A SYSTEMATIC OVERVIEW OF THE ALL DIFFERENT CONSTRAINT

M. DEVIKA[1] & Prof G. SHOBHA LATHA[2]

[1]RESEARCH SCHOLAR, DEPARTMENT OF MATHEMATICS, SRI KRISHNA DEVARAYA UNIVERSITY, ANANTAPURAMU, A. P. INDIA.
[2]PROFESSOR, DEPARTMENT OF MATHEMATICS, SRI KRISHNA DEVARAYA UNIVERSITY, ANANTAPURAMU, A.P. INDIA.

Abstract: The most significant advancements over time with relation to every type of limitation. We begin by outlining the fundamental ideas from combinatorial theory. Next, we present a summary and a comparative analysis of various propagation methods that achieve hyper-arc consistency, limits, range, and arc consistency, respectively. Furthermore, the symmetric all different constraint and the least weight all different constraint are two variations of the all different constraint that are examined.
Key words: symmetric, constraint, and combinatorial.

1. Introduction

The well-known "all different" constraint, which mandates that every variable in the constraint be pairwise different, is one of the constraints found in almost all constraint programming systems.

The significance of these "disequality constraints" was understood back in the early days of constraint programming. Lauriere [1], for instance, presented ALICE, "A language and a program for stating and solving combinatorial problems," in the 1970s. In this system, if a set of variables is given the keyword "DIS," it indicates that the variables must have distinct values. It specifies a global structure that is used to the advantage of the solution-finding process.

It was also possible to define the constraint of difference as the well-known all different constraint after the advent of constraints in logic programming, for instance in the system CHIP [Dincbas, Van Hentenryck, Simonis, Aggoun, Graf, and Berthier[2]. Wallace, Novello, and Schimpf[3] established this requirement as all distinct in the system Eclipse. Nonetheless, all of the various constraints were internally handled as a series of disequalities in the early constraint (logic) programming systems; for instance, see Van Hentenryck [4]. Sadly, in doing so, the global information is lost. The propagation algorithm proposed by Regin [5], which takes into account all disequalities at once, was used to get the global view.

The many constraints have been important throughout the development of constraint programming. This unique constraint is used in a number of articles and books to demonstrate the advantages of constraint programming, either by demonstrating the modeling power of the technique or by demonstrating how much faster issues can be solved when employing it. From a modeling perspective, the need for disjoint circuits to span a directed graph or issues based on permutations easily give rise to the all different constraint. There are many applications where each distinct restriction is crucial, such as Gomes and Shmoys' quasi-group completion problems [6], Barnier and Brisset's air traffic management [7, 8], Gronkvist's [9], and Tsang, Ford, Mills, Bradwell, Williams, and Scott's rostering difficulties [8]. Last but not least, numerous additional worldwide.

The various constraints have been thoroughly examined in constraint programming over the years. As we'll show, there are at least six distinct propagation methods for the all-different constraint, each of which achieves local consistency in a unique way or more quickly. Numerous algorithms in this list are based on operations research methodologies, such as matching theory and flow theory. With relation to the purview of this thesis, such propagation algorithms are particularly interesting.

The same underlying ideas are frequently used by the propagation algorithms. This chapter provides a comprehensive overview of all the constraints, which is likely to be considered the most well-known, influential, and extensively researched constraint in the subject of constraint programming, in an effort to make them more comprehensible, accessible, and cohesive.

We showcase findings from combinatorial theory concerning the various constraints. These results are crucial to many of the propagation methods under consideration. We provide formal definitions for various concepts of local consistency that are utilized in relation to the various constraints.

The treatment comprises an explanation of the specific concept of local consistency about the all-different constraint, in addition to an algorithmic description that accomplishes that local consistency.

Initially, a symmetric variant of every restriction is examined. Next, a linear objective function is introduced and used in conjunction with the all different constraint to exploit the weighted all different constraint.

takes into account the all different polytope, which is a specific description of the all different constraint's solution set. Constraints on combinational backgrounds. Models involving integer linear programming can make use of this explanation.

2 .Combinatorial Background

2.1 All different and Bipartite Matching

The equivalence of a solution to the all different constraint and a matching in a bipartite graph.

Definition 2.1.1 (All different constraint). Let $x_1, x_2, \ldots. x_n$ be variables with respective finite domains $D_1,D_2,\ldots..D_n$. Then all different $(x_1,\ldots..x_n) = \{(d_1,\ldots.d_n) \mid d_1 \varepsilon D_t, d_1 \neq d_j \text{ for } i \neq j\}$.

Definition 2.1..2 (Value graph). Let $X = x_1, x_2,\ldots.x_n$ be a sequence of variables with respective finite domains $D_1,D_2,\ldots\ldots.D_n$. The bipartite graph $G = (X \cup D_X,E)$ with $E = \{x_i d \mid d \varepsilon D_i\}$ is called the value graph of X.

