Finding Diverse and Similar Solutions in Constraint Programming

Emmanuel Hebrard
NICTA and UNSW
Sydney, Australia
e.hebrard@nicta.com.au

Brahim Hnich
University College Cork
Ireland
brahim@4c.ucc.ie

Barry O'Sullivan
University College Cork
Ireland
b.osullivan@4c.ucc.ie

Toby Walsh
NICTA and UNSW
Sydney, Australia
tw@cse.unsw.edu.au

Abstract

It is useful in a wide range of situations to find solutions which are diverse (or similar) to each other. We therefore define a number of different classes of diversity and similarity problems. For example, what is the most diverse set of solutions of a constraint satisfaction problem with a given cardinality? We first determine the computational complexity of these problems. We then propose a number of practical solution methods, some of which use global constraints for enforcing diversity (or similarity) between solutions. Empirical evaluation on a number of problems show promising results.

Introduction

Computational complexity deals with a variety of different problems including decision problems (e.g. "Is there a solution?"), function problems (e.g. "Return a solution"), and counting problems (e.g. "How many solutions exist?"). However, much less has been said about problems where we wish to find a *set* of solutions that are *diverse* or *similar* to each other. For brevity, we shall call these *diversity* and *similarity* problems. Existing work in this area mostly focuses on finding pairs of solutions that satisfy some distance constraint (Bailleux & Marquis 1999; Crescenzi & Rossi 2002; Angelsmark & Thapper 2004). However, there is a wider variety of problems that one may wish to consider.

In product configuration, similarity and diversity problems arise as preferences are elicited and suggestions are presented to the user. Suppose you want to buy a car. There are various constraints on what you want and what is available for sale. For example, you cannot buy a cheap Ferrari nor a convertible with a sunroof. You begin by asking for a set of solutions as diverse as possible. You then pick the most preferred car from this set. However, as not all the details are quite right, you ask to see a set of solutions as similar as possible to this car.

As a second example, suppose you are scheduling staff in a hospital. Unfortunately, the problem is very dynamic and uncertain. People are sure to phone in ill, and unforseen operations to be required. To ensure that the schedule is robust to such changes and can be repaired with minor disruption, you might look for a schedule which has many similar solutions nearby. Supermodels are based on this idea (Ginsberg, Parkes, & Roy 1998). Finally, suppose you are trying to acquire constraints interactively. You ask the user to look at solutions and propose additional constraints to rule out infeasible or undesirable solutions. To speed up this process, it may help if each solution is as different as possible from previously seen solutions.

In this paper, we introduce several new problems classes, each focused on a similarity or diversity problem associated with a NP-hard problems like constraint satisfaction. We distinguish between *offline* diversity and similarity problems (where we compute the whole set of solutions at once) and their *online* counter-part (where we compute solutions incrementally). We determine the computational complexity of these problems, and propose practical solution methods. Finally, we present some promising experimental results.

Formal background

A binary relation R over strings is polynomial-time decidable iff there is a deterministic Turing machine deciding the language $\{x; y \mid (x, y) \in R\}$ in polynomial time (Papadimitriou 1994). A binary relation R is polynomially balanced iff $(x, y) \in R$ implies $|y| < |x|^k$ for some k. A language L belongs to NP iff there is a polynomial-time decidable and polynomially balanced relation R such that $L = \{x \mid (x, y) \in R\}$. We let $Sol(x) = \{y \mid (x, y) \in R\}$. We assume that we have some symmetric, reflexive, total and polynomially bounded distance function, δ, between strings. For example, if L is SAT, this might be the Hamming distance between truth assignments. Note, however, that most of our complexity results will hold for any function that is polynomially bounded. Finally, we use 0 to denote FALSE and 1 to denote TRUE.

Offline diversity and similarity

We define two decision problems at the core of diversity and similarity problems. These ask if there is a subset of solutions of size k at least (or at most) d distance apart. For brevity, we define $max(\delta, S) = \max_{y,z \in S} \delta(y, z)$ and $min(\delta, S) = \min_{y,z \in S} \delta(y, z)$.

dDISTANTkSET (resp. dCLOSEkSET)
Instance. Given a polynomial-time decidable and polynomially balanced relation R, a distance function

δ and some string x.
Question. Does there exist S with $S \subseteq Sol(x)$, $|S| = k$ and $min(\delta, S) \geq d$ (resp. $max(\delta, S) \leq d$).

