# Tabu Search vs. Random Walk

Olaf Steinmann[1], Antje Strohmaier[2] and Thomas Stützle[1]

[1] FB Informatik, FG Intellektik, TH Darmstadt
Alexanderstr. 10, D-64283 Darmstadt, Germany
{olaf,tom}@intellektik.informatik.th-darmstadt.de
[2] KI/Wissensverarbeitung, TU Dresden
D-01062 Dresden, Germany
ast@inf.tu-dresden.de

**Abstract.** We investigate the benefit of Tabu Search for satisfiability (SAT) and constraint satisfaction problems (CSP) and compare it to the more frequently used random walk heuristic. We argue, that a more deterministic direction of search as done with Tabu Search is worth considering also for SAT and CSP. We give experimental evidence that Tabu Search can be used to efficiently guide local search procedures like GSAT and WSAT for SAT and the *min conflicts* heuristic for CSP. The algorithms are tested on randomly generated problems and hard graph coloring instances from the DIMACS benchmark test set. Additionally, we give some explanation on the value of Tabu Search.

## 1 Introduction

Constraint satisfaction problems (CSPs) and satisfiability (SAT) are central problems in Artificial Intelligence. Many practical problems like machine vision, spatial and temporal reasoning, and scheduling can be represented as CSP. SAT plays a central role in many reasoning applications and in complexity theory. Recently, SAT has also been used for the encoding of many combinatorial problems [10, 8]. These encoded problems then may be solved by general search algorithms for SAT. Traditionally, systematic complete algorithms based on backtracking were used to solve these problems. Only in recent years general local search procedures for SAT and CSPs like GSAT [16] and the *min conflicts* heuristic (MCH) [12], respectively, were proposed. These algorithms typically start with an initial complete assignment $s$ from the set of all possible assignments $\mathcal{S}$ and then perform a sequence of moves that lead from one trial solution to another. The possible moves are defined by a neighborhood function. The moves should lead to possibly better assignments satisfying more clauses or constraints. Local search algorithms for SAT or CSP can be seen as operating on an objective function or energy giving the number of unsatisfied clauses or violated constraints. Thus, these algorithms try to globally minimize the corresponding energy via local search.

One major problem of basic local search algorithms is that they can get stuck in a non-global minima of the objective function. Thus, some mechanism is required to allow the local search procedure to escape from local minima.

To this aim in GSAT restart from a new randomly generated solution is used. Another common extension to prevent the local search from getting stuck in local minima is the application of *random walk* that results in a probabilistic modification of the current solution [15]. Random walk modifies the value of a variable involved in a violated constraint or unsatisfied clause randomly by choosing some other value than the current one. This may lead to an increase of the energy function and, thus, helps to leave local minima. Another heuristic that allows to escape from local minima is Tabu Search [7, 6]. Tabu Search is, especially in the Operations Research community, regarded as one of the most powerful heuristics for the solution of hard combinatorial optimization problems. Tabu Search allows the local search heuristics to leave local minima by forbidding moves to recently visited solutions. Additionally for SAT and CSPs it has the advantage of forcing the algorithm to explore new regions of a *plateau.* This is especially helpful as in GSAT the search effort is determined by a plateau phase [5]. On a plateau many neighbored points exist that have the same objective function value. At such a point walk does not help any further, a more directed way, like offered by Tabu Search, to explore this plateau is needed. Although, initially proposed for MAX-SAT [7] and despite some recent work, see e.g. [11], it has not received very much attention for the solution of SAT and CSP compared to other heuristics like random walk. The aim of this paper is to investigate experimentally the application of Tabu Search for SAT and CSP compared to random walk option. We apply Tabu Search not only to GSAT, but also to WSAT [15] and the MCH, extending the work done in [11]. New is also that we investigate hybrid algorithms using both heuristics, Tabu Search and random walk, to see whether these heuristics may be integrated in one. Our experimental results suggest that Tabu Search behaves better than the "celebrated" random walk option on the problem classes tested in this paper.

In the next section we introduce the notion of constraint satisfaction- and SAT-problem. We describe shortly two local search heuristics for SAT and CSP and after that describe Tabu Search as an add-on heuristic to CSP- and SAT-algorithms. Section 3 gives our experimental setting and the obtained results for SAT and CSP together with an interpretation. In Section 4 we discuss some other work and conclude in the last section.

