Variable Elimination in Binary CSPs (Extended Abstract)

Martin Cooper, Achref El Mouelhi, Cyril Terrioux

Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Journal Track

Variable Elimination in Binary CSPs (Extended Abstract)*

Martin C. Cooper[1†], **Achref El Mouelhi**[2] and **Cyril Terrioux**[3]
[1]IRIT, University of Toulouse III, France
[2] H & H: Research and Training, Marseille, France
[3]Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
cooper@irit.fr, elmouelhi.achref@gmail.com, cyril.terrioux@lis-lab.fr

Abstract

We investigate rules which allow variable elimination in binary CSP (constraint satisfaction problem) instances while conserving satisfiability. We propose new rules and compare them, both theoretically and experimentally. We give optimised algorithms to apply these rules and show that each defines a novel tractable class. Using our variable-elimination rules in preprocessing allowed us to solve more benchmark problems than without.

1 Introduction

Constraint satisfaction provides a generic model for many NP-hard problems encountered in fields such as artificial intelligence, bioinformatics and operations research. In this paper, we study binary CSP instances, in which each constraint concerns at most two variables. Since the binary CSP is NP-complete, it is of practical interest to find polynomial-time operations which reduce the size of the search space. One obvious way to reduce search space size is by variable elimination.

Variable elimination is classic in those families of constraint problems in which variables can be eliminated without changing the nature of the constraints: we can cite Gaussian elimination in systems of linear equations over a field [Schrijver, 1999] or variable-elimination resolution in boolean formulae in CNF [Subbarayan and Pradhan, 2004]. Indeed, any variable x_i can be eliminated from a *general-arity* CSP instance by joining all constraints whose scope includes x_i and projecting the resulting relation R with scope Y onto the variables $Y \setminus \{x_i\}$ [Dechter, 1999; Larrosa and Dechter, 2003]. Call this relation R^{-x_i}. Unfortunately, this often introduces a high-arity constraint and this can be counterproductive in terms of both memory and time. Under certain conditions, a binary CSP instance will remain binary after this join-and-project variable elimination of x_i. For example, this is clearly the case if x_i is constrained by only two other variables since, in this case, R^{-x_i} is binary. A more interesting case is when all constraints with x_i in their scope share a majority polymorphism since, in this case, the relation R^{-x_i} is equivalent to the join of its binary projections [Jeavons *et al.*, 1998]. Fourier's algorithm for variable elimination applied to a system of binary linear inequalities [Koubarakis, 2006; Schrijver, 1999] can be viewed as just one example of this general rule, since binary linear inequalities are all closed under the majority polymorphism *median*. Another interesting case is when there is a functional constraint of the form $x_i = f(x_j)$ (where f is a function) for some other variable x_j: the relation R^{-x_i} is then equivalent to the join of its projections onto the pairs of variables (x_j, x_k) $(k \neq i, j)$ [Zhang and Yap, 2011].

Unfortunately, the introduction of a large number of new constraints, even if they are still binary, may again be counterproductive. Therefore, we concentrate in this paper on rules which do not introduce new constraints when a variable is eliminated.

Various rules have been found which allow the elimination of a variable without introducing new constraints and without changing the satisfiability of the instance [Cohen *et al.*, 2015; Cooper, 2014; Cooper *et al.*, 2010]. Such rules were used, for example, in the deep optimisation solution to the spectrum repacking problem [Newman *et al.*, 2018]. Discovery of new variable-elimination rules may have not only practical but also theoretical applications. For example, simple rules for variable or value elimination are used by Beigel and Eppstein (1995) in their algorithms with low worst-case time bounds for such NP-complete problems as 3-COLOURING and 3SAT: these simplification operations are an essential first step before the use of decompositions into subproblems with smaller domains. In the theory of fixed-parameter tractability, variable elimination is often an essential ingredient of polynomial kernalisation algorithms. For example, in the Point Line Cover problem (find k straight lines which cover n points), if at least $k+1$ points lie on a line, then they can be effectively eliminated since they must be covered by this line [Kratsch *et al.*, 2016].

We study the following generic NP-hard problem.

