On the Probabilistic Approximate Completeness of WalkSAT for 2-SAT*

Joseph Culberson[†] **Ian P. Gent**[‡] **Holger H. Hoos**[§]

February 7, 2000

Keywords: analysis of algorithms, combinatorial problems, satisfiability, local search algorithms, WalkSAT

Introduction. The WalkSAT algorithm framework is a family of highly successful stochastic local search (SLS) algorithms for the Satisfiability problem [6, 5]. WalkSAT/SKC, the first and most popular WalkSAT algorithm, has been shown to work well on SAT instances from a broad range of domains, including random 3-SAT formulae, and has been extensively studied, for example in [1]. Like many stochastic local search algorithms, WalkSAT/SKC is incomplete in the sense that it cannot be used to prove that a given problem instance is unsatisfiable and — maybe worse — for satisfiable instances, there is no guarantee that it actually finds a solution. However, for some incomplete stochastic search algorithms, such as Simulated Annealing, it can be shown that under certain conditions the probability of not solving any given soluble problem (failure probability) converges to zero as the run-time approaches infinity. In [1], this is

*Joseph Culberson's work was supported in part by an NSERC grant OGP8053 and an EPSRC visiting fellowship, GR/M54605. We thank members of APES for their help. J.C. and I.P.G. are members of the APES research group, www.apes.cs.strath.ac.uk.

[†]Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, T6G 2H1. joe@cs.ualberta.ca

[‡]School of Computer Science, University of St Andrews, St Andrews, Fife KY16 9SS, United Kingdom. ipg@dcs.st-and.ac.uk

[§]Department of Computer Science, University of British Columbia, Vancouver, B.C., Canada, V6T 1Z4. hoos@cs.ubc.ca

called the *PAC property* (probabilistic approximate completeness); incomplete algorithms which are not PAC are called *essentially incomplete*. PAC algorithms will almost certainly find a solution (if one exists) if left to run indefinitely. This property, directly only of theoretical interest, might have considerable practical importance, especially if combined with results on the failure probability's rate of convergence [2].

While for other members of the WalkSAT family, it could be shown that (when not using random restart) they are either PAC or essentially incomplete, the question whether WalkSAT/SKC (without random restart) is PAC remained open. In this article, we prove that a family of SAT algorithms including WalkSAT/SKC is PAC for 2-SAT. Unfortunately, the case for 3-SAT or general SAT seems much harder, and remains open. However, we have strong empirical evidence suggesting that even if WalkSAT/SKC is essentially incomplete for 3-SAT and general SAT, this does not have a significant affect on its behavior in practice.

Preliminaries. Let $\mathcal{S} = (V, \mathbf{C})$ be an instance of SAT containing a set of variables V and a set of clauses $\mathbf{C}$. Each clause contains a number of literals. An unnegated literal v is satisfied by $v = \mathit{true}$ while a negated literal $\widehat{v}$ is satisfied by $v = \mathit{false}$. A clause is satisfied if at least one literal in it is satisfied. Let $\mathbf{T} : V \rightarrow \{\mathit{true}, \mathit{false}\}$ be a truth assignment. The assignment $\mathbf{T}$ partitions $\mathbf{C}$ into unsatisfied and satisfied clauses, $\mathbf{C}_u$ and $\mathbf{C}_s$. $\mathcal{S}$ is satisfiable if there exists a satisfying assignment $\mathbf{T}$ for which $\mathbf{C}_u = \emptyset$. In 2-SAT, all clauses in $\mathbf{C}$ contain exactly two different literals.

We use the notation $\mathbf{T}\overline{v}$ to indicate the the truth assignment obtained from $\mathbf{T}$ by flipping the value of variable v, i.e., complementing its truth value under $\mathbf{T}$. A variable $v \in V$ is said to be *free under an assignment* $\mathbf{T}$ if v or $\widehat{v}$ occurs in a clause $C \in \mathbf{C}_u$ and for all $D \in \mathbf{C}_s$ under $\mathbf{T}$, $D \in \mathbf{C}_s$ under $\mathbf{T}' = \mathbf{T}\overline{v}$.