## 2 Local Search

### 2.1 CSP and Satisfiability

A discrete CSP is defined by a finite set of Variables $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$, every variable $X_i$ having an associated finite and discrete domain $D_i$, and a set of constraints $\mathcal{C}$. A constraint gives the allowed value combinations for the involved variables. A *k-ary constraint* between variables $\{X_{i1}, \ldots, X_{ik}\}$ is a relation $C \subseteq D_{i1} \times \ldots \times D_{ik}$. SAT may be interpreted as a special case of CSP in which the variables only take two values 1 (*true*) and 0 (*false*). We assume that the SAT problem is given in conjunctive clausal form (CNF), i.e. as a conjunction of disjunctions of literals, a literal being a variable or its negation. Usually the clauses in SAT contain more than two variables, thus the constraints are usually of higher arity.[3] We say a constraint is *violated* by a given assignment $s$ if the k-tuple of values of its variables are not contained in the relation. We denote as *conflict set* $\mathcal{K}(s)$ the set of variables that are involved in some constraint violation for the current variable assignment $s$. Anyway, SAT and CSP are two closely related problems. On the one hand, a CSP can be transformed into SAT [3], on the other hand SAT can be transformed into binary CSP [1]. This is, apart from the similarity of the basic local search procedures, the reason to treat here both problems together.

### 2.2 GSAT and MCH

GSAT and MCH are two basic greedy local search procedures for SAT and CSP, respectively. In Figure 1 we present an algorithmic frame that applies to both heuristics. Both heuristics start from an initial random assignment and perform hill climbing steps on the current solution. This is done until some upper bound of *maxTries* is reached. In every try, both aim to improve the current assignment *maxLoops* times. If no solution is found, search is restarted from a newly generated random assignment.[4] The two algorithms differ slightly in the *hillClimb*-step they perform and the neighborhood definition. In GSAT all variables are considered for a move and a variable is chosen whose flip decreases most the objective function value. MCH selects in each move one variable $X_i \in \mathcal{K}(s)$ and chooses a value for this variable among those that minimize the number of conflicts. Note that MCH may leave a variable's value unchanged[5] whereas in GSAT always one variable changes its corresponding value.

### 2.3 Random Walk, WSAT and Tabu Search

One major problem associated with MCH and GSAT is the occurrence of local minima. One common possibility to escape from local minima that is used in GSAT and MCH is to restart the algorithm with another initial random solution. Another possibility is the application of *random walk*, originally introduced in [14] and also used for CSP [20]. Random walk is applied with some fixed probability $p_w$ and usually performs by far better than restart from random solutions. When applied to GSAT with probability $p_w$ a variable appearing in a currently unsatisfied clause is flipped and with probability $1-p_w$ the usual greedy strategy is followed. For MCH a walk-step consists of picking a variable of the conflict set $\mathcal{K}(s)$ and choosing a random value of the variable's domain. By random walk the number of violated constraints or unsatisfied clauses may increase and local minima may be left. In the following, we always use GSAT with random walk,

[3] Note that SAT with clauses containing at most two variables is polynomially solvable.
[4] Initially MCH was presented without restarts to solve one specific problem instance.
[5] In case of MCH we count every variable selection as one move or flip, even if the chosen variable does not change its value.

```
procedure Local Search(𝒳, 𝒟, 𝒞)
  for i := 1 to maxTries do
    s := random assignment for X ∈ 𝒳;
    for j := 1 to maxLoops do
      if s is solution of 𝒞 then return s
      else
        possFlips := hillClimb(s, 𝒞);
        (V, d) := random element of possFlips;
        s := s[V, d];
      end if;
    end for;
  end for;
  return "no solution found";
end Local Search;
```

**Fig. 1.** General local search procedure applicable for SAT and CSPs.

abbreviated as GWSAT. We also used an extension of this strategy called WSAT [15]. This algorithm differs from GWSAT in making flips by first randomly picking a clause that is not satisfied by the current assignment, and then picking a variable within that clause to flip. The flips are made either at random (with walk probability $p_w$) or according to a greedy function.[6]

