{1, ..., h\}$. Let us name this algorithm $\mathcal{A}$.

We first check whether $\Phi^{\mathrm{DB}} \neq \emptyset$ in polynomial time (as $\mathcal{H}(\Phi)$ is acyclic). If the above check is negative, then we return the empty output and halt. Otherwise, we compute an upper bound of the maximum possible vector of values, by computing the maximum value M_i for each valuation function $\mathcal{F}_i$, $\leq i \leq h$ (over the solutions in Φ^{DB}). This is feasible in polynomial time, by Theorem 3.2. Then, starting from the upper bound vector obtained by non-decreasingly sorting these maximum values, we enumerate (going downward) the pseudo-polynomially many vectors $\bar{v} = (v_1, ..., v_h)$ that may be associated with solutions and, for each of them, we use $\mathcal{A}$ to check whether there actually exists some $\theta \in \Phi^{\mathrm{DB}}$ such that $L(\theta) = \bar{v}$. This way, we identify $\bar{v}^* = \max\{L(\theta) \mid \theta \in \Phi^{\mathrm{DB}}\}$. Then, θ is easily computed by a final loop exploiting standard self-reducibility methods and arguments, again using algorithm $\mathcal{A}$ to identify the correct variable assignments leading to the desired vector of values $\bar{v}^*$. □

We leave the section by showing that the "symmetric" scenario is intractable, too—this time in the (usual) strong sense.

Theorem 3.5. MAX-MIN-CSP$[2, \infty]$ *is* NP*-hard, even on classes of acyclic instances.*

Proof Sketch. Let $\mathcal{E} = \{1, ..., m\}$ be a set of m distinct elements (w.l.o.g., natural numbers), and let $\mathcal{S} = \{s_1, ..., s_n\}$ be a set of subsets of $\mathcal{E}$. A *set packing* (for $\mathcal{S}$) is a set $\mathcal{S}' \subseteq \mathcal{S}$ such that $s_i \cap s_j = \emptyset$, for each pair $s_i, s_j \in \mathcal{S}'$. Computing the set packing having the largest possible cardinality is a well-known NP-hard problem [Garey and Johnson, 1990].

Based on $\mathcal{E}$ and $\mathcal{S}$, we define the domain $\mathcal{U} = \{in, out\} \cup \{to_0, to_1, to_2, ..., to_n\} \cup \{0, 1, ..., n\}$ of constants and the set $Var = \{X_1, .., X_m\} \cup \{W_0, W_1, ..., W_n\} \cup \{Y_{i,e} \mid s_i \in \mathcal{S}, e \in s_i\}$ of variables. Moreover, for each set $s_i = \{e_1, ..., e_{n_i}\} \in \mathcal{S}$ where $e_1 < e_2 < ... < e_{n_i}$, we denote by ℓ_i the list $Y_{i,e_1}, ..., Y_{i,e_{n_i}}$, and by $c_{i,in}$ (resp., $c_{i,out}$) the list consisting of n_i repetitions of the constant *in* (resp., *out*).

Then, we build the (acyclic) formula Φ such that

$$\Phi = \bigwedge_{e \in \mathcal{E}} map(X_e) \wedge \bigwedge_{s_i \in \mathcal{S}} select(W_{i-1}, W_i, \ell(s_i)),$$

and the database DB such that:

- The ground atoms $map(to_0), ..., map(to_n)$ are in DB;
- For each set $s_i \in \mathcal{S}$ and each natural number $count \in \{0, 1, ..., i-1\}$, the ground atoms $select(count, count+1, c_{i,in})$ and $select(count, count, c_{i,out})$ are in DB;
- No further ground atom is in DB.

Before equipping (Φ, DB) with a set of functions, let us point out the main property of the reduction. A substitution $\theta : Var \mapsto \mathcal{U}$ is said *legal* if for each $e \in \mathcal{E}$, $\theta(X_e) = to_i$ holds if, and only if, $\theta(Y_{i,e}) = in$ holds. Moreover, let $\mathtt{sp}(\theta)$ be the set $\{s_i \in \mathcal{S} \mid \theta(W_i) = \theta(W_{i-1}) + 1\}$. It can be checked that if θ is a legal solution, then $\mathtt{sp}(\theta)$ is a set packing whose cardinality is $\theta(W_n)$. Conversely, if $\mathcal{S}'$ is a set packing, then there is a legal solution θ with $\mathtt{sp}(\theta) = \mathcal{S}'$.

