

Interleaved Alldifferent Constraints: CSP vs. SAT Approaches

Frédéric Lardeux³, Eric Monfroy^{1,2}, and Frédéric Saubion³

¹ Universidad Técnica Federico Santa María, Valparaíso, Chile

² LINA, Université de Nantes, France

³ LERIA, Université d'Angers, France

Abstract. In this paper, we want to handle multiple interleaved Alldiff constraints from two points of view: a uniform propagation framework with some CSP reduction rules and a SAT encoding of these rules that preserves the reduction properties of CSP.

1 Introduction

When modeling combinatorial problems, a suitable solution consists in expressing the problem as a constraint satisfaction problem (CSP), which is classically expressed by a set of decision variables whose values belong to finite integer domains. Constraints are used to model the relationships that exist between these variables. Another possibility is to encode the problem as a Boolean satisfiability problem (SAT), where a set of Boolean variables must satisfy a propositional formula.

The two paradigms share some common principles [2] and here, we focus on complete methods that aim at exploring a tree by enumerating variables and reducing the search space using propagation techniques.

On the CSP side, the identification of global constraints that arise in several real-world problems and the development of very specialized and efficient algorithms have considerably improve the resolution performances. The first example was certainly the Alldiff constraint [4] expressing that a set of n variables have all different values. Furthermore, one may notice that handling sets of distinct variables is often a more general problem and that, in some cases, such Alldiff constraints could be interleaved, leading to a high computational complexity. Many Alldiff constraints overlap in various problems such as Latin squares and Sudoku games. On the SAT side, no such high level modeling feature is offered to the user, who has to translate its problem into propositional logic. Systematic basic transformations from CSP to SAT have been proposed [3, 5] to ensure some consistency properties to the Boolean encodings.

Here, we want to provide, on the CSP solving side, a uniform propagation framework to handle Alldiff constraints and, in particular, interleaved Alldiff. We also want to generalize possible encodings of Alldiff and multiple Alldiff constraints in SAT (i.e., by a set of CNF formulas).

2 Encoding CSP vs. SAT

A CSP (X, C, D) is defined by a set of variables $X = \{x_1, \dots, x_n\}$ taking their values in their respective domains $D = \{D_1, \dots, D_n\}$. A constraint $c \in C$ is a relation $c \subseteq D_1 \times \dots \times D_n$. A tuple $d \in D_1 \times \dots \times D_n$ is a solution if and only if $\forall c \in C, d \in c$.

Usual resolution processes [1, 2] consists in enumerating the possible values of a given variable in order to progressively build a variables assignment and reach a solution. Reduction techniques are added at each node to reduce the search tree (local consistency mechanisms) by removing values of variables that cannot satisfy the constraints. We recall a basic consistency notion (the seminal arc consistency is the binary subcase of this definition).

Definition 1 (Generalized Arc Consistency (GAC)). *A constraint⁴ c on variables (x_1, \dots, x_m) is generalized arc-consistent iff $\forall k \in 1..m, \forall d \in D_k, \exists (d_1, \dots, d_{k-1}, d_{k+1}, \dots, d_m) \in D_1 \times \dots \times D_{k-1} \times D_{k+1} \times \dots \times D_m, s.t. (d_1, \dots, d_m) \in c$.*

Domain Reduction Rules

Inspired by [1], we abstract constraint propagation as a transition process over CSPs:

$$\frac{(X, C, D) | \Sigma}{(X, C, D') | \Sigma'}$$

where $D' \subseteq D$ and Σ and Σ' are first order formulas (i.e., conditions of the application of the rules) such that $\Sigma \wedge \Sigma'$ is consistent. We canonically generalize \subseteq to sets of domains as $D' \subseteq D$ iff $\forall x \in X D'_x \subseteq D_x$. Given a set of variables V , we also denote D_V the union $\bigcup_{x \in V} D_x$. $\#D$ is the set cardinality.

