A Soft Constraint of Equality: Complexity and Approximability[*]

Emmanuel Hebrard, Barry O'Sullivan, and Igor Razgon

Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{e.hebrard,b.osullivan,i.razgon}@4c.ucc.ie

Abstract. We introduce the SoftAllEqual global constraint, which maximizes the number of equalities holding between pairs of assignments to a set of variables. We study the computational complexity of propagating this constraint, showing that it is intractable in general, since maximizing the number of pairs of equally assigned variables in a set is NP-hard. We propose three ways of coping with NP-hardness. Firstly, we develop a greedy linear-time algorithm to approximate the maximum number of equalities within a factor of 2. Secondly, we identify a tractable (polynomial) class for this constraint. Thirdly, we identify a parameter based on this class and show that the SoftAllEqual constraint is fixed-parameter tractable with respect to this parameter.

1 Introduction

Constraints for reasoning on the number of differences within a set of variables are ubiquitous in constraint programming. One of the most commonly used global constraints is the AllDifferent constraint [11], which enforces that *all* variables take pair-wise different values. Petit et al. have introduced a soft version of the AllDifferent constraint, SoftAllDiff [10]. They proposed two types of costs, which are to be minimized: *graph*- and *variable*-based costs (see Definitions 1 and 2). The former counts the number of equalities, whilst the latter counts the number of variables violating an AllDifferent constraint. The algorithms for filtering these two constraints, introduced in the same paper, were then improved by Hoeve et al. [15]. In both cases the constraint can be represented as a flow problem, leading to polynomial time algorithms for achieving generalised arc consistency.

Another closely related constraint dealing with equalities between variables, AtMostNValue, received some attention recently. Achieving bounds consistency on this constraint can be done in polynomial time [2] whilst achieving GAC is NP-hard [3]. This latter constraint ensures that no more than k distinct values are assigned to a set of n variables. It is therefore the dual of

[*] This work was supported by Science Foundation Ireland (Grant Number 05/IN/I886). Hebrard is also supported by an Embark Initiative (IRCSET) Post-doctoral Fellowship.

Table 1. Complexity of optimizing inequalities

	Minimizing Cost	Maximizing Cost
Variable Cost	$\mathcal{O}(n\sqrt{m})$	NP-hard
Graph Cost	$\mathcal{O}(nm)$	?

SoftAllDiff for the variable-based cost, i.e. an assignment with k distinct values among n variables violates AllDifferent on $n-k$ variables. To impose a cost of k for AtMostNValue is thus equivalent to imposing a cost of $n-k$ for SoftAllDiff.

The complexities of minimizing both variable- and graph-based costs, as well as maximizing the variable-based cost, for SoftAllDiff are known. However, the complexity of maximizing the graph-based cost (i.e. maximizing the number of pairs of variables assigned with the same value) is still an open problem. Table 1 summarizes the known results for these constraints. Interestingly, whereas minimizing this cost can be mapped to a flow problem and therefore solved in polynomial time, maximizing it does not correspond in a straightforward way to any known problem. In this paper, we fill this gap by providing a number of algorithmic results on the problem of maximizing the number of pair-wise equalities amongst a set of variables. We first show that this problem is NP-hard in general, then we introduce an approximation algorithm, a tractable class and a fixed parameter tractable algorithm.

We call SoftAllEqual$_G$ the global constraint defined with the same *graph-based* cost as SoftAllDiff, albeit where this cost is to be maximized instead of minimized. This constraint has many applications. For instance, consider the problem of scheduling a number of meetings so that every person attends exactly one meeting, and the number of interactions is to be maximized. Each person is assigned to a timeslot, and two people interact only if they attend the same meeting, i.e., are assigned the same value. This can be modelled using a single SoftAllEqual$_G$ constraint. As another example, consider a map coloring problem where we want each continent to be colored as homogeneously as possible. One could post, besides inequalities corresponding to borders, as many SoftAllEqual$_G$ constraints as continents, ensuring that whilst neighboring countries are distinguishable, continents also appear as entities.

However, the original motivation for this work comes from our desire to formulate the problem of finding sets of similar and diverse solutions to CSPs as a constraint optimization problem. Similarity and diversity play fundamental roles in theories of knowledge and behaviour [14]. Reasoning about the distance between solutions is an important problem in artificial intelligence [1,4,7,8,13]. For example, in belief update one might wish to minimize Hamming distance between states [5], in case-based reasoning one often seeks solutions to similar problems while achieving diversity amongst the alternatives [13], in preference-based search one may express preferences in terms of a set of ideal or non-ideal solutions [8]. For instance, let $\mathcal{P}_1, \ldots, \mathcal{P}_m$, be m CSPs with the same number of variables n. In [7] the *diversity* of a set of solutions was defined as the sum