Now, for each $e \in \mathcal{E}$ and each set $s_i \in \mathcal{S}$ with $e \in s_i$, we define the functions $\mathcal{F}_{i,e} = \langle w_{i,e}, + \rangle$ and $\bar{\mathcal{F}}_{i,e} = \langle \bar{w}_{i,e}, + \rangle$ with $\mathrm{dom}(\mathcal{F}_{i,e}) = \mathrm{dom}(\bar{\mathcal{F}}_{i,e}) = \{X_e, Y_{i,e}\}$ and such that:

- $w_{i,e}(X_e/to_j) = n$, for each $j \neq i$; $w_{i,e}(X_e/to_i) = 0$; $w_{i,e}(Y_{i,e}/in) = n$; $w_{i,e}(Y_{i,e}/out) = 0$.
- $\bar{w}_{i,e}(X_e/to_j) = 0$, for each $j \neq i$; $\bar{w}_{i,e}(X_e/to_i) = n$; $\bar{w}_{i,e}(Y_{i,e}/in) = 0$; $\bar{w}_{i,e}(Y_{i,e}/out) = n$.

Consider also the function $\mathcal{F}_n = \langle w_n, + \rangle$ with $\mathrm{dom}(\mathcal{F}) = \{W_n\}$ and $w_n(W_n/i) = i$, for each $i \in \{0, 1, ..., n\}$.

Note that for each solution θ, θ is legal $\Leftrightarrow \min_{i,e} \mathcal{F}_{i,e}(\theta) = \min_{i,e} \bar{\mathcal{F}}_{i,e}(\theta) > 0$ (in fact, n). Hence, MAX-MIN solutions correspond to legal ones. Moreover, let L be the set with all the above functions, and let θ be a legal solution. Then, $\min_{\mathcal{F} \in L} \mathcal{F}(\theta) = \mathcal{F}_n(\theta) = \theta(W_n) \leq n$. So, θ is a MAX-MIN solution if, and only if, θ is legal and $\theta(W_n)$ is the maximum possible value over all legal solutions (hence, the corresponding set packing has the largest possible cardinality). □

4 Islands of Tractability

The source of complexity emerged in the above NP-hardness proofs ultimately lies in the interactions that occur between the atoms in Φ and the valuations functions, which are in fact not apparent from the structure of $\mathcal{H}(\Phi)$. Therefore, the natural question comes into play about whether limiting this interaction can lead to identify islands of tractability.

4.1 Revisiting the Acyclic Case

A positive answer to the above question can be given when the domain of each function is defined over one variable only.

Theorem 4.1. *On classes of acyclic instances,* LEXMAX-MIN-CSP$[1, \infty]$ *is in* P.

Proof Idea. Let (Φ, DB) be an acyclic instance, and let $L = \{\mathcal{F}_1, ..., \mathcal{F}_m\}$ be a set of valuation functions, where $|\mathrm{dom}(\mathcal{F}_i)| = 1$, for each $i \in \{1, ..., m\}$. Let T be a join tree of $\mathcal{H}(\Phi)$ (which can be computed in linear time [Gottlob *et al.*, 2001]), and let us root it at an arbitrary vertex r. Recall that each vertex v in T corresponds to a hyperedge of $\mathcal{H}(\Phi)$ and, hence, (w.l.o.g.) to an atom a_v in Φ, and define (initially) rel_v as the set of partial substitutions θ_v that are in a_v^{DB}, i.e., that are solutions to the CSP restricted to the atom a_v.