Theorem 2.2 Let $X = x_1, x_2,\ldots..,x_n$ be a sequence of variables with respective finite domains $D_1,D_2,\ldots\ldots,D_n$. Let G be the value graph of X. Then $(d_1,\ldots\ldots,d_n) \in$ all different $(x_1,\ldots..,x_n)$ if and only if $M = \{x_1d_1,\ldots..,x_nd_n\}$ is a matching in G.

Proof An edge x_id_i (for some $i \in \{1,\ldots..,n\}$) in M corresponds to the assignment $x_i = d_i$. As no edges in M share a vertex, $x_i \neq x_j$ for all $i \neq j$.

Example .1 We want to assign four tasks (1, 2, 3 and 4) to five machines (A, B, C, D and E). To each machine at most one task can be assigned. However, not every task can be assigned to every machine. Table 3.1 below presents the possible combinations. For example, task 2 can be assigned to machines B and C.

Task	Machines
1	B, C, D, E
2	B, C
3	A, B, C, D
4	B,C

Table 1. Possible task machine combinations

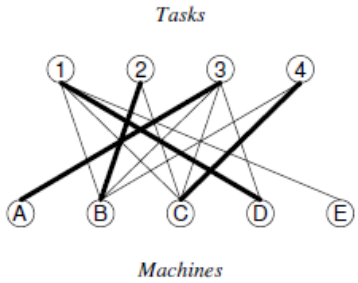


Figure .1. The value graph for the task assignment problem of Example 2.3. Bold edges form a matching covering all tasks.

This problem is modelled as follows. We introduce a variable x_i for task i = 1,……,4 whose value represents the machine to which task i is assigned.

The initial domains of the variables are defined by the possible combinations in Table .1. Since the tasks have to be assigned to different machines, we introduce an all different constraint. The problem is thus modelled as the CSP

$$x_1 \in \{B, C, D, E\}, x_2 \in \{B, C\}, x_3 \in \{A, B, C, D\}, x_4 \in \{B, C\},$$
$$\text{all different } (x_1, x_2, x_3, x_4):$$

The value graph of $X = x_1,\ldots,x_n$ is presented in Figure 3.1. The bold edges in the value graph denote a matching covering X. It corresponds to a solution to the CSP, i.e. $x_1 = D$, $x_2 = B$, $x_3 = A$ and $x_4 = C$.

2.3 Hall's Marriage Theorem

A useful theorem to derive constraint propagation algorithms for the all-different constraint is Hall's Marriage Theorem1, Hall,[14].

If a group of men and women marry only if they have been introduced to each other previously, then a complete set of marriages is possible if and only if every subset of men has collectively been introduced to at least as many women, and vice versa[2].

The following formulation is stated in terms of the all different constraint.

Theorem 2.4. Let $x_1, x_2,\ldots\ldots, x_n$ be variables with respective definite domains $D_1, D_2, \ldots\ldots, D_n$. The constraint all different $(x_1,\ldots\ldots,x_n)$ has a solution if and only if

$$|K| \leq |D_k| \qquad (1)$$

for all $K \subseteq \{x_1,\ldots..,x_n\}$.

Proof. The direct proof presented here is adapted from Schrijver [15], and originally due to Easterfield [16]. Call a set K tight if equality holds in (3.1). Necessity of the condition being obvious, we prove sufficiency. We use induction, with the hypothesis that Theorem 3.4 holds for k variables with k < n.

If there is a $d \in D_n$ such that

$$x_1 \in D_1 \setminus \{d\}, \ldots, x_{n-1} \in D_{n-1} \setminus \{d\}, \qquad (2)$$
$$\text{all different}(x_1, \ldots, x_{n-1})$$

has a solution, then we are done. Hence, we may assume the opposite, i.e. in (3.2), for each $d \in D_n$, there exists a subset $K \subseteq \{1, \ldots, n\text{-}1\}$ with $|K| > |D_K \setminus \{d\}|$. Then, by induction, all different$(x_1, \ldots, x_{n-1})$, with $x_1 \in D_1, \ldots, x_{n-1} \in D_{n-1}$, has a solution if and only if for each $d \in D_n$ there is a tight subset $K \subseteq \{1, \ldots, n\text{-}1\}$ with $d \in D_K$. Choose any such tight subset K.

