# How to tell a good neighborhood from a bad one: Satisfiability of Boolean Formulas

Tassos Dimitriou[1] and Paul Spirakis[2]

[1] Athens Information Technology
tassos@ait.gr
[2] Computer Technology Institute
spirakis@cti.gr

**Abstract.** One of the major problems algorithm designers usually face is to know in advance whether a proposed optimization algorithm is to behave as planned, and if not, what changes are to be made to the way new solutions are examined so that the algorithm performs nicely. In this work we develop a methodology for differentiating good neighborhoods from bad ones. As a case study we consider the structure of the space of assignments for random 3-SAT formulas and we compare two neighborhoods, a simple and a more refined one that we already know the corresponding algorithm behaves extremely well.
We give evidence that it is possible to tell in advance what neighborhood structure will give rise to a good search algorithm and we show how our methodology could have been used to discover some recent results on the structure of the SAT space of solutions. We use as a tool "Go with the winners", an optimization heuristic that uses many particles that independently search the space of all possible solutions. By gathering statistics, we compare the combinatorial characteristics of the different neighborhoods and we show that there are certain features that make a neighborhood better than another, thus giving rise to good search algorithms.

## 1 Introduction

Satisfiability (SAT) is the problem of determining, given a Boolean formula $f$ in conjunctive normal form, whether there exists a truth assignment to the variables that makes the formula true. If all clauses consist of exactly $k$ literals then the formula is said to be an instance of $k$-SAT. While 2-SAT is solvable in polynomial time, $k$-SAT, $k \geq 3$ is known to be $NP$-complete, so we cannot expect to have good performance in the worst case. In practice, however, one is willing to trade "completeness" for "soundness". Incomplete algorithms may fail to find a satisfying assignment even if one exists but they usually perform very well in practice. A well studied algorithm of this sort is the WalkSat heuristic[SKC93]. The distinguishing characteristic of this algorithm is the way new assignments are examined. The algorithm chooses to flip only variables that appear in *unsatisfied* clauses as opposed to flipping any variable and testing the

new assignment. Moreover, flips are made even if occasionally they increase the number of unsatisfied clauses.

In this work we make an attempt to differentiate "good" neighborhoods from "bad" ones by discovering the combinatorial characteristics a neighborhood must have in order to produce a good algorithm. We study the WALKSAT neighborhood mentioned above and a simpler one that we call GREEDY in which neighboring assignments differ by flipping any variable. Our work was inspired by a challenge reported by D. S. Johnson [J00] which we quote below:

**Understanding Metaheuristics**
"... Currently the only way to tell whether an algorithm of this sort [optimization heuristic] will be effective for a given problem is to implement and run it. We currently do not have an adequate theoretical understanding of the design of search neighborhoods, rules for selecting starting solutions, and the effectiveness of various search strategies. Initial work has given insight in narrow special cases, but no useful general theory has yet been developed. We need one."

Here, we compare the two neighborhoods by examining how the space of neighboring assignments, the search graph as we call it, decomposes into smaller regions of related solutions by imposing a quality threshold to them. Our main tool is the "Go with the winners" (GWW) strategy which uses many particles that independently search the search graph for a solution of large value. Dimitriou and Impagliazzo[DI98] were able to relate the performance of GWW with the existence of a combinatorial property of the search graph, the so called "local expansion". Intuitively, if local expansion holds then particles remain uniformly distributed and *sampling* can be used to deduce properties of the search space.

Our goal is to provide algorithm designers with a method of testing whether a proposed neighborhood structure will give rise to a good search algorithm. We do this following a two-step approach. Initially, we verify experimentally that the search graph has good local expansion. Then by gathering statistics, we compare the *combinatorial characteristics* of the neighborhood and we show that there are certain features that make a neighborhood better than another, thus giving rise to good search algorithms. Our results are in some sense complementary to the work of Schuurmans and Southey[SS01] where emphasis is given on the characteristics of successful search algorithms. Here, instead, the emphasis is on the properties of the search space itself.

## 2 Go with the winners Strategy (GWW)

Most optimization heuristics (Simulated Annealing, Genetic algorithms, Greedy, WalkSat, etc.) can be viewed as *strategies*, possibly probabilistic, of moving a particle in a search graph, where the goal is to find a solution of optimal value. The placement of the particle in a node of the search graph corresponds to examining a potential solution.

