# Reasoning About NP-complete Constraints

Emmanuel Hébrard

# Reasoning About NP-complete Constraints

**Emmanuel Hebrard**
LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
hebrard@laas.fr,

## Abstract

The concept of *local consistency* – making global deductions from local infeasibility – is central to constraint programming. When reasoning about NP-complete constraints, however, since achieving a "complete" form of local consistency is often considered too hard, we need other tools to design and analyze propagation algorithms.

In this paper, we argue that NP-complete constraints are an essential part of constraint programming, that designing dedicated methods has lead to, and will bring, significant breakthroughs, and that we need to carefully investigate methods to deal about a necessarily incomplete inference. In particular, we advocate the use of *fixed-parameter tractability* and *kernelization* to this purpose.

## 1 Introduction

Constraint programming (CP) has been a very successful framework for modeling and solving hard combinatorial problems. Many problems are naturally framed as constraint satisfaction or optimization problems, where a set of discrete valued variables is to be assigned while satisfying a set of constraints. For instance, CP has been successfully applied to managing the aftermath of natural disasters [Hentenryck, 2013], optimizing the delivery of radiotherapy in cancer treatment [Cambazard *et al.*, 2012] or scheduling the exploration of a comet [Simonin *et al.*, 2012].

A first strength of CP often put forward, is its declarative aspect. Another essential feature is the modularity and versatility of the inference mechanisms. The key principle is the notion of *local consistency*: if we can deduce that some assignments are locally infeasible, for instance with respect to a single constraint, then the same deduction holds globally. In particular, a constraint network is said *domain*[1] *consistent* [Waltz, 1975] when there is no further deduction to be made, with respect to any single constraint, about the possible values that its variables can take. It was shown that maintaining this level of consistency during search was often a good choice [Sabin and Freuder, 1994] and most constraint solvers are designed according to this principle.

[1]Or equivalently *arc* consistent

The notion of *global constraint* makes this type of reasoning extremely powerful. A constraint is said *global* when it corresponds to a class of relations on an arbitrary number of variables. For instance, the constraint ALLDIFFERENT ensures that a set of variables take all pairwise distinct values. Given a recurring subproblem for which there exists an efficient algorithm, it is natural to derive the corresponding constraint, that is, design a *propagation algorithm* responsible for making deduction locally to that constraint. For instance, the propagation algorithm for the ALLDIFFERENT constraint [Régin, 1994] is based on Hopcroft and Karp's maximum matching algorithm [Hopcroft and Karp, 1973]. Similarly, a number of constraints are propagated using results from flow theory [van Hoeve *et al.*, 2006] or dynamic programming [Pesant, 2004; Hebrard *et al.*, 2009].

Since constraints must be easily checkable, they lie within the complexity class NP. However, many recurring subproblems that might qualify as constraints are NP-complete [Bessiere *et al.*, 2006b]. Examples of NP-complete constraints are numerous. For instance for ensuring that a set of Boolean variables represent a clique of a certain size [Fahle, 2002; Régin, 2003]; that integer variables representing vertices of a graph are given a value corresponding to an isomorphic vertex in a larger graph [Régin, 1995]; that the least and highest number of occurrences of any value is within some bounds [Schaus *et al.*, 2007]; that two overlapping sets of variables both have all pairwise distinct values [Kutz *et al.*, 2008]; that there is an upper bound on the cardinality of the set of assigned values [Bessiere *et al.*, 2006a]; that a sequence of variables encode the successors of an Hamiltonian circuit of bounded length in a graph [Ducomman *et al.*, 2016]; or that a set of tasks share a disjunctive [Baptiste and Le Pape, 1995] or cumulative resource [Baptiste and Le Pape, 1997].

However, whereas one can enforce domain consistency on polynomial constraints, it is often necessary to find a compromise between inference strength and computational effort on NP-complete constraints. In this paper, we advocate the design of propagation algorithms for NP-complete constraints. Furthermore, we argue that there is a need for a theoretical characterization of the consistency enforced by these algorithms. We show that *fixed-parameter tractability* [Downey *et al.*, 1999], and in particular the notion of *kernel* are extremely relevant in this context and that the latter can be extended to fill this role of measure of propagation strength.

