A local search approach to solve incomplete fuzzy and weighted CSPs

Mirco Gelain[1], Maria Silvia Pini[1], Francesca Rossi[1], K. Brent Venable[1], and Toby Walsh[2]

[1] Dipartimento di Matematica Pura ed Applicata, Università di Padova, Italy
E-mail: {mgelain,mpini,frossi,kvenable}@math.unipd.it
[2] NICTA and UNSW Sydney, Australia,
Email: Toby.Walsh@nicta.com.au

Abstract. We consider soft constraint problems where some of the preferences may be unspecified. This models, for example, settings where agents are distributed and have privacy issues, or where there is an ongoing preference elicitation process. In this context, we study how to find an optimal solution without having to wait for all the preferences. In particular, we define local search algorithms that interleave search and preference elicitation, with the goal to find a solution which is "necessarily optimal", that is, optimal no matter what the missing data are, while asking the user to reveal as few preferences as possible. While in the past this problem has been tackled with a branch & bound approach, which was guaranteed to find a solution with this property, we now want to see whether a local search approach can solve such problems optimally, or obtain a good quality solution, with fewer resources. At each step, our local search algorithm moves from the current solution to a new one, which differs in the value of a variable. The variable to reassign and its new value are chosen so to maximize the quality of the next solution. To compute this, we elicit some of the preferences missing in the neighbor solutions. Experimental results on randomly generated fuzzy and weighted CSPs with missing preferences show that our local search approach is promising, both in terms of percentage of elicited preferences and its scaling properties.

1 Introduction

Constraint programming [1] is a powerful paradigm for solving scheduling, planning, and resource allocation problems. A problem is represented by a set of variables, each with a domain of values, and a set of constraints. A solution is an assignment of values to the variables which satisfies all constraints and which optionally maximizes/minimizes an objective function. Soft constraints [2] are a way to model optimization problems by allowing for several levels of satisfiability, modeled by the use of preference or cost values that represent how much we like a certain way to instantiate the variables of a constraint. Incomplete soft constraint problems (ISCSPs) [3–5] can model situations in which data are not completely known before solving starts by allowing some of the preferences to

be missing. Thus ISCSPs can model a form of uncertainty which is captured by the absence of some preferences in the constraints. In this scenario, the notion of optimal solution is replaced by other notions, among which the most attractive is that of a "necessarily optimal solution". They are those solutions which are optimal no matter what the missing preference values will turn out to be. These are solutions which are completely robust with respect to the uncertainty of the problem.

If an ISCSP has some variable assignments with this property, it is ideal to return one of them. However, the set of necessarily optimal solutions of an ISCSP may be empty. In this case, we may reduce the uncertainty of the problem by asking the user to reveal some of the missing preferences, and then see whether the new constraint problem has some necessarily optimal solutions. In fact, if some preferences are revealed, the new set of necessarily optimal solutions may be larger than the one we had before.

In [3] a complete branch & bound approach is used to find a necessarily optimal solution of an ISCSP, by interleaving search and elicitation. The approach was tested over both fuzzy and weighted ISCSPs, and several different elicitation strategies were considered. The experimental results showed that some of the concrete algorithms based on that approach could find a necessarily optimal solution while eliciting a small percentage of the missing preferences.

In this paper we intend to see whether ISCSPs can be better solved by a local search approach. As for the branch & bound approach, we interleave local search steps and elicitation. In this case, elicitation is used to guide the future steps. More precisely, our algorithm starts from a randomly chosen assignment to all the variables and elicits enough of the missing preferences related to this assignment to compute its preference. We then move to a new assignment which differs for the value of one variable. The variable to re-instantiate, and the new value to assign to it, are chosen as in [6], except that we compute preference values as if all the missing preferences were uninfluential. During the search, the algorithm maintains the best solution found so far. Random moves and Tabu Search are also used to avoid stagnation in local minima. The algorithm stops when a given limit on the number of steps is reached.

This basic local search approach, although not being very sophisticated, shows a very good behavior under several aspects. In our experimental tests, performed over randomly generated fuzzy and weighted ISCSPs, and considering several elicitation strategies, we measured the runtime and the percentage of elicited preferences, and we compared both of them to those of the best complete algorithm based on branch & bound.

When the number of variables is small the complete algorithm is better in terms of elicited preferences. However, the amount of elicitation needed by the local search approach is not much higher, and always less than 25% of the total number of missing preferences for fuzzy ISCSPs. In terms of runtime, the local search approach is always much better, even with small problems. In fact, the local search approach can handle problems of up to 100 variables by using the same time the complete algorithm needs to solve problems with only 25 variables.