Given a CSP (X^k, C^k, D^k) , a transition can be performed to get a reduced CSP $(X^{k+1}, C^{k+1}, D^{k+1})$ if there is an instance of a rule (i.e., a renaming without variables' conflicts):

$$\frac{(X^k, C^k, D^k) | \Sigma^k}{(X^{k+1}, C^{k+1}, D^{k+1}) | \Sigma^{k+1}}$$

such that $D^k \models \bigwedge_{x \in X} x \in D_x^k \wedge \Sigma^k$, and D^{k+1} is the greatest subset of D^k such that $D^{k+1} \models \bigwedge_{x \in X} x \in D_x^{k+1} \wedge \Sigma^{k+1}$.

In the conclusion of a rule (in Σ), we use the following notations: $d \notin D_x$ means that d can be removed from the domain of the variable x (without loss of solution); similarly, $d \notin D_V$ means that d can be removed from each domain variables of V ; and $d_1, d_2 \notin D_x$ (resp. D_V) is a shortcut for $d_1 \notin D_x \wedge d_2 \notin D_x$ (resp. $d_1 \notin D_V \wedge d_2 \notin D_V$).

Since we only consider here rules that does not affect constraints and variables, the sets of variables will be omitted and we highlight the constraints that are

⁴ This definition is classically extended to a set of constraints.

required to apply the rules by restricting our notation to $\langle C, D \rangle$. We will say that $\langle C, D \rangle$ is GAC if C is GAC w.r.t. D . The transition relation using a rule R is denoted $\langle C, D \rangle \rightarrow_R \langle C, D' \rangle$. \rightarrow_R^* denotes the reflexive transitive closure of \rightarrow_R . It is clear that \rightarrow_R terminates due to the decreasing criterion on domains in the definition of the rules (see [1]). This notion can be obviously extended to sets of rules \mathcal{R} . Note also that the result of $\rightarrow_{\mathcal{R}}^*$ is independent from the order of application of the rules [1]: from a practical point of view, it is thus generally faster to first sequence less complicated rules (or rules that execute faster).

An instance of the SAT problem can be defined by a pair (Ξ, ϕ) where Ξ is a set of Boolean variables $\Xi = \{\xi_1, \dots, \xi_n\}$ and ϕ is a Boolean formula $\phi: \{0, 1\}^n \rightarrow \{0, 1\}$. The formula is said to be satisfiable if there exists an assignment $\sigma: \Xi \rightarrow \{0, 1\}$ satisfying ϕ and unsatisfiable otherwise. The formula ϕ is in conjunctive normal form (CNF) if it is a conjunction of clauses (a clause is a disjunction of literals and a literal is a variable or its negation).

In order to transform our CSP (X, D, C) into a SAT problem, we must define how the set Ξ is constructed from X and how ϕ is obtained. Concerning the variables, we use the direct encoding [5]: $\forall x \in X, \forall d \in D_x, \exists \xi_{x,d} \in \Xi$ ($\xi_{x,d}$ is true when x has the value d , false otherwise).

To enforce exactly one value for each variable, we use the next clauses:

$$\bigwedge_{x \in X} \bigvee_{d \in D_x} \xi_{x,d} \quad \text{and} \quad \bigwedge_{x \in X} \bigwedge_{\substack{d_1, d_2 \in D_x \\ d_1 \neq d_2}} (\neg \xi_{x,d_1} \vee \neg \xi_{x,d_2})$$

Our purpose is to define uniform transformation rules for handling multiple Alldiff constraints, which are often involved in many problems.

From the resolution point of view, complete SAT solvers are basically based on a branching rule that assign a truth value to a selected variable and unit propagation (UP) which allows to propagate unit clauses in the current formula [2]. We will say that a SAT encoding preserves a consistency iff all variables assigned to false by UP have their corresponding values eliminated by enforcing GAC.

3 Alldiff Constraints: Reduction rules and Transformation

In the following, we classically note $Alldiff(V)$ the Alldiff constraint on a subset of variables V , which semantically corresponds to the conjunction of $n*(n-1)/2$ pairwise disequality constraints $\bigwedge_{x_i, x_j \in V, i \neq j} x_i \neq x_j$.