## 2 NP-complete Constraints

A *constraint network* is a triple $(\mathcal{X}, \mathcal{U}, \mathcal{C})$ where $\mathcal{X}$ is a totally ordered set of *variables*, the universe $\mathcal{U}$ is a finite set of values and $\mathcal{C}$ is a set of *constraints*. A constraint $C$ is a pair $(S_C, R_C)$ where the *scope* $S_C$ is a subset of $\mathcal{X}$ and $R_C$ is a relation on $\mathcal{U}$ of arity $|S_C|$. The *Constraint Satisfaction Problem (CSP)* asks whether a constraint network $(\mathcal{X}, \mathcal{U}, \mathcal{C})$ has a *solution*, that is, a $|\mathcal{X}|$-tuple $\sigma \in \mathcal{U}^{|\mathcal{X}|}$ such that for each $C \in \mathcal{C}$, the projection $\sigma(S_C)$ of $\sigma$ onto $S_C$ is in $R_C$.

A *global constraint* is a class of relations, one for each possible value of its parameters. The most common parameter is the number of constrained variables. For instance we can define the NVALUE [Pachet and Roy, 1999] constraint with one relation per integer $n$ given by the following predicate:

**Definition 1** (NVALUE).

$$\text{NVALUE}(x_1, \ldots, x_n, y) \iff |\{x_i \mid 1 \leq i \leq n\}| \leq y$$

During search, in order to store the decisions and deductions, we use a special type of relation over $\mathcal{X}$, the *domain* $\mathcal{D}$. There is not a unique type of relation used for that purpose, for instance the search space can be encoded as a Boolean Decision Diagram [Hadzic and Hooker, 2007]. However we consider here the canonical *finite discrete domain* where the $\mathcal{D}$ relation is restricted to conjunctions of finite unary relations, one for every variable $x \in \mathcal{X}$, which we denote $\mathcal{D}(x)$. We write $\mathcal{D}' \subseteq \mathcal{D}$ as a shortcut for $\forall x \in \mathcal{X}\ \mathcal{D}'(x) \subseteq \mathcal{D}(x)$.

**Definition 2.** *A constraint $C$ is said* domain consistent *on a finite discrete domain $\mathcal{D}$ if and only if*

$$\forall \mathcal{D}' \subseteq \mathcal{D}\ \exists \tau \in R_C \cap (\mathcal{D}(S_C) \setminus \mathcal{D}'(S_C))$$

The problem of achieving domain consistency of a domain $\mathcal{D}$ with respect to a constraint $C$ consists in finding the largest domain $\mathcal{D}' \subseteq \mathcal{D}$ such that $C$ is arc consistent on $\mathcal{D}'$. There is a unique largest arc consistent *closure* of a finite discrete domain $\mathcal{D}$ with respect to a constraint $C$.

Notice that achieving domain consistency is polynomially reducible to checking the *satisfiability* of a constraint, i.e., finding a tuple $\tau \in \mathcal{D} \cap C$. Given an algorithm for checking satisfiability, for every variable $x$ and every value $v \in \mathcal{D}(x)$ one can check the satisfiability of $C$ w.r.t. $\mathcal{D}'$ where $\mathcal{D}'(x) = \{v\}$ and $\mathcal{D}' = \mathcal{D}$ otherwise. If the constraint is not satisfiable, then the domain consistent closure of $\mathcal{D}$ is such that $v \notin \mathcal{D}(x)$ and we can change $\mathcal{D}$ accordingly. Otherwise the domain consistent closure is such that $v \in \mathcal{D}(x)$ and we leave $\mathcal{D}$ unchanged. This procedure terminates after $\sum_{x \in S_C} |\mathcal{D}(x)|$ steps. The converse is trivial, since a constraint is satisfiable if and only if its domain consistent closure is not empty. We therefore say that a constraint is NP-complete if checking its satisfiability is NP-complete, even though achieving domain consistency is not a decision problem.