We might also consider the average distance between solutions instead of the minimum or maximum. As we have two parameters, d and k, we can choose to fix one and optimize the other. That is, we can fix the size of the set of solutions and maximize their diversity (similarity). Alternatively, we can fix the diversity (similarity) and maximize the size of the set of solutions returned. Alternatively, we could look for a Pareto optimal set or combine d and k into a single metric.

MAXDIVERSEkSET (resp. MAXSIMILARkSET)
Instance. Given a polynomial-time decidable and polynomially balanced relation R, a distance function δ and some string x.
Question. Find S with $S \subseteq Sol(x)$, $|S| = k$, and for all S' with $S' \subseteq Sol(x)$, $|S'| = k$, $min(\delta, S) \geq min(\delta, S')$ (resp. $max(\delta, S) \leq max(\delta, S')$).

MAXdDISTANTSET (resp. MAXdCLOSESET)
Instance. Given a polynomial-time decidable and polynomially balanced relation R, a distance function δ and some string x.
Question. Find S with $S \subseteq Sol(x)$, $min(\delta, S) \geq d$ (resp. $max(\delta, S) \leq d$), and for all S' with $min(\delta, S') \geq d$ (resp. $max(\delta, S') \leq d$), $|S| \geq |S'|$.

For example, in the product configuration problem, we might want to see the 10 most diverse solutions. This is an instance of the MAXDIVERSEkSET problem. On the other hand, we might want to see all solutions at most 2 features different to our current best solution. This is an instance of the MAXdCLOSESET problem.

Online diversity and similarity

In some situations, we may be computing solutions one by one. For example, suppose we show the user several possible cars to buy but none of them are appealing. We would now like to find a car for sale which is as diverse from these as possible. This suggests the following two online problems in which we find a solution most distant (close) to a set of solutions. For brevity, we define $max(\delta, S, y) = \max_{z \in S} \delta(y, z)$ and $min(\delta, S, y) = \min_{z \in S} \delta(y, z)$.

MOSTDISTANT (resp. MOSTCLOSE)
Instance. Given a polynomial-time decidable and polynomially balanced relation R, a distance function δ, some string x and a subset S of strings.
Question. Find y with $y \in Sol(x) - S$, such that for all y' with $y' \in Sol(x) - S$, $max(\delta, S, y) \geq max(\delta, S, y')$ (resp. $min(\delta, S, y) \leq min(\delta, S, y')$).

We might also be interested in finding not one but a set of solutions distant from each other and from a given set. Alternatively, we might be interested in finding a larger set of solutions than the ones we currently have with a given diversity or similarity.

Computational complexity

For some of these problem classes, specifically those for which the CSP is an input (e.g. dDISTANT/CLOSEkSET), the size of the output (a set of solutions) may not be polynomially bounded by the size of the input. Therefore we will always make the assumption that k is polynomial in the size of the input. With such an assumption, dDISTANTkSET and dCLOSEkSET are in NP. Since dDISTANTkSET and dCLOSEkSET decide membership in R when $k = 1$, both problems are trivially NP-complete.

We now show that both MAXDIVERSEkSET and MAXSIMILARkSET are FP$^{NP[log\ n]}$-complete. This is the class of all languages decided by a polynomial-time oracle which on input of size n asks a total number of $O(log\ n)$ NP queries (Papadimitriou 1994).