We also measured the quality of the solutions returned by the local search approach, as the distance from the quality of the preference of the solution returned by the complete algorithm based on branch & bound. In this respect, the local search approach is very good, since such a distance appears lower than 2% in all considered cases.

By looking at the runtime behaviour of the local search algorithm, we also noticed that all the measured properties (that is, number of elicited preference values and solution quality) are achieved in the first steps of the search. Thus, we could safely stop the algorithm much earlier than the step limit, while obtaining solutions with the same quality and eliciting the same amount of missing preferences. This means that our local search algorithm can actually handle much larger instances than those considered in the experiments.

2 Background

In this section we give the basic notions about incomplete soft constraint problems and local search.

2.1 Incomplete soft constraint problems

Incomplete Soft Constraints Problems (ISCSPs) [3] extend Soft Constraint Problems (SCSPs) [7] to deal with partial information. We will focus on two specific instances of this framework, in which the soft constraints are fuzzy and weighted. In the fuzzy case, preference values are between 0 and 1, the preference of a solution is the minimum preference value contributed by the constraints, and the optimal solutions are those with the highest value. On the other hand, in weighted CSPs, preference values are interpreted as costs and range is between 0 and $+\infty$, the preference of a solution is the sum of all the costs contributed by the constraints, and the optimal solutions are those with lowest cost.

More formally, given a set of variables V with finite domain D, an *incomplete soft constraint* is a pair $\langle idef, con\rangle$ where $con \subseteq V$ is the scope of the constraint and $idef : D^{|con|} \longrightarrow (A \cup \{?\})$ is the preference function of the constraint associating to each tuple of assignments to the variables in con either a preference value that belongs to the set A, or ?. A is the set of preference values, which is $[0, 1]$ for fuzzy constraints, and $[0, +\infty]$ for weighted constraints. All tuples mapped into ? by $idef$ are called *incomplete tuples*, meaning that their preference is unspecified.

An *Incomplete Fuzzy (resp., Weighted) Constraint Problem* (IFCSP, resp, IWCSP) is a triple $\langle C, V, D\rangle$ where C is a set of incomplete fuzzy (resp., weighted) constraints over variables V with domain D.

Given an assignment s to all the variables of an IFCSP or an IWCSP P, $pref(P, s)$ is the preference of s in P. More precisely, it is defined as $pref(P, s) = \otimes_{<idef,con>\in C|idef(s_{\downarrow con})\neq ?} idef(s_{\downarrow con})$, where $\otimes$ is the minimum for fuzzy constraints and the sum for weighted constraints. It is obtained by taking the minimum or the sum of the known preferences associated to the projections of the assignment, that is, of the appropriated sub-tuples in the constraints.

Example 1. Figure 1 shows an example of an IFCSP with three variables A, B, and C, with domains $D(A) = \{a, b, c\}$, $D(B) = \{d, e\}$, and $D(C) = \{r, s\}$. The presence of the question marks identifies the missing preference values. In this example, the preference of the variable assignment $\langle A = a, B = e, C = r\rangle$ is $min(0.3, 0.4, 1, 0.9) = 0.3$. Since this variable assignment involves some missing preferences, such as the one for the tuple $\langle A = a, B = e\rangle$, its preference should be interpreted as an upper bound of the actual preference for this assignment.

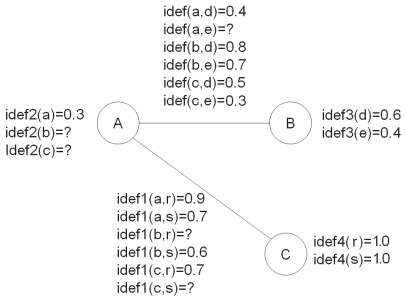


Fig. 1. An example of ISCSP

When all preferences are specified, a complete assignment of values to all the variables is an optimal solution if its preference is the best one (that is, maximal for fuzzy constraints and minimal for weighted constraints). This optimality notion is generalized to ISCSPs through the notion of *necessarily optimal solutions*, that is, complete assignments which are optimal no matter the value of the unknown preferences. Of course, these are the solutions which are most appealing, since they are completely robust w.r.t. the missing data. However, the set of necessarily optimal solutions of an ISCSP may be empty.

In Example 1, the assignment $\langle A = a, B = e, C = r\rangle$ is not necessarily optimal. In fact, if all missing preferences are 1, this assignment has preference 0.3, while the assignment $\langle A = b, B = d, C = r\rangle$ has preference 0.6. In this example, there are no necessarily optimal solutions. Consider now the same example where the preferences of both $A = b$ and $A = c$ are set to 0.2. In this new IFCSP, the assignment $\langle A = a, B = d, C = s\rangle$ has preference 0.3 and is necessarily optimal. In fact, whatever values are given to the missing preferences, all other assignments have preference at most 0.2 (if $A = b$ or c) or 0.3 (if $A = a$).