A Single Alldiff constraint We first reformulate a well known consistency property [4] w.r.t. the number of values remaining in the domain of the variables. This case corresponds of course to the fact that if a variable has been assigned then the corresponding value must be discarded from other domains.

$$[O1] \quad \frac{\langle C \wedge Alldiff(V), D \rangle \mid x \in V \wedge D_x = \{d_1\}}{\langle C \wedge Alldiff(V), D' \rangle \mid d_1 \notin D'_{V \setminus \{x\}}}$$

Property 1. If $\langle Alldiff(V), D \rangle \xrightarrow{[O1]^*} \langle Alldiff(V), D' \rangle$, then the corresponding conjunction $\bigwedge_{x_i, x_j \in V} x_i \neq x_j$ is GAC w.r.t. $\langle D' \rangle$. Note that enforcing GAC on the disequalities with [O1] reduces less the domains than enforcing GAC on the global Alldiff constraint.

This rule can be generalized when considering a subset V' of m variables with m possible values, $1 \leq m \leq (\#V - 1)$:

$$[Om] \quad \frac{\langle C \wedge Alldiff(V), D \rangle | V' \subset V \wedge D_{V'} = \{d_1, \dots, d_m\}}{\langle C \wedge Alldiff(V), D' \rangle | d_1, \dots, d_m \notin D'_{V \setminus V'}}$$

Consider $m = 2$, and that two variables of an Alldiff only have the same two possible values. Then, these two values cannot belong to the domains of the other variables.

Property 2. Given $\langle Alldiff(V), D \rangle \xrightarrow{[Om]_{1 \leq m \leq (\#V - 1)}^*} \langle Alldiff(V), D' \rangle$, then $\langle Alldiff(V), D' \rangle$ has the GAC property.

Now, the Alldiff constraints can be translated in SAT, by encoding [O1] with a set of $\#V * (\#V - 1)$ CNF clauses:

$$[SAT - O1] \quad \bigwedge_{x \in V} \bigwedge_{y \in V \setminus \{x\}} (\neg \xi_x^d \vee (\bigvee_{f \in D_x \setminus \{d\}} \xi_x^f) \vee \neg \xi_y^d)$$

This representation preserves GAC. Indeed, if $\neg \xi_x^d$ is false (i.e. when the variable x is valued to d) and $\bigvee_{f \in D_{x_1} \setminus \{d\}} \xi_{x_1, f}$ is false (i.e., when the variable x_1 is valued to d) then $\neg \xi_{x_2, d}$ must be true to satisfy the clause (x_2 cannot be valued to d). Generalized to a subset V' of m variables $\{x_1, \dots, x_m\}$ with m possible values $\{d_1, \dots, d_m\}$, $1 \leq m \leq (\#V - 1)$, the corresponding $\#(V \setminus V') * m^{m+1}$ clauses are:

$$[SAT - Om] \quad \bigwedge_{y \in V \setminus V'} \bigwedge_{k=1}^m \bigwedge_{p_1=1}^m \dots \bigwedge_{p_m=1}^m [(\bigvee_{s=1}^m \neg \xi_{x_s}^{d_{p_s}}) \vee (\bigvee_{i=1}^m \bigvee_{f \in D_{x_i} \setminus \{d_1, \dots, d_m\}} \xi_{x_i}^f) \vee \neg \xi_y^{d_k}]$$

Property 3. $\bigcup_{1 \leq m \leq \#V - 1} [SAT - Om]$ preserves the GAC property.

Multiple Overlapping Alldiff Constraints In presence of several overlapping Alldiff constraints, specific local consistency properties can be enforced according to the number of common variables, their possible values, and the number of overlaps. To simplify, we consider Alldiff constraints $Alldiff(V)$ such that $\#V = \#D_V$.