*Tabu Search* [7, 6] is a heuristic used for the guidance of an underlying local search algorithm. Tabu Search is able to leave local minima, but in contrast to random walk it relies less on probabilistic elements and guides the search in a more deterministic way. Generally, in Tabu Search the best possible move is made, also if the objective function value increases. To avoid revisiting of previously found solutions Tabu Search uses a *tabu list* of length $t$ containing the forbidden moves. The forbidden moves are identified by specific attributes characterizing previously found solutions and usually contains the reverse of the last $t$ moves.[7]

We applied Tabu Search to GWSAT, WSAT, and MCH by imposing the tabu conditions on the set of possible moves.[8] An important design criteria when using Tabu Search are the attributes that may be declared tabu. For GWSAT and WSAT the tabu list consists of the $t$ previously flipped variables, thus for them it is forbidden to flip variables currently in the tabu list. For MCH we considered two possibilities that differ in the tabu attributes used. In one version we declare variable-value pairs $(X_i, d_i)$ as tabu, in the other version complete variables are used as tabu attributes. If variable-value pairs are chosen as tabu attributes, we influence only the value selection for certain variables, when blocking whole

[6] For a more detailed description see [10].

[7] The attributes stored in the tabu list may be e.g. specific values for variables.

[8] We used the code for GWSAT and WSAT available from Bart Selman. Note, that this code also offers a Tabu Search option.

variables the selection of the next variable is modified. If we use variables as tabu attributes, this may also lead to a situation in which a variable $X_i \notin \mathcal{K}(s)$ has to be chosen to update its value, in case all variables in $\mathcal{K}(s)$ are tabu.

## 3 Experimental Results

We now report on our experimental results obtained with versions of Tabu Search. We compare GWSAT, MCH with random walk, and WSAT with extensions of these algorithms involving Tabu Search.[9] We investigate empirically the parameter combinations when random walk and Tabu Search are applied together, giving also an intuitive explanations of the role of these two components.

One part of the tests is based on randomly generated formulae. For SAT we used random 3-SAT formulae from the crossover region and for CSPs we used random binary CSPs generated according to the $< n, k, p, q >$-model [18]. Here $n$ is the number of variables, $k$ the uniform domain size of the variables, $p$ the probability that there is a constraint between a pair of variables, and $q$ is the conditional probability that a pair of values is allowed if a constraint between these variables exists.[10] We used hard binary CSP from the phase transition region [18], unsolvable instances were filtered out by a complete search procedure.

We also applied all algorithms to some graph coloring instances from the DIMACS benchmark problems. Whereas MCH may be used also for very large instances, the size of problems to which GWSAT may be applied is limited by the size of the SAT-encoded formula.

For all algorithms we calculated the number of moves needed to find a solution. Every problem instance was solved for a number of times and then the median number of moves is taken as an estimate of the solution cost. If we do not discuss specific parameter settings for an algorithm, the algorithm was run with the parameters found to be optimal in preliminary runs. New in our investigation is also that we used combinations of both strategies, random walk and Tabu Search. In Figures 2 to 5 we give the median cost surface for different combinations for $t$ and $p_w$. In all figures, the x-axis shows the tabu list length, the y-axis gives the walk probability and the z-axis shows the number of moves per solution.

### 3.1 Satisfiability

For SAT we tested random 3-SAT formula of different sizes and investigated empirically the relationship between the tabu list length $t$ and the walk probability for GWSAT and WSAT. This was done for 50 problems of size $n = 50, 100$ with a fixed clause/variable ratio of 4.3 corresponding to the phase transition region [17]. Each problem instance was run 50 times with WSAT and 10 times with

[9] We used the versions of GWSAT and WSAT supplied by Kautz and Selman. Note, that in these versions a Tabu Search option is offered.

[10] In the literature usually $p_2 = 1 - q$ is used, where $p_2$ is the probability that a pair of values is inconsistent.

GWSAT. We chose rather small problems as the experimental burden is very high for larger problems and we are more interested in the mutual influence of the parameter settings. Both heuristics were tested on various combinations of tabu list length $t$ and walk probability $p_w$. The success rates are omitted, always being near 100%. In Figures 2 and 3 the shape of the cost surface for 100-variable problems is given, for 50-variable problems and SAT-encoded graph coloring the cost surfaces are very similar, see [19].