Definition 1 *A* binary CSP instance $I = \langle X, \mathcal{D}, R \rangle$ *comprises*

- *a set* X *of* n *variables* $x_1, \ldots, x_n$,
- *a domain* $\mathcal{D}(x_i)$ *for each variable* x_i $(i = 1, \ldots, n)$, *and*
- *a binary constraint relation* R_{ij} *for each pair of distinct variables* x_i,x_j $(i,j \in \{1, \ldots, n\})$.

*This paper is an extended abstract of [Cooper *et al.*, JAIR, 2019]
†Contact Author

For notational convenience, we assume that there is exactly one binary relation R_{ij} for each pair of variables. We say that x_i *constrains* x_j if R_{ij} is different from $\mathcal{D}(x_i) \times \mathcal{D}(x_j)$, and we use e to denote the number of pairs of variables $\{x_i, x_j\}$ such that x_i constrains x_j. An assignment $\langle v_1, \ldots, v_m \rangle$ to variables $\langle x_{i_1}, \ldots, x_{i_m} \rangle$ is *consistent* if $v_j \in \mathcal{D}(x_{i_j})$ (for $j = 1, \ldots, m$) and $(v_j, v_k) \in R_{i_j,i_k}$ (for all j, k such that $1 \leq j < k \leq m$). A *solution* to I is a consistent assignment to all variables in X.

We can associate a binary CSP instance with its microstructure, a labelled graph whose vertices are the variable-value assignments and which has positive and negative edges. If $(v_i, v_j) \in R_{ij}$, we say that the assignments $\langle x_i, v_i \rangle$, $\langle x_j, v_j \rangle$ (or more simply v_i, v_j) are *compatible* and that $v_i v_j$ is a *positive edge*, otherwise v_i, v_j are *incompatible* and $v_i v_j$ is a *negative edge*. In figures, broken lines represent negative edges (incompatible pairs) and solid lines represent positive edges (compatible pairs). For simplicity of notation we can assume that variable domains are disjoint, so that using v_i as a shorthand for $\langle x_i, v_i \rangle$ is unambiguous. We say that $v_i \in \mathcal{D}(x_i)$ has a *support* at variable x_j if there exists $v_j \in \mathcal{D}(x_j)$ such that $v_i v_j$ is a positive edge. A binary CSP instance I is *arc consistent* if for all pairs of distinct variables x_i, x_j, each $v_i \in \mathcal{D}(x_i)$ has a support at x_j. Arc consistency is ubiquitous in constraint solvers: it is applied both before and during search in binary CSPs since it can be established in $O(ed^2)$ time, where e is the number of binary constraints and d the maximum domain size [Bessière *et al.*, 2005].

All proofs of results given in this extended abstract can be found in the full version of the paper [Cooper *et al.*, 2019], together with details of the experimental trials.

2 Variable-Elimination Rules

We study conditions under which a variable x_i can be eliminated from a binary CSP instance while conserving satisfiability. A simple example of such a condition is that there exists a value $v_i \in \mathcal{D}(x_i)$ which is compatible with all assignments to all other variables. Clearly any solution s to the instance I' obtained by eliminating x_i can be extended to a solution to the original instance I by setting $s(x_i) = v_i$. Another simple example is that the variable x_i has a singleton domain $\{v_i\}$. This second example demonstrates that when eliminating the variable x_i we need to retain the projections onto $X \setminus \{x_i\}$ of all constraints whose scope includes x_i, since in this example we must first eliminate from all domains $\mathcal{D}(x_j)$ ($j \neq i$) those values that are not compatible with $\langle x_i, v_i \rangle$. Thus, the instance I' obtained by *eliminating a variable* x_i from a binary CSP instance I is identical to I except that (1) $\forall j \neq i$, we have deleted from $\mathcal{D}(x_j)$ all values v_j such that $\langle x_j, v_j \rangle$ has no support at x_i in I, and (2) we have deleted the variable x_i and all constraints with x_i in their scope.

We require the following formal definition in order to study provably-correct variable-elimination rules [Cohen *et al.*, 2015].