Without loss of generality, $K = \{1, \ldots, k\}$. By induction, all different $(x_1, \ldots, x_k)$ has a solution, using all values in D_K . Moreover,

$$x_{k+1} \in D_{k+1} \setminus D_K , \ldots , x_n \in D_n \setminus D_K ,$$

$$\text{all different } (x_{k+1}, \ldots, x_n)$$

has a solution. This follows inductively, since for each $L \subseteq \{k + 1, \ldots, n\}$,

$$\left|U_{i\in L}(D_i / D_k)\right| = \left|U_{i\in KUL} D_i\right| - |D_k| \geq |KUL| - |D_k| = |K| + |L| - |D_k| = |L|$$

where the "≥" relation is obtained by applying condition (3.1). Then all different $(x_1, \ldots, x_n)$ has a solution, using all values in $D_K \cup D_L$. ¤

The following example shows an application of Theorem 3.4.

Example .2. Consider the following CSP

$$x_1 \in \{2, 3\}, x_2 \in \{2, 3\}, x_3 \in \{1, 2, 3\}, x_4 \in \{1, 2, 3\},$$

$$\text{all different } (x_1, x_2, x_3, x_4).$$

For any $K \subseteq \{x_1, x_2, x_3, x_4\}$ with $|K| \leq 3$, we have $|K| \leq |D_K|$.

For $K = \{x_1, x_2, x_3, x_4\}$ however, $|K| > |D_K|$, and by Theorem 3.4 this CSP has no solution. ¤

Theorem 3. The constraint all different $(x_1, \ldots, x_n)$ is bounds consistent if and only if $|D_i| \geq 1$ $(i = 1, \ldots, n)$ and

i) for each interval I: $|K_I| \leq |I|$,

ii) for each Hall interval I: $\{\min D_i, \max D_i\} \cap I = \emptyset$ for all $x_i \notin K_I$.

Proof. Let I be a Hall interval and $x_i \notin K_I$. If all different $(x_1, \ldots, x_n)$ is bounds consistent, it has a solution when $x_i = \min D_i$, by Definition 3.7. From Theorem 3.4 immediately follows that $\min D_i \notin I$. Similarly for $\max D_i$.

Conversely, suppose all different $(x_1, \ldots, x_n)$ is not bounds consistent. Thus, there exist a variable x_i and a value $d_i \in \{\min D_i, \max D_i\}$ for some $i \in \{1, \ldots, n\}$, such that all different $(x_1, \ldots, x_i-1, x_i+1, \ldots, x_n)$ has no solution, where $x_j \in D'_j$

$= [\min(D_j \setminus \{d_i\}), \max(D_j \setminus \{d_i\})]$ for all $j = i$.

By Theorem 3.4, there exists some $K \subseteq \{x_1,\ldots,x_n\}\setminus\{x_i\}$ such that $|K| > |D'_k|$

Choose $I=D'_k$ and consider K_I with respect to all different $(x_1, \ldots, x_n)$.

Then either I is a Hall interval and $d_i \in K_I$, or $|K| > |K_I|$.

Example .1 Consider the following CSP

$$x_1 \in \{1, 2\}, x_2 \in \{1, 2\}, x_3 \in \{2, 3\},$$
$$\text{all different } (x_1, x_2, x_3).$$

Observe that the variables x_1 and x_2 both have domain $\{1, 2\}$, Hence, values 1 and 2 cannot be assigned to any other variable and therefore, value 2 should be removed from D_3.

The algorithm detects this when the interval I is set to I = [1, 2]. Then the number of variables for which $D_i \subseteq I$ is 2. Since $|I| = 2$, I is a Hall interval. The domain of x_3 is not in this interval, and $\{\min D_3, \max D_3\} \cap I = \{\min D_3\}$. In order to obtain the empty set in the right hand side of the last equation, we need to remove $\min D_i$. The resulting CSP is bounds consistent.

Construct an algorithm that achieves bounds consistency on the all different constraint. Consider all intervals I = [l, u] where l ranges over all minimum domain values and u over all maximum do main values. There are maximally n^2 such intervals, as there are n variables. Count the number of variables that are contained in I. If a Hall interval is detected, update the bounds of the appropriate variables. This can be done in O(n) steps. Hence the time complexity of this algorithm is $O(n^3)$.

However, it is more efficient to first sort the variables. To update the minimum domain values, we sort the variables in increasing order of their maximum domain value. Then we maintain an interval by the minimum and maximum bounds of the variables, which are inserted in the above order. Whenever we detect a maximum-length Hall interval, we update the bounds of the appropriate variables, reset the interval and continue with the next variable. To update the maximum domain values, we can apply the same method after interchanging $[\min D_i, \max D_i]$ with $[-\max D_i, -\min D_i]$ for all i.

The sorting of the variables can be done in O(n log n) time. As shown by Puget [17], we can use a balanced binary tree to keep track of the number of variables within an interval, which allows updates in O(log n) per variable. Hence the total time complexity reduces to O(n log n).