In [3] several algorithms are proposed to find a necessarily optimal solution of an IFCSP or an IWCSP. All these algorithms follow a branch and bound schema where search is interleaved with elicitation. Elicitation is needed since the given problem may have an empty set of necessarily optimal solutions. By eliciting more preferences, this set eventually becomes non-empty. Several elici-

tation strategies are considered in [3] in the attempt to elicit as little as possible before finding a necessarily optimal solution.

2.2 Local search

Local search [8] is one of the fundamental paradigms for solving computationally hard combinatorial problems. Local search can also naturally be used to solve optimization problems. Given a problem instance, the basic idea underlying local search is to start from an initial search position in the space of all possible assignments (typically a randomly or heuristically generated assignment, which may be infeasible, sub-optimal or incomplete), and to improve iteratively this assignment by means of minor modifications. At each *search step* we move to a new assignment selected from a *local neighborhood*, chosen via a heuristic evaluation function. This process is iterated until a *termination criterion* is satisfied. The termination criterion is usually the fact that a solution is found or that a predetermined number of steps is reached. To ensure that the search process does not stagnate, most local search methods make use of random moves: at every step, with a certain probability a random move is performed rather than the usual move to the best neighbor. Another way to prevent local search from locking in a local minima is Tabu search [9] that uses a short term memory to prevent the search from returning to recently visited assignments for a specified amount of steps.

A local search approach has been defined in [6] to find an optimal solution in a soft constraint problem. This approach starts from a randomly chosen assignment to all the variables, say s, and at each step it moves to a new assignment which is obtained by changing the value of one variable. Such a variable is one of those whose *local preference* is minimal in s. The local preference of a variable in an assignment s is the combination of the preferences identified by s in all constraints involving the variable.

Consider for example the problem in Figure 1 where all missing preferences are set to 1. This is a classical fuzzy constraint problem. Consider now the assignment $\langle A = a, B = d, C = r\rangle$. In this assignment, the local preference of variable A is $min(0.3, 0.4, 0.9) = 0.3$, while it is 0.4 for B and 0.9 for C. Thus, variable A would be chosen by the local search algorithm.

Once the variable, say x, is chosen, its new value is identified by computing, for each new value v, the preference of the new assignment (that is, s with the new value v for x). The value which gives the best preference for the new assignment is chosen.

Consider again the example in Figure 1 where all missing preferences are set to 1, and assignment $\langle A = a, B = d, C = r\rangle$. This assignment has preference 0.3. Variable A can be changed to values b or c. With $A = b$, the new assignment has preference 0.6, while with $A = c$, the new assignment has preference 0.5, Thus the algorithm would move to the assignment $\langle A = b, B = d, C = r\rangle$ what has preference 0.6.

In the following we adapt this algorithm to deal with incompleteness.

3 Local search on ISCSPs

We will now present our local search algorithm for ISCSPs that interleaves elicitation with search. We basically follow the same algorithm as in [6], except for the following.

To start, we randomly generate an assignment of all the variables. To assess the quality of such an assignment, we compute its preference. However, since some missing preferences may be involved in the chosen assignment, we ask the user to reveal them.

In each step, when a variable is chosen, its local preference is computed by setting all the missing preferences to the best preference value (which is 1 for fuzzy constraints and 0 for weighted constraints). In other words, if there are missing preferences, they are not considered in computing the local preference of a variable in a given assignment (since the best value is the neutral element for the combination).

Consider again the example in Figure 1 and the assignment $\langle A = a, B = e, C = r\rangle$. In this assignment, the local preference of variable A is $min(0.3, 0.4, 0.9) = 0.3$, while for B is 0.4, and for C is $min(0.9, 1) = 0.9$. Thus our algorithm would choose variable A.

To choose the new value for the selected variable, we compute the preferences of the assignments obtained by choosing the other values for this variable. Since some preference values may be missing, in computing the preference of a new assignment we just consider the preferences which are known at the current point. We then choose the value which is associated to the best new assignment. If two values are associated to assignments with the same preference, we choose the one associated to the assignment with the smaller number of incomplete tuples. In this way, we aim at moving to a new assignment which is better than the current one and has the fewest missing preferences.

In the running example above, from assignment $\langle A = a, B = e, C = r\rangle$, once we know that variable A will be changed, we compute $pref(\langle A = b, B = e, C = r\rangle) = 0.4$ and $pref(\langle A = c, B = e, C = r\rangle) = 0.3$. Thus we would select the value b for A.