Definition 2 *A* satisfiability-conserving variable-elimination condition *(or a* var-elim condition*) is a polytime-computable property $P(x_i)$ of a variable x_i in a binary CSP instance I such that when $P(x_i)$ holds the instance I' obtained from I by eliminating x_i is satisfiable if and only if I is satisfiable. Such a property $P(x_i)$ is a* solution-conserving variable-elimination condition *(*sol-var-elim condition*) if it is possible to construct a solution to I from any solution s' to I' in polynomial time.*

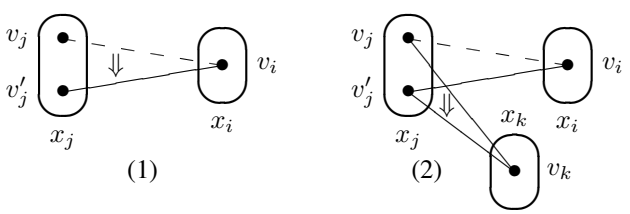

Figure 1: The DE-snake property: for some value $v_i \in \mathcal{D}(x_i)$, for each value v_j incompatible with v_i, there is a value v'_j such that (1) v'_j is compatible with v_i, and (2) v'_j is compatible with all assignments v_k to a third variable x_k which are compatible with v_j.

A sol-var-elim condition not only allows us to eliminate variables while conserving satisfiability but also allows the polynomial-time recovery of at least one solution to the original instance I from a solution to the reduced instance I'. All the var-elim properties given in this paper are also sol-var-elim properties.

3 Variable Elimination by the DE-snake Rule

Definition 3 *A variable x_i satisfies the DE-snake property if $\exists v_i \in \mathcal{D}(x_i)$ such that $\forall x_j \in X \setminus \{x_i\}$, $\forall v_j \in \mathcal{D}(x_j)$ with $(v_i, v_j) \notin R_{ij}$, $\exists v'_j \in \mathcal{D}(x_j)$ such that (1) $(v'_j, v_i) \in R_{ji}$ and (2) $\forall x_k \in X \setminus \{x_i, x_j\}$, $\forall v_k \in \mathcal{D}(x_k)$, we do not have $(v_j, v_k) \in R_{jk}$ and $(v'_j, v_k) \notin R_{jk}$.*

The DE-snake property is illustrated in Figure 1. The intuition behind this property is that any solution to the instance obtained after elimination of x_i can be extended to a solution to the original instance by assigning v_i to x_i and changing those values v_j which are incompatible with v_i to some other value v'_j.

Theorem 1 *The DE-snake property is a sol-var-elim condition in binary CSP instances.*

The following proposition shows that the worst-case complexity of applying the DE-snake rule is no worse than the complexity of applying the weaker ∃snake rule [Cohen *et al.*, 2015].

Proposition 1 *Variable eliminations by the DE-snake property can be applied until convergence in $O(ed^3)$ time and $O(ed^2)$ space.*

4 The Triangle Property

The variable-elimination rule presented in this section says that x_i can be eliminated if for some variable $x_j \neq x_i$, for all assignments $v_j \in \mathcal{D}(x_j)$ to y, in $I[\langle x_j, v_j \rangle]$ (the reduced instance consisting of the set of variable-value assignments compatible with $\langle x_j, v_j \rangle$) there is an assignment $\langle x_i, v_i \rangle$ compatible with all assignments to all variables $x_k \in X \setminus \{x_i, x_j\}$.

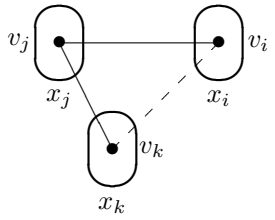

Figure 2: The open-triangle pattern.

In other words, there is some variable $x_j \neq x_i$ such that for all $v_j \in \mathcal{D}(x_j)$, there exists $v_i \in \mathcal{D}(x_i)$ such that $(v_j, v_i) \in R_{ji}$ and the open-triangle pattern shown in Figure 2 does not occur on (v_i, v_j, v_k) for any v_k.