An immediate generalization of this idea is to use many particles to explore the space of solutions. This generalization together with the extra feature of

*interaction* between the particles is essentially "Go with the Winners"[DI96]. The algorithm uses many particles that independently search the space of all solutions. The particles however interact with each other in the following way: each "dead" particle (a particle that ended its search at a local optimum) is moved to the position of a randomly chosen particle that is still "alive". The goal of course is to find a solution of optimal value. Progress is made by imposing a quality threshold to the solutions found, thus improving at each stage the average number of satisfied clauses.

In Figure 1, a "generic" version of GWW is presented without any reference to the underlying neighborhood of solutions (see Section 3). Three conventions are made throughout this work: i) creating a particle means placing the particle in a (random or predefined) node of the search graph, ii) the value of a particle is the value of the solution it represents and iii) more than one particle may be in the same node of the search graph, thus examining the same solution.

**Generic GWW**

Let $\mathcal{N}_i$ be the *subset* of all solutions during stage $i$ of the algorithm, that is assignments satisfying *at least* $i$ clauses.

- Stage 0 (*Initialization*): Generate $B$ random solutions and place one particle in each one of them.
- Stage $i$ (*Main loop*): Proceed in two phases.
  1. In the *randomization* phase, for each particle, perform an $S$ steps *random walk* in $\mathcal{N}_i$, where a step is defined as follows: Select a neighbor of the current solution. If its value is $\geq i$, move the particle to this neighbor, otherwise leave the particle in its current position.
  2. In the *redistribution* phase, if all particles have value $i$, stop. Otherwise, replace each particle of value $i$ with a copy of a randomly chosen particle whose value is $> i$.
  3. Go to the next stage, by raising the threshold $i$.
- Output the best solution found.

**Figure 1.** Description of the algorithm

The algorithm uses two parameters: the number of particles $B$ and the length $S$ of each particle's random walk. The intuition is that in the $i$-th stage and beyond we are eliminating from consideration assignments which satisfy less than $i$ clauses. This divides the search space into *components* of assignments. The redistribution phase will make sure that particles in locally optimal solutions will be distributed among non-local solutions, while the randomization phase will ensure that particles remain *uniformly distributed* inside these emerging components. This is shown abstractly in Figure 2 for a part of the search space and two thresholds $T_A$, $T_B$. When the algorithm is in stage $T_A$, all solutions of

value larger than $T_A$ form a connected component and particles are uniformly distributed inside this large component. However, when the threshold is increased to $T_B$, the search graph decomposes into smaller components and the goal is to have particles in all of them in *proportion* to their sizes.

Dimitriou and Impagliazzo [DI98] characterized the search spaces where GWW works well in terms of a combinatorial parameter, the *local expansion* of the search space. Intuitively this property suggests that if a particle starts a random walk, then the resulting solution will be uncorrelated to its start and it is unlikely that particles will be trapped into small regions of the search space.

Studying the behavior of the algorithm with respect to its two important parameters will offer a tradeoff between using larger populations or longer walks. In particular, our goal will be to show that for certain neighborhoods only *a few* particles suffice to search the space of solutions provided the space has good expansion properties and the length of the random walk is sufficiently long. When this is true, having a few particles is statistically the same as having one, therefore *this particle together with the underlying neighborhood can be thought of as defining an optimization heuristic.*

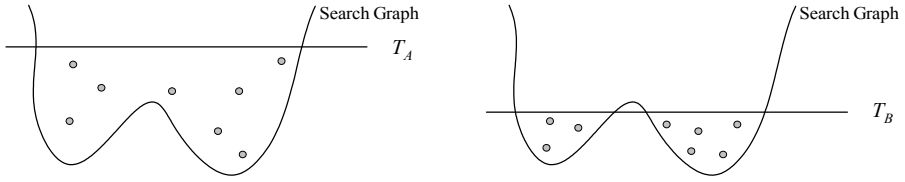



**Figure 2.** Decomposition of the search graph.

## 3 Distribution of SAT instances and Search Graphs

It is known that random 3-SAT formulas exhibit a *threshold* behavior. What this means is that there is a constant $r$ so that formulas with more than $rn$ clauses are likely to be unsatisfiable, while formulas with less than $rn$ clauses are likely to be satisfiable. In all experiments we generated formulas with ratio of clauses to variables $r = 4.26$, since as was found experimentally[MSL92,GW94] the hardest to solve instances occur at this ratio.