Since the new assignment, say s', could have incomplete tuples, we ask the user to reveal enough of this data to compute the actual preference of s'. Of course, asking for all the missing preferences is always a correct strategy; we call ALL the elicitation strategy that elicits all the missing preferences associated to the tuples obtained projecting s' on the constraints. However, depending on the class of soft constraints considered, asking for less than all the preferences could be sufficient. For fuzzy constraints, we also consider an elicitation strategy, called WORST, that asks the user to reveal only the worst preference among the missing ones, if it is less than the worst known preference. This is enough to compute the actual preference of s' since the preference of an assignment coincides with the worst preference in its constraints.

For weighted constraints, we consider the following three strategies (besides ALL):

– WW: we elicit the worst missing cost (that is, the highest) until either all the costs are elicited or the current global cost of the assignment is higher than the preference of the best assignment found so far;
– BB: we elicit the best (i.e., the minimum) cost with the same stopping condition as for WW;
– BW: we elicit the best and the worst cost in turn, with the same stopping condition as for WW.

As in many classical local search algorithms, to avoid stagnation in local minima, we employ tabu search and random moves. Our algorithm has two parameters: p, which is the probability of a random move, and t, which is the tabu tenure. When we have to choose a variable to re-assign, the variable is either randomly chosen with probability p or, with probability (1-p), we perform the procedure described above. Also, if no improving move is possible, i.e., all new assignments in the neighborhood are worse than or equal to the current one, then the chosen variable is marked as tabu and not used for t steps.

While in classical local search scenarios the underlying problem is always the same, and we just move from one of its solutions to another one, in our scenario we also change the problem via the elicitation strategies. Since the change involves only the preference values, the solution set remains the same, although the preferences of the solutions may decrease over time.

During search, the algorithm maintains the best solution found so far, which is returned when the maximum number of allowed steps is exceeded. In the ideal case, the returned solution is a necessarily optimal solution of the initial problem with the preferences added by the elicitation. However, there is no guarantee that this is so: via elicitation we can reach a problem with necessarily optimal solutions, but the algorithm may fail to find one of those. However, we will show later that, even in this case, the quality of the returned solutions is not very far from that of the necessarily optimal solutions.

4 Experimental settings and results

4.1 Problem generator

The test sets for fuzzy and weighted incomplete CSPs are created using a generator that has the following parameters:

– n: number of variables;
– m: cardinality of the variable domains;
– d: density, that is, the percentage of binary constraints present in the problem w.r.t. the total number of possible binary constraints that can be defined on n variables ($n(n-1)/2$);
– t: tightness, that is, the percentage of tuples with the worst preference (that is, 0 for the fuzzy constraints and $+\infty$ for the weighted constraints) in each constraint and in each domain w.r.t. the total number of tuples (m^2 for the constraints, since we have only binary constraints, and m in the domains);

– i: incompleteness, that is, the percentage of incomplete tuples (that is, tuples with preference ?) in each constraint and in each domain.

Given values for these parameters, we generate ISCSPs as follows. We first generate n variables and d% of the $n(n-1)/2$ possible constraints. Then, for every domain and for every fuzzy (resp., weighted) constraint, we generate a non-zero (resp., non-$+\infty$) random preference value for each of the tuples (that are m for the domains, and m^2 for the constraints); we randomly set t% of these preferences to 0 (resp., $+\infty$); we randomly set i% of the preferences in each constraint and domain as incomplete. Preference values range in $[0, 1]$ (with 101 possible different values) for IFCSPs and in $[0, 10] \cup +\infty$ (with 12 possible different values) for IWCSPs.

Our experiments measure the percentage of elicited preferences (over all the missing preferences), the solution quality (as the normalized distance from the quality of necessarily optimal solutions), and the execution time, as the generation parameters vary.

We executed our algorithm using a step limit of 100000, a random walk probability of 0.2 and tabu tenure of 1000. All results are an average over 100 problem instances.

4.2 Incomplete fuzzy CSPs

We tested the performance of our algorithm using both the ALL and the WORST elicitation strategy. We also compared the result with one of the best algorithms in [3], called here FBB (which stands for fuzzy branch and bound). In [3], this algorithm corresponds to the one called DPI.WORST.BRANCH.

We first considered the quality of the returned solution. To do this, we computed the distance between the preference of the returned solution and that of the necessarily optimal solution returned by algorithm FBB. Such a distance is measured as the percentage over the whole range of preference values. For example, if the preference of the solution returned is 0.4 and the one of the solution given by FBB is 0.5, the preference error reported is 10%. A higher error denotes a lower solution quality.