PAC Property of WalkSAT/SKC for 2-SAT. Let us now consider the algorithm family $\mathcal{WS}$ as specified in Figure 1 and make the following assumption:

Assumption 1 *The selections in step 1 and 2.1 are random (according to a*

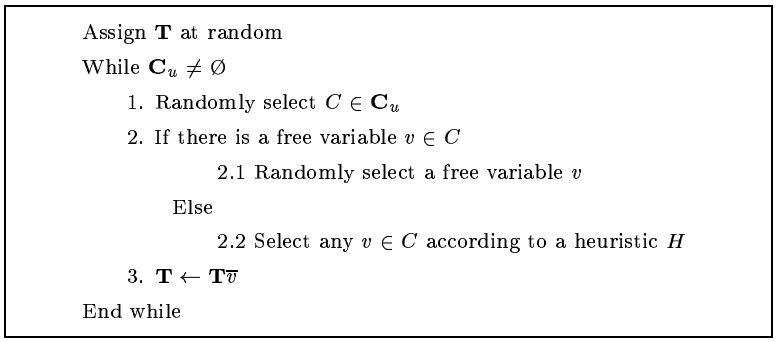

Figure 1: The $\mathcal{WS}$ family of stochastic local search algorithms for SAT.

uniform distribution), the selection in step 2.2 is according to a heuristic H such that there is a fixed lower bound > 0 for the probability of selecting any given element of C.

It is easy to see that this algorithm family contains WalkSAT/SKC with maxTries$= 1$ (i.e., random restart is not used), maxSteps$= \infty$ (i.e., the algorithm is run until a solution is found), and arbitrary settings of the noise parameter.[1] To prove the PAC property of WalkSAT/SKC for 2-SAT, we use the following lemma:

Lemma 1 *Any member of algorithm family $\mathcal{WS}$ is PAC if for every satisfiable problem instance and for every initial assignment* **T** *there exists a sequence of selections which leads to a solution.*

Proof: If such a sequence s exists, there is a lower bound > 0 of the probability that the algorithm will make the selections according to s. Thus, over all truth assignments, there is a minimal probability > 0 to find a solution within a finite number of variable flips. As the number of variable flips approaches infinity, the probability of not executing any of these sequences for any of the assignments along the search trajectory converges to zero, i.e., the algorithm is PAC. $\mathcal{QED}$

Notice that the only restriction on the algorithm's choices occurs when it selects an unsatisfied clause with a free variable. Without this restriction, it is

[1] It is easy to see that when random restart and a fixed cutoff parameter maxSteps is used, any WalkSAT algorithm is PAC. As explained in [1], this result is of very limited interest, since thus parameterized, typically the algorithms show poor performance in practice.

easy to see that the algorithm is PAC, since it can always set a variable to its final value until all clauses are satisfied.[2]

For a given satisfiable instance $\mathcal{S}$, we define a *trap* as an assignment $\mathbf{T}$ for which there is no selection sequence leading to a solution. To prove the precondition of Lemma 1 (which implies the PAC property of WalkSAT/SKC), we show that for satisfiable 2-SAT instances, traps cannot exist.

To make the presentation more succinct we restrict ourselves to satisfiable instances with a model T_m such that $\forall v \in V : \mathbf{T}_m(v) = \mathit{true}$. Note that for any instance containing a trap we can obtain an equivalent instance with this property by complementing the appropriate literals throughout. In the following, we assume without loss of generality that an arbitrary 2-SAT instance with this property is given. For any assignment $\mathbf{T}$, we define the Hamming distance $h = h(\mathbf{T}, \mathbf{T}_m)$ to be the number of variables whose assignments differ under $\mathbf{T}$ and $\mathbf{T}_m$; thus, h is just the number of incorrect variables.

Furthermore, we use the following notation for the literals evaluated under the current truth assignment $\mathbf{T}$:

v_x^u	*incorrect & unsatisfied*
v_c^s	*correct & satisfied*
$\widehat{v}_c^u$	*correct & unsatisfied*
$\widehat{v}_x^s$	*incorrect & satisfied*

where the superscript indicates whether the literal is satisfied (s) or unsatisfied (u) by $\mathbf{T}$ and the subscript c indicates that the variable has the correct value assigned in the model $\mathbf{T}_m$ while subscript x means it does not. Because we assume that the correct value for all variables is *true*, there is redundancy in the notation: for example the form $\widehat{v}_x^u$ is impossible because a negated and unsatisfied literal with is necessarily correct.

