Conflict-based Statistics*

Tomáš Müller[1], Roman Barták[1] and Hana Rudová[2]

[1]Faculty of Mathematics and Physics, Charles University
Malostranské nám. 2/25, Prague, Czech Republic
{muller|bartak}@ktiml.mff.cuni.cz
[2]Faculty of Informatics, Masaryk University
Botanická 68a, Brno 602 00, Czech Republic
hanka@fi.muni.cz

1 Introduction

Value ordering heuristics play an important role in solving various problems. They allow to choose suitable values for particular variables to compute a complete and/or optimal solution. Usually, problem-specific heuristics are applied because problem-independent heuristics are computationally expensive. Here we propose an efficient problem-independent approach to value selection whose aim is to discover good and poor values. We have already applied this so called conflict-based statistics (CBS) in our iterative-forward search algorithm [7]. This combination helped us to solve a large-scale timetabling problem at Purdue University. Here we describe a general scheme for the conflict-based statistics and apply it to local search and iterative forward search methods.

Methods similar to CBS were successfully applied in earlier works [2, 4]. In our approach, the conflict-based statistics works as an advice in the value selection criterion. It helps to avoid repetitive, unsuitable assignments of the same value to a variable. In particular, conflicts caused by this assignment in the past are memorized. In contrast to the *weighting-conflict* heuristics proposed in [4], conflict assignments are memorized together with the assignment which caused them. Also, we propose our statistics to be unlimited, to prevent short-term as well as long-term cycles.

2 General Conflict-based Statistics

The main idea behind conflict-based statistics is to memorize conflicts and discourage their future repetition. For instance, when a value is assigned to a variable, conflicts with some other assigned variables may occur. This means that there are one or more constraints which prohibit the applied assignment together with the existing assignments. A counter, tracking how many times such an event occurred in the past, is stored in memory. If a variable is selected for an assignment (or reassignment) again, the stored information about repetition of past conflicts is taken into account.

Conflict-based statistics is a data structure that memorizes hard conflicts which have occurred during the search together with their frequency and assignments which caused them. More precisely, it is an array

$$CBS[V_a = v_a \ \rightarrow \ \neg V_b = v_b] = c_{\text{ab}}.$$

It means that the assignment $V_a = v_a$ caused c_{ab} times a hard conflict with the assignment $V_b = v_b$ in the past. Note that it does not imply that these assignments $V_a = v_a$ and $V_b = v_b$ cannot be

*This work is partially supported by the Czech Science Foundation under the contract No. 201/04/1102 and by Purdue University.

used together in case of non-binary constraints. The proposed conflict-based statistics does not actually work with any constraint. It only memorizes the conflict assignments together with the assignment which caused them. This helps us capture similar cases as well, e.g., when the applied assignment violates a constraint different from the past ones, but some of the created conflicts are the same. It also reduces the total space allocated by the statistics.

The conflict-based statistics can be implemented as a hash table. Such structure is empty in the beginning. During computation, the structure contains only non-zero counters. A counter is maintained for a tuple $[A = a \ \rightarrow \ \neg B = b]$ in case that the value a was selected for the variable A and this assignment $A = a$ caused a conflict with an existing assignment $B = b$. An example of this structure

$$A = a \quad \Rightarrow \quad 3 \times \neg B = b, \quad 4 \times \neg B = c, \quad 2 \times \neg C = a, \quad 120 \times \neg D = a$$

expresses that variable B conflicts three times with its assignment b and four times with its assignment c, variable C conflicts two times with its assignment a and D conflicts 120 times with its assignment a, all because of later selections of value a for variable A. This structure can be used in the value selection heuristics to evaluate conflicts with the assigned variables. For example, if there is a variable A selected and its value a is in conflict with an assignment $B = b$, we know that a similar problem has already occurred 3 times in the past, and the conflict $A = a$ can be weighted with the number 3.

3 Conflict-based Statistics in Iterative Forward Search

The iterative forward search (IFS) algorithm [7] is based on local search methods [5]. As opposed to classical local search techniques, it operates over a feasible, though not necessarily complete, solution. In such a solution, some variables may be left unassigned. However, all hard constraints on assigned variables must be satisfied. This means that there are no violations of hard constraints like in systematic search algorithms. The difference here is that the feasibility of all hard constraints in each iteration step is enforced by the unassignments of the conflicting variables.