Theorem 1 MAXDIVERSEkSET *and* MAXSIMILARkSET *are FP$^{NP[log\ n]}$-complete.*

Proof. MAXDIVERSEkSET (resp. MAXSIMILARkSET) $\in$ FP$^{NP[log\ n]}$: The distance function is polynomial in the size of the input, n. Using binary search, we call dDISTANTkSET (resp. dCLOSEkSET) to determine the exact value of d. This requires only logarithmically many adaptive NP queries.
Completeness: We sketch a reduction from MAX CLIQUE SIZE which is FP$^{NP[log\ n]}$-complete (Papadimitriou 1994). Given a graph $G = \langle V, E \rangle$, with set of nodes V and set of edges E, MAX CLIQUE SIZE determines the size of the the largest clique in G. We define a CSP as follows. We introduce a Boolean variable B_i, called a *graph-node variable* for each node i in V. For each $a, b \in V$, if $\langle a, b \rangle \notin E$, we enforce that B_a and B_b cannot both be 1. We add an additional $\lceil \frac{n}{2} \rceil + 1$ Boolean variables, called *distance variables*, and enforce that they are 0 iff every graph-node variable is 0. This CSP always admits the solution with 0 assigned to all Boolean variables. So that MAXDIVERSEkSET finds the clique of maximum size, we define the distance between two solutions, $\delta(s_a, s_b)$, as the Hamming distance defined over all Boolean variables. Thus, MAXDIVERSEkSET(P) with $k = 2$ will always have one of the solutions assigning 0 to each Boolean variable and the other solution assigning 1 to those graph-node variables representing nodes in the largest clique possible, and 0 to all the additional distance variables, since these two solutions are maximally diverse wrt Hamming Distance. Suppose this is not the case, i.e., both solutions assign some node variables to 0 and some others to 1. Since all extra variables are set to 1, the maximum achievable distance is n. Moreover, if the clique with largest cardinality has m nodes, two such solutions cannot be more than $2m$ assignments apart. However, the solution with maximum clique and the solution with all variables assigned to 0 are $\lceil \frac{n}{2} \rceil + 1 + m$ assignments apart, which is strictly greater than $min(n, 2m)$. Hence, we have an answer to MAX CLIQUE SIZE(G).

The reduction for MAXSIMILARkSET is analogous. However, we need to define the distance between a pair of solutions, s_a and s_b as $n + \lceil \frac{n}{2} + 1 \rceil - \delta(s_a, s_b)$ if $s_a \neq s_b$, and 0 otherwise, where δ is the Hamming distance. $\square$

MAXdDISTANTSET and MAXdCLOSESET may require exponential time just to write out their answers (since the answer set may itself be exponentially large for certain queries). We therefore assume d (resp. $n-d$) to be at most $O(log\ n)$. Furthermore, we assume δ to be the Hamming distance. These assumptions are sufficient to ensure that the answer set is polynomially bounded and MAXdCLOSESET (resp. MAXdDISTANTSET) is FP$^{NP[log\ n]}$-complete.

Theorem 2 MAXdCLOSESET *and* MAXdDISTANTSET *are FP$^{NP[log\ n]}$-complete.*

Proof. MAXdCLOSESET (resp. MAXdDISTANTSET) is in FP$^{NP[log\ n]}$: With the assumptions on d and δ, the cardinality of the set of solutions returned is polynomially bounded. By binary search on k, using a dCLOSEkSET oracle, we can find the largest possible d-close set. Thus, we only need a logarithmic number of calls to an NP oracle to answer both questions. We omit the details of the proof for MAXdDISTANTSET since it is similar ($n-d$ is bounded instead of d).
Completeness: We sketch a reduction from MAX CLIQUE SIZE where $G = \langle V, E\rangle$ denotes a graph with a set of n nodes V and set of edges E. We now define a CSP as follows. We introduce n^2+1 variables. The first variable X_0 takes its value in the set $\{1, 2, \dots, n\}$, and the other variables are partitioned in n groups of n variables, $X_i \in \{0, 1, \dots, n\}$. We introduce the following constraints:

$$\forall j,\ X_{(j-1).n+X_0} = 0 \,\&\, \langle X_0, j\rangle \in E \Rightarrow X_{(X_0-1).n+j} = 0,$$

$$\forall j \neq X_0, k,\ (k \neq X_0 \vee \nexists \langle k, j\rangle \in E) \Rightarrow X_{(k-1).n+j} = X_0.$$

