The Search for Satisfaction

Ian P. Gent and Toby Walsh
Department of Computer Science
University of Strathclyde
Glasgow G1 1XH, Scotland
Email: {ipg,tw}@cs.strath.ac.uk

April 27, 1999

Abstract

In recent years, there has been an explosion of research in AI into propositional satisfiability (or SAT*). There are many factors behind the increased interest in this area. One factor is the improvement in search procedures for* SAT*. New local search procedures like* GSAT *are able to solve* SAT *problems with thousands of variables. At the same time, implementations of complete search algorithms like Davis-Putnam have been able to solve open mathematical problems. Another factor is the identification of hard* SAT *problems at a phase transition in solubility. A third factor is the demonstration that we can often solve real world problems by encoding them into* SAT*. There has also seen an improved theoretical understanding of* SAT*, particularly in the analysis of such phase transition behaviour. This paper reviews the state of the art for research into satisfiability, and discuss applications in which algorithms for satisfiability have proved successful.*

`http://dream.dai.ed.ac.uk/group/tw/sat/`

Other survey articles and many online papers can be found at the above address.

1 Introduction

Consider the following diplomatic problem proposed by Brian Hayes [45]:

> *"You are chief of protocol for the embassy ball. The crown prince instructs you either to invite Peru or to exclude Qatar. The queen asks you to invite either Qatar or Romania or both. The king, in a spiteful mood, wants to snub either Romania or Peru or both. Is there a guest list that will satisfy the whims of the entire royal family?"*

This problem can be represented as a propositional satisfiability problem. We can express the three constraints by means of a propositional formula,

$$(P \vee \neg Q) \& (Q \vee R) \& (\neg R \vee \neg P)$$

where P, Q and R are Boolean variables which are true if and only if we invite, respectively, Peru, Qatar, and Romania. A solution to the problem is a satisfying assingment, an assignment of truth values to the Boolean variables that satisfies the propositional formula. In this case, there are just two satisfying assignments (out of the eight possible). These either assign P and Q to true and R to false, or assign P and Q to false and R to true. That is, we can either invite both Peru and Qatar and snub Romania, or we can invite just Romania and snub both Peru and Qatar.

Whilst propositional satisfiability is a very simple problem, it is a cornerstone of the theory of computational complexity. Propositional satisfiability was the first problem shown to be NP-complete [10]. Despite a quarter century of effort, it remains an open question whether problems in this class can be solved in polynomial time in the worst case. Many consider this question, the P=NP problem, to be one of the most important open problem facing theoretical computer science. As the best complete algorithms are exponential, NP-completeness is generally considerable to mark the boundary between tractability and intractability. Nevertheless, a large amount of research in recent years has shown that satisfiability problems can often be solved efficiently in practice. In this paper, we survey this research, reporting the considerable improvements in our understanding of the computational complexity of propositional satisfiability, as well in algorithms to solve satisfiability problems.

2 Satisfiability

Propositional satisfiability (SAT) is the problem of deciding if there is an assignment for the variables in a propositional formula that makes the formula true. Many AI problems can be encoded quite naturally into SAT (*eg.* planning [61], constraint satisfaction, vision interpretation [82], diagnosis, ...). In addition, satisfiability is closely related to theorem proving; refutation procedures, for example, call upon the fact that a formula φ is derivable from a set of formulae Σ iff $\Sigma \cup \neg\varphi$ is not satisfiable.

Much research into SAT considers problems in conjunctive normal form (CNF). A formula is in CNF iff it is a conjunction of clauses; a clause is a disjunction of literals, where a literal is a negated or un-negated Boolean variable. A clause containing just one literal is called a unit clause. A clause containing no literals is called the empty clause and is interpreted as false. We can place restrictions on the size of clauses in the problem. For example, k-SAT is the class of decision problems in which all clauses are of length k. k-SAT is NP-complete for any $k \geq 3$ but is polynomial for $k = 2$ [26]. Other polynomial classes of SAT problems exist including Horn-SAT (in which each clauses contains no more than one positive literal), renameable Horn-SAT and several other generalizations.

Closely related to the SAT decision problem is the MAX-SAT optimization problem. Given a set of clauses, what is the maximum number of clauses that can be satisfied? Many of the complete and approximation procedures for SAT can be adapted to solve MAX-SAT problems by some simple modifications.

2.1 Random problems

There exist two popular models for random satisfiability problems: the fixed clause length and the constant probability models. In the fixed clause length model (often called the random k-SAT model), we generate problems with N variables and L clauses as follows: a random subset of size k of the N variables is selected for each clause, and each variable is made positive or negative with probability $\frac{1}{2}$. All clauses are thus the same size. In the constant probability model, we generate each of the L clauses in a problem by including each of the $2N$ possible literals with probability p. Since the inclusion of empty or unit clauses typically makes problems easier, many studies use a variant of the constant probability model proposed in [48] in which if a clause is generated containing either *no* literals or only *one* literal, it is discarded and another clause generated in its place. We call this the "CP" model.

3 Complete procedures

3.1 Davis-Putnam procedure

Despite its simplicity and age, the Davis-Putnam procedure remains one of the best complete procedure for satisfiability [19]. The original Davis-Putman procedure [17] was based on a resolution rule that eliminated the variables one-by-one and added all possible resolvents to the set of clauses. Unfortunately, this procedure requires exponential space in general. Davis, Logemann and Loveland therefore quickly replaced the resolution rule with a splitting rule which divides the problem into two smaller subproblems [16]. In much of the literature (including here), this later procedure is rather inaccurately called the "Davis-Putnam" or "DP" procedure.

procedure DP(Σ)
  (*Sat*) **if** Σ empty **then return** satisfiable
  (*Empty*) **if** Σ contains an empty clause **then return** unsatisfiable
  (*Tautology*) **if** Σ contains a tautologous clause c **then return** DP($\Sigma - \{c\}$)
  (*Unit propagation*) **if** Σ contains a unit clause $\{l\}$ **then**
    return DP(Σ simplified by assigning l to $True$)
  (*Pure literal deletion*) **if** Σ contains a literal l but not the negation of l **then**
    return DP(Σ simplified by assigning l to $True$)
  (*Split*) **if** DP(Σ simplified by assigning a literal l to $True$) is satisfiable
    then return satisfiable
    else return DP(Σ simplified by assigning the negation of l to $True$)

Figure 1: The Davis-Putnam Procedure.

After applying the split rule, we simplify the set of clauses by deleting every clause that contains the literal l assigned to $True$ (often called unit subsumption) and deleting the negation of l whenever it occurs in the remaining clauses (often called unit resolution). Note that unit subsumption can be ignored if the DP procedure is terminated when every variable has been assigned to a truth value.

Because of its cost, the pure literal rule is often deleted. Similarly, as tautologies can only be eliminated at the start of search, the tautology rule is often deleted. Neither rule is required for the soundness or completeness of the procedure. The unit propagation rule is also not necessary for the soundness or completeness of the procedure as the split and empty rules achieve the same effect. However, the unit propagation rule contributes greatly to the efficiency of the Davis-Putnam procedure. For Horn-SAT problems, the unit propagation rule is the basis for a complete procedure for determining satisfiability.

Hooker has shown how to adapt the Davis-Putnam procedure to determine incremental satisfiability efficiently [50]. Given a set of satisfiable clauses Σ, incremental satisfiability is the problem of deciding the satisfiability of $\Sigma \cup \{C\}$ for some new clause C. Incremental satisfiability problems arise in a variety of areas including hardware verification and knowledge base updating.