Lemma 2 *For all $C \in \mathbf{C}$, for all $\mathbf{T}$, there is some variable v such that either $v_c^s \in C$ or $v_x^u \in C$. In particular, the latter case must hold for all $C \in \mathbf{C}_u$.*

Proof: $\mathbf{T}_m$ must make every clause true. QED

[2]This argument is similar to the one used in [4] to show that satisfiable 2-SAT instances are solved by pure random walk in $O(n^2)$ time on average.

We notice that if a trap exists, then every $\mathbf{T}$ reachable from it is also a trap. Furthermore, it a trap exists then there must be some minimum value h' of h, where $h' > 0$, among all traps. We designate one such minimal trap state by $\mathbf{T}_\alpha$.

Lemma 3 *For each $C \in \mathbf{C}_u$ at $\mathbf{T}_\alpha$, there exists one $\widehat{v}_c^u \in C$ such that v is free and the other $w_x^u \in C$ with w not free.*

Proof: Since flipping the assignment of a variable changes h and we start with h' being the minimum over all states, every flip we are allowed to make must increase h. Each such flip must involve a correct variable, and since $C \in \mathbf{C}_u$, the corresponding literal must be unsatisfied. By Lemma 2 the other literal must be w_x^u. If w were free, or v was not free, then we could choose to flip w reducing $h < h'$, a contradiction to the minimality of h'. QED

Notice that by definition, flipping a free variable always decreases the number of unsatisfied clauses. Thus, Lemma 3 says that in the hypothesized minimal trap state $\mathbf{T}_\alpha$, we must flip a free variable, thus increasing h but decreasing the number of unsatisfied clauses. The next lemma proves that after at most one such flip, a move becomes available which decreases h and thus leads to another minimal state.

Lemma 4 *In a minimal trap state $\mathbf{T}_\alpha$, there exist $C_1, C_2 \in \mathbf{C}_u$ with free variables $\widehat{a}_c^u \in C_1$ and $\widehat{b}_c^u \in C_2$, $C_1, C_2 \in \mathbf{C}_u$, such that flipping a makes b not free.*

Proof: Let $F = \{v \in V | \exists C \in \mathbf{C}_u, \widehat{v}_c^u \in C\}$, the set of free variables at $\mathbf{T}_\alpha$. If we were to flip all $v \in F$ and they all remained continually free, then we would satisfy all clauses that were in $\mathbf{C}_u$ at $\mathbf{T}_\alpha$, yielding a solution. Thus, since $\mathbf{T}_\alpha$ is a trap state, each sequence of flips of variables in F must at some point create another unsatisfied clause. By definition, this is only possible if at some point one of the free variables becomes not free. Let state $\mathbf{T}$ be the first assignment reached in this sequence of flips such that flipping a free variable $a \in F$ makes another variable $b \in F$ change from free to not free. Notice that neither a nor b can possibly have been flipped earlier in the sequence, because both were free and after flipping a free variable it does not appear in

any unsatisfied clauses; furthermore, flipping free variables never creates new unsatisfied clauses. Therefore, both a and b are also free in $\mathbf{T}_\alpha$ and consequently. Note that there must be a clause $\langle a_c^s \vee b_c^s \rangle$ in $\mathbf{T}_\alpha$ since we are in 2-SAT and flipping a makes b not free. Since our algorithm can flip any free variable at a given state, it can flip a directly in $\mathbf{T}_\alpha$, making b not free. Furthermore, $a, b \in F$ implies that at $\mathbf{T}_\alpha$, there exist clauses $C_1, C_2 \in \mathbf{C}_u$ such that $\widehat{a}_c^u \in C_1$ and $\widehat{b}_c^u \in C_2$. QED

We are now in a position to complete the proof of:

Theorem 1 *Any member of the $\mathcal{WS}$ algorithm family for which Assumption 1 holds, is PAC for arbitrary 2-SAT instances.*