We proceed now to define the neighborhood structure of solutions. GWW is an optimization heuristic that tries to locate optimal solutions in a *search graph*, whose nodes represent all feasible solutions for the given problem. Two nodes in the search graph are connected by an edge, if one solution results from the other by making a local change. The set of neighbors of a given node defines the *neighborhood* of solutions. Such a search graph is implicitly defined by the problem at hand and doesn't have to be computed explicitly. The only operations required by the algorithm are i) generating a random node (solution)

in the search graph, ii) compute the value of a given node, and iii) list efficiently all the neighboring solutions. The two search graphs we consider are:

**GREEDY**
Nodes correspond to assignments. Two such nodes are connected by an edge if one results from the other by flipping the truth value of a single variable. Thus any node in this search graph has exactly $n$ neighbors, where $n$ is the number of variables in the formula. The value of a node is simply the number of clauses satisfied by the corresponding assignment.

**WALKSAT**
Same as above except that two such nodes are considered neighboring if one results from the other by flipping the truth value of a variable that belongs to an *unsatisfied* clause.

Our motivation of studying these search graphs is to explain why the WALKSAT heuristic performs so well in practice as opposed to the simple GREEDY heuristic (which essentially corresponds to GSAT). Does this second search graph has some interesting combinatorial properties that explain the apparent success of this heuristic? We will try to answer some of these questions in the sections that follow.

## 4 Implementation Details

We used a highly optimized version of GWW in which we avoided the explicit enumeration of neighboring solutions in order to perform the random walks for each particle.

In particular we followed a "randomized approach" to picking the right neighbor. To perform one step of the random walk, instead of just enumerating all possible neighbors we simply pick one of the $n$ potential neighbors at random and check its value. If it is larger than the current threshold we place the particle there, otherwise we repeat the experiment. How many times do we have to do this in order to perform one step of the random walk? It can be shown (see for example Figure 4 illustrating the number of neighbors at each threshold) that in the earlier stages of the algorithm we only need a few tries. At later stages, when neighbors become scarce, we simply have to remember which flips have we tried and in any case never perform more than $n$ flips.

The savings are great because as can be seen in the same figure it is only in the last stages that the number of neighbors drops below $n$, thus requiring more than one try on average. In all other stages each random step takes constant time!

## 5 Experiments

The two important parameters of GWW is population size $B$ and random walk length $S$. Large, interacting populations allow the algorithm to reach deeper levels of the search space while long random walks allow GWW to escape from

local optima that are encountered when one is using only greedy moves. The use of these two ingredients has the effect of maintaining a *uniform distribution* of particles throughout the part of the search graph being explored.

The goal of the experiments is to understand the relative importance of these parameters and the structure of the search graph. In the first type of experiments (Section 5.1) we will try to reveal the expansion characteristics of the search graph and validate the implementation choices of the algorithm. If the search graph has good expansion, this can be shown by a series of tests that indicate that sufficiently long random walks uniformly distribute the particles.

The purpose of the second type of experiments (Sections 5.2 and 5.3) is to study the quality of the solutions found as a function of $B$ and $S$. In particular, we would like to know what is more beneficial to the algorithm: *to invest to more particles or longer random walks?* Any variation in the quality of solutions returned by the algorithm as a function of these two parameters will illustrate different characteristics of the two neighborhoods. Then, hopefully, all these observations can be used to tell what makes a neighborhood better than another and how to design heuristics that take advantage of the structure of the search space.

In the following experiments we tested formulas with 200 variables and 857 clauses and each sample point on the figures was based on averaging over 500 such random formulas. To support our findings we repeated the experiments with formulas with many more variables and again the same patterns of behavior were observed (details omitted here).

### 5.1 Optimal Random Walk Length

Before we proceed with the core of the experiments we need to know whether our "randomized" version of GWW returns valid statistics. This can be tested with a series of experiments that compare actual data collected by the algorithm with data expected to be true for random formulas. For these comparisons to be valid we need to be sure that particles remain well distributed in the space of solutions. If particles are biased towards certain regions of the space this will be reflected on the statistics and any conclusions we draw may not be true. So, how can we measure the right walk length so that particles remain uniformly distributed?

One nice such test is to compute for each particle the Hamming distance of the assignments *before* and *after* the random walk, and compare with the distance expected in the random formula model. If the length $S$ of the walk is sufficiently large, these quantities should match.