The algorithm attempts to move from one (partial) feasible solution to another via repetitive assignment of a selected value to a selected variable. During each step, a variable and a value from its domain is chosen for assignment. Even if the best value is selected (whatever 'best' means), its assignment to the selected variable may cause some hard conflicts with already assigned variables. Such conflicting variables are removed from the solution and they become unassigned. Finally, the selected value is assigned to the selected variable.

Conflict-based statistics memorizes these unassignments together with the assignment which caused them. Let us expect that a value v_0 is selected for a variable V_0. To enforce feasibility of the new solution, some previous assignments $V_1 = v_1$, $V_2 = v_2$, ... $V_n = v_n$ need to be unassigned. As a consequence we increment the counters

$$CBS[V_0 = v_0 \ \rightarrow \ \neg V_1 = v_1], CBS[V_0 = v_0 \ \rightarrow \ \neg V_2 = v_2], ..., CBS[V_0 = v_0 \ \rightarrow \ \neg V_n = v_n].$$

The conflict-based statistics is being used in the value selection criterion. A trivial *min-conflict* value selection criterion selects a value with the minimal number of conflicts with the existing assignments. This heuristics can be easily adapted to a *weighted min-conflict* criterion. Here the value with the smallest sum of the number of conflicts multiplied by their frequencies is selected. Stated in another way, the weighted min-conflict approach helps to select a certain value that might cause more conflicts than another value. The point is that these conflicts are not so frequent, and therefore they have a lower weighted sum. Our hope is that it can considerably help the search to get out of a local minimum.

4 Conflict-based Statistics in Local Search

Local search algorithms [5] (e.g., min-conflict [6] or tabu search [3]) perform an incomplete exploration of the search space by repairing an infeasible complete assignment. Unlike systematic

search algorithms, local search algorithms move from one complete (but infeasible) assignment to another, typically in a non-deterministic manner, guided by heuristics.

In each iteration step, a new assignment is selected from the neighboring assignments of the current assignment. A neighborhood of an assignment can be defined in many different ways, for instance, a neighbor assignment can be an assignment where exactly one variable is assigned differently. This way, a single variable is reassigned in each move.

From the conflict-based statistics' point of view, we would like to prohibit a move (a selection of a neighboring assignment) which repetitively causes the same inconsistency. An inconsistency can be identified by a variable whose assignment becomes inconsistent with assignments of some other variables, or a constraint which becomes violated by the move.

Simply, in each iteration step, one or more variables are *reassigned*. These reassignments can cause that one or more *unchanged* variables become inconsistent with the new assignment. This means that there is a constraint which was satisfied by the previous assignment and it is violated in the new assignment. The reason is that it prohibits concurrent value assignment of some unchanged variable(s) and some reassigned variable(s). The conflict-based statistics can memorize this problem (i.e., unchanged variables become inconsistent) together with its reason (i.e., reassigned variables). Moreover, we can use the same structure of counters $CBS[V_a = v_a \rightarrow \neg V_b = v_b] = c_{\mathrm{ab}}$ as above.

More precisely, we have reassigned variables V_1, V_2, ... V_n and unchanged variables W_1, W_2, ... W_m which become inconsistent. Let us expect that v_i is a new value assigned to V_i and w_j is a value assigned to W_j. If a constraint between V_i and W_j becomes violated, the counter $CBS[V_i = v_i \rightarrow \neg W_j = w_j]$ is incremented. Note also that such constraint might operate over more than two variables and some of its variables might already be inconsistent in the prior iteration because of another constraint.

For example, there might be values v_1 and v_2 assigned to variables V_1 and V_2 respectively. As a consequence two constraints become inconsistent:

- the constraint C_1 prohibits the assignment $V_1 = v_1$ with an existing assignments $W_1 = w_1$ and $W_2 = w_2$,
- the constraint C_2 prohibits both assignments $V_1 = v_1, V_2 = v_2$ with $W_3 = w_3$ and $W_4 = w_4$, but $W_4 = w_4$ is already inconsistent because of some other constraint C_3.