The entire solution is entailed by the value i assigned to X_0, i.e. the i^{th} variable of every group must be assigned to 0, the j^{th} variable of the i^{th} group is assigned to 0 iff $\langle i, j\rangle \in E$ and to i otherwise. Finally, all other variables are set to i. It is therefore easy to see that we have exactly one solution per node in the graph. Let $\delta(s_a, s_b)$ be the Hamming distance between the solutions s_a and s_b. Let i (resp. j), be the value assigned to X_0 in s_a (resp. s_b), we show that $\delta(s_a, s_b) = 2$ iff $\langle i, j\rangle \in E$. Suppose first that $\langle i, j\rangle \in E$ and consider the variable $X_{(i-1).n+j}$. In s_a this variable is set to 0 as $\langle i, j\rangle \in E$, moreover, it is assigned to 0 in s_b, as are all j^{th} variables of each group. We can repeat this reasoning for $X_{(j-1).n+i}$, hence $\delta(s_a, s_b) \geq 2$. Now consider any other variable X_k. Either $X_k = i$ in s_a, or $X_k = j$ in s_b, however, no variable is set to j in s_a nor is assigned to i in s_b. We thus have $\delta(s_a, s_b) = 2$. Suppose now that $\langle i, j\rangle \notin E$. The last step in our reasoning is still valid, whilst now we have $X_{(i-1).n+j} = i$ in s_a and $X_{(j-1).n+i} = j$ in s_b. Therefore, $\delta(s_a, s_b) = 0$. We first showed that a set of solutions of the CSP map to a set of nodes of G, then we showed that if the distance threshold d is set to 2, then this set of nodes is a clique. Since the set with maximum cardinality will be returned, we obtain the solution to MAX CLIQUE SIZE(G) from MAXdCLOSESET. The reduction for MAXdDISTANTSET is analogous. However, here we define the distance between a pair of solutions, s_a and s_b as $n^2+1-\delta(s_a, s_b)$, if $s_a \neq s_b$, and 0 otherwise to give a reflexive distance measure. □

Finally, we show that the online diversity problem MOSTDISTANT and similarity problem MOSTCLOSE are both FP$^{NP[log\ n]}$-complete.

Theorem 3 MOSTDISTANT *and* MOSTCLOSE *are FP$^{NP[log\ n]}$-complete.*

Proof. MOSTDISTANT (resp. MOSTCLOSE) $\in$ FP$^{NP[log\ n]}$: We need to have an algorithm that calls at most a logarithmic number of NP queries. Consider the decision version of MOSTDISTANT, where the solution has to have at least k discrepancies with a given set of vectors, $\mathcal{V}$. This problem is in NP, the witness being the solution, since checking the number of discrepancies is polynomial. By binary search, we need at most $O(log\ n)$ calls to this oracle to determine the most distant solution.
Completeness: We sketch a reduction from MAX CLIQUE SIZE.We define a CSP in the same way as in the completeness proof of Theorem 1 but only use the graph node variables. We let vector $v \in \mathcal{V}$ be a string of 0s of length $n = |V|$. In order that MOSTDISTANT finds the clique of maximum size, we define the distance between two solutions, $\delta(s_a, s_b)$ as the Hamming distance. We have an answer to MAX CLIQUE SIZE(G) iff MOSTDISTANT($P, \{v\}$). The reduction for MOSTCLOSE is analogous, except we define the distance between solutions s_a and s_b as $n - \delta(S_a, S_b)$, if $s_a \neq s_b$, and 0 otherwise to give a reflexive distance measure. We have an answer to MAX CLIQUE SIZE(G) iff MOSTCLOSE($P, \{v\}$). □

Solution Methods

We have developed a number of complete and heuristic algorithms for solving similarity and diversity problems.

Complete Methods based on Reformulation

Given a CSP, we can solve similarity and diversity problems using a reformulation approach. Specifically, we build a new CSP as follows. We create k copies of the CSP, each copy with a different set of variable names. We add $k(k-1)/2$ extra variables to represent the Hamming distance d_i^j between each pair of vectors of variables. We let min be the minimum of the d_i^j's and max be the maximum of the d_i^j's. Starting with this basic model, we now describe how to extend it for each of the following problems:

dDISTANTkSET/dCLOSEkSET: We also enforce that min (resp. max) is at least (resp. at most) d.

MAXDIVERSEkSET/MAXSIMILARkSET: We introduce an objective function that maximizes (resp. minimizes) min (resp. max).

MOSTDISTANT/MOSTCLOSE: We set k to $|V|+1$ where V is the input set of solutions and assign $|V|$ sets of variables to the given solutions in V. We also introduce an objective function that maximizes (resp. minimizes) min (resp. max).