An improvement on top of this was suggested and implemented by Lopez- Ortiz, Quimper, Tromp, and van Beek [18]. They noticed that while keeping track of the number of variables within an interval, some of the used counters are irrelevant. They give two implementations, one with a linear running time plus the time needed to sort the variables, and one with an O(n log n) running time. Their experiments show that the latter algorithm is faster in practice.

A different algorithm for achieving bounds consistency of the all different constraint was presented by Mehlhorn and Thiel [19]. Instead of Hall intervals, they exploit the correspondence with finding a matching in a bipartite graph, similar to the algorithm presented in Section 3.4.4. Their algorithm runs in O(n) time plus the time needed to sort the variables according to the bounds of the domains.

Although the worst-case time complexity of the above algorithms is always O(n log n), under certain conditions the sorting can be performed in linear time, which makes the algorithms by Lopez-Ortiz et al. [20] and Mehlhorn and Thiel [21] run in linear time. This is the case in many practical instances, for example when the variables encode a permutation.

3.1 Range Consistency

Introduced an algorithm that achieves range consistency for the all different constraint. To explain this algorithm we follow the same procedure as in the previous subsection. Again we use Hall's Marriage Theorem to construct the algorithm.

Definition 3.2 (Hall set). Let $x_1, x_2, \ldots, x_n$ be variables with respective finite domains $D_1, D_2, \ldots, D_n$. Given $K \subseteq \{x_1, \ldots, x_n\}$, define the interval $I_K = [\min D_K, \max D_K]$. K is a Hall set if $|K| = |I_K|$.

Note that in the above definition IK does not necessarily need to be a Hall interval, because $K_{IK} = \{x_i \mid D_i \subseteq I_K\} \supseteq K$

Theorem 3.3. The constraint all different $(x_1, \ldots, x_n)$ is range consistent if and only if $|D_i| \geq 1$ $(i = 1, \ldots, n)$ and $D_i \cap I_K = \emptyset$ for each Hall set $K \subseteq \{x_1, \ldots x_n\}$ and each $x_i \notin K$.

Proof. Let K be a Hall set and $x_i \in/ K$. If all different $(x_1, \ldots, x_n)$ is range consistent, it has a solution when $x_i = d$ for all $d \in D_i$,

Conversely, suppose all different $(x_1, \ldots, x_n)$ is not range consistent.
Thus, there exist a variable x_i and a value $d_i \in D_i$ for some $i \in \{1, \ldots, n\}$, such that all different $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ has no solution, where $x_j \in D'_j = [\min D_j \setminus \{d_i\}, \max D_j \setminus \{d_i\}]$ for all $j \neq i$. , there is some K. Note that $D'K = I_K$.
Consider I_K with respect to all different $(x_1, \ldots, x_n)$. If K is a Hall set, then $d_i \in I_K$. Otherwise, $|K| > |I_K|$. Then either some domain is empty, or K contains a Hall set K′, and $D_j \cap I_{K'} \neq \emptyset$ for some $x_j \in K \setminus K'$.

We can deduce a first propagation algorithm in a similar way as we did for bounds consistency. Namely, consider all intervals I = [l, u] where I ranges over all minimum domain values and u over all maximum domain values. Then we count again the number of variables that are contained in I. If a Hall set is detected, we update the domains of the appropriate variables. This is one in O(n) steps. Hence the time complexity of this algorithm is $O(n^3)$, as there are again maximally n^2 intervals to consider.

A faster algorithm is presented by Leconte . We first sort (and store) the variables twice, according to their minimum and maximum domain value, respectively. The main loop considers the variables ordered by their maximum domain value. For each such variable, we maintain the interval IK and start adding variables to K. For this we consider all variables (inner loop), now sorted by their minimum domain value. When we detect a Hall set, updating the domains can be done in O(1) time, either within or after the inner loop, and we proceed with the next variable. As sorting the variables can be done in O(n log n) time, the total time complexity of this algorithm is $O(n^2)$. This time complexity is optimal, as is illustrated in the following example, taken from Leconte .

Example.3 Consider the following CSP

$$x_i \in \{2i + 1\} \quad \text{for } i = 0, 1, \ldots, n,$$
$$x_i \in \{0, 1, \ldots, 2n + 2\} \quad \text{for } i = n + 1, n + 2, \ldots, 2n,$$
$$\text{all different } (x_0, x_1, \ldots, x_{2n}).$$

In order to make this CSP range consistent, we have to remove the n + 1 first odd integers from the domains of the n variables whose domain is not yet a singleton. This takes $O(n^2)$ time. ¤

Observe that this algorithm has an opposite viewpoint from the algorithm for bounds consistency, although it looks similar. Where the algorithm for bounds consistency takes the domains (or intervals) as a starting point, the algorithm for range consistency takes the variables instead. But they both attempt to reach a situation in which the cardinality of a set of variables is equal to the cardinality of the union of the corresponding domains.