The constraint NVALUE is NP-complete since checking if there exists a tuple in a domain $\mathcal{D}$ such that $|\{\tau(x_i) \mid 1 \leq i \leq n\}| \leq y$ is equivalent to finding a hitting set of the collection $\{\mathcal{D}(x_1), \ldots, \mathcal{D}(x_n)\}$ of size $\max \mathcal{D}(y)$ [Bessiere *et al.*, 2006a]. Therefore, achieving the domain consistent closure is often considered too costly for this constraint. For NVALUE and other NP-complete constraints incomplete approaches are used instead.

**Relaxing the domain:** A common way to reduce the complexity of propagating a constraint is to relax the domain relation $\mathcal{D}$. For instance, the notion of *bounds consistency* is widely used in this situation. Let *interval domains* be the class of relations restricted to conjunctions of finite unary relations, one for every variable $x \in \mathcal{X}$, of the form $l \leq x \leq u$. Bounds consistency is the property obtained by applying Definition 2 to interval, instead of discrete, domains.

The hitting set problem has a polynomial algorithm when the sets are discrete intervals. Therefore, the relaxtion from finite discrete domains to interval domains make satisfiability checking and thus bounds consistency polynomial. The best algorithm achieves both in $\mathcal{O}(|\mathcal{X}| \log |\mathcal{X}|)$ [Beldiceanu, 2001]. This is true for many NP-complete global constraints. For instance the constraint GCC, which channels the number of occurrences of values in a set of variables $\mathcal{X}$ to another set of variables $\mathcal{Y}$ is NP-complete for discrete domains [Quimper *et al.*, 2004], but polynomial if the domain relation is relaxed to intervals for the variables $\mathcal{Y}$ [Régin, 1996].

**Decomposing the constraint:** Another common way to deal with NP-completeness is to decompose the constraint. A constraint network $\mathcal{I} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ is a *decomposition* of a constraint $C$ if and only if $S_C \subseteq \mathcal{X}$ and $\sigma$ is a solution of $\mathcal{I}$ if and only if $\sigma(S_C) \in R_C$. In other words, it is a constraint network on a superset of $S_C$ such that its set of solutions, when projected onto $S_C$ is exactly the set of tuples in $R_C$.

Decompositions are used in countless constraint programming approaches. A lot of effort has been put into the study of decompositions and in particular when it is or when it is not possible to emulate algorithms through decompositions, both for polynomial and NP-complete constraints [Narodytska, 2011]. For instance, it was shown that propagation algorithms for the NVALUE constraint can be emulated for the same time complexity, however at the cost of a much higher space complexity [Bessiere *et al.*, 2010].

Moreover, decompositions are sometimes used as a way to relax the problem of achieving the domain consistent closure, even though an efficient dedicated algorithm is then proposed to compute the closure. For instance the $\mathcal{O}(n \log n)$ "Timetabling" method for reasoning about a set of $n$ tasks consumming energy from a cumulative resource [Le Pape, 1988; Beldiceanu and Carlsson, 2001] over time can be seen as achieving bounds consistency on a decomposition involving Boolean variables $x_{ij}$ standing for whether task $i$ is processed at time $j$, however with a far lower time complexity. Similarly, "Edge Finding" algorithms [Nuijten and Aarts, 1994; Vilím, 2009] achieve bounds consistency (in polynomial time) on a decomposition (of exponential size). Lastly, Ouellet and Quimper's algorithm [Ouellet and Quimper, 2013] achieves bounds consistency on the conjunction of these two decompositions in $\mathcal{O}(kn \log n)$ time with $k$ the number of different tasks' consumptions.

**Approximation:** Often, NP-complete constraints on a set of variables $\mathcal{X}$ can be easily seen as enforcing that an $|\mathcal{X}|$-ary *cost function* $\pi : \mathcal{D}^{|\mathcal{X}|} \mapsto \mathbb{Q} \cup \{\infty, -\infty\}$ is non-positive.[2] For instance, the constraint NVALUE corresponds to the fol-

[2] This is not restrictive as any relation maps tuples to $0/1$.

lowing cost function: $|\{x_i \mid 1 \leq i \leq n\}| - y$. The intersection graph of $\{\mathcal{D}(x_1), \dots, \mathcal{D}(x_n)\}$ is the graph with one vertex $v_i$ for $i \in [1, n]$ and an edge $(v_i, v_j)$ iff $\mathcal{D}(x_i) \cap \mathcal{D}(x_j) \neq \emptyset$. The independence number of this graph is larger than or equal to the minimum hitting set of the collection. Therefore, this is a valid upper lower bound for $y$ [Bessiere *et al.*, 2006a]. In this case, the approximation offers no guarantee, yet it is a relatively effective method in practice, often outperforming the bounds consistency approach.