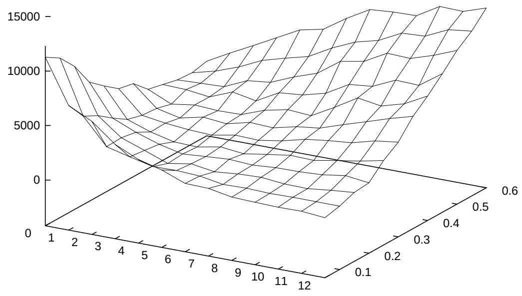


**Fig. 2.** GWSAT with Tabu Search for 3-SAT, $n = 100$, 430 clauses. The x-axis shows the tabu list length, the y-axis the walk-probability and the z-axis the number of moves per solution.

Several things are remarkable. First of all, the optimal parameter settings can not be determined by evaluating both parameters independently. Whereas for high walk probabilities $p_w$ a lower tabu list length $t$ decreases the solution cost, it is the other way round for low walk probabilities. The cost surfaces in both figures corresponding to the tabu list length and the walk probability suggest that the optimal parameter settings may not be found if one first optimizes e.g. the walk probability and after that tries to find the best parameter for the tabu length. Although this reduces the computational burden, the "best" parameter settings may be missed. This also explains, why — after fixing $p_w$ at the "optimal" value 0.5 — nearly nobody regarded Tabu Search as a valuable heuristics for SAT. There seems to be a strong dependence between the parameter values for which reasonably good results may be achieved – see the valley structure in Figures 2 and 3.

Furthermore one can obtain the optimal parameter settings to use for the solution of the specific problem class. For GWSAT the best parameter settings in the examined region are obtained with no walk and a tabu list length of 12. For WSAT the optimal parameter settings are with a tabu list length of $t = 4$

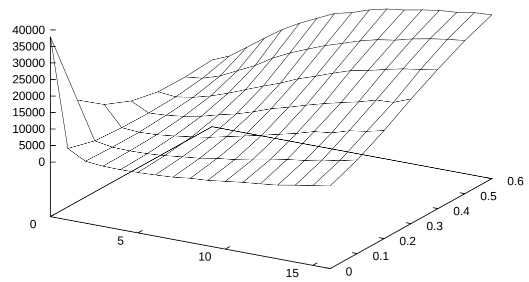


**Fig. 3.** WSAT with Tabu Search for 3-SAT, $n = 100$, clause/variable ratio 4.3.

and with walk probability $p_w = 0$. The specific parameter settings for both heuristics are qualitatively the same with respect to the effect of Tabu Search. The performance of both heuristics may be improved by adding Tabu Search. Yet, the specific tabu list length for the two heuristics is different, as basically the selection scheme of the variables to be flipped is different. We also tested both heuristics on SAT-encoded graph coloring problems, specifically on the problem instance `DSJC125.5` from [9] (also used in [16]). We solved the problem for 18 colors. As the median cost surface has the same shape as shown in Figure 3, we omit it here and only give some landmarks: On this problem, we found for WSAT with the optimal walk probability $p_{opt} = 0.1$ without Tabu Search, i.e. $t = 0$, a median cost of 33219 moves per solution resulting in a 100% success rate. The optimal common parameter setting for this problem was found to be $t_{opt} = 2, p_{opt} = 0.0$ with 100% success rate and a median of 9854 moves per solution — less than one third of the cost when using only random walk!

### 3.2 Constraint Satisfaction

For MCH we examined both variants of the tabu attributes proposed in Section 2.3. With `twmch` we denote the variant with walk in which variable-value pairs are declared tabu, with `tvwmch` we denote the variant with walk that considers variables as tabu attributes. In case only Tabu Search with variable-value tabu attributes is used, we denote this by `tmch`, `wmch` denotes MCH with walk but without any form of Tabu Search. We do not present results for `tvwmch` with $p_w = 0.0$, as this variant does not perform better than pure MCH without tabu lists and random walk.