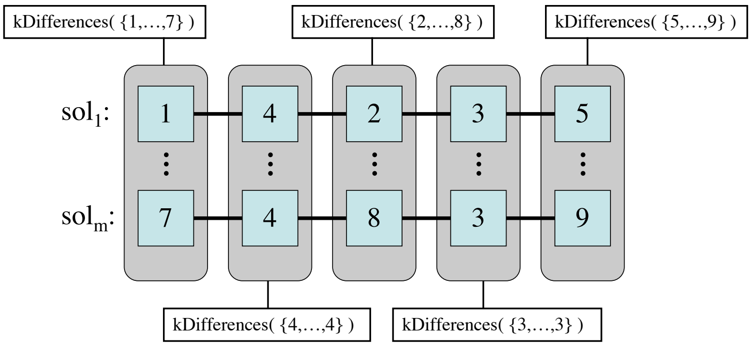
kDifferences({1,...,7})
kDifferences({2,...,8})
kDifferences({5,...,9})
sol1:
1
4
2
3
5
solm:
7
4
8
3
9
kDifferences({4,...,4})
kDifferences({3,...,3})

Fig. 1. Finding diverse solutions as a CSP

of the Hamming distances between each pair of solutions. Let *kDifferences*(M) be the number of pairs of distinct elements in the multiset M. The previously defined diversity of a set of solutions $\{sol_1, \ldots, sol_m\}$ is equivalent to the sum of *kDifferences*$(\{sol_1[i], \ldots, sol_m[i]\})$ over all indices $1 \leq i \leq n$, as illustrated in Figure 1. When either minimizing or maximizing this sum, achieving GAC on each number of differences (*kDifferences*) is enough to obtain GAC on the whole, since the hypergraph is Berge-acyclic. The complexity of computing lower and upper bounds for the number of differences on variables is, therefore, key to this problem.

Our contribution in this paper is to present an indepth study of the complexity of SoftAllEqual$_G$. While achieving generalized arc consistency on the SoftAllDiff constraint is known to be polynomial, the complexity of filtering the SoftAllEqual$_G$ constraint is more intriguing. In Section 3, we show that SoftAllEqual$_G$ is intractable in general, since maximizing the number of pairs of equally assigned variables in a set is NP-complete. We propose three ways of coping with NP-hardness. Firstly, in Section 4, we show that a natural greedy algorithm approximates the maximum number of equalities within a factor of 2, and that its complexity can be brought down to linear time. Secondly, in Section 5, we identify a polynomial class for this constraint. Thirdly, in Section 6, we identify a parameter based on this class and show that the SoftAllEqual$_G$ constraint is fixed-parameter tractable with respect to this parameter.

2 Formal Background

Constraint Satisfaction. A constraint satisfaction problem (CSP) is a triplet $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X}$ is a set of variables, $\mathcal{D}$ a mapping of variables to sets of values and $\mathcal{C}$ a set of constraints that specify allowed combinations of values for subsets of variables. A constraint $C \in \mathcal{C}$ is *generalized arc consistent* (gac) iff, when a variable in the scope of C is assigned any value, there exists an assignment of the other variables in C such that C is satisfied. This satisfying assignment is called a *support* for the value. Given a CSP $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, we shall use the following notation throughout the paper: n shall denote the number of variables, i.e., $n = |\mathcal{X}|$; m shall denote the number of distinct unary assignments, i.e., $m = \sum_{X \in \mathcal{X}} |\mathcal{D}(X)|$; λ shall denote the total number of distinct values, i.e., $\lambda = |\bigcup_{X \in \mathcal{X}} \mathcal{D}(X)|$.

Soft Global Constraints. Adding a cost variable to a constraint to represent its degree of violation is now common practice in constraint programming. This model was introduced in [12]. It offers the advantage of unifying hard and soft constraints since generalized arc consistency, along with other types of consistencies, can be applied to such constraints with no extra effort. As a consequence, classical constraint solvers can solve over-constrained problems modelled in this way without modification. This approach was refined and applied to a number of other constraints in [15].

Two natural cost measures have been explored for the AllDifferent and for a number of other constraints. The *variable-based cost* counts how many variables need to change in order to obtain a valid assignment of the hard constraint. The *graph-based cost* counts how many times a component of a decomposition of the constraint is violated. Typically these components correspond to edges of a decomposition graph, e.g. for an AllDifferent constraint, the decomposition graph is a clique and an edge is violated if and only if both variables connected by this edge share the same value (see Definitions 1 and 2). The SoftAllDiff constraint was, thus, given the following definitions in [10]:

Definition 1 (SoftAllDiff$_V$ – **variable-based cost)**

$$\text{SoftAllDiff}_V(\{X_1, ..X_n\}, N) \Leftrightarrow N \geq n - |\{v \mid X_i = v\}|.$$

Definition 2 (SoftAllDiff$_G$ – **graph-based cost)**

$$\text{SoftAllDiff}_G(\{X_1, ..X_n\}, N) \Leftrightarrow N \geq |\{\{i, j\} \mid X_i = X_j \ \& \ i \neq j\}|.$$

Consider each of the violation costs for the following two solutions of a CSP involving four variables $X_1, \ldots, X_4$ each with domain $\{a, b\}$:

$$S_1 : X_1 = a, X_2 = b, X_3 = a, X_4 = b$$

$$S_2 : X_1 = a, X_2 = b, X_3 = b, X_4 = b$$

In both solutions, at least two variables need to change (for example X_3 and X_4) to obtain a valid solution. Therefore, the variable-based cost is two for S_1 and S_2. However, in S_1 only two edges are violated $\{X_1, X_3\}$ and $\{X_2, X_4\}$ whilst in S_2, three edges are violated $\{X_2, X_3\}$, $\{X_2, X_4\}$ and $\{X_3, X_4\}$. Therefore, the graph-based cost of S_1 is two whereas it is three for S_2.