More generally, when a method with a guaranteed approximation ratio exists for a cost function $\pi$, this method provides both a primal and a dual bound which is extremely valuable. In general, however, approximation results are still largely underused within the context of constraint propagation.

## 3 Propagation via Kernels

The complexity of a problem can be more finely characterized by considering *parameters* besides the size of the input. Parameterized complexity aims at understanding which parameters are relevant to explain the hardness of a problem. Given a problem $\mathcal{P}$ and a parameter $p$, $(\mathcal{P}, p)$ is in the FPT class if there exists an algorithm that can decide an instance $\mathcal{I}$ of $\mathcal{P}$ in time $f(p)|\mathcal{I}|^{O(1)}$ where $f$ is a computable function.

A previous study [Bessiere *et al.*, 2008] of the parameterized complexity of global constraints, and of their relevant parameters, showed that this approach was promising. In particular, the NVALUE constraint is FPT when the parameter is the number of "holes" in the domains [Bessiere *et al.*, 2008].

Moreover, it is often possible to compute *kernels* of FPT problems, which are extremely relevant in this context.

**Definition 3.** *A kernelization for a problem $\mathcal{P}$ and a parameter $p$ is a polynomial-time computable function that maps each instance $x$ and parameter value $k$ to an instance $x'$ and parameter $k'$ of the same problem such that $x$ is a yes-instance if and only $x'$ is, $|x'| \leq f(k)$, and $k' \leq g(k)$ for some computable functions $f, g$.*

There is intense research both on FPT algorithms and kernelization methods [Cygan *et al.*, 2015]. Characterizing a kernel is a very significant step in understanding the combinatorial structure and efficiently reasoning about a constraint. Intuitively, the difference between the original instance and the kernel is composed of inconsistent (or entailed) values.

Consider for instance the vertex cover problem, asking whether there is a subset of at most $k$ vertices of a graph $G$, such that every edge has at least one extremity in the cover. The Buss rule consists in adding vertices of degree $k+1$ to the cover, since otherwise their neighbors would be in the cover. This very simple propagation rule yields a kernel of size $k^2$.

However, classical kernels are not always correct propagation. For instance, the smallest kernels for the vertex cover problem [Abu-Khzam *et al.*, 2007; Nemhauser and Trotter Jr, 1975] are based on *crown decompositions* of the graph.

**Definition 4.** *A* ***crown decomposition*** *of a graph $G$ is a partition $(H, W, I)$ of $V$ such that vertices in $I$ have an edge only with vertices in $W$ and there is a matching of size $|W|$ between $W$ and $I$.*

The size of a vertex cover can never be increased by removing all vertices from $I$ and adding all vertices from $W$. However, this is a correct *dominance* rule, but not a correct propagation rule: feasible or even minimal solutions might involve vertices from $I$. Similarly, an efficient kernelization using dominance was proposed for the NVALUE constraint [Gaspers and Szeider, 2011]. For the same reason, this method is to be used within the *probing* procedure described in Section 2 to achieve domain consistency through satisfiability checks: for every variable-value pair, the problem of deciding if the constraint is satisfiable when restricting the domain accordingly is solved on the kernel. In other words, several exponential satisfiability checks have to be performed.