We then solve the reformulated problem using off-the-shelf constraint programming solvers. For MAXdDISTANTSET (resp. MAXdCLOSESET, we find all solutions and construct a graph whose nodes are the solutions. There is an edge between two nodes iff the pairwise Hamming distance is at least d (resp. at most d). We then find a maximum clique in the resulting graph. For problems with many solutions, the constructed graph may be prohibitively large.

Complete Method for MOSTDISTANT/MOSTCLOSE

We will show in the next section that the problems MOSTDISTANT/MOSTCLOSE can be used to approximate all the others. Therefore, we introduce here an efficient complete algorithm for MOSTDISTANT/MOSTCLOSE based on global diversity/similarity constraints. We define and study four such global constraints that use Hamming distance. Let d be an integer, $V = \{v_1, \dots, v_k\}$ be a set of vectors of size n, and $v_j[i]$ be the i^{th} element of v_j. We define the following global constraints.

$$Similar_{\Sigma}(X_1, \dots, X_n, V, d) \Leftrightarrow \sum_{i=1,j=1}^{i=n,j=k} (X_i \neq v_j[i]) \leq d$$

$$Diverse_{\Sigma}(X_1, \dots, X_n, V, d) \Leftrightarrow \sum_{i=1,j=1}^{i=n,j=k} (X_i \neq v_j[i]) \geq d$$

$$Similar_{max}(X_1, \dots, X_n, V, d) \Leftrightarrow max_{j=1}^{j=k} \sum_{i=1}^{i=n} (X_i \neq v_j[i]) \leq d$$

$$Diverse_{min}(X_1, \dots, X_n, V, d) \Leftrightarrow min_{j=1}^{j=k} \sum_{i=1}^{i=n} (X_i \neq v_j[i]) \geq d$$

For reasons of space, we only consider here the diversity constraints. The results are analogous for the similarity ones. We propose a Branch & Bound algorithm for MOSTDISTANT that uses these global constraints as follows. We first post a global $Diverse$ constraint with $d = 1$. For every solution s that is found during search, we set d to the Hamming distance between V and s plus one, and continue searching. When the algorithm eventually proves unsatisfiability, the last solution found is optimal. We expect propagation on the $Diverse$ constraint to prune parts of the search space.

$Diverse_{\Sigma}$: A constraint like the global $Diverse_{\Sigma}$ constraint is *generalized arc consistent* (GAC), if every value for every variable can be extended to a tuple satisfying the constraint. Consider first the case where there is only one vector V as argument of $Diverse_{\Sigma}$. As long as the number of variables X_i still containing $V[i]$ in their domain is greater than d, the constraint is GAC. If this number is less than d, then we fail. Finally, when there are exactly d such variables, we can remove $V[i]$ from $D(X_i)$ for all i.

The situation is more subtle when we look for assignments close to several vectors at once. For instance, consider the following two vectors:

$$V_1 = \langle 0, 0, \dots, 0 \rangle, \ \ V_2 = \langle 1, 1, \dots, 1 \rangle$$

Even if all the domains contain the values 0 and 1, the maximum distance to V_1 and V_2 is n, as any solution has to disagree with either V_1 or V_2 on every variable. Moreover, if we add $V_3 = \langle 0, 1, 0, 1 \dots, 0, 1 \rangle$ to this example, then setting a variable with even index to 1 implies that the distance grows by 1, whereas it grows by 2 if we set it to 0, and by 3 if we use any other value. Suppose that all variables take their values in $\{0, 1, 2\}$. The assignment with maximum Hamming distance sets to 2 every variable. The distance between this assignment and V_1, V_2, V_3 is $3n$. Now suppose that $d = 3n - 1$, then for any i, $X_i = 0$ and $X_i = 1$ are arc inconsistent, whilst $X_i = 2$ is GAC.