Parameterized Complexity. We shall use the notion of parameterized complexity in the last section of this paper. For a comprehensive introduction the reader is referred to [9]. Given a problem **A**, a parameterized version of **A** is obtained by specifying a parameter of this problem and getting as additional input a non-negative integer k which restricts the value of this parameter. The resulting parameterized problem $\langle \mathbf{A}, k \rangle$ is *fixed-parameter tractable* (FPT) with respect to k if it can be solved in time $f(k) * n^{\mathcal{O}(1)}$, where $f(k)$ is a function depending only on k.

3 The SoftAllEqual Constraint

We define the constraint that we study in this paper, SoftAllEqual$_G$, by reversing the graph-based cost of the SoftAllDiff constraint (Definition 2).

Definition 3 (SoftAllEqual$_G$ – **dual of** SoftAllDiff$_G$**)**

$$\textsc{SoftAllEqual}_G(\{X_1, ..X_n\}, N) \Leftrightarrow N \leq |\{\{i,j\} \mid X_i = X_j \ \& \ i \neq j\}|.$$

Interestingly, the same inversion of the definition for the variable-based cost of SoftAllDiff leads to the AtMostNValue constraint [2]. The focus of this paper, however, is on the graph-based cost. We first show that solving a CSP with the SoftAllEqual$_G$ constraint is intractable using a reduction from **3dMatching** [6].

Definition 4 (3dMatching)
Data: *An integer K, three disjoint sets X, Y, Z, and $T \subseteq X \times Y \times Z$.*
Question: *Does there exist $M \subseteq T$ such that $|M| \geq K$ and $\forall m_1, m_2 \in M, \forall i \in \{1,2,3\},\ m_1[i] \neq m_2[i]$.*

Theorem 1 (The Complexity of SoftAllEqual$_G$**).** *Finding a satisfying assignment for the* SoftAllEqual$_G$ *constraint is NP-complete even if no value appears in more than three domains.*

Proof. The problem is clearly in NP: checking the number of equalities in an assignment can be done in $\mathcal{O}(n^2)$ time.

We use a reduction from **3dMatching** to show completeness. Let $P{=}(X, Y, Z, T, K)$ be an instance of **3dMatching**, where: K is an integer; X, Y, Z are three disjoint sets such that $X \cup Y \cup Z = \{x_1, \ldots x_n\}$; and $T = \{t_1, \ldots t_m\}$ is a set of triplets over $X \times Y \times Z$. We build an instance I of SoftAllEqual$_G$ as follows:

1. Let $n = |X| + |Y| + |Z|$, we build n variables $\{X_1, \ldots, X_n\}$.
2. For each $t_l = \langle x_i, x_j, x_k \rangle \in T$, we have $l \in \mathcal{D}(X_i)$, $l \in \mathcal{D}(X_j)$ and $l \in \mathcal{D}(X_k)$.
3. For each pair (i, j) such that $1 \leq i < j \leq n$, we put the value $(|T| + (i-1) * n + j)$ in both $\mathcal{D}(X_i)$ and $\mathcal{D}(X_j)$.

We show there exists a matching of P of size K if and only if there exists a solution of I with $\lfloor \frac{3K+n}{2} \rfloor$ equalities. We refer to "a matching of P" and to a "solution of I" as "a matching" and "a solution" throughout this proof, respectively.

$\Rightarrow$: We show that if there exists a matching of cardinality K then there exists a solution with at least $\lfloor \frac{3K+n}{2} \rfloor$ equalities. Let M be a matching of cardinality K.

We build a solution as follows. For all $t_l = \langle x_i, x_j, x_k \rangle \in M$ we assign X_i, X_j and X_k to l (2). Observe that there remain exactly $n-3K$ unassigned variables after this process. We pick an arbitrary pair of unassigned variables and assign them with their common value (3), until at most one variable is left (if one variable is left we assign it to an arbitrary value). Therefore, the solution obtained in this way has exactly $\lfloor \frac{3K+n}{2} \rfloor$ equalities, $3K$ from the variables corresponding to the matching and $\lfloor \frac{n-3K}{2} \rfloor$ for the remaining variables.