Definition 4 *A variable x_i satisfies the* triangle property *if $\exists x_j \in X \setminus \{x_i\}$ such that $\forall v_j \in \mathcal{D}(x_j)$, $\exists v_i \in \mathcal{D}(x_i)$ with $(v_i, v_j) \in R_{ij}$ such that $\forall x_k \in X \setminus \{x_i, x_j\}$, $\forall v_k \in \mathcal{D}(x_k)$, $(v_j, v_k) \in R_{jk}$ implies that $(v_i, v_k) \in R_{ik}$.*

Theorem 2 *The triangle property is a sol-var-elim condition in binary CSP instances.*

Proposition 2 *Variable eliminations by the triangle property can be applied until convergence in $O(end^3)$ time and $O(end^2)$ space.*

5 Variable Elimination by Broken Polyhedra

The broken-triangle property is a property of the microstructure of instances of the binary CSP which when satisfied allows either value merging [Cooper *et al.*, 2016], variable elimination or the definition of a tractable class [Cooper *et al.*, 2010]. In this section, we generalise the notion of broken triangle to broken polyhedron, which allows us to define rules for variable elimination parameterised by the dimension k of the polyhedron. We begin by recalling the definition of the broken-triangle property (BTP) [Cooper *et al.*, 2010].

Definition 5 *Let $I = \langle X, \mathcal{D}, R\rangle$ be a binary CSP instance. A pair of values $v'_k, v''_k \in \mathcal{D}(x_k)$ satisfies BTP if for each pair of variables (x_i, x_j) (with $i, j \neq k$), $\forall v_i \in \mathcal{D}(x_i)$, $\forall v_j \in \mathcal{D}(x_j)$, if $(v_i, v_j) \in R_{ij}$, $(v_i, v'_k) \in R_{ik}$ and $(v_j, v''_k) \in R_{jk}$, then $(v_i, v''_k) \in R_{ik}$ or $(v_j, v'_k) \in R_{jk}$. A variable x_k satisfies BTP if each pair of values of $\mathcal{D}(x_k)$ satisfies BTP. If I is equipped with an order $<$ on its variables, then I satisfies BTP for the variable order $<$ if each variable x_k satisfies BTP in the sub-instance of I restricted to the variables x_i such that $x_i \leq x_k$.*

If $(v_i, v_j) \in R_{ij}$,$(v_i, v'_k) \in R_{ik}$, $(v_j, v''_k) \in R_{jk}$, $(v_i, v''_k) \notin R_{ik}$ and $(v_j, v'_k) \notin R_{jk}$ (as in Figure 3), then the quadruple (v'_k, v_i, v_j, v''_k) constitutes a *broken triangle* on x_k. I satisfies the BTP on x_k if no broken triangles occur on x_k.

We now generalise the notion of broken triangle to broken polyhedron and show that this notion can be used to define variable elimination rules that are stronger than BTP. A broken triangle is a broken polyhedron of dimension 2. We now define a broken k-dimensional polyhedron for $k \geq 2$.

Definition 6 A broken k-dimensional polyhedron *on x_m consists of*

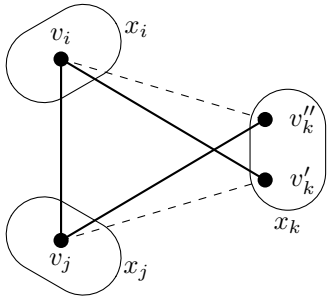

Figure 3: A broken triangle (v'_k, v_i, v_j, v''_k).

- *a consistent assignment $\langle v_1, \ldots, v_k\rangle$ to distinct variables $\langle x_{i_1}, \ldots, x_{i_k}\rangle$ (where each x_{i_j} $(j = 1, \ldots, k)$ is distinct from x_m),*
- *k distinct values $u_1, \ldots, u_k \in D_m$,*

such that

- $\forall j \in \{1, \ldots, k\}$, $(v_j, u_j) \notin R_{i_j m}$,
- $\forall h, j \in \{1, \ldots, k\}$, *if* $h \neq j$ *then* $(v_h, u_j) \in R_{i_j m}$,

The assignment $\langle v_1, \ldots, v_k\rangle$ to variables $\langle x_{i_1}, \ldots, x_{i_k}\rangle$ is known as the base *of the broken polyhedron, and each assignment $\langle x_m, u_j\rangle$ $(j = 1, \ldots, k)$ is an* apex.

