Variable-Selection Heuristics in Local Search for SAT

Alex S. Fukunaga*
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, MS 525-3660
Pasadena, CA 91109-8099
alex.fukunaga@jpl .nasa.gov

Abstract

One of the important components of a local search strategy for satisfiability testing is the variable selection heuristic, which determines the next variable to be flipped. In a greedy local search such as GSAT, the major decision in variable selection is the strategy for breaking ties between variables that offer the same improvement in the number of unsatisfied clauses. In this paper, we analyze a number of tie-breaking strategies for GSAT and evaluate the strategies empirically using randomly generated 3-SAT instances from a hard distribution of random instances. We find that the property of fairness, which was proposed in the literature as being the critical property of a successful variable strategy, is not a sufficient property, and show that randomness plays a significant role in the success of variable selection heuristics.

Introduction

Local search algorithms for propositional satisfiability such as GSAT (Selman, Levesque, & Mitchell 1992) have received much attention in recent years because of the discovery that it is possible to find solutions to difficult problems which are much larger than those which are solvable with conventional, systematic approaches such as the Davis-Putnam procedure (Davis & Putnam 1960), at the cost of completeness.

The basic schema for a local search algorithm for satisfiability testing is the following: Initially, a complete assignment of truth values to variables is generated randomly. Then, the truth values of the variables are repeatedly flipped in order to find a satisfying solution, usually applying some hill-climbing technique that tries to minimize the number of clauses that are left unsatisfied. Since hill-climbing approaches are susceptible to getting stuck at local minima, various heuristics for escaping local minima are applied to resume progress once the hill-climbing search becomes stuck.

The best-known local search algorithm for satisfiability testing is GSAT (Selman, Levesque, & Mitchell 1992), shown in Figure 1. A greedy hill-climbing procedure is repeatedly applied to a randomly generated initial assignment. In order to escape local minima, after each MAXFLIPS flips, a new random assignment is generated, and the greedy-hill climbing procedure begins again. This is repeated MAXTRIES times (or until time runs out).

```
GSAT

Input: set of clauses α, MAXFLIPS, and MAXTRIES
Output: a satisfying truth assignment of α, if found
for i := 1 to MAXTRIES
  T := a randomly generated truth assignment
  for j := 1 to MAXFLIPS
    if T satisfies α then return T
    p := Choose a propositional variable to flip that
      maximizes the increase in total number of clauses
      of α satisfied by T.
    T := T with the truth assignment of p reversed
  end for
end for
return failure (no satisfying assignment found)
end
```

Figure 1: The GSAT Procedure.

A distinctive feature of GSAT is the particular method by which a variable is chosen to be flipped. In the original formulation of GSAT, Selman et al. specified that "... the variable whose assignment is to be changed is chosen at random from those that would give an equally good improvement ." Their justification for this choice was that "such non-determinism makes it very unlikely that the algorithm makes the same sequence of changes over and over." (Selman, Levesque, & Mitchell 1992). However no empirical support of this conjecture was presented.

Gent and Walsh (Gent & Walsh 1992; 1993b) later questioned the importance of randomness in the method of picking the variable to be flipped, and presented experimental results using 50-100 variable random problems, as well as N-queeens problem instances

and Walsh (Gent & Walsh 1993 b), proposed that a strategy is fair "...if it is guaranteed to eventually pick any variable that is offered continually." They note that this is a weak definition, since it allows a variable not to be picked if it presented, say, only every other time. With respect to the terminology used in this paper, we define fairness as follows:

Definition 1 *A variable selection heuristic H is* fair *if the following is* true: Given a *variable v which is inserted into a gain bucket b, H guarantees that if b is* continually *selected as the highest gain bucket and v* remains *in b, then v* will eventually *be selected* (with probability approaching *1).*

The Random heuristic uses a gain bucket that is implemented as an array in which access to any element takes constant time, and the number of elements in the array is maintained. At each iteration, an element of the array is picked randomly and chosen to be flipped. This element is then deleted, and resulting "hole" in the array is filled in by moving the last nonempty element of the array into the hole. New elements are inserted into the first empty element of the array. Thus, insertions and removals are done in constant time. This is a fair strategy, since the probability of eventually picking a variable approaches 1 if the bucket is randomly sampled infinitely many times,

The First-In-First-Out *(FIFO)* strategy uses a linked list gain bucket implementation. Variables are inserted at the tail of the list, and are chosen (removed) from the head of the list, so both insertion and removal are constant time operations. FIFO is clearly fair, since a variable that is inserted into a gain bucket with N variables is guaranteed to be picked after the gain bucket is chosen at most N + 1 times by GSAT.

