Breaking Out CSPs

Julien Vion
vion@cril.univ-artois.fr

CRIL-CNRS FRE 2499,
Université d'Artois
Lens, France

Abstract. In this paper, we present the very efficient and unparametrized local search algorithm called Weighted Min-Conflicts (WMC), based on Morris' Breakout Method, and show how the relationship existing between WMC and the $dom/wdeg$ heuristic used with MGAC can be used to design a simple hybrid algorithm. An exhaustive experimentation will be performed, considering in particular the instances used as benchmarks during the second competition of CSP solvers. The results show how WMC and hybrid algorithms outperforms standard Min-Conflicts, Tabu search and MGAC-$dom/wdeg$ on various sets of random and structured instances.

1 Introduction

Search methods usually fall into one of two main families: systematic algorithms and local search algorithms. When dealing with discrete CSP instances, a usual approach to solve them is to use the MGAC-$dom/wdeg$ algorithm. MGAC is a systematic algorithm which maintains Generalized Arc Consistency (GAC), a powerful inference property, during search [9]. $dom/wdeg$ is an adaptive variable ordering heuristic that try to focus the search on harder parts of the problem [2]. Local search algorithms such as *Hill-Climbing Min-Conflicts* [7] or *Tabu Search* [4] start from an incorrect, usually randomly generated, complete assignment, and then perform an incomplete exploration of the search space by repairing this assignment. Although such algorithms cannot generally take profit of inference algorithms such as GAC, they may be far more efficient in terms of response time than systematic ones to find a first solution. However, they cannot guarantee that they find a solution or prove that none exists. There has been recently some effort to develop powerful hybrid algorithms, considering that the duality of local search and backtracking search should be combined to develop superior search methods. This as been formalized as a challenge in both satisfiability and CSP [10].

In this paper, we present *Weighted Min-Conflicts* (WMC), a powerful local search algorithm designed to quickly find feasible solutions to satisfiable CSP. We show the relation existing between WMC and MGAC-$dom/wdeg$ that allows to use very naturally WMC as an oracle to identify the hard parts of the problems, as proposed in [6].

2 Technical Background

A Constraint Network (CN) is a pair $(\mathscr{X}, \mathscr{C})$. $\mathscr{X}$ is a finite set of n variables such that each variable $X \in \mathscr{X}$ has an associated domain $\mathrm{dom}(X)$ denoting the set of values

Algorithm 1: update$\gamma(X$: Variable, v_{old}: Value)

```
1 foreach C ∈ 𝒞 | X ∈ scp(C) do
2     foreach Y ∈ scp(C) | X ≠ Y do
3         foreach v_y ∈ dom(Y) do
4             if check(C|_{Y=v_y}) ≠ check(C|_{Y=v_y ∧ X=v_old}) then
5                 if check(C|_{Y=v_y}) then γ(Y, v_y) ← γ(Y, v_y) − wght[C]
6                 else γ(Y, v_y) ← γ(Y, v_y) + wght[C]
```

allowed for X. $\mathscr{C}$ is a finite set of e constraints such that each constraint $C \in \mathscr{C}$ denotes the set of tuples allowed for the variables $\mathrm{scp}(C) \subseteq \mathscr{X}$ involved in the constraint C. The degree of a variable corresponds to the number of constraints involving the variable. In the remaining of this article, d will denote the size of the largest domain, Γ_{max} the maximal degree of the variables, and r will denote the maximal arity of the constraints. A solution to a CN is an assignment of values to all the variables such that all the constraints are satisfied. A CN is said to be satisfiable iff it admits at least one solution. The Constraint Satisfaction Problem (CSP) is the NP-complete task of determining whether a given CN is satisfiable. A CSP instance is then defined by a CN, and solving it involves either finding at least one solution or determining its unsatisfiability.

Although there also has been some interest in using Local Search techniques to solve the CSP problem [7, 4], these algorithms have not be studied a fraction as much as MGAC. A local search algorithm works on *complete assignments*: each variable is assigned with some value, then the assignment is iteratively *repaired* until a solution is found. A repair generally involves changing the value assigned to a variable so that as few constraints as possible are violated. The initial variable assignments may be randomly generated.