3.2 Branching heuristics

The Davis-Putnam algorithm is non-deterministic as we can choose the literal upon which to branch. A popular and cheap branching heuristic is MOM's heuristic. This picks the literal that occurs most often in the minimal size clauses. Ties are usually broken with a static or random ordering. In some cases, computing the literal that occurs most often is considered too expensive and we simply branch on the first literal of a shortest size clause.

The Jeroslow-Wang heuristic [54] estimates the contribution each literal is likely to make to satisfying the clause set. Each literal is scored as follows: for each clause c the literal appears in, $2^{-|c|}$ is added to the literal's score, where $|c|$ is the number of literals in c. The split-rule is then applied to the literal with the highest score.

Hooker and Vinay have investigated the motivation behind the scoring function used in the Jeroslow-Wang heuristic [51]. They propose the "satisfaction hypothesis", that it is best to branch into subproblems that are more likely to be satisfiable, but reject this in favour of the "simplification hypothesis", that it is best to branch into simpler subproblems with fewer and shorter clauses after unit propagation. The simplification hypothesis suggests a "two-sided" Jersolow-Wang rule which performs better than the original Jersolow-Wang rule.

One of the best current implementations of the Davis-Putnam procedure, *satz* uses a branching heuristic based on maximizing the amount of unit propagation [69]. This adds further support to Hooker and Vinay's simplification hypothesis.

3.3 Data structures and implementation

The performance of the Davis-Putnam procedure critically depends upon the care taken in the implementation. Crawford and Auton present an algorithm that does both unit resolution and subsumption in linear time [12]. For each variable, there are lists of the clauses that contain the variable poistively and negatively. Each clause has a counter which contains either the number of literals that have not yet been assigned or an `inactive` flag if the clause has been subsumed. When a variable is assigned to *true*, the counters of active clauses in which the variable appears negatively are decremented, and of active clauses in which the variable appears positively are set to `inactive`. When a variable is assigned to *false*, the analogous changes are performed. Zhang and Stickel show how to improve this slightly by performing unit resolution in amortized linear time, but unit subsumption in amortized constant time [102]. This improvement can offer a speedup of two on average over Crawford and Auton's algorithm on various SAT problems.

Discrimination trees (often called "tries") have proved useful for rapid indexing in a variety of areas of automated reasoning like term rewriting and resolution theorem proving. Zhang and Stickel have compared their use in the Davis-Putnam procedure with the list based methods described above [101]. In a trie-based procedure, the set of clauses is stored as a tree data structure in which edges are labelled by literals, and each path represents a the set of literals in one of the clauses. Unit resolution can then be performed by a merging operation.

Zhang and Stickel's experimental and theoretical results suggest a hybrid scheme in which clauses are represented as tries but lists are used to identify rapidly clauses in which the variable occurs positively and negatively. This scheme is implemented in the SATO procedure [98].

3.4 Non-clausal problems

Our description of the Davis-Putnam procedure has assumed that problems are in CNF or clausal form. This is not essential. For example, Otten has used a matrix representation to extend the Davis-Putnam procedure to non-clausal formulae [78]. Experiments on non-clausal problems show that this non-clausal Davis-Putnam procedure performs better than a standard Davis-Putnam procedure using the direct translation into clausal form (which is exponential in the worst case) as well as the definitional translation (which is quadratic, but introduces extra variables).

3.5 Intelligent backtracking and learning

The standard Davis-Putnam procedure performs chronological backtracking, exploring one branch of the search tree completely before backtracking and exploring the other. We can improve upon this by adapting some of the well-developed techniques from the constraint satisfaction community like conflict-directed backjumping and nogood learning. Conflict-directed backjumping backs up the search tree to the cause of failure, skipping over irrelevant variable assignments. Nogood learning records the cause of failure to prevent similar mistakes being made down other branches. Bayardo and Schrag have described how both of these mechanisms can be implemented within the Davis-Putnam procedure [3, 4]. They show that, with these enhancements (especially with relevance-bounded learning), the Davis-Putnam procedure can perform as well or better than local search procedures like GSAT and WALKSAT on a wide variety of benchmark problems.

3.6 Early mistakes

One problem with a complete procedure like Davis-Putnam is that an early mistake can be very costly. In [33], we identified SAT problems which were typically easy but could occassionally trip up the Davis-Putnam procedure. For example, on one CP problem, the first split led immediately to an unsatisfiable subproblem which took over 350 million branches to solve. On backtracking to the root of the search tree, a satisfying assignment was found down the other branch without any search. Gomes, Selman and Kautz have shown that a strategy of randomization and rapid restarts can often be effectively at tackling such early mistakes [43]. Meseguer and Walsh show that other modifications of the depth-first search strategy like limited discrepancy search and interleaved depth-first search can also tackle these early mistakes [74].

3.7 Parallelization

The non-deterministic split rule suggests a simple mechanism for parallelizing the Davis-Putnam procedure. Rather than explore one branch, backtrack, and explore the other, we can explore the two branches in parallel. To be effective, however, we must decide when to explore branches sequentially and when to explore branches in parallel. We must also balance loads on the different processors since the exploration of one branch can terminate before the exploration of the other. Zhang, Bonacina and Hsiang have implemented a simple master-slave distributed Davis-Putnam procedure and used it to solve some open quasigroup existence problems [100]

3.8 Upper bounds

Various upper bounds have been derived for the performance of the Davis-Putnam procedure and of other complete procedures on 3-SAT problems. The worst-case time complexity of Davis-Putnam is $O(1.696^N)$, and a small modification improves this to $O(1.618^N)$ [11]. Rodosek presents an algorithm that takes $O(1.476^N)$ by making binary decisions that eliminate at least two variables [83]. Beigel and Eppstein give an algorithm that reduces the 3-SAT problem to a 2-SAT problem, and then tries to extend the solution of this to the original problem [5]. In the worst case, this algorithm runs in $O(1.381^L)$. However, since the number of clauses L is usually much larger than the number of variables N, this is often not an improvement. Indeed, the $O(1.476^N)$ bound of Rodosek's algorithm is better for $L/N > 1.206$.

3.9 Resolution

van Gelder and Tsuji have augmented the Davis-Putnam procedure with a 2-closure rule which uses a limited version of resolution to generate all possible resolvents with 2 or fewer literals [94]. This algorithm was very competitive on hard random 3-SAT problems and circuit-fault analysis problems. More recently, Dechter and Rish have championed the original Davis-Putnam procedure [18]. They show that a variant of this procedure, which they call "directional resolution", performs well on various tractable classes like 2-SAT problems as well as on certain types of structured problems. They also propose an approximation procedure called "bounded directional resolution" which limits the size of resolvents to make the space complexity polynomial. This approximation procedure can be useful as a pre-processing step for the DP procedure.

3.10 Binary decision diagrams

Ordered binary decision diagrams (or OBDDs) are a mechanism for efficiently representing and manipulating Boolean expressions [6]. They have proved highly successful in the model checking of finite-state hardware. Uribe and Stickel found that for solving satisfiability problems, OBDDs were often more sensitive to the variable ordering than the Davis-Putnam procedure, and were less good on problems which have few solutions [93].