A broken 3-dimensional polyhedron (i.e. a broken tetrahedron) is shown in Figure 4. A broken triangle (Figure 3) is a broken 2-dimensional polyhedron. We show that the notion of broken k-dimensional polyhedron allows us to define novel variable-elimination rules and tractable classes.

The broken-triangle property (BTP) has been generalised to the $\forall\exists$BTP rule for variable elimination which allows us to eliminate more variables [Cooper, 2014] than BTP. Eliminating a variable satisfying the $\forall\exists$BTP rule is strictly stronger than the BTP rule. This is demonstrated by the fact that $\forall\exists$BTP, but not BTP, subsumes the rule that allows us to eliminate a variable x_m when an assignment to x_m is compatible with all assignments to all other variables. Another generic example is when all occurrences of the broken-triangle pattern on variable x_m occur on pairs of values $v_m, v'_m \in S \subset \mathcal{D}(x_m)$ and each assignment v_i to each other variable $x_i \neq x_m$ has a support at x_m in $\mathcal{D}(x_m) \setminus S$.

We first given the definition of the $\forall\exists$ broken-triangle property [Cooper, 2014], in order to generalise it to k dimensions.

Definition 7 *A binary CSP instance satisfies the* $\forall\exists$ broken-triangle property *on variable x_m if for all $i_1 \neq m$, for all $v_1 \in \mathcal{D}(x_{i_1})$, there exists $v_m \in \mathcal{D}(x_m)$ such that*

1. *$\langle v_1, v_m\rangle$ is a consistent assignment to variables $\langle x_{i_1}, x_m\rangle$, and*
2. *for all $i_2 \notin \{i_1, m\}$, for all $v_2 \in \mathcal{D}(x_{i_2})$, there is no broken triangle on x_m with base the assignment $\langle v_1, v_2\rangle$ to variables $\langle x_{i_1}, x_{i_2}\rangle$ and with an apex $\langle x_m, v_m\rangle$.*

We now generalise the $\forall\exists$BTP rule for variable elimination to the case of broken polyhedra of any dimension $k \geq 2$. When $k = 2$ the following definition coincides with Definition 7 of the $\forall\exists$BTP rule.

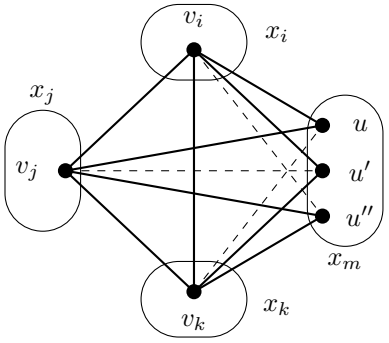

Figure 4: A broken tetrahedron.

Definition 8 *A binary CSP instance satisfies the* $\forall\exists$ broken k-dimensional polyhedron property *on variable* x_m *if for all distinct* $i_1, \ldots, i_{k-1} \neq m$*, for all consistent assignments* $\langle v_1, \ldots, v_{k-1}\rangle$ *to variables* $\langle x_{i_1}, \ldots, x_{i_{k-1}}\rangle$*, there exists* $v_m \in \mathcal{D}(x_m)$ *such that*

1. $\langle v_1, \ldots, v_{k-1}, v_m\rangle$ *is a consistent assignment to variables* $\langle x_{i_1}, \ldots, x_{i_{k-1}}, x_m\rangle$*, and*
2. *for all* $i_k \notin \{i_1, \ldots, i_{k-1}, m\}$*, for all* $v_k \in \mathcal{D}(x_{i_k})$*, there is no broken* k*-dimensional polyhedron on* x_m *with base the assignment* $\langle v_1, \ldots, v_k\rangle$ *to variables* $\langle x_{i_1}, \ldots, x_{i_k}\rangle$ *and with an apex* $\langle x_m, v_m\rangle$*.*

The first condition of Definition 8 is a k-consistency condition on variable x_m with respect to all other variables [Lecoutre, 2009]. The second condition guarantees that this k-consistency condition is sufficient for any consistent assignment to the variables $X \setminus \{x_m\}$ to be extendible to a consistent assignment to all variables.