Designing efficient local search algorithms for CSP requires the use of clever data structures and powerful incremental algorithms in order to keep track of the efficiency of each repair. In [4], it is proposed to use a data structure $\gamma(X, v)$ which at any time contains the number of conflicts a repair would lead to. Algorithm 1 describes the management of γ ($check(C)$ controls whether C is satisfied by the current assignments of $\mathrm{scp}(C)$). Since each assignation has an impact only on the constraints involving the selected variable, we can perform the selection of the pair as well as counting conflicts incrementally, with a worst-time complexity of $O(\Gamma_{max} r d)$ at each iteration. The space complexity of γ is obviously in $O(nd)$.

There are many cases where no value change can improve the current assignment in terms of constraint satisfaction. In this case, we have reached a *local minimum*. The main challenge over local search techniques is to find the best way to avoid or escape local minima and carry on the search. Two such techniques have already been thoroughly studied in previous works: *Random Walks* (with a probability p, the repair is chosen randomly instead of being selected into the set of repairs that improves the current assignment [7]) and *Tabu Search* (previous repairs are recorded so that we can avoid repairs that lead back to an already visited assignment [4]). The local search algorithm implementing Random Walks will be called *Min-Conflicts Random Walk* (MCRW) in the remaining of this paper.

Algorithm 2: WMC($P = (\mathscr{X}, \mathscr{C})$: CN, $maxIterations$: Integer): Boolean

```
1  nbConflicts ← initP(P) ; initγ(P) ; nbIterations ← 0
2  while nbConflicts > 0 do
3      select (X, v) | γ(X, v) is minimal
4      v_old ← current value for X
5      if γ(X, v) ≥ γ(X, v_old) then
6          foreach C ∈ 𝒞 | C is in conflict do
7              wght[C]++ ; nbConflicts++
8              foreach Y ∈ scp(C) do
9                  foreach w ∈ dom(Y) do if ¬check(C|_{Y=w}) then γ(Y, w)++
10     else
11         P ← P|_{X=v} ; nbConflicts ← γ(X, v) ; updateγ(X, v_old)
12     if nbIterations++ > maxIterations then throw Expiration
13 return true
```

If no solution is found after $maxIterations$ iterations, the search is restarted with a new initial assignment. The best value for $maxIterations$ is highly dependent on the size and nature of the problem. If the value is too small, the search is unlikely to last enough to reach a solution. If the value is too high, much time may be lost in large local minima. MCRW and Tabu search use an additional parameter, respectively the probability p to perform a random walk, and the size of the Tabu list. Again, the performance of these algorithms is highly dependent on these parameters.

3 The WMC Local Search algorithm

Another efficient way to escape from local minima, called the Breakout method, has also been proposed [8]. We propose to use this method to design a local search algorithm aimed to find solutions to satisfiable CSPs. The resulting algorithm, *Weighted Min-Conflicts* (WMC) is described in Algorithm 2. $initP$ generates the initial random assignment. Line 5 detects local minima. When a local minimum is encountered, all conflicting constraints are weighted (line 7). Updating γ is then done in $O(erd)$. Note that a main advantage of WMC over Tabu search or MCRW is that it involves no parameter outside of $maxIterations$.

Incrementing the weight of constraints permits to effectively and durably escape from local minima. Incrementing the constraints "fills" the local minimum until another parts of the search space are reached. Constraints that are heavily weighted are expected to be the "hardest" constraints to satisfy, and the algorithm will try to satisfy them in priority. Incoherent problems usually contains a smaller set of constraints that form an incoherent sub-problem. When trying to solve such a problem, at each iteration, at least one of the constraints of this sub-problem will be in conflict. We thus expect that the constraints of this sub-problem to be heavily weighted. This observation has also be done recently on graph coloring problems [3]. We show that this assertion is also true on large real-life problems, such as RLFAP (Radio Link Frequency Assignment Problem). $scen11$-$f8$ is the RLFAP $scen11$ problem from which 8 frequencies were removed. The resulting problem is unsatisfiable. It involves 680 variables and $4,103$ constraints and contains at least one MUC of 28 constraints. After $50,000$ iterations of

WMC on $scen11\text{-}f8$. The average weight of all the $4,103$ constraints is 136, whereas the average weight of the 28 constraints from the known MUC is $2,590$.

4 WMC and MGAC-$dom/wdeg$: designing an hybrid algorithm

It is well known that the main drawback of systematic backtracking strategies such as MGAC is that an early bad choice may lead to explore a huge sub-tree that could be avoided if the heuristic had lead to focus on a rather small, very hard or even inconsistent sub-problem. In this case, the solver is said to be subject to "thrashing": it rediscovers the same inconsistencies multiple times. The $dom/wdeg$ heuristic was designed to avoid thrashing by focusing the search on one hard sub-problem [2]. This technique is reported to work quite well on structured problems.