Figure 2 shows the preference error when density, incompleteness, tightness, and the number of variables vary (please notice that the y-axis ranges from 0% to 10%). We can see that the error is always very small and its maximum value is 3.5% when we consider problems with 20 variables. In most of the other cases, it is below 1.5%. We also can notice that the solution quality is practically the same for both elicitation strategies.

If we look at the percentage of elicited tuples (Figure 3), we can see that the WORST strategy elicits always less tuples than ALL, eliciting only 20% of incomplete tuples in most of the cases. When tightness is above 40%, WORST elicits very few tuples since the algorithm discovers soon that there are no solutions. The FBB algorithm elicits about half as many preferences as WORST. Thus, with 10 variables, FBB is better than our local search approach, since it

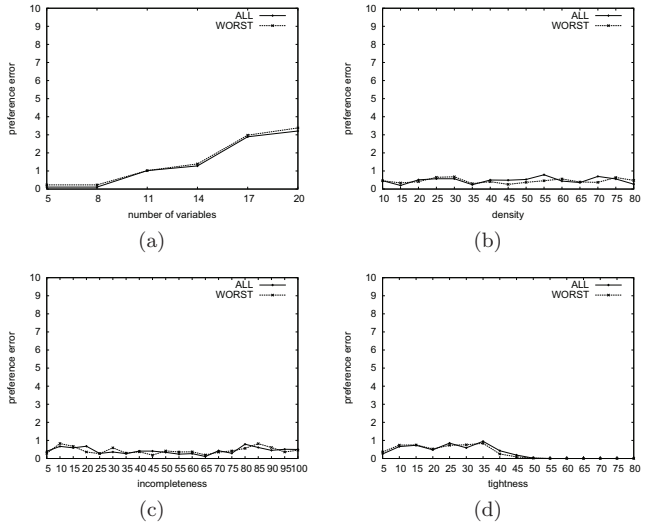


Fig. 2. Solution quality. When a parameter is fixed, its value is: n=10, m=5, d=50%, i=30%, t=10%.

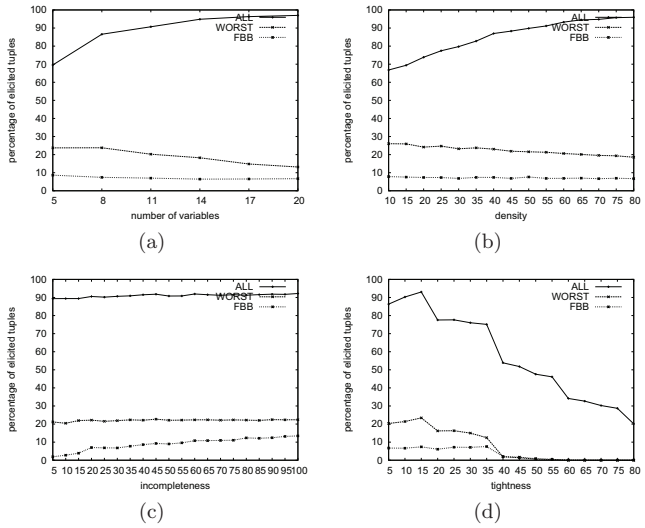
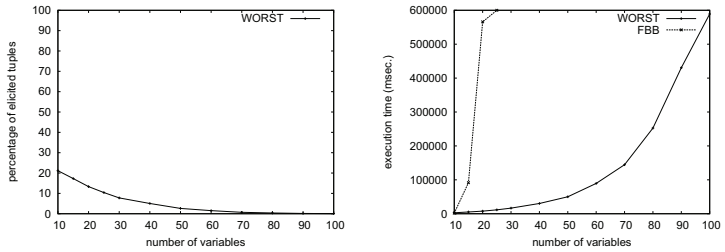


Fig. 3. Percentage of elicited tuples. When a parameter is fixed, its value is: n=10, m=5, d=50%, i=30%, t=10%.

guarantees to find a necessarily optimal solution while eliciting a smaller number of preferences.

We also tested the WORST strategies varying the number of variables from 10 to 100. In Figure 4(a) we show how the elicitation varies up to 100 variables. It is easy to notice that with more than 70 variables the percentage of elicited tuples decreases. This is because the probability of a complete assignment with a 0 preference arises (since density remains the same). Moreover, we can see how the local search algorithms can scale better than the branch and bound approach. In Figure 4(b) the FBB reaches a time limit of 10 minutes with just 25 variables, while the WORST algorithm needs the same time to solve instances of size 100.



(a) percentage of elicited tuples (b) execution time

Fig. 4. Values of the fixed parameters: m=10, d=35%, i=30%, t=5%.