Let us fix some notations. For any substitution $\theta : \mathcal{X} \mapsto \mathcal{U}$, define $L^{\downarrow}(\theta)$ as the projection of the sorted vector $L(\theta)$ to the components of those functions $\mathcal{F} \in L$ with $\mathrm{dom}(\mathcal{F}) \subseteq \mathcal{X}$. For a set of substitutions $\mathcal{S}$ with domain $\mathcal{X}$, we say that $\theta \in \mathcal{S}$ is $\succ$-maximal in $\mathcal{S}$ if $L^{\downarrow}(\theta) = \max\{L^{\downarrow}(\theta') \mid \theta' \in \mathcal{S}\}$.

Moreover, for the vectors of real numbers $(v_1, ..., v_h)$ and $(w_1, ..., w_k)$, denote by $(v_1, ..., v_h) \odot (w_1, ..., w_k)$ the non-decreasingly sorted vector including the components from both vectors. It can be shown that a useful property of this operator is that it distributes over $\max$: If $\mathcal{V}_1$ and $\mathcal{V}_2$ are two sets of non-decreasingly ordered vectors, then $\max\{\bar{v}_1 \odot \bar{v}_2 \mid \bar{v}_1 \in \mathcal{V}_1, v_2 \in \mathcal{V}_2\} = \max \mathcal{V}_1 \odot \max \mathcal{V}_2$.

In order to compute a LEXMAX-MIN solution to (Φ, DB) w.r.t. L, we traverse T from the leaves to the root (according to a topological ordering): At each vertex v, for each substitution $\theta_v \in rel_v$ and for each child c of v in T, we focus on those substitutions $\bar{\theta}_c$ in rel_c that *conform* with θ_v, denoted by $\theta_v \approx \bar{\theta}_c$, that is, for each variable X that the domains of $\bar{\theta}_c$ and θ_v have in common, it must be the case that $\bar{\theta}_c(X) = \theta_v(X)$. If there is no substitution in rel_c conforming with θ_v, then we conclude that θ_v cannot be extended to a solution in Φ^{DB} and it is removed from rel_v. Otherwise, θ_v is enlarged by adding to it any $\succ$-maximal substitution θ_c in the set of substitutions from rel_c conforming with θ_v. Note that

$$L^{\downarrow}(\theta_c \cup \theta_v) = L^{\downarrow}(\theta_c \setminus \theta_v) \odot L^{\downarrow}(\theta_v). \quad (1)$$

Eventually, when the root r is reached, any substitution $\succ$-maximal in rel_r is returned as the output.

It is easy to see that the proposed algorithm runs in polynomial time. The formal proof of correctness is based on a structural induction over the join tree T, whose underlying idea is the following. Consider a vertex v and a substitution $\theta_v \in rel_v$, and assume inductively that, for each child c_i of v, each substitution $\theta_i \in rel_{c_i}$ is an optimal one for the problem induced by the variables occurring in the subtree of T rooted at c_i, and where the variables occurring in c_i are fixed according to θ_i. Recall that T is a join tree and thus the domains of the substitutions in the children of v can overlap only on the variables they have in common with v. Therefore, the same holds for the unary valuation functions. Then, by Eq. (1) and the fact that $\odot$ distributes over $\max$, the composition of θ_v with the best substitutions conforming with θ_v over all the children of v in T provides the best solution for the larger subproblem including the variables occurring in the atom associated with v, fixed according to θ_v. After this step, rel_v contains one best partial substitution for the subtree rooted at v, for each partial substitution for the variables in v. □

4.2 Guards and Tractable Acyclic Instances

Our goal is now to generalize the above result, by defining some specific notion of LEXMAX-MIN CSP structure where such interactions are considered and exploited in a suitable way, in order to identify larger islands of tractability.

The first step is to take under control the variables in the domain of valuation functions. To this end, we adopt a method inspired by the characterization of acyclic instances in terms of *guarded formulas*, that is, the fragment of positive existential first-order queries where the free variables of every subformula occur together in some atom, called its *guard*.