Theorem 3 *The* $\forall\exists$ *broken* k*-dimensional polyhedron property is a sol-var-elim condition in binary CSP instances* I *with at least* k *variables.*

The $\forall\exists$ broken k-dimensional polyhedron property is interesting from a theoretical point of view. However, from a practical point of view, the time complexity of detecting whether variables can be eliminated is likely to be prohibitive. Indeed, for $k = 3$, a naive exhaustive search for broken tetrahedra has time complexity $\Theta(n^4d^6)$.

6 Theoretical Comparison of Rules

We say that two variable-elimination rules are incomparable if neither is subsumed by the other.

Proposition 3 *The following three variable-elimination rules are all pairwise incomparable: the DE-snake property, the triangle property and* $\forall\exists$*BTP.*

7 Variable-Elimination Rules and Tractability

We now investigate the possibility of defining tractable classes based on our variable-elimination rules. As is the case for BTP [Cooper *et al.*, 2010], the rules we have presented in this paper also define tractable classes that can be detected in polynomial time by successive elimination of variables.

Definition 9 *For a variable-elimination property* P*, we say that a binary CSP instance* I satisfies P for the variable order $<$ *if for each variable* x_m*, except for the first variable according to the order* $<$*,* I *satisfies the property* P *on* x_m *in the sub-instance of* I *restricted to the variables* x_i *such that* $x_i \leq x_m$*.*

Theorem 4 *The class of binary CSP instances* I *satisfying any of the following properties (for a possibly unknown ordering of its variables) can be detected and solved in polynomial time: the* $\forall\exists$ *broken* k*-dimensional polyhedron property (for any fixed* $k \geq 2$*), the DE-snake property,* $\forall\exists$*BTP, and the triangle property.*

When not all variables can be eliminated, we are interested in maximising the number of eliminated variables.

Theorem 5 *Maximising the number of variables that can be eliminated by any of the following rules can be achieved in polynomial time: the* $\forall\exists$ *broken* k*-dimensional polyhedron property (for any fixed* $k \geq 2$*), the DE-snake property and* $\forall\exists$*BTP. Maximising the number of variable eliminations by combining the triangle property and neighbourhood substitution [Freuder, 1991] can also be achieved in polynomial time.*

8 Discussion and Conclusion

In this paper we have given novel satisfiability-conserving variable-elimination rules for binary CSPs. In each case, if the instance is satisfiable, then a solution to the original instance can be recovered in low-order polynomial time from a solution to the reduced instance. The DE-snake rule can be applied until convergence in $O(ed^3)$ time, whereas the corresponding time complexity for the triangle rule is $O(end^3)$. However, it should be pointed out that the DE-snake rule inherits the disadvantage of the $\exists$snake rule that the number of solutions may actually increase after elimination of a variable [Cohen *et al.*, 2015]: for example, it allows us to eliminate the central variable in the two-colouring of a star graph which increases the number of solutions from 2 to 2^{n-1}.

Extensive experimental trials (reported in [Cooper *et al.*, 2019]) have confirmed that because of relatively high time complexity of each of the variable-elimination rules, they may only be tested exhaustively during preprocessing. Applying them in preprocessing allowed us to solve more benchmark instances than without, the triangle rule allowing us to eliminate more variables and hence solve more instances than the other rules. We observed that most variables eliminated by our rules have small domain size and/or small degree.

We generalised the notion of broken triangle to broken polyhedron, which may be of theoretical interest.

We have also shown that each of the variable-elimination rules allows us to define a novel hybrid tractable class by successive elimination of almost all variables. For each rule, this elimination order can be found in polynomial time, which we found surprising in the case of the triangle property.

Acknowledgments

This work was funded by the Agence Nationale de la Recherche projects ANR-16-C40-0028 and ANR-19-PI3A-0004 (Artificial and Natural Intelligence Toulouse Institute).