We now give an algorithm for enforcing GAC on $Diverse_{\Sigma}(X_1, \dots, X_n, V_1, \dots, V_k, d)$:

Algorithm 1 $Diverse_{\Sigma}(X_1, \dots, X_n, V_1, \dots, V_k, d)$

```
   Dist ← 0;
1  foreach X_i do
       foreach v_j ∈ D(X_i) do
           Occ[i][j] ← |{l | V_l[i] = v_j}|;
       Dist ← Dist + k − min(Occ[i]);
       if Dist < d then Fail;
       Best[i] ← min(Occ[i]);
2  foreach X_i do
3      D(X_i) ← {v_j | v_j ∈ D(X_i) ∧ d ≤ (Dist+Best[i]−Occ[i][j])};
```

Theorem 4 *Algorithm 1 maintains GAC on $Diverse_{\Sigma}(X_1, \dots, X_n, V_1, \dots, V_k, d)$ and runs in $O(n(d + k))$ where d is the maximum domain size.*

Proof. Soundness: suppose that $v_j \in D(X_i)$ after propagation. We construct an assignment satisfying the constraint, and involving $X_i = v_j$ as follows: For each variable X_l other than X_i we assign this variable to the value v_m such that $Occ[l][m]$ is minimum, that is, equal to $Best[l]$. We therefore have $\sum_{x=1,y=1}^{x=n,y=k} (X_x \neq v_y[x]) = Dist - Best[i] + k - Occ[i][j]$ for this assignment, however, line 3 ensures that this value is greater than d. Hence $X_i = j$ is GAC.

Completeness: Suppose that $v_j \notin D(X_i)$ after propagation. Then we have $Dist - Best[i] + k - Occ[i][j] < d$, where $Dist$ is the sum of $Best[l]$ for all $l \in [1..n]$. It therefore means that the assignment where every variable but X_i takes the value occurring the least in $V_1, \dots, V_k$, is not a support (a valid extension) for $X_i = j$. Moreover any other assignment would have a greater distance to $V_1, \dots, V_k$, and would therefore not be a support. Hence $X_i = j$ is not GAC.

Worst case time complexity: The loop 1 has complexity $O(nd + kd)$. The values in $Occ[i][j]$ can be computed in two passes. The first pass set $Occ[i][j]$ to 0 for every value j of every variable j ($O(nd)$). The second pass increments $Occ[i][j]$ by one for each vector V_l such that $V_l[i] = j$ ($O(nk)$). The second loop (2) is in $O(nd)$. □

$Diverse_{min}$: Unfortunately, when we use the minimum (resp. maximum) distance, maintaining GAC on a global $Diverse$ ($Similar$) constraint becomes intractable. We therefore should look to enforce a lesser level of consistency.

Theorem 5 *GAC is NP-hard to propagate on $Diverse_{min}$.*

Proof. We reduce 3SAT to the problem of finding a consistent assignment for $Diverse_{min}$. Given a

Boolean formula in n variables (x_i, for all $i \in [1..n]$) and m clauses (c_i, for all $i \in [1..m]$), we construct the $Diverse_{min}(X_1, \ldots, X_n, V_1, \ldots, V_m, d)$ constraint in which $X_i = \{i, \neg i\}$ for all $i \in [1..n]$ and each V_j for all $j \in [1..m]$ represents one of the m clauses. If the clause c_l is $\{x_i, \neg x_j, x_k\}$ then $V_l[i] = i$, $V_l[j] = \neg j$ and $V_l[k] = k$. All other values are set to $n + 1$. If $d = n - 2$, then the constructed $Diverse_{min}$ constraint has a satisfying assignment iff the 3SAT formula has a model. The solution we find corresponds to an assignment of the variables. Moreover, the dummy values "consume" exactly $n - 3$ differences. Therefore the solution has to share at least one element with each clause, and thus satisfies it. Notice that in this case the solution we find corresponds to a model where all literal are negated. $\square$

Heuristic Approaches

Methods based on reformulation may give prohibitively large CSPs. We therefore propose an approximation scheme that is based on the Branch & Bound algorithm for MOSTDISTANT/MOSTCLOSE, that approximates the others. Variants of the following greedy heuristic can approximate dCLOSEkSET and MAXdCloseSet problem by using the Branch & Bound algorithm for MOSTCLOSE.

Algorithm 2 `greedy`

```
  Data    : P = (X, D, C), K, d
  Result  : V
  find a solution v;
  V ← {v};
  while |V| < K & min_{x,y∈V} δ(x, y) > d do
1     find most diverse solution u to previous solutions V';
      V ← V ∪ {u};
```

Finding a solution that maximizes similarity to previous solutions (Line 1) corresponds to solving MOSTDISTANT. Observe that if we keep only the first test for the "while" loop, then we approximate MAXDIVERSEkSET. Similarly, if we keep only the second test, we approximate MAXdDistantSet. This method easily applies to diversity.