4. Hyper-arc Consistency

A hyper-arc consistency propagation algorithm for the all different constraint was proposed by Regin[5]. The algorithm is based on matching theory. We first give a characterization in terms of Hall's Marriage Theorem.

Definition 4.1 (Tight set). Let $x_1, x_2, \ldots, x_n$ be variables with respective finite domains $D_1, D_2, \ldots, D_n$. $K \subseteq \{x_1, \ldots, x_n\}$ is a tight set if $|K| = |DK|$.

Theorem .4.2 The constraint all different $(x_1, \ldots, x_n)$ is hyper-arc consistent if and only if $|D_i| \geq 1$ $(i = 1, \ldots, n)$ and $D_i \cap D_K = \emptyset$ for each tight

set $K \subseteq \{x_1, \ldots x_n\}$ and each $x_i \in/ K$.

Proof. Let K be a tight set and $x_i \notin K$. If all different $(x_1, \ldots, x_n)$ is hyper- arc consistent, it has a solution when $x_i = d$ for all $d \in D_i$, by Definition 3.6.

immediately follows that $D_i \cap D_K = \emptyset$.

Conversely, suppose all different $(x_1, \ldots, x_n)$ is not hyper-arc consistent.

Thus, there exist a variable x_i and a value $d_i \in D_i$ for some $i \in \{1, \ldots, n\}$, such that all different $(x_1,\ldots,x_{i-1}, x_{i+1},\ldots\ldots,x_n)$ has no solution, where

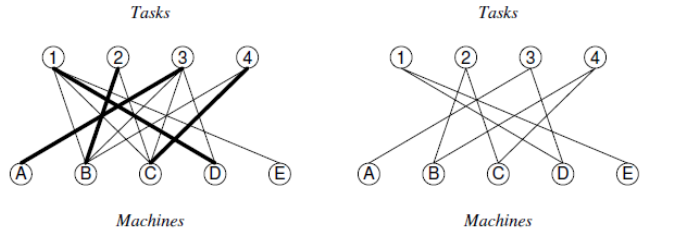
$X_j \in D'_j = D_j \setminus \{d_i\}$ for all $j \neq i$. By theorem 3.4 $|K| > |D'_j|$

$K \subseteq \{x_1, \ldots, x_n\} \setminus \{x_i\}$. If K is a tight set with respect to all different $(x_1, \ldots, x_n)$, then $d_i \in D_K$. Otherwise, $|K| > |D_K|$. Then either some domain is empty, or K contains a tight set K_0, and $D_j \cap D_{K0} = \emptyset$ for some $x_j \in K \setminus K_0$.

The all different constraint can be made hyper- arc consistent by generating all tight sets K and updating $D_i = D_i \setminus D_K$ for all $x_i \in/ K$. This approach is similar to the algorithms for achieving bounds consistency and range consistency. For bounds consistency and range consistency we could generate the respective Hall intervals and Hall sets rather easily because we were dealing with intervals containing the domains. In order to generate tight sets similarly we should consider all possible subsets $K \subseteq \{x_1, \ldots, x_n\}$. As the number of subsets is exponential in n, this approach is not practical. A different, more constructive, approach makes use of matching theory to update the domains, and was introduced by Regin [5].

Theorem4.3. Let G be a graph and M a maximum-size matching in G. An edge belongs to a maximum-size matching in G if and only if it either belongs to M, or to an even M alternating path starting at an M free vertex, or to an M alternating circuit.

Proof. Let M be a maximum-size matching in G=(V, E). Suppose edge e belongs to a maximum-size matching N, and $e \notin M$. The graph $G^0=(V, M \cup N)$ consists of even paths (possibly of length 0) and circuits with edges alternatingly in M and N. If the paths are not of even length, M or N can be made larger by interchanging edges in M and N along this path (a contradiction because they are of maximum size).
Conversely, let M be a maximum-size matching in G. By interchanging edges in M and not in M along even M alternating paths starting at an M free vertex and M alternating circuits we obtain matching of maximum size again. ¤
consider a tight set $K \subset \{x_1, \ldots, x_n\}$ of minimum size. The edges between vertices in K and in D_K form M -alternating circuits in the value graph. we remove those edges xid with xi K and $d \in D_K$, i.e. $D_i \cap D_K = \emptyset$. This corresponds to applying Theorem 3.20.



a. A matching covering all tasks. b. The bipartite graph after propagation.

Figure .2. The bipartite graph

5. Variants of the All different Constraint

This section presents two variants of the all different constraint: the symmetric all different constraint and the weighted all different constraint.