We also investigated the runtime behavior of our local search algorithm. Surprisingly, the algorithm elicits almost all the tuples it needs within the first 10000 steps. Moreover the distance from the necessarily optimal preference decreases significantly during the first 20000 steps and then it decreases slightly until the end of the search. This behavior is the same no matter which parameter is varying. Hence we can stop our algorithm after 20-30000 steps whilst still ensuring a good solution quality.

4.3 Incomplete weighted CSPs

We tested the performance of our algorithm using the ALL, BB, WW and BW elicitation strategies. We also compared the result with one of the best algorithms in [3] for solving incomplete weighted CSPs, which we call WBB here (in [3] it was called DPI.BW.TREE). Figure 6 shows the solution quality, in terms of the preference error, which is measured similarly to the fuzzy context but with a specific treatment to deal with $+\infty$. More precisely, the error is the percentage difference between the solution preference found by the local search algorithm and the one found by WBB over the preference range. Notice that, with costs in $[0, 10] \cup +\infty$, solutions have preferences that may be between 0 and

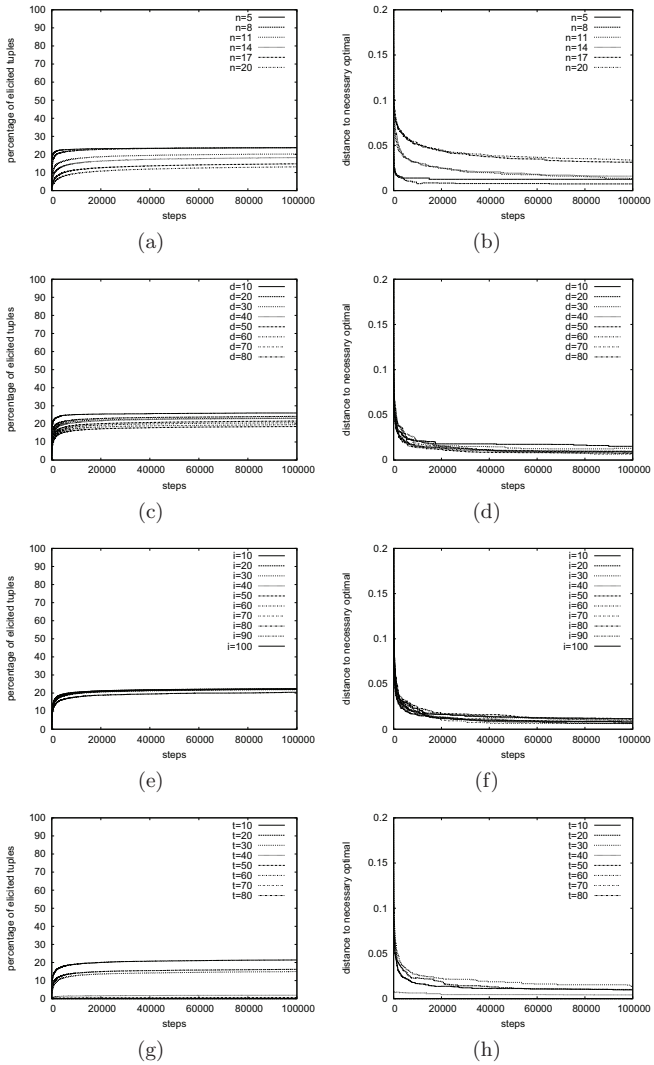


Fig. 5. Runtime behavior of WORST. When a parameter is fixed, its value is: n=10, m=5, d=50%, i=30%, t=10%.

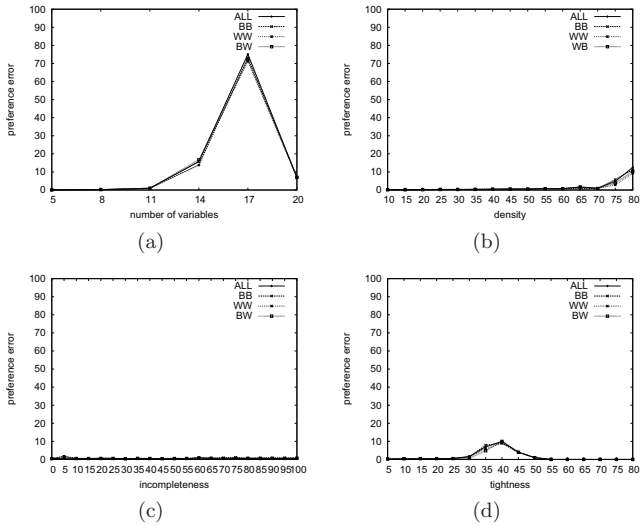


Fig. 6. Solution quality. When a parameter is fixed, its value is: n=10, m=5, d=50%, i=30%, t=10%.