Mazure et al. [6] report that statistics earned during a failed run of local search can be successfully as an oracle to guide a systematic algorithm in the search of a solution or to extract an incoherent core. Eisenberg & Faltings [3] have designed a simple hybrid algorithm based on this assumption, using the weights obtained with WMC. We propose to use directly the weights of the constraints obtained at the end of a WMC run to initiate $dom/wdeg$ weights. Since most complete and incomplete solvers use restarts (by limiting to $maxBT$ backtracks or $maxIterations$ iterations), we propose here to launch them sequentially in turn. The different algorithms are independent and each one keep its main advantages. If there exists a search strategy which is particularly efficient for a given type of problem (i.e. local search for large or dense problems, systematic search for unsatisfiable problems), we have only lost a limited amount of time when considering bad search strategies for this problem. Moreover, much interesting information can be kept from one run to another, even if the search fails. Outside of the constraint weights, MGAC can determine inconsistent values or tuples of values that can be recorded as additional constraints [5], as well as generating partial arc-consistent assigments.

Even though WMC and MGAC-$dom/wdeg$ do not need any additional parameter outside of $maxIterations$ and $maxBT$, tuning these was an important issue in the design of the Hybrid algorithm. We tried to make sure that both algorithms use approximately the same amount of resources during the search by measuring the speed of each solver internally and dynamically adapting $maxBT$. At each run, WMC is tried $maxTries$ times, then MGAC is tried once. After each run, $maxTries$ and $maxBT$ are increased by a factor α so that MGAC eventually becomes complete. This is reached in the worst case when $maxBT \geq d^n$.

5 Experiments

Experiments where run on a farm of Linux machines, each of which is equipped with 3 GHz x86-64 processors and 2 GiB of RAM. We compare the performance of MGAC-$dom/wdeg$ and the different local search algorithms (with GAC enforced during a preprocessing phase), and the hybrid algorithm. The maximum iterations for MCRW and Tabu search was fixed to $150,000$, and $2,000$ for WMC. MGAC-$dom/wdeg$ implements restarts with similar parameters as the hybrid algorithm. MCRW has p fixed to

		MGAC-$dom/wdeg$		MCRW		Tabu		WMC		Hybrid	
$instance$	nb	$solved$	$time$	$solved$	$time$	$solved$	$time$	$solved$	$time$	$solved$	$time$
qcp-qwh-bqwh	253	248	0.78	**253**	0.60	**253**	**0.41**	246	0.76	247	2.23
fapp	146	**141**	11.04	**141**	4.16	61	>600.00	131	**2.37**	**141**	119.27
shop	128	104	1.93	76	51.65	56	>600.00	**111**	**1.19**	103	5.98
rlfap	25	**25**	2.68	**25**	1.97	19	1.71	21	**1.14**	**25**	17.44
other structured	321	308	0.57	272	1.45	268	1.39	297	**0.51**	**310**	2.79
rand-d>n	94	**94**	3.89	31	>600.00	16	>600.00	33	>600.00	42	>600
rand-d<n	672	561	22.32	550	37.98	**592**	**19.7**	589	22.62	582	22.41

Table 1. Results on satisfiable instances (median time shown in seconds)

	pure MGAC-$dom/wdeg$			hybrid					
instance	$runs$	$assgns$	CPU	WMC		MGAC-$dom/wdeg$			$total\ CPU$
				$runs$	CPU	$runs$	$assgns$	CPU	
scen11-f8	1	13,596	**107.9**	1	126.4	1	**470**	19.1	145.5
scen11-f5	3	68,307	713.3	2	258.3	2	**47,692**	378.6	**634.3**
os-5-95-2	7	79,599	105.8	1	20.8	1	**3,148**	10.7	**31.5**
os-5-95-5	4	30,262	**52.1**	2	39.5	2	**11,372**	29.2	68.7
qK-50-5-mul	7	13,482	**452.3**	2	398	2	**2,495**	73.0	525.3
qK-80-5-mul	>9	>25,000	>6,000.0	2	1,666.2	2	**6,395**	971.6	**2,637.8**

Table 2. Results on various unsatisfiable problems

0.04 and Tabu size is fixed to 30. These parameters were found to be optimal for random problems near the threshold with 50 variables and 23 values. MCRW and Tabu search could benefit very much from parameter adjustments specific to each problem, but this is still an open question outside the scope of this paper.