**Random CSPs** For the results in Figures 4 and 5 we performed experiments with random binary CSP from the crossover region with 20 variables and domain size 10. The parameter values for $p, q$ correspond to the crossover region. In Figure 4 and 5 the results for parameter combinations of $p_w$ and $t$ are presented for one specific combination of $p$ and $q$. Both figures exhibit a similar behavior of the cost surfaces as the ones for GWSAT and WSAT, although the differences between the two strategies are not as marked as for SAT. The optimal parameter settings for twmch were found to be $t_{opt} = 2$ and $p_{opt} = 0.025$ with a median of 5868 moves, MCH using only random walk for $p_{opt} = 0.1$ had a cost of 7545 moves and MCH using only Tabu Search for $t_{opt} = 2$ a cost of 6217. For `tvwmch` using only Tabu Search gives rather poor results, but in combination with the random walk strategy yields considerable improvements over MCH using only random walk. The search cost for the optimal parameter settings $t_{opt} = 4, w_{opt} = 0.075$ yielded a median search effort of 5711, whereas using only walk for $w_{opt} = 0.125$ resulted in a median search cost of 8166.[11]

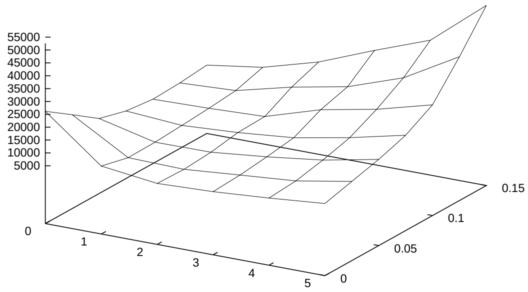


**Fig. 4.** `twmch` for random binary CSP, $p = 0.5$, $q = 0.62$.

Other experiments with values for $p \in \{0.1,\ 0.3,\ 0.5,\ 0.7,\ 0.9\}$, and corresponding values for $q$, such that the problem instances are from the crossover region, varying $q$ in steps of 0.01, showed that the variants of MCH using elements of Tabu Search performed all better than MCH using only Random Walk.[12]

The same observations as for WSAT and GWSAT on SAT problems also hold for CSP. In general, by adding elements of Tabu Search the search cost can be reduced considerably and the optimal parameter settings for both heuristics may not be determined independently. Additionally it seems that in the version `tvwmch` elements of Tabu Search and random walk may be integrated successfully.

[11] Note, that the number of problem instances used for `twmch` and `tvwmch` were not the same.

[12] We also experimented with MCH using other variants of random walk steps than the one presented here. Yet, none of those provided further significant improvements.

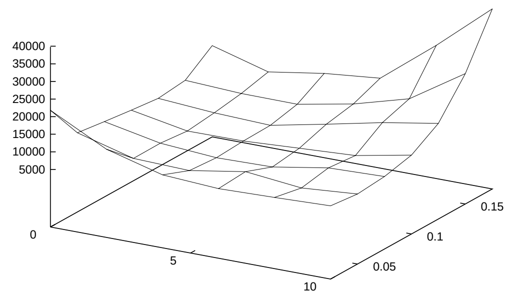


**Fig. 5.** `tvwmch` for random binary CSP, $p = 0.5$, $q = 0.62$.

**Graph Coloring Problems** We also tested MCH on hard graph coloring problems contained in the DIMACS benchmark set[13]. We present some of our results in Table 1. In the experiments for most problems a pure Tabu Search without random walk performed best, always yielding a lower search cost than MCH with random walk. Additionally MCH with random walk has the disadvantage that the optimal walk parameter depends strongly on the problem instance at hand. On the other side for all of the problems a tabu list length of $t = 2$, if variable-value pairs are used as tabu attributes, worked very well.

**Table 1.** Results for some graph coloring examples. $k$ is the number of used colors. The number of nodes in the problem instance are represented by the first number appearing in the problem instance descriptor. For success rates of 100% , we give the median of the number of moves, otherwise (x%) is the success rate.

| Problem / $k$ | `tmch` | `twmch` | `wmch` | `tvwmch` |
|---|---|---|---|---|
| DSJC125.1 / 5 | 33,452 | 26,243 | 67,879 | 43,846 |
| DSJC125.5 / 18 | 12,980 | 15,815 | 25,290 | 19,773 |
| DSJC125.5 / 17 | (72%) | (60%) | (34%) | (24%) |
| DSJC125.9 / 44 | 17,026 | 15,161 | 18,704 | 13,770 |
| DSJC250.5 / 30 | 70,570 | 84,530 | 462,421 | 460,533 |
| DSJC250.5 / 29 | (52% ) | (36%) | (4%) | (4%) |
| DSJC500.5 / 48 | 61,479 | 74,418 | (80%) | 191,460 |
| flat300_28_0 / 34 | 39,699 | 45,088 | 90,249 | 69,920 |
| le450_15c / 22 | 21,870 | 22,418 | 39,139 | 36,957 |