Proof: We assume that a trap, and therefore a minimal trap state $\mathbf{T}_\alpha$ exists. Using Lemma 4, at this state there are clauses $C_1, C_2 \in \mathbf{C}_u$ with free variables $\widehat{a}_c^u \in C_1$ and $\widehat{b}_c^u \in C_2$, $C_1, C_2 \in \mathbf{C}_u$, such that choosing to flip a makes b not free. According to Lemma 3, there are non-free variables d, e such that $C_1 = \langle \widehat{a}_c^u \vee d_x^u \rangle$ and $C_2 = \left\langle \widehat{b}_c^u \vee e_x^u \right\rangle$. Note that a and e cannot be identical since they have complementary settings at $\mathbf{T}_\alpha$, the same holds for b and d. Now our algorithm can first select C_1 and flip a making b not free. (Note that according to Assumption 1, any unsatisfied clause can be chosen without loss of generality.) Then, C_2 is selected. Since b is not free any longer, e can be flipped, leading into a new minimal trap state $\mathbf{T}'$ with $h = h'$.

Since e was not free at $\mathbf{T}_\alpha$ there must be a clause C_3 in which variable e appears. We must have $g_x^u \in C_3$ by Lemma 2 and the fact that only the literal on e made the clause *true* (else e would be free). Furthermore, the status of e in C_3 at $\mathbf{T}_\alpha$ must be $\widehat{e}_x^s$, since flipping e decreases h. Note that g must be different from a, since their truth values differ at $\mathbf{T}_\alpha$. After flipping e at $\mathbf{T}_\alpha$, the status of C_3 at $\mathbf{T}'$ is $\langle \widehat{e}_c^u \vee g_x^u \rangle \in \mathbf{C}_u$ with e not free. (The variable most recently flipped cannot be free since flipping is only possible if it occurred in a false clause.) But this means, selecting C_3, g can now be flipped to its correct value, getting a new state with $h = h' - 1$, a contradiction to the assumed minimality of h'. Thus, minimal trap states,and hence, traps cannot exist under the assumptions stated in the theorem. QED

Since WalkSAT/SKC without restart and infinite cutoff parameter satisfies the precondition of the theorem, its PAC property for 2-SAT is a simple

corollary. Unfortunately, this proof fails for 3-SAT, essentially because at a hypothetical minimal trap state $\mathbf{T}_\alpha$, we can have three unsatisfied clauses with free variables, and one other clause containing all three free variables complemented. Thus, Lemma 4 does not hold, and it seems that the algorithm can be forced to make at least two upward steps in h before another downward step becomes possible. This considerably complicates any attempt to prove the existence of a sequence of flips which can escape from such a trap by reducing h beneath $h' = h(\mathbf{T}_\alpha, \mathbf{T}_m)$. At the same time it appears to be very hard to construct a trap for the 3-SAT case, since it seems to be very difficult to restrict the structure of the trap enough that it becomes feasible to show that all possible escape routes lead into another trap state in which all clauses contain correctly assigned, free variables. Faced with these difficulties, so far we have not succeeded in proving or disproving the PAC property for this cases.

Empirical Evidence. For other algorithms of the WalkSAT family, in particular the more recent Novelty and R-Novelty algorithms [5], instances containing traps could be relatively easily constructed, after the essential incompleteness of these algorithms became apparent when analysing their empirical performance applied to various sets of benchmark problems [1]. Although in general, experimental results can neither prove nor disprove the PAC property of an algorithm, they can give highly suggestive evidence. Such evidence is collected by applying the algorithm in question to a large set of satisfiable problem instances and checking whether for a large number of runs on each instance, solutions can be found in each individual run, when the restart mechanism is disabled. A huge amount of such experiments were performed on the problem instances available through SATLIB,[3] a public resource for SAT benchmark instances and solvers. We evaluated WalkSAT/SKC on a total of 3,700 hard Random-3-SAT instances and several thousand SAT-encoded problem instances from other domains, including graph coloring, planning, and circuit fault analysis. In every instance, WalkSAT/SKC was able to find a model in each of at least 100 runs (see also [2]). Other WalkSAT variants, which are provably essentially incomplete, however, failed to find models in some of the runs, especially for SAT-encoded problems

[3]available on the WWW at www.informatik.tu-darmstadt.de/AI/SATLIB

from other domains. As argued in [1], this indicates that the PAC property is not only of theoretical interest, but has also (at least in cases where it does not hold) significant impact on the performance of state-of-the-art stochastic SLS algorithms for SAT. Our empirical evidence summarized above suggests that if WalkSAT/SKC is essentially incomplete, the practical impact of this property is much smaller than for other high-performing WalkSAT variants which are essentially incomplete.