3.11 OR techniques

A variety of OR techniques have been used to determine the satisfiability of a set of clauses. For example, Hooker has developed a cutting plane algorithm for solving SAT problems [47]. He combined this with a branch-and-bound algorithm to obtain a branch-and-cut algorithm that shows some promise [49]. Kamath *et al.* have also had success with an interior point method [59].

3.12 Logical approaches

Much work on SAT solving techniques has been done by those implementing first order theorem provers. Since SAT is a subclass of (classical) first order logic, any theorem prover must inevitably contain a SAT solver, even if little conscious thought has been put into its construction. Some of these provers, for example the resolution based Otter [73], have had significant successes and must be dealing with propositional logic effectively. Often the line between propositional and first order reasoning is blurred, for example in systems that allow quantified formulae over only finite domains.

One appealing approach for dealing with propositional logic is Smullyan's method of analytic tableaux [90]. This has the advantage of not needing input in clausal form, and is extremely easy for humans reasoning on small formulas with pen and paper. A related technique is the KE calculus, which can sometimes improve exponentially on tableaux [15]. It is likely that there will be 'convergent evolution' of efficient SAT methods based on other logical calculi. For example, Crawford and Auton [12] report that, having started with a tableau calculus, their program 'tableau' could best be seen as an efficient implementation of Davis-Putnam.

4 Approximation procedures

4.1 A semi-decision procedure

Papadimitriou has proposed a semi-decision procedure for 2-SAT that repeatedly flips the truth assignment for a variable in an unsatisfied clause [79]. This satisfies the clause but may make other clauses unsatisfied. Using an argument based upon the gambler's ruin, Papadimitriou proves that, if the problem is satisfiable then this procedure will find a satisfying assingment in $O(N^2)$ flips with a probability approaching 1.

procedure 2SAT(Σ)
    T := *random(Σ)* ; random truth assignment
    repeat until T satisfies Σ
        v := variable in an unsatisfied clause
        T := T with truth assignment for v flipped
    end

Figure 2: Papadimitriou's semi-decision procedure for 2-SAT

It is less well known that there is also a decision procedure for Horn-SAT problems which flips variables in unsatisfiable clauses and which takes at most N flips [80]. This procedure has two phases. In the first phase, we start with a truth assignment that assigns $False$ to every variable. We then satisfy the Horn clauses that contain a positive literal by flipping the truth assignments to the positive literals. In the second phase, we simply test the remaining purely negative clauses to determine if we have a satisfying assignment.

procedure HornSAT(Σ)
    T := *all-false(N)* ; all N vars set to $False$
    for v := positive literals in Horn clause **do**
        T := T with truth assignment for v flipped
    end
    if T satisfies Σ **return** "satisfiable"
        else "unsatisfiable"

Figure 3: Papadimitriou's decision procedure for Horn-SAT

4.2 A simple greedy algorithm

Koutsoupias and Papadimitriou have introduced a simple approximation procedure for satisfiability which they call the "greedy algorithm" [67]. In this procedure, we choose a truth assignment at random. We then pick a variable such that flipping its truth value increases the number of satisfied clauses. We do not allow sideways moves. We repeat until no improvement is possible. If this is a solution, we report success. Otherwise, we simply give up. Koutsoupias and Papadimitriou prove that this procedure will find a satisfying assignment for almost all satisfiable problems with $O(N^2)$ clauses.

procedure greedy(Σ)
  T := *random(Σ)* ; random truth assignment
  repeat until no improvement possible
    T := T with a truth assignment flipped that increases
      number of satisfied clauses
  end
  if T satisfies Σ **return** "satisfiable"
    else "no satisfying assignment found"

Figure 4: Koutsoupias and Papadimitriou's greedy algorithm

Gent has proposed that an even simpler greedy algorithm and shown an even better bound on performance. In this algorithm, we simply assign a variable according to which polarity it has most often. This will find a satisfying assignment for almost all satisfiable problems with $O(N \log(N))$ clauses [29].

procedure greedy(Σ)
  T := *nil* ; no truth assignments
  for v := 1 **to** N
    if v occurs more often positively in Σ **then** T := T $\cup\{v/True\}$
      else T := T $\cup\{v/False\}$
  end
  if T satisfies Σ **return** "satisfiable"
    else "no satisfying assignment found"

Figure 5: Gent's greedy algorithm

4.3 GSAT

The GSAT procedure [88] essentially adds restarts and sideways flips to Koutsoupias and Papadimitriou's greedy algorithm. Despite its simplicity, GSAT and its variants give good performance on a wide variety of satisfiability problems. GSAT starts with a randomly generated truth assignment, and hill-climbs by changing the variable assignment which gives the largest increase in the number of clauses satisfied. If there exists no variable assignment that can be changed to

```
procedure GSAT(Σ)
    for i := 1 to Max-tries
        T := random(Σ) ; random truth assignment
        for j := 1 to Max-flips
            if T satisfies Σ then return T
            else T := T with truth assignment flipped to maximize
                    number of satisfied clauses
        end
    end
    return "no satisfying assignment found"
```

Figure 6: The GSAT local search procedure

increase the score, a variable is flipped which does not change the score. Without such sideways flips, the performance of GSAT degrades greatly. In [32] it is shown that much of the search consists of the exploration of large "plateaus" where sideways flips predominate and only the occasional up flip is possible.

Local search procedures like GSAT's need to be implemented with some care. Naively, one simply iterates through the variables, and for each one calculate the number of clauses satisfied if that variable is flipped. For N variables and L clauses, the work involved is O(NL). However, if we store for each variable the change in score that flipping it would make, it is possible to update this information after each flip just by examining each clause that the flipped variable appears in.

Sebastiani has described how to modify GSAT so that it can be used with non-clausal formulae [39]. He shows how to compute the number of clauses that would be satisfied if the formulae was converted into CNF without constructing the CNF conversion itself. This computation takes time linear in the size of the input formula.

4.4 Gu's procedures

At around the same time as GSAT was introduced, several families of local search procedures for SAT were proposed by Gu [44]. These procedures share many similarities with GSAT. However, they have a different control structure which allows them, for instance, to make sideways moves when upwards moves are possible. No direct comparison has yet been made between Gu's procedures and the GSAT procedure and its variants.

```
procedure SAT1.0(Σ)
    T := random(Σ) ; random truth assignment
    while T does not satisfy Σ do
        for i := 1 to number of variables
            if T with v_i flipped does not decrease score
                then T := T with v_i flipped
        if local then flip some variables at random
    end
```

Figure 7: The SAT 1.0 family of procedures introduced by Gu.

4.5 GenSAT

To study GSAT in more detail, we proposed the GENSAT family of local search procedures [28] (see Figure 4). GSAT is an instance of GENSAT in which *hill-climb* returns those variables which increase the score most by being flipped, while *pick* chooses a random member of the set given to it, and *initial* simply sets each variable independently to true or false with equal probability. We used the GENSAT family to support the conjectures made in [27] that neither greediness nor randomness are essential for the effectiveness of GSAT. We also proposed a new procedure, HSAT which often performs much better than GSAT, and which has since been used by several other researchers [95, 60, 21]. HSAT modifies GSAT to take account of historical information concerning when variables were last flipped. When offered a choice of variables, HSAT always picks the one that was flipped longest ago (in the current try): if two variables are offered which have never been flipped in this try, an arbitrary (but fixed) ordering is used to choose between them. HSAT is otherwise like GSAT.