In [Carbonnel and Hebrard, 2016; 2017] we proposed an approach to propagating NP-complete constraints based on a new definition of "loss-less" kernelization tailored for constraint propagation. Intuitively, $z$-loss-less kernels are kernels with an extra *loss-lessness* property: Whereas solving a classical kernel is sufficient to solve the original instance, *achieving domain consistency* on a loss-less kernel is sufficient to achieve domain consistency on the original instance. In other words, there exists a polynomial algorithm, which, given the domain consistent closure of the loss-less kernel, computes it for the original instance. This is much more consistent with the spirit of kernelization, extends smoothly constraints with polynomial-time propagators and yields a propagation algorithm with running time $O(g(p) + |I|^{O(1)}p^{O(1)})$, instead of $O(|I|^{O(1)}g(p))$ for probing plus classical kernelization.

There is caveat, however. Consider again constraints defined as minimizing a cost function, for instance the NVALUE constraint which ensures that the cardinality of the set of values taken by the variables $x_1, \dots, x_n$ is at most $y$. As long as the domain of $y$ contains high values, the constraint is unlikely to propagate much. In fact, if $n \in \mathcal{D}(y)$ then the constraint is completely inoperant. More generally, kernels whose size decrease when the constraint get *tighter* should be priotirized since they are the most useful with respect to propagation. When defining constraints as minimizing a cost function such as the NVALUE constraint, the tightness is very clearly related to the *gap* $z$ from 0 to the minimum of $\pi_{\mathcal{D}}(\mathcal{X})$ under the current domain $\mathcal{D}$. We therefore use this extra parameter $z$ in the definition of loss-less kernels:

**Definition 5.** *Let $\Pi$ be a set of cost functions, and $C$ a global constraint defined as the relations $\pi(\mathcal{X}) \leq 0$ for $\pi \in \Pi$.*

*A* $z$-loss-less kernelization *of $C$ with parameter $p$ is a polynomial-time computable function mapping each instance $(\pi, \mathcal{X}, \mathcal{D})$ and parameter value $k$ to an instance $(\pi', \mathcal{X}', \mathcal{D}')$ and parameter $k'$ of the same constraint such that* ***if*** $-\min(\pi_{\mathcal{D}}(\mathcal{X})) \leq z$, ***then*** *there is a polynomial algorithm achieving domain consistenty of $\mathcal{D}$ w.r.t. $\pi(\mathcal{X}) \leq 0$ given the the domain consistent closure of $\mathcal{D}'$ w.r.t. $\pi'(\mathcal{X}') \leq 0$, $|\pi'| + |\mathcal{D}'| + |\mathcal{X}'| \leq f(k)$, and $k' \leq g(k)$ for some computable functions $f, g$.*

The results in [Carbonnel and Hebrard, 2017] show that loss-less kernels exist: There is a $(z+2)k$ $z$-loss-less kernel for the vertex cover problem parameterized by the size $k$ of the cover, thus matching the result of Nemhauser [Nemhauser and Trotter Jr, 1975] for $z = 0$.Similarly, there is a $\infty$-loss-less kernel $\max(6k, k^2/2 + 7k/2)$ kernel for the *edge dominating set* parameterized by the size of the dominating set, thus matching the result of Hagerup [Hagerup, 2012].

Moreover, they are not mere theoretical curiosities: kernel-based propagation was successfully applied to a constrained vertex cover problem [Carbonnel and Hebrard, 2016].

## 4 Conclusions

In this paper we have surveyed different approaches to propagating NP-complete constraints and argued that designing dedicated methods is extremely valuable. Moreover, among the possible ways of tackling the propagation of NP-complete constraints (approximation, relaxation, decomposition, etc.) we argue that fixed parameterized complexity and in particular kernelization offers several extremely relevant features:

Firstly, the value of the parameter, as well as the size of the gap $z$, changes during search. Therefore, it is possible to target when kernelization is most likely to being beneficial and thus use it in an opportunistic way.

Secondly, loss-less kernels are designed to achieve domain consistency of the full instance with a single call to a possibly exponential algorithm to compute the closure of the kernel. However, in practice, the kernelization procedure in itself makes some incomplete inference while being polynomial. The kernelization process in itself can therefore be used as a polynomial propagation procedure.

Finally, the guarantee on the size of the kernels entails a guarantee on the strength of this inference. In other words, the size of the kernel is a valuable criterion to compare the achieved level of consistency.