5.1 The Symmetric All different Constraint

Regin [5] developed the symmetric all different constraint, a specific instance of the all different constraint. Assumed to represent the same set of elements are the variables and their respective domain values. If the variable representing element i is assigned to the value representing element j, then the variable representing element j must be assigned to the value representing element i, according to the symmetric all different constraint, which mandates that all variables take different values. Below is a more formal definition that is provided.

It is very appropriate to use the symm all different restriction on round-robin tournament problems. In situations like this, like a sporting event, every team needs to be paired with another team. The problems are frequently very hard to solve since there are typically many more constraints involved than just the symm all different constraint. For real-world problem examples, the propagation of the all different constraint and the symm all different constraint have been examined. They demonstrate that constraint programming performs several orders of magnitude better than operations research techniques when employing the symm all different constraint.

Definition5.2 (Symmetric all different constraint). Let $x_1, x_2, \ldots, x_n$ be variables with respective finite domains $D_1, D_2, \ldots, Dn \subseteq \{1, 2, \ldots, n\}$. Then symm all different $(x_1, \ldots, x_n) = \{(d_1, \ldots, d_n) \mid d_i \in D_i, d_i = d_j \text{ for } i = j, d_i = j \Leftrightarrow d_j = i \text{ for } i = j\}$.

The symm all different constraint in a CSP can also be written as an all different constraint plus one or more symmetry-preserving constraints. There is also an alternative form that makes advantage of the so-called cycle constraint, which states that every cycle has to have two vertices. Then, x is assigned to y and vice versa, as indicated by a cycle on two vertices, x and y. Nonetheless, compared to the common all different constraint with extra restrictions, the symm all different constraint captures more global information. Therefore, a stronger propagation method can be obtained by applying the symm all different constraint.

Example .1 Consider a set of three people that have to be grouped in pairs. Each person can only be paired to one other person. This problem can be represented as a CSP by introducing a set of people $S = \{p_1, p_2, p_3\}$ that are pairwise compatible. These people are represented both by a set of variables x_1, x_2 and x_3 and by a set of values v_1, v_2 and v_3, where x_i and v_i represent p_i. Then the CSP

$$x_1 \in \{v_2, v_3\}, x_2 \in \{v_1, v_3\}, x_3 \in \{v_1, v_2\},$$

$$x_1 = v_2 \Leftrightarrow x_2 = v_1,$$

$$x_1 = v_3 \Leftrightarrow x_3 = v_1,$$

$$x_2 = v_3 \Leftrightarrow x_3 = v_2,$$

$$\text{all different } (x_1, x_2, x_3)$$

is hyper-arc consistent. However, the following CSP

$$x_1 \in \{v_2, v_3\}, x_2 \in \{v_1, v_3\}, x_3 \in \{v_1, v_2\},$$

$$\text{symm all different } (x_1, x_2, x_3)$$

is inconsistent. Indeed, there exists no solution to this problem, as the number of variables is odd.

Suppose there exists a value $j \in D_i$, while $i \neq D_j$. Then we can immediately remove value j from D_i. Hence we assume that such situations do not occur in the following.

Similar to the common all different constraint, the symm all different constraint can be expressed by a graph. Given a constraint symm all different $(x_1, \ldots, x_n)$, construct the graph G symm = (X, E), with vertex set $X = \{x_1, x_2, \ldots, x_n\}$ and edge set $E = \{x_i x_j \mid x_i \in D_j, x_j \in D_i, i < j\}$. Note that we identify each variable x_i with value i for i = 1, . . . , n. We denote the numbers of edges in G_{symm} by m, i.e. $m = \left(\sum_{i=1}^{n} |D_i|\right)/2$. An illustration of G_{symm} is given in the next example.

Example .2 Consider the following CSP

$x_a \in \{b, c, d, e\}$, $x_b \in \{a, c, d, e\}$, $x_c \in \{a, b, d, e\}$,

$x_d \in \{a, b, c, e\}$, $x_e \in \{a, b, c, d, i, j\}$, $x_f \in \{g, h\}$,

$x_g \in \{f, h\}$, $x_h \in \{f, g, i, j\}$, $x_i \in \{e, h, j\}$, $x_j \in \{e, h, i\}$,

symm all different $(x_a, x_b, \ldots, x_j)$.

Theorem 5.3. Let $x_1, x_2, \ldots, x_n$ be a sequence of variables with respective finite domains $D_1, D_2, \ldots, Dn$. Then

$$(d_1, \ldots, d_n) \in \text{symm all different } (x_1, \ldots, x_n)$$

if and only if $M = \{x_i d_i \mid i < d_i\}$ is a matching in G.