4.6 Weights

One modification of the GSAT procedure that can be very effective on certain classes of problems are the "clause weights" introduced in [84] and [76]. Associated with each clause is a weight which is varied during search to focus attention on the hardest clauses. The score function is the sum of the weights of the satisfied clauses. Selman and Kautz originally only modified weights at the end of each try. Frank has shown that both GSAT and HSAT can be improved if weights are modified after every flip [21].

```
procedure GenSAT(Σ)
    for i := 1 to Max-tries
        T := initial(Σ) ; initial truth assignment
        for j := 1 to Max-flips
            if T satisfies Σ then return T
            else Poss-flips := hill-climb(Σ, T)
                 v := pick(Poss-flips)
                 T := T with truth assignment to v flipped
        end
    end
    return "no satisfying assignment found"
```

Figure 8: The GenSat family of local search procedures.

4.7 Random walk

The success of Papadimitriou's semi-decision procedure for 2-Sat and his decision procedure for Horn-Sat illustrate the benefits of focusing search on variables in unsatisfied clauses (called, from now on, unsatisfied variables). A variant of Gsat, called Gsat with random walk [86] focuses some of its search effort on unsatisfied variables. With probability p, Gsat with random walk flips a variable in an unsatisfied clause, and otherwise hill-climbs normally. Gsat with random walk can be seen as an instance of the GenSat family in which *hill-climb* returns the unsatisfied variables with probability p, or the variables that maximize the score when flipped otherwise. Flipping an unsatisfied variable can actually decrease the number of satisfied clauses. Nevertheless random walk improves the performance of Gsat considerably. Flipping an unsatisfied variable appears to be a better strategy than flipping a variable at random with probability p. In all satisfying assignments, at least one of the variables in an unsatisfied clause must have the opposite truth value. We must therefore flip at least one of them en route to a satisfying assignment. Flipping an unsatisfied variable is also guaranteed to change the set of unsatisfied clauses. Since Gsat's search is governed by the variables in this set, flipping an unsatisfied variable introduces diversity into the search. In [36], we showed that one of the largest benefits of adding random walk is the reduction in the sensitivity of performance to the Max-Flips parameter.

4.8 WalkSAT

The WalkSat procedure introduced in [86] takes the random walk idea one step further and makes it the central component of the algorithm. WalkSat essentially adds restarts and a heuristic to choose between unsatisfied variables to Papadmitriou's semi-decision procedure for 2-Sat . WalkSat chooses an

```
procedure WalkSAT(Σ)
    for i := 1 to Max-tries
        T := random(Σ) ; random truth assignment
        for j := 1 to Max-flips
            if T satisfies Σ then return T
            else pick an unsatisfied clause
                    v := var in clause chosen either greedily or at random
                    T := T with truth assignment to v flipped
        end
    end
    return "no satisfying assignment found"
```

Figure 9: The WalkSat local search procedure.

unsatisfied clause at random, and then picks an unsatisfied variable to flip from this clause using either a greedy or a random heuristic. WalkSat is an instance of the GenSat family in which *hill-climb* returns the variables in an unsatisfied clause, and pick chooses between then either randomly or greedily. The exact choice is as follows: if a variable can be flipped to satisfy the clause without breaking any other clauses, then it is flipped, otherwise, with a probability p we flip the variable that breaks the fewest number of clauses else we flip a variable at random. Note that, unlike GSat, the score function does not consider the number of clauses satisfied, only the number of clauses that are currently satisfied which become unsatisfied.

4.9 Novelty

Inspired perhaps by the good performance of WalkSat and HSat, the Novelty algorithm [72] combines together a focus on unsatisfied clauses with a mechanism similar to that in HSat for flipping variables that have not been flipped recently. The Novelty algorithm also returns to GSat's score function of the number of satisfied clauses. The Novelty algorithm choses an unsatisfied clause at random. It then flips the variable with the highest score provided it is not the last variable in the clause to be flipped. If it is, then with probability p, Novelty flips the variable with the next highest score, and with probability $1-p$, the variable with the highest score. The R-Novelty algorithm (also introducted in [72]) takes into account the difference between the two highest scores. This makes the procedure entirely deterministic for $p \in \{0, 0.5, 1\}$. To inhibit loops, the unsatisfied variable is therefore selected at random every 100 flips.

4.10 Simulated annealing

Simulated annealing has not proved as popular as local search methods like GSAT. Comparisons between local search methods like GSAT and simulated annealing paint a rather muddled picture. For example, Selman and Kautz report that they are unable to find an annealing schedule that performed better than GSAT [85]. On the other hand, Spears presents results that suggest that simulated annealing scales better than GSAT on hard random 3-SAT problems [92]. We are unaware of any comparisons between simulated annealing with local search methods like WALKSAT and Novelty which tend to perform better than GSAT on this problem class.

4.11 Tabu search

Given the success of historical information in HSAT and Novelty, we might expect tabu search to perform well on SAT problems. This is indeed the case. Mazure, Saïs and Grégoire propose TSAT, a basic tabu search algorithm for SAT [71]. This algorithm keeps a fixed-length chronologically ordered FIFO-list of flipped variables. On hard random 3-SAT problems, they found that the optimal length of this tabu list increased linearly with N. With such a list, they showed that TSAT was highly competitive with WALKSAT.

4.12 Hybrid methods

There has been some interest in hyrbid methods that combine together the best feature of approximate and complete methods for SAT. Local search methods can traverse the search space rapidly. Complete methods, on the other hand, tend to follow much more restricted trajectories. They compensate for this by powerful constraint propagation rules like unit propagation. Zhang and Zhang have therefore proposed a hybrid method for SAT that combines together a local search method based on GSAT and a systematic procedure based on the Davis-Putnam procedure [104]. They report some promising result on certain satisfiable quaisgroup existence problems.

4.13 Other approaches

There has been a limited amount of work using neural networks and genetic algorithms to solve SAT. One exception is [91] in which Spears proposes a Hopfield network algorithm for solving SAT problems. The network is based on the

AND/OR parse tree of the SAT problem. Spears reports favourable results compared to GSAT on hard random 3-SAT problems. Spears and de Jong have also proposed a simple genetic algorithm for solving SAT problems [56]. However, insufficient results are reported to determine if this approach is competitive with local search methods like GSAT.

Hogg and Williams have proposed quantum algorithms for solving SAT problem. For example, Hogg has proposed a simple quantum algorithm for satisfiability and constraint satisfaction problems [46]. He has observed similar phase transition on random 3-SAT problems as seen in classical computational methods.

Finally, Lipton has shown how to SAT problems using DNA-based computing methods [70].

5 Phase transition behaviour

Following [8], there has been considerable interest in "phase transition" behaviour in many different NP-complete problems. Problems which are very over-constrained are insoluble and it is usually easy to determine this. Problems which are very under-constrained are soluble and it is usually easy to guess one of the many solutions. A phase transition tends to occur inbetween when problems are "*critically constrained*" and it is difficult to determine if they are soluble or not. Such behaviour has been extensively studied in SAT over the last five years.