In Figure 3 we show the results of this experiment for a medium population size of $B = 100$ particles and various walk lengths ($S = 100, 200, 300, 500, 1000$). For each stage (shown are stages where the number of satisfied formulas is $\geq 600$) we computed the average Hamming distance between solutions (assignments) at start and end of the random walk, and we plotted the average over all particles. As we can see the required length to achieve uniformity in the GREEDY neighborhood is about 1000 steps. For this number of steps the

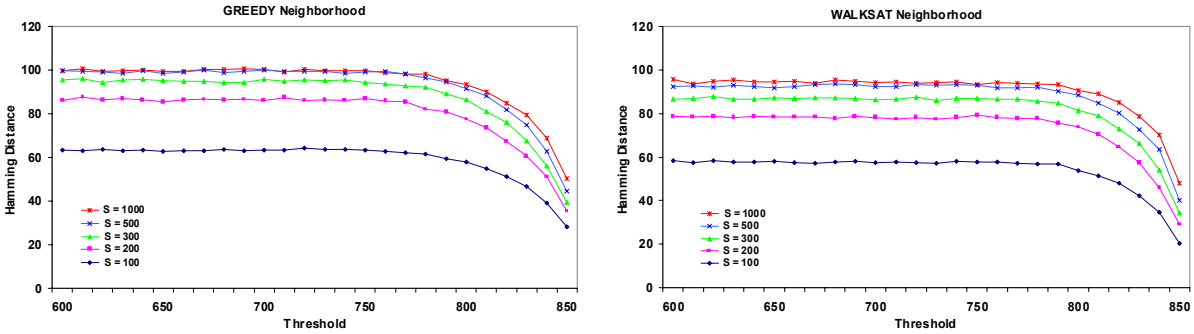



**Figure 3.** Choosing the right walk length that achieves uniformity for GREEDY (*left*) and WALKSAT (*right*) neighborhoods: Average Hamming distance between start and end of the random walk for $S = 100, 200, 300, 500, 1000$.

average Hamming distance matches the one expected when working with random formulas, which is exactly $n/2$. It is only in the last stages of the algorithm that the Hamming distance begins to drop below the $n/2$ value as not all flips give rise to good neighbors (compare with Figure 4). The same is true for the WALKSAT neighborhood. The required length is again about 1000 steps. Here however, the Hamming distance is slightly smaller as flips are confined only to variables found in unsatisfied clauses. Thus we conclude that 1000 steps seems to be a sufficient walk length so that particles remain uniformly distributed in the space of solutions.

A second test we performed to compute the appropriate walk length was to calculate the average number of neighbors of each particle at each threshold and compare it with the expected values in the random formulas model. The results are shown in Figure 4(left) for a population size $B = 100$ and the optimal random walk length ($S = 1000$). For the GREEDY neighborhood, at least in the early stages of the algorithm, all the flips should lead to valid neighbors, so their number should be equal to $n$, the number of variables. For the WALKSAT neighborhood the number of neighbors should be smaller than $n$ as flips are confined to variables in unsatisfied clauses. As it can be seen, the collected averages match with the values expected to be true for random formulas, proving the validity of the randomized implementation and the uniform distribution of particles.

Finally in Figure 4(right), we performed yet another test to examine the hypothesis that 1000 steps are sufficient to uniformly distribute the particles. In this test we examined the average number of satisfied clauses of the GWW population at each threshold and compared them with those expected for random formulas. Again the two curves matched showing that particles remain well distributed. In this particular plot it is instructive to observe a qualitative difference between the two neighborhoods. In the GREEDY one, the algorithm starts with 1/8 of the clauses unsatisfied and it is only in the last stages that this number begins to drop. In the WALKSAT neighborhood however, this number is much

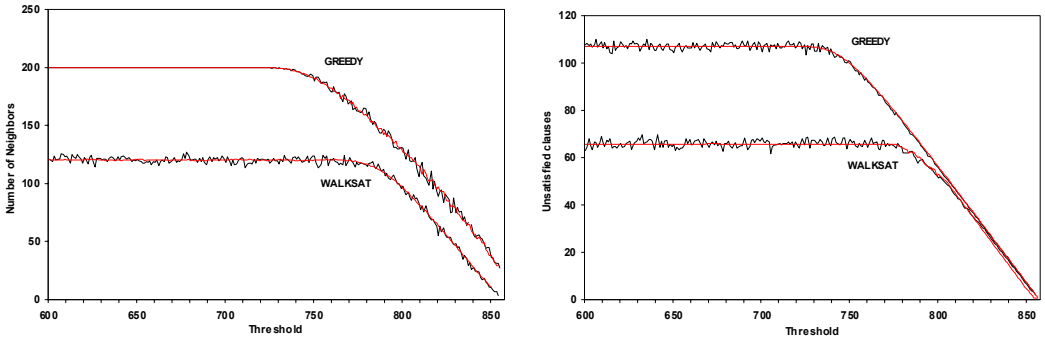



**Figure 4.** Comparing the number of neighbors (*left*) and unsatisfied clauses (*right*) at various thresholds with those *expected* for random formulas, for both neighborhoods. The collected data *matched* the expectations.