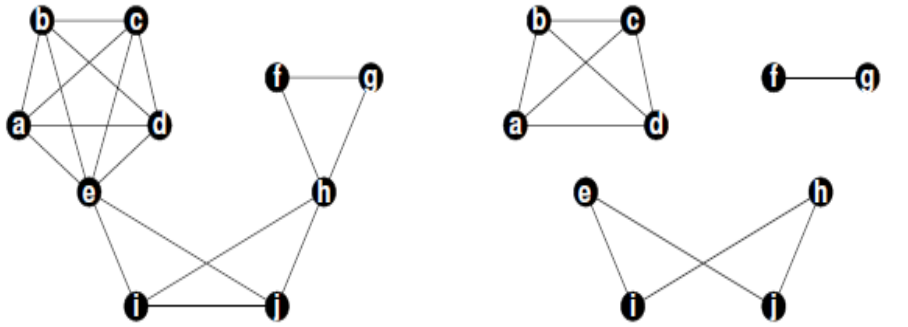
Proof. An edge $x_i x_j$ in G_{symm} corresponds to the assignments $x_i = d_i$ and $xd_i = i$, which are equivalent with $x_j = d_j$ and $xd_j = j$, because $d_i = j$ and $d_j = i$. As no edges in a matching share a vertex, all endpoints are different. Finally, as M covers X, to all variables a value is assigned. ¤

We use Theorem 3.24 to make the symm all different constraint hyper- arc consistent.

Example. 3.. Consider for example the sub graph induced by vertices f, g and h. obviously vertices f and g have to be paired in order to satisfy the symmetric all different constraint.

Hence, vertex h cannot be paired to f nor g, and the corresponding edges can be removed. The same holds for the sub graph induced by a, b, c, d and e, where the vertices a, b, c and d must form pairs.

we already know how to identify edges that belong to a maximum-size matching. Namely, given an arbitrary maximum-size matching M, they belong either to M, or to an even M -alternating path starting at a free vertex, or to an even M -alternating circuit. However, in the cur- rent case, G_{symm} does not need to be bipartite, so we cannot blindly apply the machinery symmetric all different constraint.



a. The graph G_{symm}. b. The graph G_{symm} after propagation.

Figure 3. Propagation of the symmetric all different constraint of Example 3

The following algorithm to achieve hyper-arc consistency was proposed by Regin [5]. First, we compute a maximum size matching M in the (possibly non-bipartite) graph G_{symm}. If $|M| < n/2$, there is no solution. Otherwise, we need to detect all edges that can never belong to a maximum-size matching. Since there are no M free vertices, we only need to check whether an edge that does not belong to M is part of an even M - alternating circuit. This can be done as follows. If for an edge $uv \in M$ we find an M -alternating path u, . . . , w, v (with $u \neq w$), we know that the edge wv is on an even M -alternating circuit. Moreover, we can compute all possible M -alternating paths from u to v, that avoid edge uv. All edges wv that are not on such a path cannot belong to an even M -alternating circuit. Namely, all M -alternating circuits through wv should also contain the matching edge uv. Hence, by. This procedure should be repeated for all vertices in G_{symm}.

Hence the total time complexity of the algorithm achieving hyper-arc consistency for the symm all different constraint is O(nm).

Note that the above algorithm is not incremental, and may not be effective in all practical cases. For this reason, also an incremental algorithm was proposed by Regin [5], that does not ensure hyper-arc consistency, however. The algorithm computes once a maximum-size matching, and each incremental step has a time complexity of O(m).

6. Conclusion

We have provided an overview of the key findings from the years pertaining to all the various constraints. In order to achieve this, we have first presented the fundamental combinatorial ideas—matchings, flows, and Hall's Theorem—that underpin the findings. These ideas have allowed us to explain the many ideas of local consistency in a methodical manner, along with the propagation algorithms that go along with them, all of which have been used to address various constraints.

The following observation is crucial. Constraint programming requires efficient and effective propagation techniques in order to be applied to real-world issues. It is true that the most effective propagation method for all constraints—that is, the one that yields hyper-arc consistency—is highly effective. We can use operations research's matching theory, which explains why. Effective and efficient propagation methods are also available for the symmetric all different constraint and the weighted all different constraint, both of which are based on operations research methodologies.

These findings demonstrate the potential advantages of using operations research methods in constraint propagation algorithms.