5.1 Random k-SAT phase transition

For random 2-SAT, the phase transition has been proven to occur at $L/N = 1$ [9, 42]. For random 3-SAT, the phase transition has been shown to occur experimentally around $L/N = 4.3$ [75, 12]. Theoretical bounds put the phase transition for random 3-SAT in the interval $3.003 < L/N < 4.598$ [23, 66]. For random 4-SAT, the phase transition has been shown to occur experimentally around $L/N = 9.8$ [35]. For large k, the phase transition for random k-SAT occurs at a value of L/N close to $-1/\log_2(1-\frac{1}{2^k})$ [65]. A recent result of Friedgut proves that the width of the phase transition in the random 3-SAT model narrows as problems increases in size [22]. Kamath *et al.* show that at the random 3-SAT transition, the distribution in the number of satisfying assignments is very skewed [58]. An exponentially small number of problems have an exponentially large number of satisfying assignments. This makes it hard to calculate the exact location of the transition.

A technique borrowed from statistical mechanics called finite size scaling [1] models the shape of the satisfiability transition [65, 87]. Around a critical value of L/N, finite size scaling predicts that maroscopic properties like the probability of satisfiability are indistinguishable except for a simple power law scaling in N.

The scaling of search costs can also be accurately modelled using this technique [30].

5.2 Random k-SAT search cost

For complete methods like Davis-Putnam, median search cost has been observed to peak in the random 3-SAT phase transition where approximately 50% of problems are satisfiable [75]. Crawford and Auton report simple exponential growth for their Davis-Putnam procedure on the random 3-SAT problem class at $l/n = 4.2$ (i.e. at the phase transition) and at $l/n = 10$ (i.e. in the over-constrained region) [13]. They also record roughly linear growth at $l/n = 1, 2$ and 3 in the under-constrained region, but caution that "below the cross-over point ... there are some problems that are as hard as those at the cross-over point" [13]. We have reported similar behaviour in the constant probability model [35, 34].

The theoretical analysis of the Davis-Putnam procedure has largely been restricted to the easier constant probability problem class. One exception is Yugami who has developed an approximate theoretical model for the average case complexity of the Davis-Putnam procedure across the random 3-SAT phase transition [97]. Whilst the results agree well with experiment, the derivation is complex and only yields recursion equations.

For local search methods, we reported a possible cubic growth in search cost for GSAT at the random 3-SAT phase transition in [28]. Parkes and Walser also record sub-exponential growth for several variants of the GSAT procedure at the random 3-SAT phase transition [81]. Gent et al. extend these results to model search cost across the whole random 3-SAT phase transition [31] They were unable to fit their data for GSAT to an exponential model of search growth, and used instead a simple two parameter power law model which grew no faster than n^4.

Recently, there has been more detailed analysis of the distribution of search costs. Frost and Rish have proposed modelling discrete run-time distributions with continuous probability functions [24]. They show that on hard unsatisfiable random 3-SAT problems, the run-time distribution of the Davis Putnam procedure can be modelled by a lognormal density function. Hoos and Stutzle have also modelled run-time distributions, focusing on individual problem instances to reduce variance [53, 52]. They showed that the run-time distribution for algorithms like WALKSAT can often be modelled with an exponential distribution. This supports experimental results in [36] that suggests that random restarting offers little or no advantage with algorithms that perform random walk as such algorithms tend not to be stuck in a local minimum.

5.3 Constant probability model

Various theoretical results have been shown for the constant probability model without the deletion of empty and unit clauses. For example, Purdom and Brown [57] show that average running time is polynomial if any of the following hold asymptotically: (1) $L \leq MlnN$ for some constant M; (2) $L \geq e^{\epsilon N}$ for some constant ϵ; (3) $p \geq \epsilon$; or (4) $p \leq M(lnN/N)^{3/2}$. It is important to note, however, that these results do not cover all possible functions of p and L, and do not in the limit cover the phase transition between satisfiable and unsatisfiable.

Most experimental studies have used the CP model in which empty and unit clauses are deleted to prevent trivially insoluble problems. If the expected clause length is kept roughly constant by varying p as $1/n$ then the solubility phase transition occurs around an approximately constant value of L/N [35]. Whilst problems from the CP model are typically much easier than random k-Sat problems of a similar size [75], occasional problems in the "easy" soluble region can be much harder for the Davis-Putnam procedure than the hardest problems of the "hard" region of the random k-Sat model [35, 34]. Such problems occur in a region where constraint propagation is least effectively, search depth is maximal and branching heuristics can make occasional very expensive mistakes [37].

6 Applications

If a problem is NP-complete, we can in theory translate it into Sat using an encoding that is polynomially bounded in size. Research over the last decade has shown that this approach is often surprisingly effective in practice.

6.1 Graph colouring

Graph colouring problems can be easily encoded into Sat. To encoding the problem of k-colouring a graph of n nodes, we use kn variables, with v_{ij} true iff node i takes the jth colour. A more compact encoding is possible that uses $n \log_2(k)$ variables in which v_{ij} is true iff node i has a colour with the jth bit set. However, this encoding appears to be much harder to reason about. In addition, we may choose not to specify the constraints that each node takes only one colour. If a node is given two or more colours, we can simply pick one at random. This is especially useful for local search methods. Although Selman *et al.* report that Gsat is competitive with specialized graph colouring procedures [88], there has been little other work reported in this area. One exception is Hoos, who has looked at the effect of different encodings of graph colouring problems into Sat on local search algorithms like WalkSat [52]. Another exception is the solution of quasigroup existence problems, which can be viewed as a specialized type of graph colouring problem.

6.2 Quasigroup existence

A quasigroup is a Latin square, a n by n multiplication table in which each entry appears once in every row and column. Quasigroups model a variety of practical problems like tournament scheduling and designing drug tests. A variety of automated reasoning programs have been used to answer open questions about the existence and non-existence of quasigroups with particular properties of interest to mathematicians [25]. For example, the QG5 family of problems concern the existence of idempotent quasigroups (those in which $a \cdot a = a$ for each element a) in which $(ba \cdot b)b = a$.

An order n problem can be encoded into SAT using n^3 variables, with v_{ijk} true iff $i \cdot j = k$. Symmetry breaking to eliminate isomorphic solutions is crucial for the efficient solution of these problems. One symmetry breaking constraint often used is that that $a \cdot n \geq a-1$. Zhang has had considerable success solving encodings into SAT of quasigroup existence problems using SATO, his efficient implementation of the Davis-Putnam procedure [99, 98]. Zhang, Bonacina and Hsiang have also solved several open quasigroup existence problems with PSATO, a parallel version of the Davis-Putnam procedure [100]. Finally, Stickel has solved some open problems with various implementations of the Davis-Putnam procedure [89].

6.3 Hardware

Problems like the verification and diagnosis of faults in hardware have proved popular targets for encoding into SAT. Such problems often map into SAT in a very direct and natural way, especially if the circuit model ignores timing issues. The signal on a wire can be modelled with a Boolean variable which is $True$ if the signal is Hi and $False$ otherwise. Logic gates like $NAND$ are then modelled by the logical connectives of the same name. For example, Larrabee has generated some challenging SAT problems based on a variety of faults in synchronous digital circuits including "single stuck-at" and "bridge-faults" [68]. As another example, Kamath, Karmarkar, Ramakrishnan and Resende have proposed some benchmarks which encode circuit synthesis problems into SAT [59]. Whilst these circuit synthesis problems have been used in a number of studies of local search methods (for example, [88], [84] and [86]), they pose few challenges to complete methods like Davis-Putnam. Even with a simple-minded branching heuristic, our results show that the Davis-Putnam procedure solves these problems with little search.