$\Leftarrow$: We show that if the cardinality of the maximal matching is K, then there is no solution with more than $\lfloor \frac{3K+n}{2} \rfloor$ equalities. Let S be a solution. Furthermore, let L be the number of values appearing three times in S. Observe that this set of values corresponds to a matching. Indeed, a value l appears in three domains $\mathcal{D}(X_i), \mathcal{D}(X_j)$ and $\mathcal{D}(X_k)$ if and only if there exists a triplet $t_l = \langle x_i, x_j, x_k \rangle \in T$ (2). Since a variable can only be assigned to a single value, the values appearing three times in a solution form a matching. Moreover, since no value appears in more than three domains, all other values can appear at most twice. Hence the number of equalities in S is less than or equal to $\lfloor \frac{3L+n}{2} \rfloor$, where L is the size of a matching. It follows that if there is no matching of cardinality greater than K, there is no solution with more than $\lfloor \frac{3K+n}{2} \rfloor$ equalities. □

It is worth noting that if one similarly reverses the alternative, variable-based, cost of SoftAllDiff, the result corresponds to the AtMostNValue constraint. Interestingly, this is *not* equivalent to applying the variable-based cost on a constraint AllEqual. For instance, consider n variables $X_1, \dots, X_n$ and suppose that half are assigned to a whilst the other half are assigned to b. One needs to change $n/2$ variables in order to make them all equal, and $n-2$ to make them all different. In this paper we consider only the costs as defined for SoftAllDiff in [10], when reasoning about both lower an upper bounds. On the other hand, the graph-based cost on AllEqual is indeed equivalent to the opposite of SoftAllDiff$_G$.

4 Approximation Algorithm

In this section and in the rest of the paper we consider the optimization version of SoftAllEqual$_G$ where the objective is to *maximize* the number of pairs of variables assigned with the same value. We first study a natural greedy algorithm for approximating the maximum number of equalities in a set of variables. This algorithm picks the value that occurs in the largest number of domains, and assigns as many variables as possible to this value (this can be achieved in $\mathcal{O}(m)$). Then it recursively repeats the process on the resulting sub-problem until all variables are assigned (at most $\mathcal{O}(n)$ times). We show that this algorithms approximates the maximum number of equalities with a factor 2 in the worst case. Moreover, we it can be implemented to run in in linear amortized time (that is, $\mathcal{O}(m)$) by using the following data structures:

- $var : \Lambda \mapsto 2^{\mathcal{X}}$ maps every value v to the set of variables whose domains contain v.
- $val : \mathbb{N} \mapsto 2^{\Lambda}$ maps every integer $i \in [0..n]$ to the set of values appearing in exactly i domains.

These data structures are initialized in Lines 1 and 2 of Algorithm 1, respectively.

Algorithm 1. `GreedyValue`

```
  Data: A set of variables $\mathcal{X}$
  Result: Lower bound on the maximum number of equalities
1 $var(v) \leftarrow \emptyset,\ \forall v \in \bigcup_{X \in \mathcal{X}} \mathcal{D}(X)$;
  foreach $X \in \mathcal{X}$ do
      foreach $v \in \mathcal{D}(X)$ do
          add $X$ to $var(v)$;
2 $val(k) \leftarrow \emptyset,\ \forall k \in [0..|\mathcal{X}|]$;
  foreach $v \in \bigcup_{X \in \mathcal{X}} \mathcal{D}(X)$ do
      add $v$ to $val(|var(v)|)$;
  return AssignAndRecurse($var, val, |\mathcal{X}|$);
```

Algorithm 2. `AssignAndRecurse`

```
  Data: a mapping $var : \Lambda \mapsto 2^{\mathcal{X}}$, a mapping $val : \mathbb{N} \mapsto 2^{\Lambda}$, an integer $k$;
1 while $val(k) = \emptyset$ do $k \leftarrow k - 1$;
  if $k \leq 1$ then
      return 0;
  else
2     pick and remove any $v \in val(k)$;
3     foreach $X \in var(v)$ do
          foreach $w \neq v \in \mathcal{D}(X)$ do
              $occ_w \leftarrow |var(w)|$;
              remove $w$ from $val(occ_w)$;
              add $w$ to $val(occ_w - 1)$;
4             assign $X$ with $w$ and remove $X$ from $var(w)$;
      return $\frac{k(k-1)}{2}$+AssignAndRecurse($var, val, k$);
```

The above algorithm returns the number of pairs of equal values of an assignment of the given CSP, which is constructed on Line 4.

Theorem 2 (Algorithm Correctness). *The algorithm* `GreedyValue` *approximates the optimal satisfying assignment of the* SoftAllEqual$_G$ *constraint within a factor of* 2 *and runs in* $\mathcal{O}(m)$.

Proof. We first prove the correctness of the approximation ratio, the soundness of the algorithm and then the complexity of the algorithm.

Approximation Factor. We proceed using induction on n. Let lb be the value returned by `GreedyValue` and let eq^* be the maximum possible number of equalities. We denote $P(n)$ the proposition "If there are no more than n values in the union of the domains of $\mathcal{X}$, then $lb \geq eq^*/2$". $P(1)$ implies that every variable can all be assigned to a unique value v. Algorithm `GreedyValue` therefore chooses this value and assigns all variables to it. In this case $lb = eq^*$.