7.References

[1] J.-L. Lauriere. A language and a program for stating and solving combinatorial problems. Artificial Intelligence. An International Journal, 10(1): 29{127, 1978. Cited on page 27.
[2] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier. The Constraint Logic Programming Language CHIP. In ICOT,editor, Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS'88), pages 693{702. Springer, 1988. Cited on page 27.
[3] M. Wallace, S. Novello, and J. Schimpf. ECLiPSe: A platform for constraint logic programming. Technical report, IC-Parc, Imperial College, London,1997. Cited on page 27.
[4] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier. The Constraint Logic Programming Language CHIP. In ICOT, editor, Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS'88), pages 693{702. Springer, 1988. Cited on page 27.
[5] J.-C. Regin. A Filtering Algorithm for Constraints of Difference in CSPs.In Proceedings of the Twelfth National Conference on Artificial Intelligence(AAAI), volume 1, pages 362{367. AAAI Press, 1994. Cited on page 27,
39, 40, 42, 61.
[6] C.P. Gomes and D. Shmoys. Completing Quasigroups or Latin Squares: A Structured Graph Coloring Problem. In Proceedings of the Computational Symposium on Graph Coloring and its Generalizations, 2002. Cited on page
28, 97.
[7] N. Barnier and P. Brisset. Graph Coloring for Air Tra±c Flow Management. In Proceedings of the Fourth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR 2002), pages 133{147, 2002. Cited on page 28.
[8] M. Gronkvist. A Constraint Programming Model for Tail Assignment. In J.-C. Regin and M. Rueher, editors, Proceedings of the First International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004), volume 3011 of LNCS, pages 142{156. Springer, 2004. Cited on page 28.
[9] E. Tsang, J. Ford, P. Mills, R. Bradwell, R. Williams, and P. Scott. ZDCRostering: A Personnel Scheduling System Based On Constraint Programming. Technical Report 406, University of Essex, Colchester, UK, 2004.Cited on page 28.
[10] W.J. Older, G.M. Swinkels, and M.H. van Emden. Getting to the Real Problem: Experience with BNR Prolog in OR. In Proceedings of the Third International Conference on the Practical Applications of Prolog (PAP'95).
Alinmead Software Ltd, 1995. Cited on page 28.
[11] J. Zhou. A permutation-based approach for solving the job-shop problem. Constraints, 2(2):185{213, 1997. Cited on page 28.
[12] N. Beldiceanu and E. Contejean. Introducing Global Constraints in CHIP.Mathematical and Computer Modelling, 20(12):97{123, 1994. Cited on page 28.
[13] J.-C. Regin. Generalized Arc Consistency for Global Cardinality Constraint.In Proceedings of the Thirteenth National Conference on Arti¯cial Intelligenceand Eighth Innovative Applications of Arti¯cial Intelligence Conference
(AAAI / IAAI), volume 1, pages 209{215. AAAI Press / The MITPress, 1996. Cited on page 28, 65, 68.
[14] P. Hall. On representatives of subsets. Journal of the London Mathematical Society, 10:26{30, 1935. Cited on page 30.

[15] A. Schrijver. Combinatorial Optimization - Polyhedra and Efficiency. Springer,2003. Cited on page 7, 8, 9, 10, 30, 31, 40, 110.
[16] T.E. Easterfield. A combinatorial algorithm. Journal of the London Mathematical Society, 21:219{226, 1946. Cited on page 31.
[17] J.-F. Puget. A fast algorithm for the bound consistency of alldiff constraints.In Proceedings of the Fifteenth National Conference on Artifcial Intelligence and Tenth Innovative Applications of Artifcial Intelligence Conference(AAAI / IAAI), pages 359{366. AAAI Press / The MIT Press, 1998.Cited on page 36, 37, 42.
[18] A. Lopez-Ortiz, C.-G. Quimper, J. Tromp, and P. van Beek. A fast and simple algorithm for bounds consistency of the alldi®erent constraint. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03), pages 245{250. Morgan Kaufmann, 2003. Cited on page 37,38, 42, 43.
[19] K. Mehlhorn and S. Thiel. Faster algorithms for bound-consistency of the sortedness and the alldifferent constraint. In R. Dechter, editor, Proceedings of the Sixth International Conference on Principles and Practice of
Constraint Programming (CP 2000), volume 1894 of LNCS, pages 306-319.Springer, 2000. Cited on page 37, 38, 42.
[20] A. Lopez-Ortiz, C.-G. Quimper, J. Tromp, and P. van Beek. A fast and simple algorithm for bounds consistency of the alldifferent constraint. In Proceedings of the 18th International Joint Conference on Artificial Intelligence
(IJCAI-03), pages 245{250. Morgan Kaufmann, 2003. Cited on page 37,38, 42, 43.
[21] K. Mehlhorn and S. Thiel. Faster algorithms for bound-consistency of thesortedness and the alldiferent constraint. In R. Dechter, editor, Proceedings of the Sixth International Conference on Principles and Practice of
Constraint Programming (CP 2000), volume 1894 of LNCS, pages 306-319.Springer, 2000. Cited on page 37, 38, 42.