Constraint Reductions

Olivier Bailleux[*,1] and Yacine Boufkhad[†,2]

[1]LIB Université de Bourgourgne
[2]IRIF CNRS-Université Paris Diderot

Abstract

In this commentary on the [BB03] paper, after recalling its context, we outline a classification of Constraints with respect to their deductive power regarding General Arc Consistency (GAC).

1   Introduction

It is not usual to be asked to write something on a subject on which you have worked fifteen years before and on which you have remained far from subsequent developments. Yet this is the challenge we humbly accepted to undertake in this paper with the limitation that our expertise in this research area have diminished regarding the developments made by many others during the ten previous years.

We begin with the easy part consisting in recalling the context of the paper [BB03].

In the sequel the word *constraint* will be used in two meanings: a type of constraints, such as propositional clause, Boolean cardinality constraint, ALLDIFF..., and a constraint instance, i.e., the formal representation of a relation on several variables each having a domain. We will use the word *Constraint* in the first case and *constraint* in the second case. For example, CNF is a Constraint that denotes the set of propositional clauses, whereas $(a \vee \neg b \vee c)$ is a CNF constraint.

The main result in [BB03] was twofold: the description of a general method of solving constraints satisfaction problems through translation and the application of this method to the translation of Boolean cardinality Constraint into CNF.

The general method can be quickly described as follows: We have a constraint network based on one or more types of so called *source Constraints*, and we want to solve this network using a solver dedicated to a type of so called *target Constraint*. Then we need to translate each source constraint to an equivalent set of target constraints. It is therefore desirable that the problem can be solved at least as effectively with the target solver as it would have been with an up to date source solver. The three most obvious criteria for achieving this objective are:

1. The translation process must be fast compared to the assumed resolution time of the source solver, namely, the translation algorithms must have a polynomial time complexity.

2. The target constraint network must not be of a prohibitive size.

3. All deductions that could be made by an up to date solver from source constraints must be feasible by the target solver.

Let us call for the moment a translation that fulfills the three above criteria a good translation. Regarding the third criterion, almost all modern constraint solvers at least enforce domain consistency, also called generalized arc consistency or GAC. We therefore considered, as [Gen02] before us, and as many authors thereafter, that it was desirable that enforcing domain consistency on target constraints produces at least the same deductions as enforcing domain consistency on source constraints.

We applied this method to the case of cardinality constraints as a source Constraint and clauses in CNF form as a target Constraint. Most of SAT solvers use unit propagation as a deduction method and therefore we required for the

---

[*]olivier.bailleux@u-bourgogne.fr
[†]yacine.boufkhad@irif.fr

third criterion that the translation restores the generalized arc consistency through unit propagation. At the time, it was known that a direct translation of cardinality constraints to CNF that preserves the power of deduction without adding any new variable requires an exponential number of clauses. Then a linear time translation was proposed in [War98], producing a polynomial number of clauses, but at the cost of a loss of deduction power since the unit propagation does not restore arc consistency.

We proposed then in [BB03] a good encoding (thus fulfilling the three above criteria) of cardinality constraints to CNF based on boolean adders using the unary representation of integers.

Many developments have been done since : refinements of the translation of cardinality constraints, CNF translation of general arithmetic constraints... It would be difficult to cite them without omitting some. Some papers studied the existence of good translations to CNF as a target Constraint. That is the subject we choose to tackle and generalize in this paper through some incomplete thoughts.

A breakthrough was made in [BKNW09] by giving a necessary and sufficient condition for a source Constraint to have a good translation to CNF. The condition is that a propagator that enforces generalized arc consistency on this source Constraint must be computed by a polynomial size monotone circuit. As a consequence of this result, a superpolynomial lower bound for the monotone circuit that encodes a propagator for some Constraint implies that there is no good translation to CNF for this Constraint. It is the case of ALLDIFF as mentioned [BKNW09] and also for its generalization GCC.

In the same vein, [GK14b] proves that there exists no GAC translation of systems of an arbitrary number of XOR equations. By contrast, it gives a good translation to CNF for XOR systems having a constant number of equations.

These results draw the outlines of classes of Constraints according to the existence of good translations to CNF. We describe in the following a classification that does not consider necessarily CNF as a target Constraint and we discuss it.

2   Comparing constraints according to their propagation power

Polynomial time constraints translation, in its simplest acceptance, i.e., the target constraint network and the source constraint network must be equiconsistent, is a Karp-reduction in the sense of complexity theory. It allows to polynomialy translate a source problem into an equivalent target problem. We propose to call *GAC-reduction* any Constraint reduction that preserves the deductive power of domain consistency enforcement. Such a reduction allows to categorize Constraints with respect to their propagation powers.