Experiments

We ran experiments using the Renault Megane Configuration benchmark (Amilhastre, Fargier, & Marguis 2002), a large real-world configuration problem. The variables represent various options such as engine type, interior options, etc. The problem consists of 101 variables, domain size varies from 1 to 43, and there are 113 table constraints, many of them non-binary. The number of solutions to the problem is over 1.4×10^{12}. We report the results obtained using the greedy algorithm and a Branch & Bound procedure taking advantage of the $Diverse_{\Sigma}$ constraint. We solved MAXDIVERSEkSet for k set to 3, motivated by work in recommender systems that argues that the optimal sized set to present to a user contains 3 elements (Shimazu 2001). The same variable ordering heuristic (`H_1_dom/deg_x` in (Bessière, Chmeiss, & Saïs 2001)) was used for all instances. The values that were used the least in the previous solutions are chosen first. We also ran the same experiment without a specific value ordering. We also report results by simply shuffling the values in the domains and starting a new search without using a value ordering.

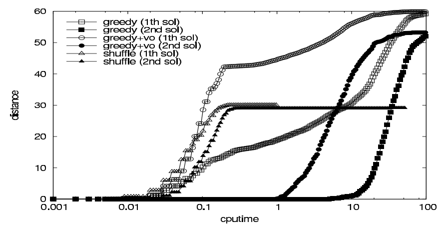

Figure 1: Solving MAXDIVERSE3SET for the Renault Megane configuration benchmark.

The results for the Renault benchmark are given in Figure 1. The distance between solutions, averaged over all instances, is plotted on the y-axis, against the cpu-time required to obtain it. To obtain different instances, upon which to base an average, we simply shuffled the values. For each method (i.e., greedy with value ordering, greedy without value ordering, and shuffling), the same initial solution is found, then the first curve corresponds to the first diverse solution from it, and the second curve to the next diverse solution (from both previous solutions).

We also ran some experiments on random binary CSPs, to observe how performance changes with the constrainedness of the problem. Results are given in Figure 2. All instances have 100 variables, domain size is 10, and 275 binary constraints. We generated 500 instances with tightness equal to 0.5 and 0.52 (0.52 is the phase transition), and 1500 instances with tightness equal to 0.53, as few of them were satisfiable.

The results on random CSPs show that our approximation method is more efficient as the constrainedness of the problem increases. For loosely constrained problems the simply shuffling method can be competitive since many possible diverse solutions exist. We also observe that choosing the value that is used the least in previous solutions tends to improve greatly performance on loosely constrained problems, but when the tightness of the constraints increases this value heuristic can slow the algorithm down as it chooses "successful" values last.

Overall, the optimization method without value ordering is more efficient on more tightly constrained problems. However, the same method with value ordering is consistently the best choice. Indeed, on easier problems the heuristic finds a distant solution quickly, whilst on harder problems the the Branch & Bound procedure combined with the $Diverse_{\Sigma}$ constraint can find far more distant solutions than the shuffling approach.

The Renault Megane configuration problem is loosely constrained and thus admits a large number of solutions. It is not surprising therefore that the best method uses the value ordering heuristic. We are able to find 2 diverse solutions

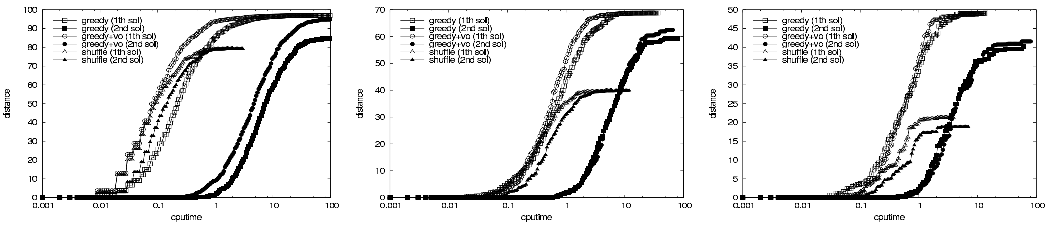

Figure 2: Solving MAXDIVERSE3SET for random CSPs (100 variables, 10 values, 275 constraints). Tightness is 0.5/0.52/0.53,respectively.

such that half of the variables are assigned differently within 3 seconds, or 3 such solutions within 15 seconds.