Now we suppose that $P(n)$ holds and we show that $P(n+1)$ also holds. Let $\mathcal{X}$ be a set of variables such that $|\bigcup_{X \in \mathcal{X}} \mathcal{D}(X)| = n+1$ and let v be the first value chosen by `GreedyValue`. We denote by $\mathcal{X}_v = \{X \in \mathcal{X} \mid v \notin \mathcal{D}(X)\}$ the set of variables whose domains do not contain v, $\bar{\mathcal{X}_v} = \mathcal{X} \setminus \mathcal{X}_v$ the complementary set of variables assigned to v and we let $k = |\bar{\mathcal{X}_v}|$. We denote eq_v^* the maximal number of equalities on the set of variables $\mathcal{X}_v$, that is where both variables in the equality belong to $\mathcal{X}_v$. Consider any variable $X \in \bar{\mathcal{X}_v}$. Given any value w in $\mathcal{D}(X)$, there are no more than k variables in $\mathcal{X}$ containing w. Indeed, v was chosen for maximizing this criterion and belongs to the domains of exactly k variables. Therefore, the total number of equalities involving at least one variable in $\bar{\mathcal{X}_v}$ is at most $k(k-1)$, since there are k variables in $\bar{\mathcal{X}_v}$ and each can only be involved in $k-1$ equalities. Since an equality either involves at least one variable $\bar{\mathcal{X}_v}$, or none of them, we can bound the maximal total number of equalities as follows:

$$eq^* \leq k(k-1) + eq_v^*.$$

Now, observe that the total number of values in $\mathcal{X}_v$ is less than or equal to n since every variable whose domain contains v is in $\bar{\mathcal{X}_v}$. Since we suppose that $P(n)$ holds, we know that the value returned by `GreedyValue` for $\bar{\mathcal{X}_v}$ is greater than or equal to $eq_v^*/2$. Hence the value returned for $\mathcal{X}$ is greater than or equal to $k(k-1)/2 + eq_v^*/2$. We can therefore conclude that $lb \geq eq^*/2$ and hence $P(n+1)$ is true.

Correctness. Here we show that the mapping var and val are correctly updated in a call to `AssignAndRecurse`. Let $\mathcal{X}$ be the set of variables given as initial input of `GreedyValue`. We define $\mathcal{X}_V$, as the set of variables remaining after greedily choosing and assigning the set of values V. ($\mathcal{X}_V = \{X \mid X \in \mathcal{X} \,\&\, \mathcal{D}(X) \cap V = \emptyset\}$). We say that val and var are correct for $\mathcal{X}_V$ iff both of the following invariants hold:

$$\forall v \in \bigcup_{X \in \mathcal{X}_V} \mathcal{D}(X),\ var(v) = \{X \mid v \in \mathcal{D}(X)\} \tag{1}$$

$$\forall k \in [1..n],\ val(k) = \{v \mid k = |var(v)|\} \tag{2}$$

This is clearly the case after the initialisation phase. Now we suppose that it is the case at the i^{th} call to `AssignAndRecurse` and we show that it still holds at the $(i+1)^{th}$ call. We assume that value w is chosen in Line 2.

We suppose first that invariant 1 does not hold. That is, there exists $X \in \mathcal{X}_V, v$ such that either $v \in \mathcal{D}(X)$ and $X \notin var(v)$ or $v \notin \mathcal{D}(X)$ and $X \in var(v)$. The latter case is not possible since we only remove values from $var(v)$. The former case can only arise if X was removed from $var(v)$ in `AssignAndRecurse` (Line 4). However this can only happen if $X \in var(w)$, hence $X \notin \mathcal{X}_{V \cup \{w\}}$.

Then we suppose that invariant 2 does not hold, i.e., there exists k, v such that either $v \in val(k)$ and $k \neq |var(v)|$ or $v \notin val(k)$ and $k = |var(v)|$. However, the cardinality k of $var(v)$ can only decrease by one at Line 4, and in that case v is removed from $val(k)$ and added to $val(k-1)$.

Complexity. The mapping var is in $\mathcal{O}(m)$ space, and val is in $\mathcal{O}(\lambda)$, where λ denotes the number of distinct values. Initialising both mappings is done in linear time since exactly one element is added to either val or var at every step. Hence the initialisation is in $\mathcal{O}(m) + \mathcal{O}(\lambda)$ time, i.e., $\mathcal{O}(m)$. In Line 1 in `AssignAndRecurse`, k can be decremented at most n times in total. In Loop 3, every iteration remove exactly one element in var, the amortised time complexity for this loop therefore is $\mathcal{O}(m)$. The overall time complexity is thus $\mathcal{O}(m)$. □

Theorem 3 (Tightness of the Approximation Ratio). *The approximation factor of* 2 *for* `GreedyValue` *is tight.*

Proof. Let $\{X_1, \dots X_4\}$ be a set of four variables with domains as follows:

$$X_1 \in \{a\};\ X_2 \in \{b\};\ X_3 \in \{a, c\};\ X_4 \in \{b, c\}.$$

Every value appears in exactly two domains, hence `GreedyValue` can choose any value. We suppose that the value c is chosen first. At this point no other value can contribute to an equality, hence `GreedyValue` returns 1. However, it is possible to achieve two equalities with the following solution: $X_1 = a,\ X_3 = a,\ X_2 = b,\ X_4 = b$. □

5 Tractable Class

In this section we explore further the connection between the SoftAllEqual$_G$ constraint and vertex matching. We showed earlier that the general case was linked to **3dMatching**. We now show that the particular case where no value appears in more than two domains solving the SoftAllEqual$_G$ constraint is equivalent to the vertex matching problem on general graphs, and therefore can be solved by a polynomial time algorithm. We shall then use this tractable class to show that SoftAllEqual$_G$ is NP-hard only if an unbounded number of values appear in more than two domains.