13 Available via: ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color/.

### 3.3 Interpretation of Results

As we have seen in the experimental results, variants of GSAT using elements of Tabu Search perform better than the variants of GSAT in which only random walk is used. Often Tabu Search alone performs best. In case of random walk for GSAT, Tabu search can best be interpreted as a heuristic that allows GSAT to leave local minima and that additionally helps to pass the plateau phase of the search. It forces GSAT-like algorithms to explore plateaus more systematically and therefore improves the performance significantly. On the other hand, random walk can be best seen as some kind of diversification strategy for Tabu Search, leading it into different regions of the search space. So, if the walk probability is high, Tabu Search is not able to exploit a specific region of the search space any more.

For MCH we used two different kinds of Tabu attributes, on the one side variable-value pairs, on the other whole variables. When considering a Tabu Search variant in which variable-value pairs are used as tabu attributes, our experimental results show that this version behaves better than pure MCH with random walk. Essentially, when combining both heuristics in `twmch` the walk-probability $p_w$ should be chosen very small, as for larger values, the search cost increases again. Thus, the addition of walk may be interpreted again as the application of a diversification strategy to explore different solutions. If we use variables as tabu attributes in `tvwmch`, the walk probability should be chosen as a slightly lower value than $w_{opt}$ in `wmch`. The effect of the tabu list then is to diversify the selection of the variables and to avoid that variables are chosen several times in a row.

## 4 Related Work

Tabu Search is a general heuristic for optimization problems. One of its first applications has been on MAX-SAT problems [7]. Since then it has not received much attention for the solution of hard SAT-problems, but GSAT and variants of GSAT involving random walk seemed to perform best. We know of only few works in which "memory-based" heuristics like Tabu Search were investigated for SAT and CSP. In [4] a version of GSAT with "memory", called MSAT, is reported to show "improved performance over GSAT ... although the improvement declines as problems grow larger". MSAT essentially equals GSAT with tabu list length 1. Another variant is HSAT which "always picks the (variable) that was flipped longest ago". This can also be best seen as a kind of tie-breaking rule if more than one variable is allowed to flip its value. HSAT is reported to "perform significantly better than GSAT", but has no specific mechanism to leave local minima and, thus, differs from Tabu Search.

In [11] positive results for the application of Tabu Search to SAT are reported. The authors have shown at hand of random 3-SAT instances that Tabu Search applied to GSAT behaves better than GWSAT. Additionally they investigated the dependence of the optimal tabu list length $t$ on the problem size and showed that a linear dependence holds. We extend the results presented in [11] in that we explicitly studied the relationship between the two strategies Tabu Search and random walk and we also show for WSAT that Tabu Search is to be preferred over the random walk option.

In the Constraint Satisfaction community applications of Tabu Search also seem to be rare. An exception here is [21] in which Tabu Search was used for MAX-CSP problems. In this application the selection scheme of the next value is a little bit different and it seems that no systematic investigation of the optimal tabu list length was done, as the authors note that they used usual values given in the literature, i.e. $t = 7$. In contrast to this, when using Tabu Search to solve CSP the tabu list length should be very low, e.g. $t = 2$ showed always a reasonably good performance. Note, that our version of Tabu Search using variables as tabu attributes has some similarity to HSAT [4] in that it also influences the variable selection. Future work will be to compare our variants involving Tabu Search also to some other search methods like the breakout-method [13] as well as to GENET [2], that relies heavily on the breakout method.