Let us illustrate the notions of reduction and GAC-reduction with an example involving two Constrains : CNF, namely the propositional clauses, and $\neq$, namely the binary difference constraints on finite domains.

Figure 1 shows a reduction from CNF to $\neq$ in the case where the source constraint is a clause of 3 literals, but which can be generalized to a clause of any length.
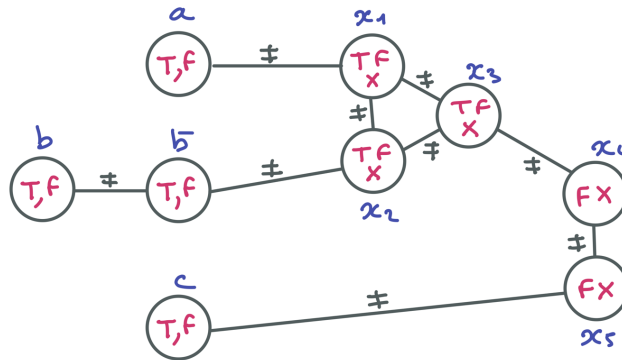

Figure 1: A (non-GAC) reduction from the clause $(a \vee \neg b \vee c)$ to binary difference constraints.

The variables a, b, c of the resulting target constraint network correspond to the variables of the source clause $(a \vee \neg b \vee c)$, with domains $\{F, T\}$, where T stand for true and F for false. The variables x_1 to x_5 are called auxiliary variables. If $a = F$, $b = T$ and $c = F$ the target network cannot be satisfied. For all other assignations of a, b, c, it can be satisfied.

But this reduction is *not* a GAC-reduction. For example, with $c = F$, $b = T$ and a unassigned, unit propagation, which enforces domain consistency on the source clause, assigns $a = T$. But on the target network, enforcing domain

consistency on the difference constraints does *not* assign a.

On the other hand, the reduction from CNF to $\neq$ presented figure in 2 on the same example *is* a GAC-reduction. If the source clause is falsified, then the target network in inconsistent and enforcing domain consistency on its constraints produces an empty domain. In addition to that, in the three cases where unit propagation on the source clause assigns a variable, enforcing domain consistency on constraints of the target network do the same assignation.

Incidentally, if we want to solve SAT instances with a graph coloring solver, it might be more efficient to use the second reduction rather than the first one.
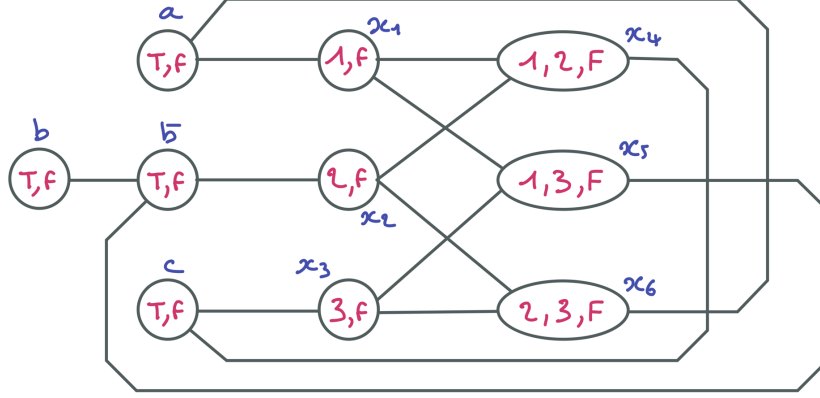

Figure 2: A GAC-reduction from the clause $(a \vee \neg b \vee c)$ to binary difference constraints.

GAC-reduction may involve different domains for the source variables and the related target variables. For example, to reduce a constraint $A \neq B$ in CNF, where A and B have domains $1..n$, the source variable A (B) can be represented in the target network by n variables $a_1, ..., a_n$ $(b_1, ..., b_n)$ with domains $\{\mathsf{T}, \mathsf{F}\}$, which is expressive enough to represent not only all possible assignments of A and B, but also all possible restrictions of their domains.

The GAC-reduction can be completed as follow:

1. A set $\mathrm{card}[1..1](a_1, ..., a_n)$ of clauses encoding that exactly one variable among $a_1, ..., a_n$ must be set to 1, see for example [Sin05] [HN13] for different GAC-compliant solutions.

2. A set $\mathrm{card}[1..1](b_1, ..., b_n)$ of clauses encoding that exactly one variable among $b_1, ..., b_n$ must be set to 1.