The *Last-In-First- Out (LIFO)* heuristic uses a linked list gain bucket implementation. Variables are inserted and removed from the head of the list. Both insertion and removal take constant time. Intuitively, this is a poor strategy, since there is too much locality in the choice of variable – it is possible to cycle among a small subset of variables at the head of the gain bucket list, while other variables are continually "stuck" near the tail of the list and never chosen. LIFO is not a fair strategy, because even if a variable v is offered continually, it is possible to never pick v if at every step a new element is added to the gain bucket.

Experimental Results

The tie-breaking heuristics described above (Random, FIFO, LIFO) were evaluated experimentally using formulas generated randomly using the fixed clause length model (Mitchell, Selman, & Levesque 1992). Three parameters are controlled: the number of variables N, the number of literals per clause k, and the number of clauses M. For a given N and M, a random problem instance is created by generating M clauses of k variables, where each clause contains k distinct variables from N which are negated with probability 0.5. 500 instances of 3-SAT ($k = 3$ variables per clause) problem instances were generated, where the $R = M/N$, the ratio of the number of clauses to the number of variables was fixed at 4.3. As shown in (Mitchell, Selman, & Levesque 1992), this generates a distribution of instances that are difficult to solve. Problems of up to 500 variables were used.

	Parameters		Number Solved		
Vars	MAXFLIPS	MAXTRIES	Random	FIFO	LIFO
50	250	10	**246**	**232**	**23**
100	500	50	**174**	**138**	**0**
150	1500	100	**170**	**84**	**0**
200	2000	250	**144**	**58**	**0**
250	2500	250	**130**	**37**	**0**
300	6000	250	**152**	**12**	**0**
400	8000	450	**83**	**2**	**0**
500	10000	1000	**18**	**1**	**0**

Table 1: Performance of the Random, FIFO, and LIFO variable selection heuristics on difficult "random 3-SAT instances. The number of instances (out of 500) that were solved by each strategy given a computational resource bound of MAXTRIES and MAXFLIPS is shown above.

For each randomly generated problem, GSAT was run using each of the tie-breaking strategies, where each run consisted of up to MAXTRIES iterations of MAXFLIPS flips (the solution was completely randomized after every MAXFLIPS flips).

Table 1 shows the results of this experiment. The number of instances for which satisfying assignments were found using each tie-breaking strategy is shown. Note that not all of the problems in the randomly generated set are satisfiable; according to previous studies (Mitchell, Selman, & Levesque 1992; Crawford& Auton 1996), approximately half of the 500 instances are expected to be satisfiable.[5]

As predicted, the LIFO strategy performed very poorly. Although the FIFO strategy was competitive with the randomized strategies for the smaller problems (50 and 100 variables), it performed relatively poorly on the larger problem instances. Overall, the Random strategy performed significantly better than the two non-random strategies.

Next, we tested the tie-breaking heuristics in the context of a slightly different local search, GSAT with random walk (Selman & Kautz 1993). This is a simple extension of GSAT which works as follows:

- With probability p, pick a variable occuring in some unsatisfied clause and flip its truth assignment.
- With probability $1 - p$, follow the standard GSAT scheme, i.e., pick randomly from the list of variables

[5] It would be possible to run a efficient systematic procedure such as Tableau (Crawford & Auton 1996) to determine exactly how many of the smaller instances are actually satisfiable; however, this was not currently feasible for a large set of the largest problem instances.