Definition 5 (The VertexMatching Problem)
Data: *An integer K, an undirected graph $G = (V, E)$.*
Question: *Does there exist $M \subseteq E$ such that $|M| \geq K$ and $\forall e_1, e_2 \in M$, e_1 and e_2 do not share a vertex.*

Theorem 4 (Tractable Class of SoftAllEqual$_G$). *If all triplets of variables $X, Y, Z \in \mathcal{X}$ are such that $\mathcal{D}(X) \cap \mathcal{D}(Y) \cap \mathcal{D}(Z) = \emptyset$ then finding an optimal satisfying assignment to* SoftAllEqual$_G$ *is in P.*

Proof. In order to solve this problem, we build a graph $G = (V, E)$ with a vertex x_i for each variable $X_i \in \mathcal{X}$, that is, $V = \{x_i \mid X_i \in \mathcal{X}\}$. Then for each pair

$\{i, j\}$ such that $\mathcal{D}(X_i) \cap \mathcal{D}(X_j) \neq \emptyset$, we create an undirected edge $\{i, j\}$; let $E = \{\{i, j\} \mid i \neq j \,\&\, \mathcal{D}(X_i) \cap \mathcal{D}(X_j) \neq \emptyset\}$.

We first show that if there exists a matching of cardinality K, then there exists a solution with at least K equalities. Let M be a matching of cardinality K of G, for each edge $e = (i, j) \in M$ we assign X_i and X_j to any value $v \in \mathcal{D}(X_i) \cap \mathcal{D}(X_j)$ (by construction, we know that there exists such a value). Observe that no variable is counted twice since it would mean that two edges of the matching have a common vertex. The obtained solution therefore has at least $|M|$ equalities.

Now we show that if there are K equalities in S, then there exists a matching of cardinality K. Let S be a solution, and let $M = \{\{i, j\} \mid S[X_i] = S[X_j]\}$. Observe that M is a matching of G. Indeed, suppose that two edges sharing a vertex (say $\{i, j\}, \{j, k\}$) are both in M. It follows that $S[X_i] = S[X_j] = S[X_k]$, however this is in contradiction with the hypothesis. We can therefore compute a solution S maximizing the number of equalities by computing a maximal matching in G. $\square$

This tractable class can be generalized by restricting the number of occurrences of values in the domains of variables. The notion of *heavy values* is key to this result.

Definition 6 (Heavy Value). *A heavy value is a value that occurs more than twice in the domains of the variables of the problem.*

Theorem 5 (Tractable Class with Heavy Values). *If the domain $\mathcal{D}(X_i)$ of each variable X_i contains at most one heavy value then finding an optimal satisfying assignment of* SoftAllEqual$_G$ *is in P.*

Proof. Consider a two stage algorithm. In the first stage it explores all values w that have three or more appearances and assigns w to all the variables whose domains contain it. Notice that no variable will be assigned with two values. In the second stage the CSP created by the domains of unassigned variables consists of only values having at most two occurrences, so we solve this CSP by transforming it to the matching problem as suggested in the proof of Theorem 4.

We show that there exists an optimal solution where each variable that can be assigned to an heavy value is assigned to this value. Let s^* be an optimal solution and w be an heavy value over a set T of variables of cardinality t. We suppose that only $z < t$ of them are assigned to w in s^*. Consider the solution s' obtained by assigning all these t variables to w: we add exactly $t(t-1)/2 - z(z-1)/2$ equalities. However, we potentially remove $t - z$ equalities since values other than w do not appear more than twice. We therefore have $obj(s') - obj(s^*) \geq t^2 - 3t - z^2 + 3z$, which is non-negative for $t \geq 3$ and $z < t$. By iteratively applying this transformation, we obtain an optimal solution where each variable that can be assigned to an heavy value is assigned to this value. The first stage of the algorithm is thus correct. The second stage is correct by Theorem 4. $\square$

6 Parameterized Complexity

We further advance our analysis of the complexity of the SoftAllEqual$_G$ constraint by introducing a fixed-parameter tractable (FPT) algorithm with respect to the number of values. This result is important because it shows that the complexity of propagating this constraint grows only polynomially in the number of variables. It may therefore be possible to achieve GAC at a reasonable computational cost even for a very large set of variables, providing that the total number of distinct values is relatively small.

We first show that the SoftAllEqual$_G$ problem is FPT with respect to the number of values λ. We use the tractable class introduced in the previous section to generalize this result, showing that the problem is FPT with respect to the number of *heavy* values occurring in domains containing two or more heavy values. We begin with a definition.

Definition 7 (Solution from a Total Order). *A solution $s_{\prec}$ is induced by a total order $\prec$ over the values if and only if*

$$s[X] = v \;\Rightarrow\; \forall w \prec v,\; w \notin \mathcal{D}(X).$$

We now prove the following key lemma.