6.4 Planning

Kautz and Selman have had considerable success solve planning problems by encoding them into SAT and using both local search methods like GSAT and complete procedures like Davis-Putnam with a randomization and restart strategy

[61, 62, 43]. In general, planning is a more complex problem than SAT. With no restrictions on the operators, planning is undecidable as we can represent a Turing machine using plan operators. Even restricted to propositional STRIPS-like operators, planning is PSPACE-complete. To encode planning problems into SAT, we must therefore make additional restrictions. Kautz and Selman's proposal is to bound the length of plan. This bound can be iteratively increased to retain completeness.

A planning problem can be represented as a set of axioms, for which any satisfying assignment corresponds to a valid plan. Each predicate in the planning language is augmented with an extra time index. The bound on plan length limits the number of time indices needed. For example, the predicate $move(x, y, z, i)$ is true iff we move x from y to z at time step i. Frame axioms are needed to ensure that predicates continue to hold if they are not affected by the actions. Kautz and Selman hand-crafted their original encodings with additional domain-specific information that is not-expressible in the STRIPS action language (for example, they add axioms to state that the fluent *on* is irreflexive and non-commutative). Ernst, Millstein and Weld therefore proposed the MEDIC planning system in which the encoding is automatic and approaches the efficiency of Kautz and Selman's hand-crafted encodings [20]. Kautz and Selman build upon this work, showing highly competitive performance with encodings into SAT that take advantage of a simple declarative specification of domain-specific knowledge [64]. Kautz and Selman have also investigated a different way of encoding planning problems into SAT based upon the planning representation used in the ground-breaking Graphplan system [63].

6.5 Steiner trees

Network Steiner tree problems have been extensively studied in operations research. A network Steiner tree problem consists of an undirected graph, where each edge is assigned a positive integer cost, and a subset of the nodes, called the Steiner nodes. The aim is to find a subtree of the graph that spans the Steiner nodes, such that the sum of the costs of the edges of the tree is minimal. Such problems have many applications in network design and routing. A simple encoding into SAT uses a number of variables that is quadratic in the number of edges. Since problem instances of interest have thousands of edges, such an encoding is not of practical use. Jiang, Kautz, and Selman therefore developed an approximate encoding that is essentially linear in the number of edges in the graph [55]. The encoding is approximate since it does not preserve all possible solutions. Nevertheless, WALKSAT on the encoding produced solutions that were competitive with the best current specialized OR algorithms. Davenport (personal communication) subsequently showed that even a very simple Davis-Putnam procedure found solutions to these encodings.

6.6 Scheduling

Crawford and Baker have encoded machine-shop scheduling problems into SAT [14]. The machine-shop scheduling problem consists of scheduling a set of jobs subject to a set of resource constraints, start and due dates, and sequencing constraints. Crawford and Baker reject a naive encoding of this problem in which the Boolean variable $v_i t$ is $True$ iff operation i starts at time t as it gives a much larger search space than necessary. Instead, they propose an encoding in which the Boolean variable $v_i j$ is $True$ iff operation i starts before operation j. They observed that Sadeh's sceduling benchmarks encode into large, under-constrained SAT problems. Whilst their Davis-Putnam procedure performed poorly on some of these encodings due to early-mistakes, a simple iterative sampling algorithm was able to solve all of Sadeh's problems with very little search.

6.7 Diagnosis

Pandurang and Williams have constructed a propositional theory to monitor, diagnose and repair the Deep Space One spacecraft [96, 77]. Using some complex and domain-dependent compilation techniques, they are able to construct plans of action in essentially constant time per plan step.

6.8 Benchmark libraries and other resources

Many of these problems have been collected into benchmark libraries. One of the oldest is the benchmark library that was developed for the Second DIMACS International Algorithm Implementation Challenge in 1993. Several hundred problems from this challenge can be found at:

`ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf/`

This challenge also proposed a simple plain text format for specifying satisfiability problems which has since become a standard. A benchmark library was also created for the International Competition and Symposium on Satisfiability Testing held in Beijing in March 1996. Problems from this library are available from:

`http://www.cirl.uoregon.edu/jc/beijing`

More recently, Hoos and Stutzle have created the SATLIB library at

`http://www.informatik.tu-darmstadt.de/AI/SATLIB`.

This contains several benchmark problems, as well as links to various solvers and other resources.

7 Beyond the propositional

Many of the algorithms developed for SAT have been extended to languages which are richer than the purely propositional.

7.1 Linear 0-1

Linear 0-1 problems are constraints of the form,

$$\sum_i c_i \,.\, x_i \;\sim\; d$$

where x_i are 0-1 variables, $c_i, d \in Z$, and $\sim$ is one of the relations, $\{=, \leq, <, \geq, >\}$. SAT problems can be easily encoded as a set of linear 0-1 constraints. Each clause,

$$x_0 \vee \ldots x_i \vee \neg x_{i+1} \vee \ldots \neg x_n$$

is converted into the constraint,

$$x_0 + \ldots + x_i + (1 - x_{i+1}) \ldots + (1 - x_n) \geq 1.$$

On the other hand, converting linear 0-1 constraints into SAT can result in a large (but polynomially bounded) number of clauses. For instance, problems like the pigeonhole problem which involve counting can often be compacting represented using linear 0-1 constraints but tend to need much larger encodings in SAT. Optimization problems with linear objective functions are also easily represented as a sequence of linear 0-1 decision problems Both complete and approximation methods for SAT have been generalized to deal with linear 0-1 constraints. For example, Barth has extended the Davis-Putnam procedure to linear 0-1 constraints [2]. Walser has extended the WALKSAT procedure to linear 0-1 constraints [95]. Both of these approaches show promise on benchmark problems.

7.2 Modal logics

Propositional modal logics extend the propositional language of SAT with modal operators like $\Box$ and $\Diamond$. The exact interpretation of these operators depends on the particular modal logic. For example, in the modal logic K, if $\Box(\varphi \rightarrow \psi)$ and $\Box\varphi$ then $\Box\psi$ also holds. There has been revived interest in the satisfiability of such logics since it was shown that the terminological logic ALC is a notational variant of K(m), the propositional modal logic K with m modalities [40]. Giunchiglia and Sebastiani have shown how procedures like Davis-Putnam can be extended to deal with many different types of propositional modal logics [39]. They report orders of magntidue improvement over more traditional tableau based procedures for ALC like Kris. The key observation behind their extension of the Davis-Putnam procedure is that the propositional connectives retain the same semantics. We can therefore use the Davis-Putnam procedure to determine possible truth values for (potentially boxed or diamond) formulae within a disjunction. We then recursively call the Davis-Putnam procedure on the formulae that appear within the scope of these boxes and diamonds to check that the boxes and diamonds can be appropriately satisfied. Giunchiglia and Sebastiani also extend the random k-SAT model to modal logics [41]. They identify similar phase transition behaviour in the modal logics K and S5 to that observed the random k-SAT model.