If a value appears in variables of the intersection of two Alldiff, and that it does not appear in the rest of one of the Alldiff, then it can be safely removed from the other variables' domains of the second Alldiff.

$$[OI2] \quad \frac{\langle C \wedge Alldiff(V_1) \wedge Alldiff(V_2), D \rangle | d \in D_{V_1 \cap V_2} \wedge d \notin D_{V_2 \setminus V_1}}{\langle C \wedge Alldiff(V_1) \wedge Alldiff(V_2), D' \rangle | d \notin D'_{V_1 \setminus V_2}}$$

[OI2] is coded in SAT as $\#D_{V_1 \cap V_2} * \#(V_1 \cap V_2) * \#(V_1 \setminus V_2)$ clauses:

$$[SAT - OI2] \quad \bigwedge_{d \in D_{V_1 \cap V_2}} \bigwedge_{x_1 \in V_1 \cap V_2} \bigwedge_{x_2 \in V_1 \setminus V_2} \bigvee_{x_3 \in V_2 \setminus V_1} (\neg \xi_{x_1}^d \vee \xi_{x_2}^d \vee \neg \xi_{x_3}^d)$$

[OI2] can be extended to [OIm] to handle m ($m \geq 2$) *Alldiff* constraints connected by one intersection. Let denote by V the set of variables appearing in the common intersection: $V = \bigcap_{i=1}^m V_i$

$$[OIm] \quad \frac{\langle C \bigwedge_{i=1}^m \text{Alldiff}(V_i), D \rangle \mid d \in D_V \wedge d \notin D_{V_1 \setminus V}}{\langle C \bigwedge_{i=1}^m \text{Alldiff}(V_i), D' \rangle \mid d \notin \bigcup_{i=1}^m D'_{V_i \setminus V}}$$

Note that this rule can be implicitly applied to the different symmetrical possible orderings of the m Alldiff.

[OIm] is translated in SAT as $\#D_V * \#V * \sum_{i=2}^m (\#(V_i \setminus V))$ clauses:

$$[SAT - OIm] \quad \bigwedge_{d \in D_V} \bigwedge_{x_1 \in V} \bigwedge_{i=2}^m \bigwedge_{x_3 \in V_i \setminus V} \bigvee_{x_2 \in V_1 \setminus V} (\neg \xi_{x_1}^d \vee \xi_{x_2}^d \vee \neg \xi_{x_3}^d)$$

Property 4. Consider $m > 2$ Alldiff with a non empty intersection. Given $\langle C, D \rangle \xrightarrow{[OIm]} \langle C, D' \rangle$ and $\langle C, D \rangle \xrightarrow{[OI2]} \langle C, D'' \rangle$, then $D'' \subseteq D'$.

Other rules can be defined for several Alldiff connected by several intersections.

4 Conclusion

We have defined a set of consistency rules for general Alldiff constraints that can be easily implemented in usual constraint solvers. Recent works deal with the combination of several global constraints. Nevertheless, these approaches require some specialized and complex algorithms for reducing the domains, while our approach allows us to simplify and unify the presentation of the propagation rules and attempts at addressing a wider range of possible combinations of Alldiff. The basic encodings of CSP into SAT have been fully studied to preserve consistency properties and induce efficient unit propagation in SAT solvers. Our transformation is based on the reduction rules and extended to multiple connected Alldiff. As some of these works we proved that it is correct w.r.t. GAC. This work provides then an uniform framework to handle interleaved Alldiff and highlights the relationship between CSP and SAT in terms of modeling and resolution when dealing with global constraints.

References

1. K. Apt. *Principles of Constraint Programming*. Cambridge University Press, 2003.
2. L. Bordeaux, Y. Hamadi, and L. Zhang. Propositional satisfiability and constraint programming: A comparative survey. *ACM Comput. Surv.*, 38(4):12, 2006.
3. I. Gent. Arc consistency in SAT. Technical Report APES-39A-2002, University of St Andrews, 2002.
4. J.C. Régin. A filtering algorithm for constraint of difference in csp. In *National Conference of Artificial Intelligence*, pages 362–367, 1994.
5. T. Walsh. SAT v CSP. In *Proc. of CP 2000*, volume 1894 of *LNCS*, pages 441–456. Springer, 2000.