Lemma 1. *Let s^* be an optimal solution, v be a value, and $occ(s^*, v)$ be the number of variables assigned to v in s^*. Moreover, let $\prec_{occ}$ be a total order such that values are ranked by decreasing number of occurrences (ties are broken arbitrarily). We claim that $\prec_{occ}$ induces s^*.*

Proof. Consider, without loss of generality, a pair of values v, w such that $v \prec_{occ} w$. By definition we have $occ(s^*, v) \geq occ(s^*, w)$. We suppose that the hypothesis is falsified and show that this leads to a contradiction. Suppose that there exists a variable X such that $\{v, w\} \subseteq \mathcal{D}(X)$ and $s^*[X] = w$ (that is, $\prec_{occ}$ does not induce s^*). The objective value of the solution s' such that $s'[X] = v$ and $s'[Y] = s^*[Y]\, \forall y \neq x$ is given by: $obj(s') = obj(s^*) + occ(s^*, v) - (occ(s^*, w) - 1)$. Therefore, $obj(s') > obj(s^*)$. However, s^* is optimal, hence this is a contradiction. □

This lemma has two interesting consequences. The first consequence is expressed by the following corollary.

Corollary 1. *There exists a total order $\prec$ over the set of values, such that the solution $s_{\prec}$ induced by $\prec$ is optimal.*

Proof. Direct consequence of Lemma 1.

The fixed-parameter tractability of the SoftAllEqual$_G$ constraint follows easily from Corollary 1.

Theorem 6 (FPT – number of values). *Finding an optimal satisfying assignment of the SoftAllEqual$_G$ constraint is fixed-parameter tractable with respect to λ, the number of values in the domains of the constrained variables.*

Proof. Explore all possible $\lambda!$ permutations of values. For each permutation create a solution induced by this permutation. Compute the cost of this solution. Return the solution having the highest cost. According to Corollary 1, this solution is optimal. Creating an induced solution can be done by selecting for each domain the first value in the order. Clearly, this can be done in $\mathcal{O}(m)$. Computing the cost of the given solution can be done by computing the number of occurrences $occ(w)$ and then summing up $occ(w) * (occ(w) - 1)/2$ for all values w. Clearly, this can be done in $\mathcal{O}(m)$ as well. Hence the theorem follows. □

The second corollary from Lemma 1 is much more surprising.

Corollary 2. *The number of optimal solutions of the CSP with the* SoftAllEqual$_G$ *is at most* $\lambda!$

Proof. According to Lemma 1, each optimal solution is induced by an order over the values of the given problem. Clearly each order induces exactly one solution. Thus the number of optimal solution does not exceed the number of total orders which is at most $\lambda!$. □

Corollary 2 claims that the number of optimal solutions of the considered problem *does not* depend on the number of variables and they all can be explored by considering all possible orders of values. We believe this fact is interesting from the practical point of view because in essence it means that even enumerating all optimal solutions is *scalable* with respect to the number of variables. Moreover, we can show that SoftAllEqual$_G$ is fixed-parameter tractable with respect to the number of bad values, defined as follows.

Definition 8 (Bad Value). *A value w of a given CSP is a* bad value *if and only if it is an heavy value and there is a domain $\mathcal{D}(X)$ that contains w and another heavy value.*

Theorem 7 (FPT – number of bad values). *Let k be the number of bad values of a CSP comprising only one* SoftAllEqual$_G$ *constraint. Then the CSP can be solved in time $\mathcal{O}(k! * n^2 * \lambda)$, hence* SoftAllEqual$_G$ *is fixed-parameter tractable with respect to k.*

Proof. Consider all the permutations of the bad values. For each permutation perform the following two steps. In the first step for each variable X where there are two or more bad values, remove all the bad values except the one which is the first in the order among the bad values of $\mathcal{D}(X)$ according to the given permutation. In the second stage we obtain a problem where each domain contains exactly one heavy value. Solve this problem polynomially by the algorithm provided in the proof of Theorem 5.

Let s be the solution obtained by this algorithm. We show that this solution is optimal. Let p^* be a permutation of *all* the values of the considered CSP so that the solution s^* induced by p^* has the highest possible cost. By Corollary 1, s^* is an optimal solution. Let p_1 be the permutation of the bad values which is induced by p^* and let s_1 be the solution obtained by the above algorithm with respect to p_1. By definition of s, $obj(s) \geq obj(s_1)$. We show that $obj(s_1) \geq obj(s^*)$ from which the optimality of s immediately follows.