7.3 QSAT

QSAT is the problem of deciding the satisfiability of a propositional formula in which the Boolean variables are either existentially or universally quantified. For example, $\forall x \exists y \ (x \vee \neg y) \wedge (\neg x \vee y)$ is satisfiable since whatever truth value, $True$ or $False$ we give to x, there is a truth value for y, namely the same value as x, which satisfies the quantified formula. QSAT is a more complex decision problem than SAT. Whilst SAT is NP-complete, QSAT is PSPACE-complete[80]. Many two person games like the generalized versions of checkers, Go, Hex, and Othello are PSPACE-complete. Indeed, we can see QSAT as a game between the existential quantifiers, which try to pick instantiations that give a satisfied formula, and the universal quantifiers, which try to pick instantiations that give an unsatisfied formula. Cadoli *et al.* have extended the Davis-Putnam to decide the satisfiability of QSAT problems [7]. They report qualitiatively different behaviour across the phase transition for a simple generalization of the random k-SAT model to QSAT. In [38], we show that, if trivially insoluble problems are eliminated from this model, behaviour across the phase transition is very similar to that in the random k-SAT model.

7.4 First-order logic

Propositional reasoning can be used to find finite models of statements in a richer first-order logic. We may then combine the efficiency of propositional reasoning with the parsimony of a first-order specification. For example, Zhang and Zhang have developed SEM, a system for generating finite models of first-order many sorted theories [103]. The input to SEM is a many-sorted first-order logic with functions and equality. We can express concepts in this logic like $\forall x \ . \ f(x, x) = x$ where x ranges over the elements in a quasigroup, and f is the quasigroup multiplication function. This equality constraint enforces idempotency. This can be instantiated to give n ground equality constraints, where n is the order of the group.

8 Conclusions

As this survey article has shown, there has been considerable progress made in the study of propositional satisfiability over the last decade. Progress has been both theoretical and practical. On the practical side, we now have complete and local search procedures able to solve problems with hundreds and, in some cases, thousands of variables. Many problems of practical interest have also been solved by encoding them into SAT. On the theoretical side, we have a more sophisticated picture of what makes SAT problems hard to solve. Research into phase transition behaviour has been particularly informative. What can we expect in the future? SAT remains a field in which experiment is perhaps well ahead of theory. We can therefore expect theory to do some catching up, especially in the theoretical analysis of local search methods. We can also expect more applications as the message spreads that encoding problems into SAT can be a viable solution method. Finally, encoding problems into SAT will re-focus the research community onto representation and problem reformulation. These are topics that go back to the earliest days of AI and which remain central to the solution of difficult search problems.

Acknowledgements

We thank the other members of the APES research group at the Universities of Strathclyde and Leeds, and the members of the Mathematical Reasoning Group at Edinburgh University. The second author is supported by EPSRC award GR/K/65706.

References

[1] Michael N. Barber. Finite-size scaling. In *Phase Transitions and Critical Phenomena, Volume 8*, pages 145–266. Academic Press, 1983.

[2] P. Barth. A davis-putnam based enumeration algorithm for linear pseudo-boolean optimization. Research report mpi-i-95-2-003, Max Plack Institut fur Informatik, Saarbrucken, 1995.

[3] Roberto Bayardo and Robert Schrag. Using CSP look-back techniques to solve exceptionally hard SAT instances. In *Proceedings of Second International Conference on Principles and Practice of Constraint Programming (CP96)*, 1996.

[4] Roberto Bayardo and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on AI*. American Association for Artificial Intelligence, 1997.

[5] R. Beigel and D. Eppstein. 3-coloring in time $o(1.3446^n)$: a No-MIS algorithm. Technical report, ECCC, 1995. TR95-33.

[6] R.E. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.

[7] M. Cadoli, A. Giovanardi, and M. Schaerf. Experimental analysis of the computational cost of evaluation quantified Boolean formulae. In *Proceedings of the AI*IA-97*, pages 207–218. Springer-Verlag, 1997. LNAI-1321.

[8] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proceedings of the 12th IJCAI*, pages 331–337. International Joint Conference on Artificial Intelligence, 1991.

[9] V. Chvatal and B. Reed. Mick gets some (the odds are on his side). In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 620–627. IEEE, 1992.

[10] S.A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computation*, pages 151–158, 1971.

[11] S.A. Cook and D.G. Mitchell. Finding hard instances of the satisfiability problem: A survey. In *Satisfiability Problem: Theory and Applications.* Dimacs Series in Discrete Mathamatics and Theoretical Computer Science, Volume 35, 1997.

[12] J.M. Crawford and L.D. Auton. Experimental Results on the Cross-Over Point in Satisfiability Problmes. In *Proceedings of AAAI 1993 Spring Symposium on AI and NP-Hard Problems*, 1993.

[13] J.M. Crawford and L.D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81:31–57, 1996.

[14] J.M. Crawford and A.D. Baker. Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems. In *Proceedings of 12th National Conference on AI*, pages 1092–1097. American Association for Artificial Intelligence, 1994.

[15] Marcello D'Agostino. Are tableaux an improvement on truth-tables? *Journal of Logic, Language, and Information*, to appear.

[16] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Comms. ACM*, 5:394–397, 1962.

[17] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, pages 201–215, 1960.

[18] R. Dechter and I. Rish. Directional resolution: The davis-putnam procedure, revisited. In *Proceedings of KR-94*, pages 134–145, 1994. A longer version is available from `http://www.ics.uci.edu/ irinar`.

[19] O. Dubois, P. Andre, Y. Boufkhad, and J. Carlier. SAT versus UNSAT. In *Proceedings of the Second DIMACS Challenge*, 1993.

[20] G.W. Ernst. A definition-driven theorem prover. In *Proceedings of the 3rd IJCAI*, pages 51–55. International Joint Conference on Artificial Intelligence, 1973.

[21] J. Frank. Learning short term weights for GSAT. In *Proceedings of the 14th IJCAI*. International Joint Conference on Artificial Intelligence, 1995.

[22] E. Friedgut. Sharp thresholds for graph properties and the k-SAT problem, 1998. Unpublished manuscript.

[23] A. Frieze and S. Suen. Analysis of two simple heuristics on a random instance of k-SAT. *Journal of Algorithms*, 20:312–355, 1996.

[24] D. Frost, I. Rish, and L. Vila. Summarizing CSP hardness with continuous probability distributions. In *Proceedings of the 14th National Conference on AI*, pages 327–333. American Association for Artificial Intelligence, 1997.

[25] Masayuki Fujita, John Slaney, and Frank Bennett. Automatic generation of some results in finite algebra. In *Proceedings of the 13th IJCAI*, pages 52–57. International Joint Conference on Artificial Intelligence, 1993.

[26] M.R. Garey and D.S. Johnson. *Computers and intractability : a guide to the theory of NP-completeness*. W.H. Freeman, 1979.

[27] I. Gent and T. Walsh. The Enigma of SAT Hill-climbing Procedures. Technical Report 605, Dept. of Artificial Intelligence, University of Edinburgh, 1992.

[28] I. Gent and T. Walsh. Towards an Understanding of Hill-climbing Procedures for SAT. In *Proceedings of the 11th National Conference on AI*. American Association for Artificial Intelligence, 1993.

[29] I.P. Gent. On the stupid algorithm for satisfiability. Technical report, Technical report APES-03-1998, 1998. available from http://www.cs.strath.ac.uk/ apes/reports/apes-03-1998.ps.gz.

[30] I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The constrainedness of search. In *Proceedings of AAAI-96*, pages 246–252, 1996.

[31] I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The scaling of search cost. In *Proceedings of the 14th National Conference on AI*, pages 315–320. American Association for Artificial Intelligence, 1997.

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

[33] I.P. Gent and T. Walsh. Easy problems are sometimes hard. *Artificial Intelligence*, pages 335–345, 1994.