3. The clauses $(\neg a_1 \vee \neg b_1), ..., (\neg a_n \vee \neg b_n)$.

Here, not only GAC-reduction can introduce auxiliary variables (depending of the encoding of the card[1..1] constraints), but it may also use different representations of the possible restrictions of source domains and the corresponding target domains.

A formal definition of a GAC-reduction - which we will not give here - should express the idea, summarized in figure 3 that if, from a certain source knowledge K, the GAC-propagation of the source constraint produces a deduction D, then from the same knowledge expressed in the target representation, the GAC-propagation of the target constraint network should produce a deduction at least as strong as D. K and D are restrictions of the domains of the source variables. The corresponding target knowledge and deduction are restrictions of domains of target variables. A domain d is a restriction of another domain k if $d \subseteq k$.

The main result in [BKNW09] can be reformulated in terms of GAC-reduction this way: for any finite domains Constraint S, the GAC-propagator of S can be implemented by a monotone Boolean circuit if and only if there is a GAC-reduction from S to CNF. We identify here a class of Constraints that we call MPC (for Monotone Polynomially Propagatable Constraints), which is the set of all the Constraints that can be GAC-reduced to CNF. De facto, CNF is a MPC-complete Constraint.

All the generalizations of CNF belonging to MPC, such as Boolean cardinality Constraint (Card) and pseudo-Boolean inequalities ($\mathrm{PB}_>$), are obviously MPC-complete. (Card $\in$ MPC follows from the GAC-reduction proposed in [BB03] and $\mathrm{PB}_> \in$ MPC follows from the one proposed in [BBR09]).

It can be noticed that a necessary condition for a type Constraint Q be MPC-complete is that Q is NP-complete - in the sense that the problem of determining the consistency of a constraint network of type Q is NP-complete - because the existence of a GAC-reduction from Q to CNF implies the existence of a Karp-reduction from Q to CNF. So the XOR Constraint, which belongs to MPC but is not NP-complete, is therefore not MPC-complete.
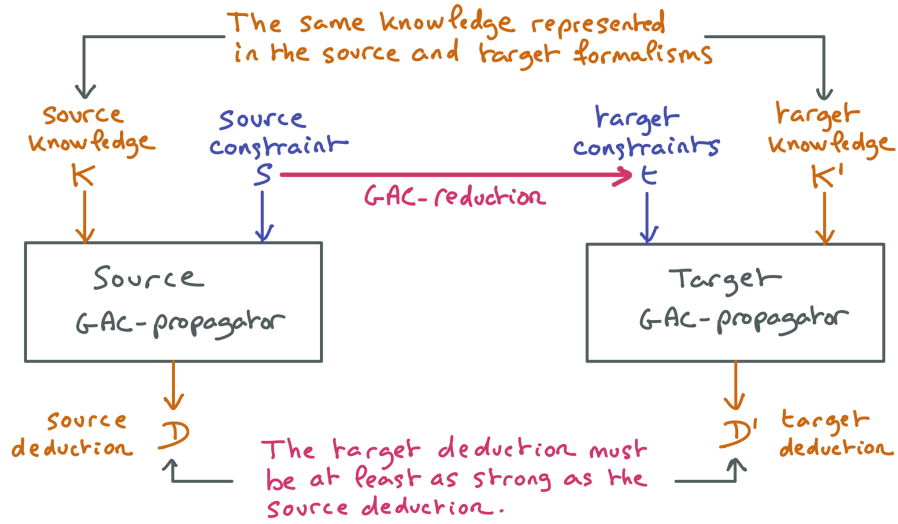
Figure 3: A schematic representation a GAC-reduction.

Also according to [BKNW09], there are Constraints, such as ALLDIFF and its generalization GCC (Global Cardinality Constraint [Ré96]), whose CAG-propagators can be computed in polynomial time, but which are not in MPC. We propose to call PPC (for Polynomially Propagatable Constraints) the set of all finite domain Constraints whose GAC-propagators have polynomial time complexity, and then can be implemented as polynomialy sized Boolean circuits. De facto, MPC is strictly included in PPC, and ALLDIFF and GCC are in PPC \ MPC. We do not know if there are PPC-complete Constraints, and the question seems interesting to us insofar as a solver optimized for such a Constraint could perhaps have a broader application scope than any solver dedicated to a MPC Constraint, such as SAT solvers or even pseudo-Boolean solvers.

Finally, it should be noted that there are Constraints whose satisfaction is verified in polynomial time, but for which enforcing domain consistency is a NP-hard problem. This holds with pseudo-Boolean inequalities ($PB_=$), for example. Under the assumption $P \neq NP$, these Constraints are not in PPC. We propose to call PVC the set of polynomialy verifiable Constraints, in the sense that a finite domain Constraint Q is in VPC if and only if there is a polynomial time complexity algorithm that checks whether a complete assignment satisfies a constraint q of type Q.