Observe that there is no X such that $s^*[X] = w$ and w was removed from $\mathcal{D}(X)$ in the first stage of the above algorithm where the permutation p_1 is considered. Indeed, w can only be removed from $\mathcal{D}(X)$ if it is preceded in p_1 by a value $v \in \mathcal{D}(X)$. It follows that w is also preceded in p^* by v and consequently $s^*(X) \neq w$. Thus s^* is a solution of the CSP obtained as a result of the first stage. However s_1 is an *optimal* solution of that CSP by Theorem 5 and, consequently, $obj(s_1) \geq obj(s^*)$ as required. $\square$

This result shows that the complexity of propagating the SoftAllEqual$_G$ constraint comes primarily from the number of (bad) values, whereas other factors, such as the number of variables, have little impact. Observe that the "exponential" part of this algorithm is based on the exploration of all possible orders over the given set of bad values. In fact the ordering relation between two values matters only if these values belong to a domain of the same variable. In other words consider a graph H on values of the given CSP instance. Two values a and b are connected by an edge if and only if they belong to the domain of the same variable. Instead of considering all possible orders over the given set of values we may consider all possible ways of transforming the given graph into an acyclic digraph. The upper bound on the number of possible transformations is $2^{E(H)}$ where $E(H)$ is the number of edges of H. For sparse graphs such a bound is much more optimistic that $k!$. For example, if the average degree of a vertex is 4 then the number of considered partial orders is $2^{2k} = 4^k$.

7 Conclusion and Future Work

We showed that achieving GAC for the SoftAllEqual$_G$ constraint is NP-complete. Then we introduced a simple linear greedy algorithm and showed that it approximates the maximum number of pairs of variables that can be assigned equally within a factor of 2. Moreover, we showed that the hardness of the problem could be encapsulated by the number of "bad" values, irrespective of the size of the instance.

We believe one can combine our parameterized and approximation algorithms in order to design a practical algorithm for solving the SoftAllEqual$_G$ constraint. Firstly, Theorem 7 allows us to search in the space of permutations of *a subset of values*, which is usually much smaller than the space of partial assignments. Therefore we can design a branch-and-bound algorithm searching in the space of permutations. Secondly, we can use our approximation algorithm to more effectively prune the branches of the search tree. In particular, if the number of equalities guessed by the approximation algorithm is at most half of the current *upper bound* maintained by the branch-and-bound algorithm, then the algorithm may safely backtrack.

To the best of our knowledge, the problem we are tackling in this paper does not have a straightforward equivalent formulation in the algorithmic literature.

We therefore focused on complexity and approximability issues, laying down theoretical foundations for future work on this constraint. The design of a filtering algorithm for SoftAllEqual$_G$, besides the trivial application of the bounds provided in this paper, is left as a challenge. A very important avenue of research is to study the complexity of achieving *bounds consistency* on SoftAllEqual$_G$. The complexity of SoftAllEqual$_G$ when domains are intervals on $\mathbb{N}$ is still open, while bounds consistency on the AtMostNValue constraint can be done in polynomial time [2]. The problem is equivalent to finding a *clique cover* of minimal cardinality for the intersection graph of the domains [3], which is by definition an interval graph. The restriction of SoftAllEqual$_G$ to intervals, on the other hand, leads to a similar problem, but requiring a clique cover of the same graph that maximizes the sum of cardinalities of the cliques.

References

1. Bailleux, O., Marquis, P.: Some Computational Aspects of Distance-SAT. Journal of Automated Reasoning 37(4), 231–260 (2006)
2. Beldiceanu, N.: Pruning for the Minimum Constraint Family and for the Number of Distinct Values Constraint Family. In: Walsh, T. (ed.) CP 2001. LNCS, vol. 2239, pp. 211–224. Springer, Heidelberg (2001)
3. Bessière, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: Filtering Algorithms for the NValue Constraint. In: Barták, R., Milano, M. (eds.) CPAIOR 2005. LNCS, vol. 3524, pp. 79–93. Springer, Heidelberg (2005)
4. Crescenzi, P., Rossi, G.: On the Hamming Distance of Constraint Satisfaction Problems. Theor. Comput. Sci. 288(1), 85–100 (2002)
5. Forbus, K.D.: Introducing Actions into Qualitative Simulation. In: IJCAI, pp. 1273–1278 (1989)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W.H. Freeman and Company, New York (1979)
7. Hebrard, E., Hnich, B., O'Sullivan, B., Walsh, T.: Finding Diverse and Similar Solutions in Constraint Programming. In: AAAI, pp. 372–377 (2005)
8. Hebrard, E., O'Sullivan, B., Walsh, T.: Distance Constraints in Constraint Satisfaction. In: IJCAI, pp. 106–111 (2007)
9. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, Oxford (2006)
10. Petit, T., Régin, J.-C., Bessière, C.: Specific Filtering Algorithms for Over-Constrained Problems. In: CP, pp. 451–463 (2001)
11. Régin, J.-C.: A Filtering Algorithm for Constraints of Difference in CSPs. In: AAAI, pp. 362–367 (1994)
12. Régin, J.-C., Petit, T., Bessière, C., Puget, J.-F.: An Original Constraint Based Approach for Solving over Constrained Problems. In: CP, pp. 543–548 (2000)
13. Smyth, B., McClave, P.: Similarity vs. Diversity. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 347–361. Springer, Heidelberg (2001)
14. Tversky, A.: Features of Similarity. Psychological Review 84, 327–352 (1977)
15. van Hoeve, W.-J., Pesant, G., Rousseau, L.-M.: On Global Warming: Flow-Based Soft Global Constraints. Journal of Heuristics 12(4-5), 347–373 (2006)