Related work

A number of researchers have studied the complexity of finding *another* solution to a problem. For example, given a graph and a Hamiltonian cycle with this graph, it is NP-complete to decide if the graph contains another Hamiltonian cycle. However, constraints are not typically placed on the relationship between the two solutions. One exception is work in constraint satisfaction where we ensure that the Hamming distance between the two solutions is maximized (Angelsmark & Thapper 2004; Crescenzi & Rossi 2002) or that the solutions are within some distance of each other (Bailleux & Marquis 1999). The problem classes introduced here subsume these problems.

Diversity and similarity are also important concepts in case-based reasoning and recommender systems (Bridge & Ferguson 2002; Shimazu 2001; Smyth & McClave 2001). Typically, we want chose a diverse set of cases from the case-base, but retrieve cases based on similarity. Also more recent work in decision support systems has focused on the use of diversity for finding diverse sets of good solutions as well as the optimum one (Løkketangen & Woodruff 2005).

Conclusions

Motivated by a number of real-world applications that require solutions that are either diverse or similar, we have proposed a range of similarity and diversity problems. We have presented a detailed analysis of their computational complexity (see Table 1), and developed a number of algorithms and global constraints for solving them. Our experimental results on some real-world problems are very encouraging. In future work, we will study problems that combine both similarity and diversity. Specifically, given a solution (or set of solutions), we may wish to find a set of solutions that are as similar as possible to the given one(s), but are as mutually diverse as possible.

Table 1: A summary of the complexity results.

	complexity class
dDISTANTkSET	NP-complete
dCLOSEkSET	NP-complete
MAXDIVERSEkSET	FP$^{NP[log\ n]}$-complete
MAXSIMILARkSET	FP$^{NP[log\ n]}$-complete
MAXdDISTANTSET	FP$^{NP[log\ n]}$-complete
MAXdCLOSESET	FP$^{NP[log\ n]}$-complete
MOSTDISTANT	FP$^{NP[log\ n]}$-complete
MOSTCLOSE	FP$^{NP[log\ n]}$-complete

Acknowledgements. Hebrard and Walsh are supported by the National ICT Australia, which is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. Hnich is supported by Science Foundation Ireland (Grant 00/PI.1/C075). O'Sullivan is supported by Enterprise Ireland (Grant SC/02/289).

References

Amilhastre, J.; Fargier, H.; and Marguis, P. 2002. Consistency restoration and explanations in dynamic CSPs – application to configuration. *AIJ* 135:199–234.

Angelsmark, O., and Thapper, J. 2004. Algorithms for the maximum hamming distance problem. In *Proceedings of CSCLP-04*, 128–141.

Bailleux, O., and Marquis, P. 1999. DISTANCE-SAT: Complexity and algorithms. In *AAAI/IAAI*, 642–647.

Bessière, C.; Chmeiss, A.; and Saïs, L. 2001. Neighborhood-based variable ordering heuristics for the constraint satisfaction problem. In *Proceedings of CP-2001*, 565–569.

Bridge, D., and Ferguson, A. 2002. Diverse product recommendations using an expressive language for case retrieval. In *Procs. of EC-CBR*, 43–57.

Crescenzi, P., and Rossi, G. 2002. On the hamming distance of constraint satisfaction problems. *TCS* 288:85–100.

Ginsberg, M. L.; Parkes, A. J.; and Roy, A. 1998. Supermodels and robustness. In *AAAI/IAAI*, 334–339.

Løkketangen, A., and Woodruff, D. L. 2005. A distance function to support optimized selection decisions. *Decision Support Systems* 39(3):345–354.

Papadimitriou, C. 1994. *Computational Complexity*. Addison-Wesley.

Shimazu, H. 2001. Expertclerk: Navigating shoppers' buying process with the combination of asking and proposing. In *Proceedings of IJCAI-2001*, 1443–1448.

Smyth, B., and McClave, P. 2001. Similarity vs diversity. In *Procs of IC-CBR*, 347–361.