## References


[Abu-Khzam *et al.*, 2007] Faisal N Abu-Khzam, Michael R Fellows, Michael A Langston, and W Henry Suters. Crown Structures for Vertex Cover Kernelization. *Theory of Computing Systems*, 41(3):411–430, 2007.

[Baptiste and Le Pape, 1995] Philippe Baptiste and Claude Le Pape. A Theoretical and Experimental Comparison of Constraint Propagation Techniques for Disjunctive Scheduling. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 600–606, 1995.

[Baptiste and Le Pape, 1997] Philippe Baptiste and Claude Le Pape. Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems. In *Proceedings of the 3rd International Conference on Principles and Practice of Constraint Programming*, pages 375–389, 1997.

[Beldiceanu and Carlsson, 2001] Nicolas Beldiceanu and Mats Carlsson. Sweep as a Generic Pruning Technique Applied to the Non-overlapping Rectangles Constraint. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, pages 377–391, 2001.

[Beldiceanu, 2001] Nicolas Beldiceanu. Pruning for the *Minimum* Constraint Family and for the *Number of Distinct Values* Constraint Family. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, pages 211–224, 2001.

[Bessiere *et al.*, 2006a] Christian Bessiere, Emmanuel Hebrard, Brahim Hnich, Zeynep Kiziltan, and Toby Walsh. Filtering Algorithms for the NValue Constraint. *Constraints*, 11(4):271–293, 2006.

[Bessiere *et al.*, 2006b] Christian Bessiere, Emmanuel Hebrard, Brahim Hnich, and Toby Walsh. The Complexity of Reasoning with Global Constraints. *Constraints*, 12(2):239–259, 2006.

[Bessiere *et al.*, 2008] Christian Bessiere, Emmanuel Hebrard, Brahim Hnich, Zeynep Kiziltan, Claude-Guy Quimper, and Toby Walsh. The Parameterized Complexity of Global Constraints. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, pages 235–240, 2008.

[Bessiere *et al.*, 2010] Christian Bessiere, George Katsirelos, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. Decomposition of the NValue Constraint. In *Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming*, pages 114–128, 2010.

[Cambazard *et al.*, 2012] Hadrien Cambazard, Eoin O'Mahony, and Barry O'Sullivan. A Shortest Path-based Approach to the Multileaf Collimator Sequencing Problem. *Discrete Applied Mathematics*, 160(1-2):81–99, 2012.

[Carbonnel and Hebrard, 2016] Clément Carbonnel and Emmanuel Hebrard. Propagation via Kernelization: the Vertex Cover Constraint. In *Proceedings of the 22nd International Conference on Principles and Practice of Constraint Programming*, pages 147–156, 2016.

[Carbonnel and Hebrard, 2017] Clément Carbonnel and Emmanuel Hebrard. On the kernelization of global constraints. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 578–584, 2017.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 2015.

[Downey *et al.*, 1999] Rod G. Downey, Michael R. Fellows, and Ulrike Stege. Parameterized Ccomplexity: A Framework for Systematically Confronting Computational Intractability. *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, pages 49–99, 1999.

[Ducomman *et al.*, 2016] Sylvain Ducomman, Hadrien Cambazard, and Bernard Penz. Alternative Filtering for the Weighted Circuit Constraint: Comparing Lower Bounds for the TSP and Solving TSPTW. In *Proceedings of the 30th National Conference on Artificial Intelligence*, pages 3390–3396, 2016.

[Fahle, 2002] Thorsten Fahle. *Integrating concepts from constraint programming and operations research algorithms*. PhD thesis, University of Paderborn, 2002.

[Gaspers and Szeider, 2011] Serge Gaspers and Stefan Szeider. Kernels for Global Constraints. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 540–545, 2011.

[Hadzic and Hooker, 2007] Tarik Hadzic and John N. Hooker. Cost-Bounded Binary Decision Diagrams for 0-1 Programming. In *Proceedings of the 4th International Conference on Integration of AI and OR Techniques in CP*, pages 84–98, 2007.

[Hagerup, 2012] Torben Hagerup. Kernels for Edge Dominating Set: Simpler or Smaller. In *Proceedings of the 31st International Symposium on Mathematical Foundations of Computer Science*, pages 491–502, 2012.