$(((n*(n-1))/2)+n)*10$, or $+\infty$. When the returned preference is $+\infty$ and the correct preference is different, we report an error of 100%. For example, with 10 variables and preferences in $[1,10] \cup +\infty$, if the local search algorithm returns a solution with preference 120 and the necessarily optimal preference is 100, then the error is (120-100)*100/550 = 3.64%.

In most cases, all the local search algorithms find a solution with a preference very close to the necessarily optimal one. The peaks at 17 variables in Figure 6(a), at $d = 80\%$ in Figure 6(b), and at $t = 40\%$ in Figure 6(d) show a phase transition where the number of solutions with infinite cost increase significantly. This affects the quality of the solutions found by local search, while the branch and bound approach is able to find an optimal solution with a finite cost.

As in the fuzzy case, we measured the percentage of elicited tuples. As before, the local search approach elicits more preferences than the branch and bound based algorithms in [3]. However, the difference is fairly small and independent of the amount of incompleteness (see Figure 7(c)). Moreover, it is small also with density is below 60%, or when tightness is less than 30% or greater than 65% (Figure 7(b)). Among the local search algorithms, as expected, the algorithms BB, WW, and BW elicit less preferences than ALL.

To study the runtime behavior of our local search approach, we focused on algorithm BB, which is one of the best algorithms. Figure 8 shows how the percentage of elicited tuples and the distance from the necessarily optimal preference vary as the execution proceeds. From Figures 8(a), 8(c), 8(e) and 8(g) we

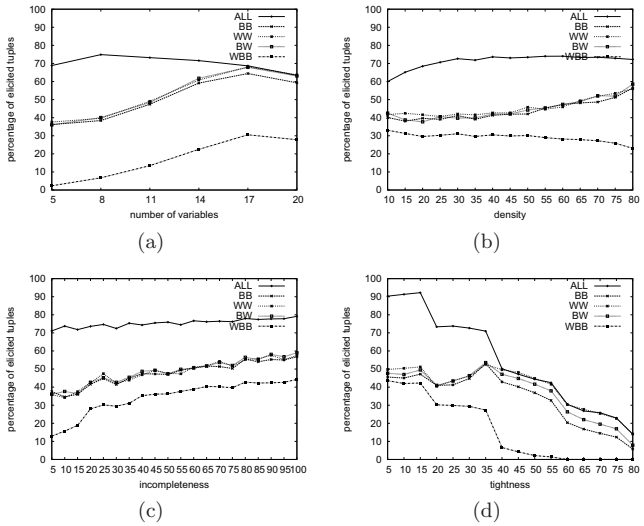


Fig. 7. Percentage of elicited tuples. When a parameter is fixed, its value is: n=10, m=5, d=50%, i=30%, t=10%.

can see that the elicitation process takes place mainly in the first 10000 steps. This behavior does not depend on the parameter that is varying.

Furthermore, if we consider the distance from the necessarily optimal preference, we can see that the algorithm finds the best solution in the first 20000 steps (see Figures 8(b), 8(d), 8(f) and 8(h)). Exceptions occur around the peaks of Figure 6. For example, in Figure 8(b) with $n = 17$ or $n = 20$ the solution preference is improved during the whole execution and not only in the first steps. Other examples are for $d = 80\%$ in Figure 8(d) and for $t = 40\%$ in Figure 8(h).

As in the fuzzy case, we tested our local search algorithms (using the WB strategy in this case) on instances up to 100 variables. From Figure 8(a) we can see that the algorithm elicits around 30% of incomplete preferences from 10 to 90 variables. From 90-100 variables, the instances start to have solutions equal to $+\infty$ and the algorithm elicits more preferences. In Figure 9(b) we measured the execution time of WB compared with WBB. The branch and bound algorithm reaches the time limit of 10 minutes per instance with just 15 variables so we stopped the execution at 30 variables. On the other hand, the local search algorithm, can solve instances up to 90 variables taking less time.

Summarizing, we can see that our local search approach finds a solution with a quality which is very close (with an error of at most 2%) to the quality of the necessarily optimal ones. Moreover, such a quality can be obtained also if execution is stopped after only 10000 steps.