smaller. It seems that the algorithm already has an advantage over the GREEDY implementation as it has to satisfy fewer unsatisfied clauses (approximately 5/64 of the clauses against 8/64 of the GREEDY neighborhood). Does this also mean that the algorithm will need fewer "resources" to explore the space of solutions? We will try to answer this next.

### 5.2 Characterizing the two Neighborhoods

In this part we would like to study the quality of the solutions found by GWW as a function of its two mechanisms, population size and random walk length. Studying the question of whether to invest to more particles or longer random walks will illustrate different characteristics of the two neighborhoods. In particular, correlation of the quality of the solutions with population size will provide information about the *connectedness* of the search graphs, while correlation with random walk length will tell us more about the *expansion characteristics* of these search spaces.

We first studied (not shown due to space restrictions) the effect of varying the population size $B$ while keeping the number of random steps $S$ constant, for various walk lengths ($S = 4, 8, 16, 32$). The number of satisfied clauses increased as a function of $B$ and the curves looked asymptotic, like those of Figure 5. In general, the plots corresponding to the two neighborhoods were similar with the exception that results were slightly better for the WALKSAT neighborhood.

In Figure 5 we studied the effect of varying the number of steps while keeping the population size constant, for various values of $B$ ($B = 2, 4, 8, 16, 32$). As it can be seen, increasing the population size resulted in better solutions found. But there is a distinctive difference between the two neighborhoods. While in the first increasing the particles had a clear impact on the quality of solutions found, in the second, having 8 particles was essentially the same as having 32 particles. Furthermore, 2 particles searching the WALKSAT neighborhood ob-

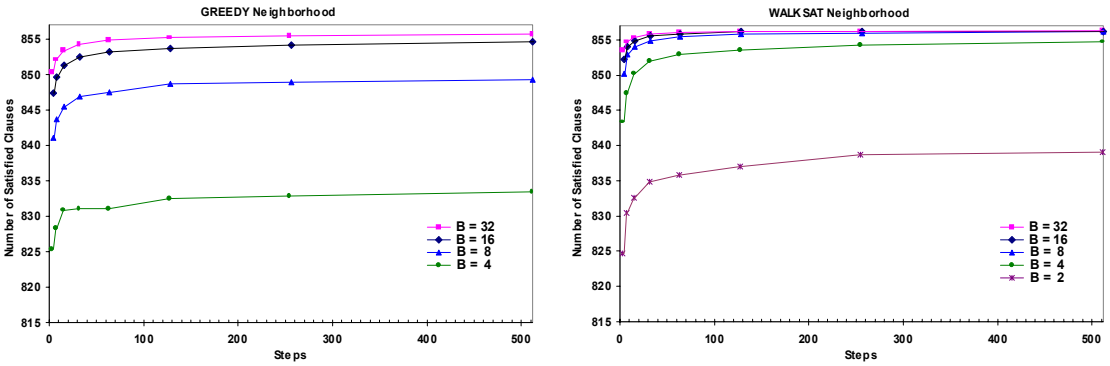



**Figure 5.** Average number of clauses satisfied vs random walk length for various population sizes. Results shown are for GREEDY (*left*) and WALKSAT neighborhoods (*right*).

tained far better results than 4 particles searching the GREEDY one, and in general the GREEDY neighborhood required twice as many particles to achieve results comparable to those obtained in the WALKSAT one. Thus we note immediately that the second neighborhood is *more suited for search with fewer particles*, reaching the same quality on the solutions found provided the walks are kept sufficiently long. This agrees with results found by Hoos[Ho99] where a convincing explanation is given for this phenomenon: Greedy (i.e. GSAT) is essentially incomplete, so numerous restarts are necessary to remedy this situation. Since a restart is like introducing a new search particle, although without the benefit of interaction between particles as in GWW, we see that our findings come to verify this result.

It is evident from these experiments that increasing the computational resources (particles or walk steps) improves the quality of the solutions. We also have an indication that the number of particles is not so important in the WALKSAT neighborhood. So, to make this difference even more striking, we proceeded to examine the effect on the solutions found when the product $B \times S$, and hence the running time, was kept *constant*[3]. The results are shown in Figure 6.