Figure 4 shows the classes MPC, PPC and VPC, as well as some Constraints belonging to these classes.
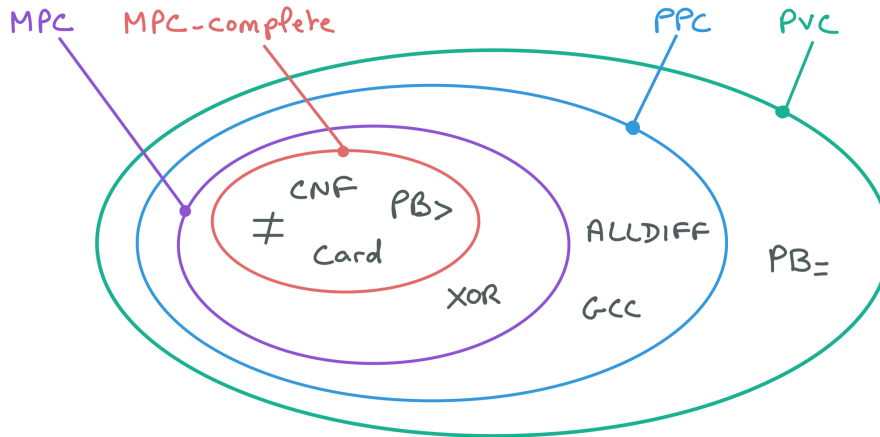

Figure 4: A classification of Constraints

3 Concluding remarks and questions

In a world where solvers are based on enumeration and domain consistency enforcement, a solver dedicated to a Constraint of the PPC \ MPC class has a higher deductive power than a solver dedicated to an MPC Constraint, such as a SAT solver, a graph coloring solver, or even a pseudo-Boolean solver. And in such a world, CNF is a poor candidate for the role of universal Constraint into which all polynomially propagatable Constraints could be translated

without loss of resolution efficiency. If we were to look for a universal solver, we would first have to identify the complete constraints of the PPC class. In the current state of our young reflections, we do not even know if such Constraints exist and so one question that could be addressed would be: are there Constraints complete for the class PPC?

But in a world where solvers use more sophisticated deductive systems, things are becoming more complex. In [GK14a] is addressed the question of power of deduction restricted to the representations of boolean functions but they consider also more sophisticated systems of deduction than GAC. The same can be done with Constraints with general domains. For example, the GAC-propagation of Constraints beyond MPC, such as GCC, cannot be polynomially simulated by unit propagation in a SAT solver or by GAC-propagation in a pseudo-Boolean solver. But what if the SAT solver replaces unit propagation with bounded resolution or if the pseudo-Boolean solver uses cutting planes?

Finally, we think that the study of good practices of constraint translation will have to be strongly connected to the study of solvers in order to progress.

References

[BB03]   Olivier Bailleux and Yacine Boufkhad. Efficient cnf encoding of boolean cardinality constraints. In *CP*, 2003. 1, 2, 3

[BBR09]   Olivier Bailleux, Yacine Boufkhad, and Olivier Roussel. New encodings of pseudo-boolean constraints into cnf. volume 5584, pages 181–194, 06 2009. 3

[BKNW09] Christian Bessiere, George Katsirelos, Nina Narodytska, and Toby Walsh. Circuit complexity and decompositions of global constraints. In *IJCAI*, 2009. 2, 3, 4

[Gen02]   Ian P. Gent. Arc consistency in sat. In *ECAI*, 2002. 1

[GK14a]   Matthew Gwynne and Oliver Kullmann. A framework for good SAT translations, with applications to CNF representations of XOR constraints. *CoRR*, abs/1406.7398, 2014. 5

[GK14b]   Matthew Gwynne and Oliver Kullmann. On sat representations of xor constraints. In Adrian-Horia Dediu, Carlos Martín-Vide, José-Luis Sierra-Rodríguez, and Bianca Truthe, editors, *Language and Automata Theory and Applications*, pages 409–420, Cham, 2014. Springer International Publishing. 2

[HN13]   Steffen Hölldobler and Van-Hau Nguyen. An efficient encoding of the at-most-one constraint. 2013. 3

[Ré96]   Jean-Charles Régin. Generalized arc consistency for global cardinality constraint. pages 209–215, 01 1996. 4

[Sin05]   Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In *CP*, 2005. 3

[War98]   Joost P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Information Processing Letters*, 68(2):63 – 69, 1998. 2