Then, the following counters are incremented:

- $CBS[V_1 = v_1 \rightarrow \neg W_1 = w_1]$ and $CBS[V_1 = v_1 \rightarrow \neg W_2 = w_2]$ wrt. C_1
- $CBS[V_1 = v_1 \rightarrow \neg W_3 = w_3]$ and $CBS[V_2 = v_2 \rightarrow \neg W_3 = w_3]$ wrt. C_2

The conflict-based statistics can be used in the move selection criterion. For example, if there is a reassignment $V_a = v_a$ contained in the move, and it causes an unchanged assignment $V_b = v_b$ to become inconsistent, such move can be weighted by the counter $CBS[V_a = v_a \rightarrow \neg V_b = v_b]$.

5 Experiments

We compare various local search algorithms. Basic instance of hill climbing [5] is extended by our conflict-based statistics and tabu search [3]. Also random walk min-conflict [6] is included in our comparison. Some experimental results for iterative forward search can be found in [7]. We present results achieved on a Random Binary CSP with uniform distribution [1]. A random CSP is defined by a four-tuple (n, d, p_1, p_2), where n denotes the number of variables and d denotes the domain size of each variable, p_1 and p_2 are two probabilities. They are used to generate randomly the binary constraints among the variables. p_1 represents the probability that a constraint exists between two different variables (tightness) and p_2 represents the probability that a pair of values in the domains of two variables connected by a constraint are incompatible (density).

Figure 1 presents the number of conflicting constraints wrt. the probability p_2 representing tightness of the generated problem $CSP(100, 20, 15\%, p_2)$. The average values of the best achieved solutions from 10 runs on different problem instances within the 60 second time limit are presented.

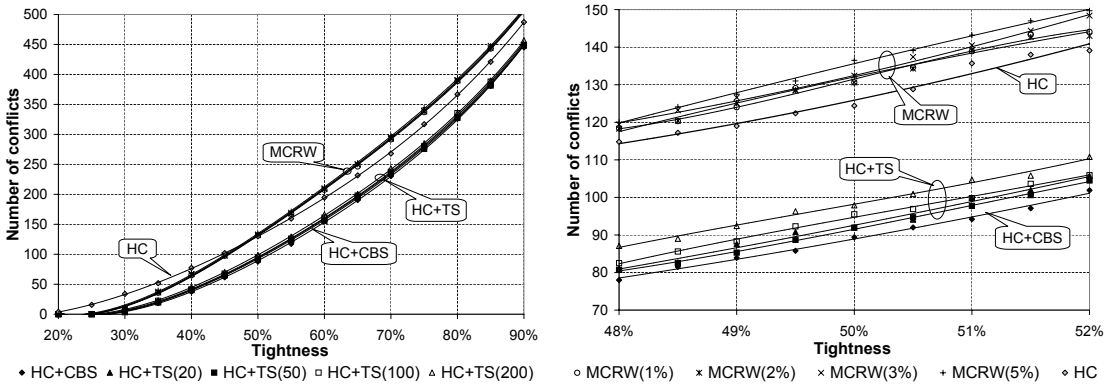

Figure 1: $CSP(100, 20, 15\%, p_2)$, the number of conflicting constraints in the best achieved solution within 60 seconds, average from 10 runs.

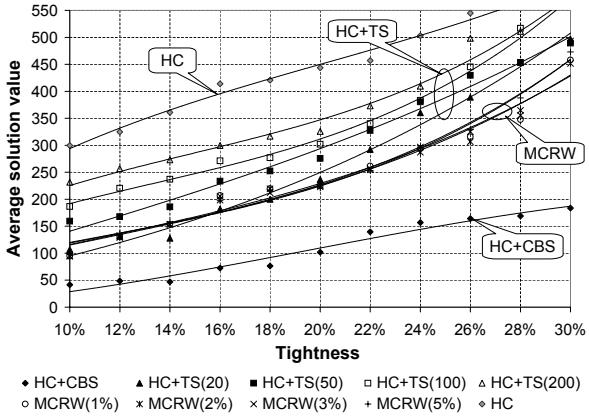

Figure 2: $minCSP(40, 30, 43\%, p_2)$, the sum of all assigned values of the best solution within 60 seconds wrt. problem tightness, average from 10 runs.