References

[Beigel and Eppstein, 1995] Richard Beigel and David Eppstein. 3-Coloring in Time $O(1.3446^n)$: A No-MIS Algorithm. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA*, pages 444–452. IEEE Computer Society, 1995.

[Bessière *et al.*, 2005] Christian Bessière, Jean-Charles Régin, Roland H. C. Yap, and Yuanlin Zhang. An optimal coarse-grained arc consistency algorithm. *Artif. Intell.*, 165(2):165–185, 2005.

[Cohen *et al.*, 2015] David A. Cohen, Martin C. Cooper, Guillaume Escamocher, and Stanislav Zivny. Variable and value elimination in binary constraint satisfaction via forbidden patterns. *J. Comput. Syst. Sci.*, 81(7):1127–1143, 2015.

[Cooper *et al.*, 2010] Martin C. Cooper, Peter G. Jeavons, and András Z. Salamon. Generalizing constraint satisfaction on trees: Hybrid tractability and variable elimination. *Artif. Intell.*, 174(9-10):570–584, 2010.

[Cooper *et al.*, 2016] Martin C. Cooper, Aymeric Duchein, Achref El Mouelhi, Guillaume Escamocher, Cyril Terrioux, and Bruno Zanuttini. Broken triangles: From value merging to a tractable class of general-arity constraint satisfaction problems. *Artif. Intell.*, 234:196–218, 2016.

[Cooper *et al.*, 2019] Martin C. Cooper, Achref El Mouelhi, and Cyril Terrioux. Variable elimination in binary CSPs. *J. Artif. Intell. Res.*, 66:589–624, 2019.

[Cooper, 2014] Martin C. Cooper. Beyond Consistency and Substitutability. In Barry O'Sullivan, editor, *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France*, volume 8656 of *Lecture Notes in Computer Science*, pages 256–271. Springer, 2014.

[Dechter, 1999] Rina Dechter. Bucket Elimination: A Unifying Framework for Reasoning. *Artif. Intell.*, 113(1-2):41–85, 1999.

[Freuder, 1991] Eugene C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In Thomas L. Dean and Kathleen R. McKeown, editors, *Proceedings of the 9th National Conference on Artificial Intelligence, Anaheim, CA, USA, 1991, Volume 1.*, pages 227–233. AAAI Press / The MIT Press, 1991.

[Jeavons *et al.*, 1998] Peter Jeavons, David A. Cohen, and Martin C. Cooper. Constraints, Consistency and Closure. *Artif. Intell.*, 101(1-2):251–265, 1998.

[Koubarakis, 2006] Manolis Koubarakis. Temporal CSPs. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 665–697. Elsevier, 2006.

[Kratsch *et al.*, 2016] Stefan Kratsch, Geevarghese Philip, and Saurabh Ray. Point Line Cover: The Easy Kernel is Essentially Tight. *ACM Trans. Algorithms*, 12(3):40:1–40:16, 2016.

[Larrosa and Dechter, 2003] Javier Larrosa and Rina Dechter. Boosting Search with Variable Elimination in Constraint Optimization and Constraint Satisfaction Problems. *Constraints*, 8(3):303–326, 2003.

[Lecoutre, 2009] Christophe Lecoutre. *Constraint Networks Techniques and Algorithms*. ISTE/Wiley, 2009.

[Newman *et al.*, 2018] Neil Newman, Alexandre Fréchette, and Kevin Leyton-Brown. Deep optimization for spectrum repacking. *Commun. ACM*, 61(1):97–104, 2018.

[Schrijver, 1999] Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, 1999.

[Subbarayan and Pradhan, 2004] Sathiamoorthy Subbarayan and Dhiraj K. Pradhan. NiVER: Non-increasing Variable Elimination Resolution for Preprocessing SAT instances. In *SAT 2004 - The 7th International Conference on Theory and Applications of Satisfiability Testing, Vancouver, Canada, Online Proceedings*, 2004.

[Zhang and Yap, 2011] Yuanlin Zhang and Roland H. C. Yap. Solving functional constraints by variable substitution. *TPLP*, 11(2-3):297–322, 2011.