A Generalized Family of Algorithms. We can view $\mathcal{WS}$ as an example of a broader family of algorithms. Within this class, we show that $\mathcal{WS}$ is the only one in which the proof of PAC property or essential incompleteness is difficult. To do so, we vary the definition of a free variable such that it covers variables which, when flipped, cause at most δ currently satisfied clauses to become unsatisfied. Based on this new definition, we obtain a generalized algorithm family $\mathcal{WS}'$ from the family $\mathcal{WS}$ considered before. Only for $\delta = 0$, i.e. the existing class $\mathcal{WS}$, is there any difficulty.

For $\delta = -1$, $\mathcal{WS}'$ algorithms have a free choice of the variable to flip in the selected clause at each step, since free variables do not exist. As argued before, this allows in each step a variable to be set to its correct value until all clauses are satisfied. Consequently, these algorithms are all PAC. This includes as a special case the well-known GWSAT algorithm [6]. The $\mathcal{WS}'$ algorithms for $\delta = 0$ form the family $\mathcal{WS}$ considered before. We have shown that these algorithms are PAC for the 2-SAT case, but for more general cases, it is not known whether the PAC property holds. For $\delta = 1$, the following simple example shows that the corresponding $\mathcal{WS}'$ algorithms are essentially incomplete for 2-SAT:

$$C = \{\langle a \vee c_x^u \rangle, \langle \widehat{a} \vee c_x^u \rangle, \langle \widehat{c}_x^s \vee d_x^u \rangle, \langle \widehat{c}_x^s \vee e_x^u \rangle\}.$$

Here, c is incorrectly set to *false* and not free. But since a is free, and remains free at $\delta = 1$ when flipped, c can never be changed to its correct truth value. This example can be easily generalized to longer clauses and for $\delta > 1$, which shows that $\mathcal{WS}'$ algorithms for $\delta \geq 1$ are essentially incomplete in general.

Conclusions. We have shown that WalkSAT/SKC, one of the most prominent and successful SLS algorithms for SAT, is probabilistically approximately

complete for 2-SAT. That is, applied to any satisfiable 2-SAT instance, it will find a model almost certainly when given enough time. This result complements a series of related results for other prominent SLS algorithms for SAT [1] and might be regarded as a first step in proving (or refuting) the PAC property of WalkSAT/SKC in the NP-complete cases of 3-SAT and general clausal SAT. There is substantial empirical evidence that the PAC property can have a significant impact on the performance of SLS algorithms for SAT, such as WalkSAT/SKC and related algorithms, in practice.

References

[1] H.H. Hoos. On the run-time behaviour of stochastic local search algorithms for sat. In *Proceedings of AAAI-99*, 1999.

[2] H.H. Hoos and T. Stützle. Towards a characterisation of the behaviour of stochastic local search algorithms for sat. *Artificial Intelligence*, **112**, pages 213–232, 1999.

[3] H.H. Hoos and T. Stützle. Local search algorithms for SAT: An empirical evaluation. *J. Automated Reasoning*, to appear, 2000.

[4] C. Papadimitriou. On Selecting a Satisfying Truth Assignment. In *Proc. 32nd IEEE Symposium on the Foundations of Computer Science*, pages 163–169, 1991.

[5] D. McAllester, B. Selman, and H. Kautz. Evidence for invariants in local search. In *Proceedings of AAAI-97*, pages 321–326, 1997.

[6] B. Selman, H. Kautz, and B. Cohen. Noise Strategies for Improving Local Search. In *Proceedings of the 12th National Conference on AI*, pages 337–343. American Association for Artificial Intelligence, 1994.