We have also solved other problems with different characteristics (sparse problems, dense problems, larger problems) but the results were similar.

For all the compared local search algorithms, a neighbor assignment is defined as an assignment where exactly one variable is assigned differently. *Min-conflict random walk* (marked MCRW(P_{rw})) algorithm selects a variable in a violated constraint randomly. Its value which minimizes the number of conflicting constraints is chosen. Furthermore, with the given probability P_{rw} a random neighbor is selected. For *hill-climbing* (marked HC) and *tabu search* (marked HC+TS(l), where l is the length of the tabu list), the best assignment among all the neighbor assignments is always selected. This means an assignment of a value to a variable which minimizes the number of conflicting constraints. Moreover, in tabu-search, if such an assignment is contained in the tabu list (a memory of the last l assignments), the second best assignment is used and vice versa (except of an aspiration criteria, which allows to select a tabu neighbor when the best ever found assignment is found). For the conflict-based statistics (marked HC+CBS), the best assignment is always selected as well, but the newly created conflicts are weighted by the frequencies of their previous occurrences.

As we can see from Figure 1, the conflict-based statistics approach is much better than the min-conflict random walk and the hill-climbing algorithms and slightly better than the tabu-search on the tested problem. Moreover, there is no algorithm specific parameter (which usually depends on the solved problem) unlike in the compared methods (e.g., the length of the tabu-list).

For the results presented in Figure 2, we turned the random CSP into an optimization problem. The goal is to minimize the total sum of values for all variables. Note that each variable has d

generated values from $0, 1, ...d-1$. For the comparison, we used $CSP(40, 30, 43\%, p_2)$ problems with the tightness p_2. The problems were chosen such that each algorithms was able to find a complete solution for each of 10 different generated problems within the given 60 second time limit[1].

All algorithms can be easily adopted to solve this *minCSP* problem by selecting an assignment with the smallest value among the neighbors minimizing the number of conflicts. But, the conflict-based statistics can do better. Here, we can add the value of the assignment to the number of conflicts (weighted by CBS). Then, a neighbor assignment with the smallest sum of the values and the conflicts weighted by their previous occurrences is selected. We can afford this approach because the weights of repeated conflicts are being increased during the search, and the algorithm is much more likely to escape from a local minima than the other compared algorithms.

For this problem, the presented conflict-based statistics was able to give much better results than other compared algorithms. The algorithm is obviously trying to stick much more with the smallest values than the others, but it is able to find a complete solution since the conflict counters are rising during the search. Such behavior can be very handy for many optimization problems, especially when optimization criteria (expressed either by some optimization function or by soft constraints) go against the hard constraints.

6 Conclusions and Future Work

We presented a general schema for value selection heuristics called conflict-based statistics. We applied it in iterative forward search and local search algorithms. Study of iterative forward search was already presented in our former paper [7]. Here we concentrated on the local search extension and we presented new experimental results with promising behavior for our new approach.

Our future study will include comparison of the described approaches on the various sets of experimental problems. We will also analyze complexity of particular extensions. As another step towards generalization of the conflict-based statistics, we would like to explore its possible extension for systematic search algorithms.

References

[1] Christian Bessière. Random uniform csp generators, 1996. `http://www.lirmm.fr/~bessiere/generator.html`.

[2] Rina Dechter and Daniel Frost. Backjump-based backtracking for constraint satisfaction problems. *Artificial Intelligence*, 136(2):147–188, 2002.

[3] Philippe Galinier and Jin-Kao Hao. Tabu search for maximal constraint satisfaction problems. In *Proceedings 3rd International Conference on Principles and Practice of Constraint Programming*, pages 196–208. Springer-Verlag LNCS 1330, 1997.

[4] Narendra Jussien and Olivier Lhomme. Local search with constraint propagation and conflict-based heuristics. *Artificial Intelligence*, 139(1):21–45, 2002.

[5] Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*. Springer, 2000.

[6] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[7] Tomáš Muller and Hana Rudová. Minimal perturbation problem in course timetabling. In *PATAT 2004 — Proceedings of the 5th international conference on the Practice And Theory of Automated Timetabling*, pages 283–303, 2004.

[1] Except of the hill-climbing which is not able to find a complete solution from tightness at around 16%.