Let $\mathcal{H} = (V, H)$ be a hypergraph, and let W and $\{X, Y\}$ be sets of nodes. A $[W]$-path from X to Y is a sequence $X = X_0, \ldots, X_\ell = Y$ of nodes such that, for each $i \in \{0, ..., \ell\text{-}1\}$, $\{X_i X_{i+1}\} \subseteq (h_i \setminus W)$, for some hyperedge $h_i \in H$. We say that C is $[W]$-connected if $\forall X, Y \in C$ there is a

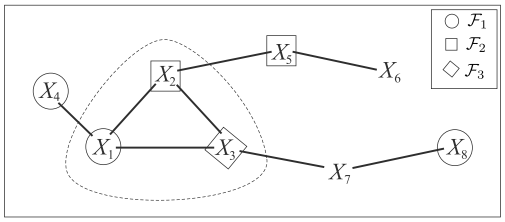

Figure 3: The (hyper)graph in Example 4.3

$[W]$-path from X to Y. A $[W]$-component (of $\mathcal{H}$) is a maximal $[W]$-connected non-empty set of nodes $C \subseteq (V \setminus W)$.

Definition 4.2. Let Φ be a constraint formula, let L be a set of valuation functions, and let $\mathcal{F} \in L$. A set W of nodes of $\mathcal{H}(\Phi)$ (i.e., variables in Φ) is a *guard* for $\mathcal{F}$ (w.r.t. L) if every $[W]$-component C of $\mathcal{H}(\Phi)$ with $C \cap \mathrm{dom}(\mathcal{F}) \neq \emptyset$ is such that $C \cap \mathrm{dom}(\mathcal{F}') = \emptyset$, for each $\mathcal{F}' \in L$ with $\mathcal{F}' \neq \mathcal{F}$. □

Intuitively, a guard W for $\mathcal{F}$ may directly cover some variables of $\mathrm{dom}(\mathcal{F})$, which will not appear in any $[W]$-component, and separates the remaining variables from the variables occurring in the other valuation functions.

Example 4.3. Consider the (hyper)graph $\mathcal{H}(\Phi)$ shown in Figure 3 (by assuming, for simplicity, that variables have numerical domains), and the valuation functions $\mathcal{F}_1, \mathcal{F}_2$, and $\mathcal{F}_3$ such that $\mathcal{F}_1(\theta) = \theta(X_1) + \theta(X_4) + \theta(X_8)$, $\mathcal{F}_2(\theta) = \theta(X_2) \times \theta(X_5)$, and $\mathcal{F}_3(\theta) = 5 \times \theta(X_3)$. In this example, the set $\{X_1, X_2, X_3\}$ is a guard for $\mathcal{F}_i$, $\forall i \in \{1, 2, 3\}$. ◁

The second step is to deal with the intricacy of the CSP, measured in terms of "degree of cyclicity" of the associated hypergraph. In fact, we next provide tractability results for LEXMAX-MIN-CSP$[\infty, \infty]$ over classes that are strictly more general than those considered in the previous section. To this end, recall (see, e.g., [Gottlob *et al.*, 2000]) that a *structural decomposition method* Ψ can be viewed as a function associating each CSP instance (Φ, DB) with a set $\Psi(\Phi, \mathrm{DB})$ of *equivalent* acyclic CSP instances, that is, $\bar{\Phi}^{\bar{\mathrm{DB}}} = \Phi^{\mathrm{DB}}$, for each $(\bar{\Phi}, \bar{\mathrm{DB}}) \in \Psi(\Phi, \mathrm{DB})$.

Definition 4.4. Let Ψ be a structural decomposition method and let $I = (\Phi, \mathrm{DB}, L)$ be an instance of (LEX)MAX-MIN-CSP. A CSP instance $(\bar{\Phi}, \bar{\mathrm{DB}}) \in \Psi(\Phi, \mathrm{DB})$ is a $[\Psi, L]$-*decomposition* (of I) if, for each $\mathcal{F} \in L$, there is a hyperedge W of $\mathcal{H}(\bar{\Phi})$ that is a guard for $\mathcal{F}$. Whenever such a decomposition exists, we say that I is *guarded via* Ψ. □

As a "baseline" method, let Ψ_a be the acyclic method returning the instance itself when this is acyclic, and $\emptyset$ otherwise. Then, the following is immediate by Definition 4.4.