## 5 Conclusion

In this paper we presented the application of Tabu Search as a heuristic for greedy local search procedures used for SAT and CSP. As opposed to probabilistic elements like random walk, Tabu Search offers a more deterministic way of directing an underlying search procedure towards good solutions. We performed extensive tests on random 3-SAT problems at the crossover point, hard binary CSPs and hard graph coloring problems. We considered also hybrids of the two strategies and investigated experimentally the dependency between the tabu list length $t$ and the walk probability $p_w$. We found that optimal parameter settings may not be determined independently for both heuristics. Thus, both parameters have to be varied to find the best combination of both. Yet, in general we found that the variants using only Tabu Search significantly improve on the variants that rely only on the random walk option.

As we already pointed out, random walk may be interpreted as some special form of diversification of search. We think that by a stronger usage of information gathered during the run of the algorithm we are able to devise more directed diversification strategies improving the performance even more. Further work also will include a more theoretical explanation of the performance of Tabu Search. Obviously, the role of Tabu Search for SAT and CSP has been underestimated. This work aims to be an additional indication that it may be worthwhile to have a closer look at this heuristic that in other fields — e.g. scheduling or quadratic sum assignment problems — is applied with great success.

## References


1. H. Bennaceur. The Satisfiability Problem Regarded as a Constraint Satisfaction Problem. In Wolfgang Wahlster, editor, *ECAI'96*, pages 155–160. John Wiley & Sons, Chichester, 1996.

2. A. Davenport, E. Tsang, C.J. Wang, and K. Zhu. GENET: A Connectionist Architecture for Solving Constraint Satisfaction Problems by Iterative Improvement. In *AAAI'94*, 1994.
3. J. de Kleer. A Comparison of ATMS and CSP Techniques. In *IJCAI'89*. Morgan Kaufmann, 1989.
4. I. P. Gent and T. Walsh. Towards an understanding of hill–climbing procedures for SAT. In *AAAI'93*, pages 28–33. MIT press, 1993.
5. I.P. Gent and T. Walsh. An empirical analysis of search in GSAT. *Journal of Artificial Intelligence Research*, 1:47–59, 1993.
6. F. Glover. Tabu Search – Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.
7. P. Hansen and B. Jaumard. Algorithms for the Maximum Satisfiability Problem. *Computing*, 44:279–303, 1990.
8. H. H. Hoos. Solving hard combinatorial problems with GSAT — a case study. In *KI-96*, volume 1137 of *LNAI*, pages 107–119. Springer Verlag, 1996.
9. D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning. *Operations Research*, 39(3):378–406, 1991.
10. H. Kautz and B. Selman. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *AAAI'96*, volume 2, pages 1194–1201. MIT Press, 1996.
11. B. Mazure, L. Sais, and E. Gregoire. TWSAT: A New Local Search Algorithm for SAT – Performance and Analysis. In *CP'95 Workshop on Solving Hard Problems*, 1995.
12. 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*, 52:161–205, 1992.
13. P. Morris. The Breakout Method for Escaping from Local Minima. In *AAAI'93*, pages 40–45, 1993.
14. B. Selman and Henry A. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *IJCAI'93*, pages 290–295, 1993.
15. B. Selman, Henry A. Kautz, and Bram Cohen. Noise Strategies for Improving Local Search. In *AAAI'94*, pages 337–343. MIT press, 1994.
16. B. Selman, H. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In *AAAI'92*, pages 440–446. MIT press, 1992.
17. Bart Selman, David G. Mitchell, and Hector J. Levesque. Generating Hard Satisfiability Problems. *Artificial Intelligence*, 81(1–2):17–30, 1996.
18. B.M. Smith. Phase Transitions and the Mushy Region in Constraint Satisfaction Problems. In *ECAI'94*, pages 100–104, 1994.
19. Olaf Steinmann. Kombinatorische Probleme, Optimierung und Parallelisierung: Eine experimentelle Analyse von Multi–Flip–Netzwerken. Master's thesis, TH Darmstadt, FB Informatik, FG Intellektik, 1997.
20. Richard J. Wallace. Analysis of Heuristic Methods for Partial Constraint Satisfaction Problems. In *Constraint Programming 96*, LNCS, pages 482–496. Springer Verlag, 1996.
21. R.J. Wallace and E. Freuder. Heuristic Methods for Over-Constrained Constraint Satisfaction Problems. In *Over-Constrained Systems*, volume 1106 of *Lecture Notes in Computer Science*. Springer Verlag, 1996.

This article was processed using the LaTeX macro package with LLNCS style