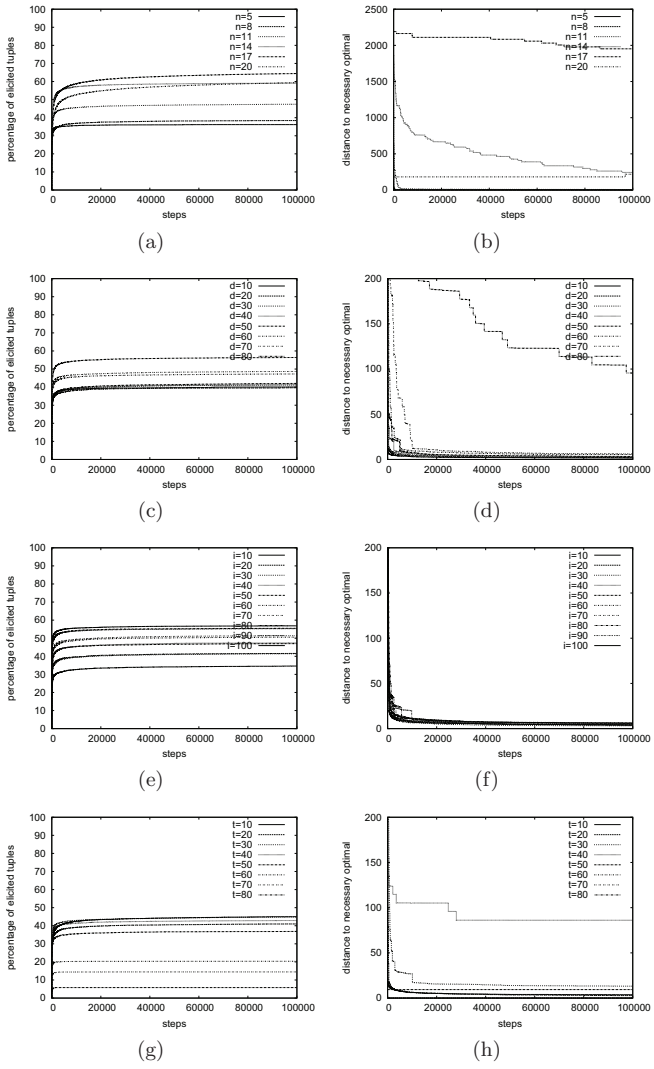
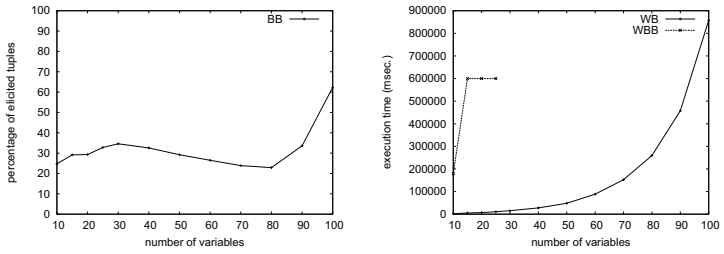


Fig. 8. Runtime behavior of WB. Values of the fixed parameters: m=10, d=35%, i=30%, t=5%.

5 Conclusions

We developed and tested a local search algorithm to solve incomplete fuzzy and weighted CSPs. We tested different elicitation strategies and, in both cases, our



(a) percentage of elicited tuples (b) execution time

Fig. 9. Values of the fixed parameters: m=10, d=35%, i=30%, t=5%.

best strategies have shown good results compared with the branch and bound solver described in [3]. More precisely, our local search approach shows a very good solution quality when compared with complete algorithms. In addition it shows better scaling properties than such complete methods.

6 Acknowledgements

Research partially supported by the Italian MIUR PRIN project 20089M932N: "Innovative and multi-disciplinary approaches for constraint and preference reasoning".

References

1. Dechter, R.: Constraint processing. Morgan Kaufmann (2003)
2. Pedro Meseguer, Francesca Rossi, T.S.: Soft constraints. In Rossi, F., Beek, P.V., Walsh, T., eds.: Handbook of Constraint Programming. Elsevier (2006)
3. Gelain, M., Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies. AI Journal **174**(3-4) (2010) 270–294
4. Gelain, M., Pini, M.S., Rossi, F., Venable, K.B.: Dealing with incomplete preferences in soft constraint problems. In: Proc. CP'07. Volume 4741 of LNCS., Springer (2007) 286–300
5. Gelain, M., Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Dealing with incomplete preferences in soft constraint problems. In: Proc. CP'08. Volume 5202 of LNCS., Springer (2008) 402–417
6. Codognet, P., Diaz, D.: Yet another local search method for constraint solving. In: Proc. SAGA 2001. (2001)
7. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint solving and optimization. Journal of the ACM **44**(2) (mar 1997) 201–236
8. Hoos, H.H., Stutzle, T.: Stochastic Local Search: Foundations and Applications. Elsevier - Morgan Kaufmann (2004)
9. Glover, F., Laguna, M.: Tabu Search. Kluwer, Norwell, MA (1997)