[Hebrard *et al.*, 2009] Emmanuel Hebrard, Daniel Marx, Barry O'Sullivan, and Igor Razgon. Constraints of Difference and Equality: A Complete Taxonomic Characterisation. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, pages 424–438, 2009.

[Hentenryck, 2013] Pascal Van Hentenryck. Computational Disaster Management. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2013.

[Hopcroft and Karp, 1973] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[Kutz *et al.*, 2008] Martin Kutz, Khaled M. Elbassioni, Irit Katriel, and Meena Mahajan. Simultaneous matchings: Hardness and approximation. *J. Comput. Syst. Sci.*, 74(5):884–897, 2008.

[Le Pape, 1988] Claude Le Pape. *Des systèmes d'ordonnancement flexibles et opportunistes*. PhD thesis, Université Paris XI, 1988.

[Narodytska, 2011] Nina Narodytska. *Reformulation of global constraints*. PhD thesis, University of New South Wales, Sydney, Australia, 2011.

[Nemhauser and Trotter Jr, 1975] George L Nemhauser and Leslie E Trotter Jr. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

[Nuijten and Aarts, 1994] Wim Nuijten and Emile Aarts. Constraint satisfaction for multiple capacitated job shop scheduling. In *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 635–639, 1994.

[Ouellet and Quimper, 2013] Pierre Ouellet and Claude-Guy Quimper. Time-Table Extended-Edge-Finding for the Cumulative Constraint. In *Proceedings of the 19th International Conference on Principles and Practice of Constraint Programming*, pages 562–577, 2013.

[Pachet and Roy, 1999] François Pachet and Pierre Roy. Automatic Generation of Music Programs. In *Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming*, pages 331–345, 1999.

[Pesant, 2004] G. Pesant. A Regular Language Membership Constraint for Finite Sequences of Variables. In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming*, pages 482–495, 2004.

[Quimper *et al.*, 2004] Claude-Guy Quimper, Alejandro López-Ortiz, Peter van Beek, and Alexander Golynski. Improved Algorithms for the Global Cardinality Constraint. In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming*, pages 542–556, 2004.

[Régin, 1994] Jean-Charles Régin. A Filtering Algorithm for Constraints of Difference in CSPs. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 362–367, 1994.

[Régin, 1995] Jean-Charles Régin. *Développement d'outils algorithmiques pour l'Intelligence Artificielle. Application à la chimie organique*. PhD thesis, Université Montpellier II, 1995.

[Régin, 1996] Jean-Charles Régin. Generalized Arc Consistency for Global Cardinality Constraint. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 209–215, 1996.

[Régin, 2003] Jean-Charles Régin. Using Constraint Programming to Solve the Maximum Clique Problem. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming*, pages 634–648, 2003.

[Sabin and Freuder, 1994] Daniel Sabin and Eugene C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In Alan Borning, editor, *Proceedings of the 2nd Workshop on Principles and Practice of Constraint Programming*, pages 10–20, 1994.

[Schaus *et al.*, 2007] Pierre Schaus, Yves Deville, and Pierre Dupont. Bound-Consistent Deviation Constraint. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming*, pages 620–634, 2007.

[Simonin *et al.*, 2012] Gilles Simonin, Christian Artigues, Emmanuel Hebrard, and Pierre Lopez. Scheduling Scientific Experiments on the Rosetta/Philae Mission. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming*, pages 23–37, 2012.

[van Hoeve *et al.*, 2006] Willem-Jan van Hoeve, Gilles Pesant, and Louis-Martin Rousseau. On Global Warming: Flow-Based Soft Global Constraints. *Journal of Heuristics*, 12(4-5):347–373, 2006.

[Vilím, 2009] Petr Vilím. Max Energy Filtering Algorithm for Discrete Cumulative Resources. In *Proceedings of the 6th International Conference on Integration of AI and OR Techniques in CP*, pages 294–308, 2009.

[Waltz, 1975] David Waltz. Understanding Line Drawings of Scenes with Shadows. In *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, 1975.