Fact 4.5. *Let I be any acyclic* LEXMAX-MIN-CSP$[\mathrm{D}, \mathrm{F}]$ *instance, with* $\mathrm{D} = 1$ *or* $\mathrm{F} = 1$. *Then, I is guarded via* Ψ_a.

Therefore, Theorem 4.1 can be seen now as singling out a class of guarded tractable instances. We next generalize this result to more powerful structural methods (using Theorem 4.1 as a technical tool), by showing that guardedness is sufficient to ensure tractability without any further restriction.

Theorem 4.6. *Given any instance* (Φ, DB, L) *with a* $[\Psi, L]$-*decomposition, a solution of* LEXMAX-MIN-CSP *(if any) can be computed in polynomial time.*

Proof Idea. Let $(\bar{\Phi}, \bar{\mathrm{DB}})$ be the given $[\Psi, L]$-decomposition of (Φ, DB, L), with $L = \{\mathcal{F}_1, ..., \mathcal{F}_m\}$. We build in polynomial time an instance $(\Phi', \mathrm{DB}', L')$ as follows. For each function $\mathcal{F}_i$, $1 \le i \le m$, chose one atom $r_i(\mathbf{u_i})$ of $\bar{\Phi}$ such that its variables are a guard for $\mathcal{F}_i$. Such an atom exists by definition of $[\Psi, L]$-decomposition. The constraint formula Φ' is the same as $\bar{\Phi}$ but, for each each $\mathcal{F}_i$, $1 \le i \le m$, it contains a fresh atom $r'_i(\mathbf{u_i}, X_i)$ having the same terms as the guard r_i plus an additional fresh variable X_i. Correspondingly, DB' is the same as $\bar{\mathrm{DB}}$ but, for each $i \in \{1, ..., m\}$ and partial substitution θ such that $r_i(\theta(\mathbf{u_i})) \in \bar{\mathrm{DB}}$, we have the ground atom $r'_i(\theta(\mathbf{u_i}), v^*_\theta)$ in DB', where v^*_θ is the optimal value of the LEXMAX-MIN-CSP$[\infty, 1]$ (acyclic) CSP instance $(\bar{\Phi}, \bar{\mathrm{DB}})$ equipped only with function $\mathcal{F}_i$ and where the variables in $\mathbf{u_i}$ are fixed to $\theta(\mathbf{u_i})$. Note that these values, which form the domain of variable X_i in the new instance, can be computed in polynomial time by Theorem 3.2. Finally, let $L' = \{\mathcal{F}'_1, ..., \mathcal{F}'_m\}$, where $\mathcal{F}'_i = (w_i, +)$, $1 \le i \le m$, is a unary function with $\mathrm{dom}(\mathcal{F}'_i) = X_i$ and $w_i(X_i/c) = c$.

Note that new instance is acyclic, too. Moreover, $(\Phi', \mathrm{DB}', L')$ and (Φ, DB, L) are equivalent since, by the guardedness, the substitutions of X_i stored in the ground atoms of DB' encode all possible values of $\mathcal{F}'_i$ that may be associated with solutions of Φ^{DB}. The result then follows as $(\Phi', \mathrm{DB}', L')$ can be solved in P, by Theorem 4.1. □

Well-known decompositions methods that are tractable, in that any of their output instances may be computed in polynomial time, are the bounded treewidth (tw) [Robertson and Seymour, 1984] and the bounded hypertree width (htw) methods [Gottlob *et al.*, 2002]. Let k be any fixed natural number, representing intuitively the maximum degree of cyclicity we want to deal with. Then, $\Psi_{tw\text{-}k}$ returns the set of all equivalent acyclic instances where each atom contains $k-1$ variables at most, while $\Psi_{hw\text{-}k}$ returns the set of all the equivalent acyclic instances whose hypergraphs are width-k hypertree decompositions in normal form of the given instance. Basically, in $\Psi_{hw\text{-}k}$, we consider instances whose atoms are subproblems comprising at most k constraints of the original instance. We refer the interested reader to [Gottlob *et al.*, 2002], for further background on $\Psi_{hw\text{-}k}$.