Parameters			Number Solved							
			FIFO-Random Hybrid				LIFO-Random Hybrid			
Vars	MAXFLIPS	MAXTRIES	**FR-0.1**	FR-O.25	FR-O.5	**FR-0.75**	**LR-0.1**	LR-O.25	LR-O.5	**LR-0.75**
50	250	10	246	251	255	251	146	196	244	243
100	500	50	169	167	189	182	36	110	156	181
150	1500	100	136	156	166	170	20	91	156	172
200	2000	250	112	126	145	153	9	66	132	150
250	2500	250	85	102	122	136	1	35	101	130
300	6000	250	82	104	134	159	1	47	132	160
400	8000	450	32	46	70	86	0	17	63	94
500	10000	1000	4	9	14	15	0	2	10	20

Table 4: Performance of the FIFO-Random and LIFO-Random hybrid variable selection strategies, for $p \in \{0.1,, 0.25, 0.5, 0.75\}$ on difficult random 3-SAT instances. (In the table **FR-0.1** denotes the FIFO-Random hybrid with $p = 0.1$, LR-0.5 denotes the LIFO-Random hybrid with $p = 0.5$, and so on.) The number of instances (out of 500) that were solved by each strategy given a computational resource bound of MAXTRIES and M AX FLIPS is shown above.

non-zero are moved from their current gain bucket to a new gain bucket.

These *side effect* movements of the variables among gain buckets can significantly disrupt the FIFO/LIFO orderings in the gain buckets.

For example, suppose that we are using a FIFO selection strategy, and a gain bucket g contains variables $v_1, v_2, \ldots v_n$ (where v_1 was inserted into g first, v2 was inserted second, and so on). Suppose g is repeatedly offered as the bucket with the highest gain. If there were no side effects, then the variables would be selected in the order inserted $(v_1, v2, \ldots v_n)$. However, when side effects are present, then with every flip, it is possible that some variables are moved from g, and/or some new variables are added to g. Obviously, the lower the average number of side effects per move, the more likely it is that g is undisturbed by a side effect and that the variables are selected in the original (FIFO) order.

Note that if the number of side effects is large with respect to the number of variables, then the variable selection heuristic becomes relatively unimportant - the fixed orderings imposed by the FIFO/LIFO strategies are irrelevant when a large fraction of the variables are moving in and out of the gain buckets at every flip due to side effects. The converse of this observation is that the fixed orderings (FIFO/LIFO) play a more significant role when the average fraction of the variables moving due to side effects is low.

It can be shown that on average, the maximum number of variables that move between gain buckets when a variable is flipped is independent of the size of the problem (the number of variables), and is relatively small for large problems. More formally:

Claim 1 *For the class of* **random** *k-SAT instances generated by the fixed clause length model, the average* **number** *of variables* **moving** *between gain buckets is less than* $(k-1)kR + 1$.

Proof:

Consider the class of random k-SAT instances generated using the fixed clause length model (defined in Section). Let v be a variable that is flipped. Let C_v be the set of clauses that contain v. Each clause has $k-1$ variables other than v, so clearly, the number of variables that move is bounded by $(k-1)|C_v| + 1$.

The expected value of $|C_v|$ is M x $(k$ x l/N) = k x $M/N = kR$, so the average number of variables that move between gain buckets is less than $k(k-1)R + 1$.□

This bound of $k(k-1)R + 1$ is independent of N. As the problem size grows (i.e., N increases), the fraction of the variables that move between gain buckets on each flip (which is bounded by $((k-1)kR)/N$) decreases. For small N, each flip moves a relatively large fraction of the variables between gain buckets, compared to large N. This partially explains why the performance differences between the variable selection heuristics depends on N: The smaller N is, the more disruptive each flip is to the gain table structure, and hence, the less important it is which of the Random/FIFO/LIFO strategies are being used.

Finally, it is important to note that although the analysis above uses the terminology introduced in Section (e.g., gain buckets), the results are independent of the particular implementation, and apply in general to similar data structures are used in order to enable the incremental updating of variable gains.[6]

Discussion

In this paper, we evaluated several tie-breaking heuristics for selecting variables to flip in GSAT. Randomly generated formulas from a difficult distribution were used to empirically compare the performances of the strategies.

In the context of standard GSAT (without random walk), we found that randomized tie-breaking heuristics performed best, while the FIFO strategy did relatively poorly, and the LIFO strategy performed very poorly. It is interesting to compare our results with that of (Gent & Walsh 1992; 1993b), who concluded that ".. there is nothing essential about randomness of picking in GSAT (although fairness is important)...". The poor performance of LIFO (an unfair strategy) is

[6] All efficient implementations of GSAT which we are aware of use a similar incremental updating framework.