It is clear that as the number of particles increases the average value also increases, but there is a point where this behavior stops and the value of solutions starts to decline. This is well understood. As the population becomes large, the number of random steps decreases (under the constant time constraint) and this has the effect of not allowing the particles to escape from bad regions.

Perhaps what is more instructive to observe is the point in the two neighborhoods when this happens. In the GREEDY neighborhood this behavior is more balanced between particles and length of random walk, as one resource does not

[3] The running time of GWW is $O(BSm)$, where $m$ is the maximum number of stages (formula clauses)

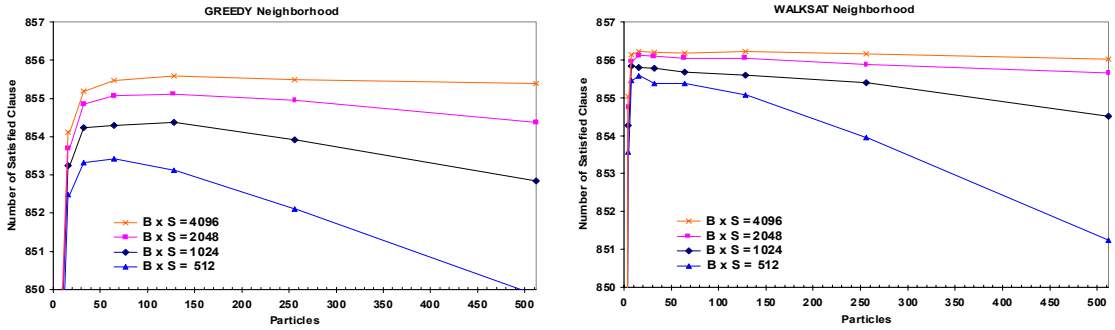



**Figure 6.** Average solution value vs number of particles for different values of $B \times S$. Results shown are for GREEDY (*left*) and WALKSAT neighborhoods (*right*).

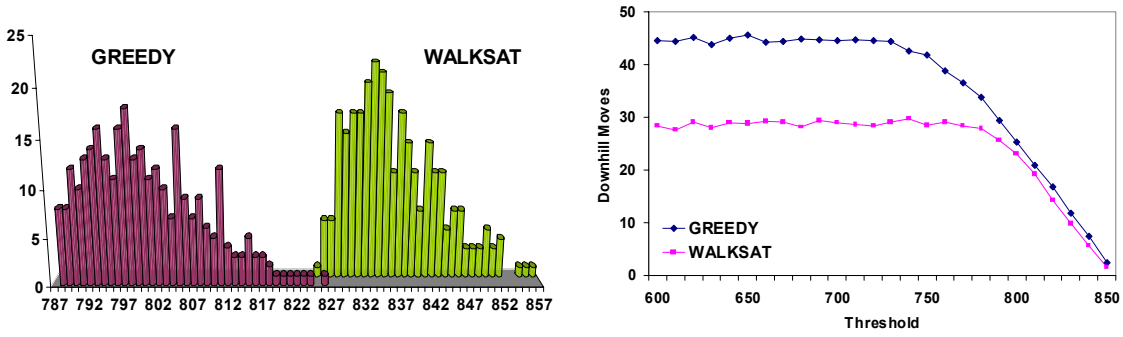
seem more important than the other. In the WALKSAT neighborhood however, things change drastically as the quality of the solutions found degrades when fewer steps and more particles are used! Thus *having longer random walks is more beneficial to the algorithm than having larger populations*. This comes to validate the observation we made in the beginning of this section that the second neighborhood is more suited for search with fewer particles. When taken to the extreme this explains the apparent success of WALKSAT type of algorithms in searching for satisfying assignments as they can be viewed as strategies for moving just one particle around the WALKSAT neighborhood.

To illustrate the difference in the two neighborhoods we performed 300 runs of GWW using $B = 2$ and $S = 1000$ and we presented the results in Figure 7(a) as a histogram of solutions. The histograms of the two neighborhoods were displayed on the same axis as the overlap of values was very small. There was such strong separation between the two neighborhoods that even when we normalized for running time the overlap was still very small. Indeed the best assignment found in the GREEDY neighborhood satisfied only 827 clauses, which is essentially the worst case for the WALKSAT neighborhood. One can thus conclude that the second implementation is intrinsically more powerful and more suited for local optimization than the first one, even when running time is taken into account.