Example 4.7. Consider again Figure 3. The acyclic instance whose hypergraph comprises the hyperedge $\{X_1, X_2, X_3\}$ is a $[\Psi_{tw\text{-}2}, L]$-decomposition, as the additional hyperedge uses $2+1$ variables and is a guard for the functions. In fact, it is also a $[\Psi_{htw\text{-}2}, L]$-decomposition, as the set $\{X_1, X_2, X_3\}$ can be covered by 2 (binary) original constraints. ◁

As a further example, we point out that the case of acyclic instances with a bounded number of valuation functions with bounded domains is generalized by the class of guarded bounded hypertree-width instances. This can be shown by noticing that $h \times k$ variables (hence, constraints) are sufficient to cover the union of all the domains of the given functions.

Fact 4.8. *Let I be any acyclic* LEXMAX-MIN-CSP[h, k] *instance. Then, I is guarded via $\Psi_{hw\text{-}(h \times k+1)}$.*

Turning to computational aspects, we observe that while computing an arbitrary decomposition according to the above

[D, F]	1	h	∞
1	in P [▷3.2]	in P [▷4.1]	in P [4.1]
k	in P [▷3.2]	in P [4.8+4.10]	NP-hard [3.5]
∞	in P [3.2]	weaky-NP-hard [3.3]	NP-hard [▷3.5]

Table 1: Tractability frontier for acyclic instances.

methods is tractable, in order to solve (LEX)MAX-MIN-CSP instances, we should be able to filter, out of exponentially many possible candidates, a decomposition with all the guards needed to take care of the valuation functions. Of course, a naïve enumeration approach is not feasible here, and a more involved approach is needed to get tractability.

Theorem 4.9. *Deciding whether an instance is guarded via $\Psi_{tw\text{-}k}$ (resp., $\Psi_{hw\text{-}k}$) is feasible in polynomial time.*

Proof Idea. We consider the more general case of $\Psi_{hw\text{-}k}$. We use the notion of weighted hypertree decompositions [Scarcello *et al.*, 2007], with a suitable function ξ weighing any normal form width-k decomposition, usually represented through a labeled join tree, called decomposition tree. The function ξ is such that, whenever an optimal decomposition has $-|L|$ as its (minimum) weight, the given instance is guarded via $\Psi_{hw\text{-}k}$. Moreover, such an optimal decomposition can be computed in polynomial time. The first key property is that the subset of variables that are possible candidates to be guards in such decompositions can be enumerated in polynomial time. The second property is that, whenever an instance is guarded, for each function $\mathcal{F} \in L$, there must be a guard p for $\mathcal{F}$ in the decomposition tree such that its subtree rooted at p covers some variable occurring in $\mathrm{dom}(\mathcal{F}') \setminus \mathrm{dom}(\mathcal{F})$, for some (other) function $\mathcal{F}' \in L$. Then, the weighing function ξ assigns a weight -1 to such a vertex p (more precisely, -1 for each function for which p may play this role). From the latter property and the definition of guards, it can be seen that at most one guard for each valuation function may get weight -1 in any hypertree decomposition. Hence, decompositions having total weight $-|L|$ must have a guard for each valuation function. □

An easy consequence is the desired islands of tractability.

Corollary 4.10. *On classes of instances that are guarded via $\Psi_{hw\text{-}k}$ (and hence via $\Psi_{tw\text{-}k}$), solving* LEXMAX-MIN-CSP$[\infty, \infty]$ *is feasible in polynomial time.*

5 Conclusion

In this paper, we have described a framework for "fair" CSP optimization, and we have provided a clear picture of the tractability frontier for (LEX)MAX-MIN-CSP[D, F] over acyclic instances, as summarized in Table 1 (where note that hardness results hold even over MAX-MIN-CSP[D, F]). Moreover, we have identified larger classes of tractable instances based on a new structural decomposition approach, which is able to deal with the interactions among the constraints and the valuation functions to be optimized. This is accomplished via a suitable notion of guard of a function that is able to separate its variables from the domains of the other functions, in any decomposition of the (possibly cyclic) given instance. By Fact 4.5 and Fact 4.8, tractable results for acyclic instances are just special cases of such guarded classes.