We ran the algorithms on all instances from the Second International Solver Competition [1] available in the XCSP 2.0 format. Only instances that did not involve constraints with arity higher than 4 (dealing with constraints of high arity efficiently would need a non-boolean management of constraint checks [4]) and that could be proved satisfiable by at least one algorithm are taken into consideration. Results are summarized in Table 1. For each algorithm, the $solved$ column indicates how much problems were solved in less than 600 seconds. The $time$ column shows the median cpu time exhausted to solve the problems, in seconds. All these problems are of reasonable size and are treatable with MGAC-$dom/wdeg$ (local search algorithms are able to treat much larger problems). There are noticeable series, where local search is superior even though the problems are of relatively small size: shop problems (job-shop and open-shop) and hard random problems with smaller domain sizes ($d < n$). Tabu and MCRW show interest over WMC when their parameters match the right problems. In particular, random problems and qwh/qcp problems from the competition all have comparable domain sizes and number of variables. WMC is also often the fastest to reach a solution. The Hybrid approach seems to succeed in combining the advantages of WMC and MGAC-$dom/wdeg$ and is the approach that solves the largest number of problems, even though it often remains the slowest.

Table 2 shows that when running our hybrid solver on various unsatisfiable structured problems, in **all** cases the total number of assignments and cpu time for MGAC-$dom/wdeg$ is much lower when used in cooperation with WMC than alone. This shows

the effectiveness of the weight transfer. On hardest problems, the overall CPU Time is also better.

6 Conclusion and future work

In this paper, we presented *Weighted Min-Conflicts*, a local search algorithms that builds over Morris' Breakout Method. We studied the computing costs of iterating local search algorithms by using incremental evaluation of the repairs to perform at each iteration. We introduced an hybrid approach between systematic and local search, respectively represented by the MGAC-$dom/wdeg$ and WMC algorithms. We shown the relationship existing between the Breakout Method and the $dom/wdeg$ heuristic, and used this relationship to improve the efficiency of the search.

Experiments on various available benchmarks have shown the versatility of our approach, with good results over structured instances, even though the size and the nature of these are extremely variable. Our hybrid algorithm confirms results obtained by Mazure et al. on the SAT problem [6]. Moreover, with our approach the two algorithms are much more tied together since they both use heuristics that naturally use the same, shared information.

Much work still has to be done around the way to use weights both in local and systematic searches: in particular, latest refinements of constraint weighting used in SAT solvers use advanced strategies to lower the weights of the constrains periodically or automatically. We also believe that much more complementary information may be learned from one run of local search.

References

1. Second International CSP Solvers Competition. http://cpai.ucc.ie/06/Competition.html, 2006.
2. F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais. Boosting systematic search by weighting constraints. In *Proceedings of ECAI'04*, pages 146–150, 2004.
3. C. Eisenberg and B. Faltings. Using the Breakout Algorithm to Identify Hard and Unsolvable Subproblems. *the Proceedings of Principles and Practice of Constraint Programming CP-2003, LNCS*, 2833:822–826, 2003.
4. P. Galinier and J.K. Hao. A General Approach for Constraint Solving by Local Search. *Journal of Mathematical Modelling and Algorithms*, 3(1):73–88, 2004.
5. C. Lecoutre, L. Sais, S. Tabary, and V. Vidal. Nogood recording from restarts. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'2007)*, 2007.
6. B. Mazure, L. Sais, and E. Gregoire. Boosting complete techniques thanks to local search methods. *Annals of Mathematics and Artificial Intelligence*, 22:319–331, 1998.
7. S. Minton, M.D. Johnston, A.B. Philips, and P. Laird. Minimizing conflicts: a heuristic repair method for constraint-satisfaction and scheduling problems. *Artificial Intelligence*, 58(1-3):161–205, 1992.
8. P. Morris. The breakout method for escaping from local minima. In *Proceedings of AAAI'93*, pages 40–45, 1993.
9. D. Sabin and E. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *Proceedings of CP'94*, pages 10–20, 1994.
10. B. Selman, H. Kautz, and D. McAllester. Ten challenges in propositional reasoning and search. *Proc. IJCAI'97*, 1997.