### 5.3 Further Properties of the Search Graphs

The results of the previous section suggest that the WALKSAT neighborhood is more suited for local search as only a few particles can locate the optimal solutions. So, we may ask what is the reason behind this discrepancy between the two neighborhoods?

We believe the answer must lie in the structure of the search graphs. As the GREEDY space decomposes into components by imposing the quality threshold




(a) (b)

**Figure 7.** (a) Histogram of satisfied clauses for 300 runs of GWW with $B = 2$ and $S = 1000$. Results shown are for GREEDY and WALKSAT neighborhoods. (b) Results of performing simple greedy optimization starting from particles at various thresholds. The plots show the average number of downhill moves required before getting trapped into a local optimum in both neighborhoods.

to solutions, good solutions must reside in pits with high barriers that render long random walks useless. So the algorithm requires more particles as smaller populations usually get trapped. In the case of the WALKSAT neighborhood, the search graph must be " smooth" in the sense that good solutions must not be hidden in such deep pits. Thus the algorithm needs only a few particles to hit these regions.

We tried to test this hypothesis with the following experiment: Once the particles were uniformly distributed after the randomization phase, we performed simple *greedy optimization* starting from that particle's position and recorded the number of downhill moves (we use this term even if this is a maximization problem) required before the particle gets trapped into a local optimum. Then we averaged over all particles and proceeded with the next stage. So, essentially we counted the average (downhill) distance a particle has to travel before gets trapped. Since by assumption the particles are uniformly distributed, this reveals the depth of the pits.

As it can be seen in Figure 7(b), particles in the WALKSAT implementation had to overcome smaller barriers than those in the GREEDY one. (One may object by saying that the average improvement per move may be larger in the WALKSAT neighborhood. But this is not the case as we found that improvement is $\approx 2$ clauses/move in the GREEDY space against $\approx 1.6$ clauses/move in the WALKSAT one). Moreover, the same patterns were observed when we repeated the experiment with formulas having 500 and 1000 variables. This is a remarkable result. It simply says that the WALKSAT neighborhood is in general smoother than the GREEDY one and easier to be searched with only a few particles, which perhaps also explains why GSAT doesn't meet the performance

of WalkSat. Again our findings come to verify some old results: Schuurmans and Southey[SS01] identify three measures of local search effectiveness, one of which is *depth* corresponding to number of unsatisfied clauses. Similarly, the depth of GSAT is intensively studied by Gent and Walsh[GW93].

Finally we performed another experiment similar to the previous one (not shown here due to space restrictions) in which we counted the average number of neighbors of each *greedily* obtained solution. A number smaller than the number of neighbors found by GWW (Figure 4) would be an indication that solutions lied inside deep pits. The results again supported the "smoothness" conjecture; the number of neighbors of greedily obtained solutions in the WALKSAT space was much closer to the expected curve than those of the GREEDY one.

## 6 Conclusions and Future Research

Many optimizations algorithms can be viewed as strategies for searching a space of potential solutions in order to find a solution of optimal value. The success of these algorithms depends on the way the underlying search graph is implicitly defined, that is on the way a new potential solution is generated by making local changes to the current one. As was mentioned in [J00], currently the only way to tell whether an algorithm of this sort will be effective for a given problem is simply to implement and run it. So, one of the challenges algorithm designers face is to design the right search neighborhood so that the corresponding optimization heuristic behaves as expected.

In this work we considered the problem of differentiating good neighborhoods from bad ones. In particular, we studied two search neighborhoods for the 3-SAT problem, a simple one which we called GREEDY and a more refined one that we called WALKSAT. In the first one, neighboring assignments were generated by flipping the value of *any* of the $n$ variables, while in the second one only variables that belong to *unsatisfied* formulas were flipped. Our motivation for this work was inspired by the challenge mentioned above and the need to explain the apparent success of WALKSAT type of algorithms in finding good satisfying assignments since all of them are based on the WALKSAT neighborhood.

We gave evidence that it is possible to tell in advance what neighborhood structure will give rise to a good search algorithm by comparing the *combinatorial characteristics* of the two neighborhoods. We used as a platform for testing neighborhoods "Go with the winners", an algorithm that uses many particles that independently search the space of solutions. By gathering statistics we showed that there are certain features that make one neighborhood better than another, thus giving rise to a good search algorithm.