References

[Bansal and Sviridenko, 2006] Nikhil Bansal and Maxim Sviridenko. The Santa Claus problem. In *Proc. of the 38th Annual ACM Symposium on Theory of Computing (STOC'06)*, pages 31–40, 2006.

[Bernstein and Goodman, 1981] Philip A. Bernstein and Nathan Goodman. Power of natural semijoins. *SIAM Journal on Computing*, 10(4):751–771, 1981.

[Bezáková and Dani, 2005] Ivona Bezáková and Varsha Dani. Nobody left behind: fair allocation of indivisible goods. *ACM SIGecom Exchanges*, 5, 2005.

[Bistarelli *et al.*, 1999] Stefano Bistarelli, Ugo Montanari, Francesca Rossi, Thomas Schiex, Gérard Verfaillie, and Hélène Fargier. Semiring-based csps and valued csps: Frameworks, properties, and comparison. *Constraints*, 4(3):199–240, 1999.

[Bouveret and Lematre, 2009] Sylvain Bouveret and Michel Lemaître. Computing leximin-optimal solutions in constraint networks. *Artificial Intelligence*, 173(2):343–364, 2009.

[Brandt *et al.*, 2012] Felix Brandt, Vincent Conitzer, and Ulle Endriss. Computational Social Choice, in *Multiagent Systems*. MIT Press, 2012.

[Dechter, 2003] Rina Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[Dubois and Fortemps, 1999] Didier Dubois and Philippe Fortemps. Computing improved optimal solutions to max-min flexible constraint satisfaction problems. *European Journal of Operational Research*, 118(1):95–126, 1999.

[Fagin, 1983] Ronald Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *Journal of the ACM*, 30(3):514–550, 1983.

[Freuder *et al.*, 2010] Eugene C. Freuder, Robert Heffernan, Richard J. Wallace, and Nic Wilson. Lexicographically-ordered constraint satisfaction problems. *Constraints*, 15(1):1–28, 2010.

[Garey and Johnson, 1990] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[Gottlob *et al.*, 2000] Georg Gottlob, Nicola Leone, and Francesco Scarcello. A comparison of structural csp decomposition methods. *Artificial Intelligence*, 124(2):243–282, 2000.

[Gottlob *et al.*, 2001] Georg Gottlob, Nicola Leone, and Francesco Scarcello. The complexity of acyclic conjunctive queries. *Journal of the ACM*, 48(3):431–498, 2001.

[Gottlob *et al.*, 2002] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64(3):579–627, 2002.

[Gottlob *et al.*, 2009] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Tractable optimization problems through hypergraph-based structural restrictions. In *Proc. of the 36th International Colloquium on Automata, Languages and Programming (ICALP'09)*, pages 16–30, 2009.

[Greco and Scarcello, 2011] Gianluigi Greco and Francesco Scarcello. Structural tractability of constraint optimization. In *Proc. of the 17th International Conference on Principles and Practice of Constraint Programming (CP'11)*, pages 340–355, 2011.

[Kolaitis and Vardi, 2000] Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61(2):302–332, 2000.

[Robertson and Seymour, 1984] Neil Robertson and P.D Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

[Rossi *et al.*, 2006] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

[Scarcello *et al.*, 2008] Francesco Scarcello, Georg Gottlob, and Gianluigi Greco. Uniform Constraint Satisfaction Problems and Database Theory. In *Complexity of Constraints*, Springer, 2008.

[Scarcello *et al.*, 2007] Francesco Scarcello, Gianluigi Greco, and Nicola Leone. Weighted hypertree decompositions and optimal query plans. *Journal of Computer and System Sciences*, 73(3):475–506, 2007.

[Snow and Freuder, 1990] Paul Snow and Eugene C. Freuder. Improved relaxation and search methods for approximate constraint satisfaction with a maximin criterion. In *Proc. of the 8th biennal CSCSI conference*, pages 227–230, 1990.

[Torrens and Faltings, 2002] Marc Torrens and Boi Faltings. Using soft csps for approximating pareto-optimal solution sets. In *AAAI Workshop Proceedings. Preferences in AI and CP: Symbolic Approaches*, AAAI Press, 2002.