In particular, we noticed that the WALKSAT neighborhood was more suited for search with fewer particles, statistically the same as one. We expected this to be true since we knew that the WalkSat heuristic performs extremely well, but we were surprised by the extend to which this was true. We thought that having a more balanced setting between particles and walk lengths would be more beneficial to GWW but this was the case only for the GREEDY neighborhood.

Although we studied only one type of problem (SAT), we believe that search spaces for which good heuristics exist must have similar "characteristics" as the WALKSAT space and can be verified using our approach. Specifically, *to test if a neighborhood will give rise to a good search algorithm run GWW and study the tradeoff between particles and random walk steps. If GWW can discover good solutions with just a few particles and long enough walks, then this space is a candidate for a good search heuristic.* These observations lead to some interesting research directions:

- Provide more evidence for the previous conjecture by trying to analyze the search spaces of other optimization problems for which good algorithms exist. Do the corresponding search graphs have similar properties to the WALKSAT one? This line of research would further validate GWW as a tool for collecting valid statistics.
- What are the properties of search spaces for which *no good* algorithms exist? Are in any sense complementary to those defined here?
- Understand how the WALKSAT space decomposes into components by imposing the quality threshold to solutions. We believe that the WALKSAT space decomposes into only *a few* components (which also explains why one doesn't need many particles) but this remains to be seen.
- Can we find a simple neighborhood that behaves even better than WALKSAT? If this neighborhood has similar characteristics to WALKSAT's (sufficiency of a few particles to locate good solutions, small barriers, etc.) it will probably give rise to even better satisfiability algorithms. (Introducing weights[F97,WW97] smooths out the space but does not meet our definition of a neighborhood.)
- More ambitiously, try to analyze WalkSat and its variants and prove that they work in polynomial time (for certain ratios of clauses to variables, of course). This ambitious plan is supported by the fact that similar findings for graph bisection [DI98,Ca01]) ultimately led to a polynomial time, local search algorithm for finding good bisections[CI01].

## Acknowledgements

The authors wish to thank Christos Papadimitriou for some helpful discussions.

## References

[Ca01] T. Carson. *Empirical and Analytic Approaches to understanding Local Search Heuristics*. PhD thesis, University of California, San Diego, 2001.

[CI01] T. Carson and R. Impagliazzo. Hill climbing finds random planted bisections. In *Proc. 12th Annual ACM Symposium on Discrete Algorithms (SODA)*, 2001.

[DI96] T. Dimitriou and R. Impagliazzo. Towards an analysis of local optimization algorithms. In *Proc. 28th Annual ACM Symposium on Theory of Computing (STOC)*, 1996.

[DI98] T. Dimitriou and R. Impagliazzo. Go with the Winners for Graph Bisection. In *Proc. 9th Annual ACM Symposium on Discrete Algorithms (SODA)*, 510–520, 1998.

[F97] Frank, J. Learning weights for GSAT. In *Proc. IJCAI-97*, 384-391.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, San Francisco, CA, 1979.

[GW93] I. Gent and T. Walsh. An empirical analysis of search in GSAT. *Journal of Artificial Intelligence Research*, 1:23-57, 1993.

[GW94] I. Gent and T. Walsh. The SAT Phase Transition. In *Proc. of ECAI-94*, 105-109, 1994.

[Ho99] Hoos, H. On the run-time behavior of stochastic local search algorithms for SAT. In *Proc. AAAI-99*, 661–666, 1999.

[J00] D. S. Johnson. Report on Challenges for Theoretical Computer Science. Workshop on Challenges for Theoretical Computer Science, Portland, 2000, `http://www.research.att.com/~dsj/`

[MSL92] Mitchell, D., Selman, B., and Levesque, H.J. Hard and easy distributions of SAT problems. In *Proc. AAAI-92*, pp. 459–465, San Jose, CA, 1992.

[SKC93] B. Selman, H. A. Kautz and B. Cohen. Local search strategies for satisfiability testing. In *Second DIMACS Challenge on Cliques, Coloring and Satisfiability*, October 1993.

[SLM92] Selman, B., Levesque, H.J. and Mitchell, D. A new method for solving hard statisfiability problems. In *Proc. AAAI-92*, San Jose, CA, 1992.

[SS01] D. Schuurmans, F. Southey. Local search characteristics of incomplete SAT procedures. *Artif. Intelligence*, 132(2), 121-150, 2001.

[WW97] Wu, Z. and Wah, W. Trap escaping strategies in discrete Langrangian methods for solving hard satisfiability and maximum satisfiability problems. In *Proc. AAAI-99*, 673-678.