[34] I.P. Gent and T. Walsh. The hardest random SAT problems. In *Proceedings of KI-94, Saarbrucken.*, 1994.

[35] I.P. Gent and T. Walsh. The SAT phase transition. In A G Cohn, editor, *Proceedings of 11th ECAI*, pages 105–109. John Wiley & Sons, 1994.

[36] I.P. Gent and T. Walsh. Unsatisfied variables in local search. Research Paper 721, Dept. of Artificial Intelligence, University of Edinburgh, 1994. Presented at AISB-95.

[37] I.P. Gent and T. Walsh. The satisfiability constraint gap. *Artificial Intelligence*, 81(1–2), 1996.

[38] I.P. Gent and T. Walsh. Beyond np: the qsat phase transition. Technical report, Technical report APES-05-1998, 1998. available from http://www.cs.strath.ac.uk/ apes/reports/apes-05-1998.ps.gz.

[39] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal k. In *Proceedings of the 13th International Conference on Automated Deduction (CADE-13)*. Springer Verlag, 1996.

[40] F. Giunchiglia and R. Sebastiani. A sat-based decision procedure for alc. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning - KR'96*, 1996.

[41] F. Giunchiglia and R. Sebastiani. A new method for testing decision procedures in modal logics. In *Proceedings of the 14th International Conference on Automated Deduction (CADE-14)*. Springer Verlag, 1997.

[42] A. Goerdt. A threshold for unsatisfiability. In I. Havel and V. Koubek, editors, *Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, pages 264–274. Springer Verlag, 1992.

[43] C. Gomes, B. Selman, and H. Kautz. Boosting combinatorial search through randomization. In *Proceedings of 15th National Conference on Artificial Intelligence*, pages 431–437. AAAI Press/The MIT Press, 1998.

[44] J. Gu. Efficient local search for very large-scale satisfiability problems. *SIGART Bulletin*, 3(1):8–12, 1992.

[45] P. Hayes. The naive physics manifesto. In D. Michie, editor, *Expert Systems in the Microelectronic Age*. Edinburgh University Press, 1979.

[46] T. Hogg. Quantum computing and phase transitions in combinatorial search. *Journal of Artificial Intelligence Research*, 4:91–128, 1996.

[47] J. N. Hooker. Resolution *vs.* cutting plane solution of inference problems: some computational experience. *Operations Research Letters*, 7(1):1–7, 1988.

[48] J. N. Hooker and C. Fedjki. Branch-and-cut solution of inference problems in propositional logic. *Annals of Mathematics and Artificial Intelligence*, 1:123–139, 1990.

[49] J.N. Hooker. Branch-and-cut solution of inference problems in propositional logic. *Annals of Mathematics and Artificial Intelligence*, 1:123–139, 1990.

[50] J.N. Hooker. Solving the incremental satisfiability problem. *Journal of Logic Programming*, 15:177–186, 1993.

[51] J.N. Hooker and V. Vinay. Branching Rules for Satisfiability. *Journal of Automated Reasoning*, 15:359–383, 1995.

[52] H. Hoos. *Stochastic Local Search - Methods, Models, Applications.* PhD thesis, TU Darmstadt, 1998. Available from `http://www.cs.ubc.ca/spider/hoos/publ-ai.html`.

[53] H. Hoos and T. Stutzle. Evaluating las vegas algorithms - pitfalls and remedies. In *Proceedings of 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, 1998. Available from `http://www.sis.pitt.edu/~dsl/uai98.html`.

[54] R. E. Jeroslow and J. Wang. Solving propositional satisfiability problems. *Annals of Mathematics and Artificial Intelligence*, 1:167–187, 1990.

[55] Y. Jiang, H. Kautz, and B. Selman. Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT. In *First International Joint Workshop on Artificial Intelligence and Operations Research*, 1995.

[56] K.A. De Jong and W.M. Spears. Using Genetic Algorithms to Solve NP-Complete Problems. In *Third International Conference on Genetic Algorithms*. Morgan Kaufmann, 1989.

[57] Paul Walton Purdom Jr. and Cynthia A. Brown. The pure literal rule and polynomial average time. *SIAM Journal of Computing*, 14(4):943–953, November 1985.

[58] A. Kamath, R. Motwani, K. Palem, and P. Spirakis. Tail bounds for occupancy and the satisfiability threshold conjecture. *Randomized Structure and Algorithms*, 7:59–80, 1995.

[59] A.P. Kamath, N.K. Karmarkar, K.G. Ramakrishnan, and M.G.C. Resende. An interior point approach to Boolean vector function synthesis. In *Proceedings of the 36th MSCAS*, 1993.

[60] K. Kask and R. Dechter. Gsat and local consistency. In *Proceedings of the 14th IJCAI*, pages 616–622. International Joint Conference on Artificial Intelligence, 1995.

[61] H. Kautz and B. Selman. Planning as Satisfiability. In *Proceedings of the 10th ECAI*, pages 359–363. European Conference on Artificial Intelligence, 1992.

[62] H. Kautz and B. Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the 13th National Conference on AI*, pages 1194–1201. American Association for Artificial Intelligence, 1996.

[63] H. Kautz and B. Selman. BLACKBOX: A new approach to the application of theorem proving to problem solving. In *Working notes of the Workshop on Planning as Combinatorial Search*, 1998. Held in conjunction with AIPS-98, Pittsburgh, PA, 1998.

[64] H. Kautz and B. Selman. The role of domain-specific knowledge in the planning as satisfiability framework. In *Proceedings of AIPS-98, Pittsburgh, PA*, 1998.

[65] S. Kirkpatrick and B. Selman. Criticial behaviour in the satisfiability of random boolean expressions. *Science*, 264:1297–1301, 1994.

[66] L.M. Kirousis, E. Kranakis, and D. Krizanc. Approximating the unsatisfiability threshold of random formulas. In *Proceedings of the 4th Annual European Symposium on Algorithms (ESA'96)*, pages 27–38, 1996.

[67] E. Koutsoupias and C.H. Papadimitriou. On the greedy algorithm for satisfiability. *Information Processing Letters*, 43:53–55, 1992.

[68] T. Larrabee. Test pattern generation using Boolean satisfiability. *IEEE Transactions on Computer-Aided Design*, pages 4–15, 1992.

[69] C.M. Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proceedings of the 15th IJCAI*, pages 366–371. International Joint Conference on Artificial Intelligence, 1997.

[70] R.J. Lipton. Using DNA to solve NP-complete problems. *Science*, 268:542–545, 1995.

[71] B. Mazure, L. Sais, and E. Gregoire. Tabu search for SAT. In *Proceedings of 14th National Conference on Artificial Intelligence*, pages 281–2858. American Association for Artificial Intelligence, AAAI Press/The MIT Press, 1997.

[72] D. McAllester, B. Selman, and H. Kautz. Evidence for invariants in local search. In *Proceedings of the 14th National Conference on AI*, pages 321–327. American Association for Artificial Intelligence, 1997.

[73] W. W. McCune. Otter users' guide 0.91, 1988. Maths and CS. Division, Argonne National Laboratory, Argonne, Illinois.

[74] P. Meseguer and T. Walsh. Interleaved and discrepancy based search. In *Proceedings of the 13th ECAI*. European Conference on Artificial Intelligence, Wiley, 1998.

[75] D. Mitchell, B. Selman, and H. Levesque. Hard and Easy Distributions of SAT Problems. In *Proceedings of the 10th National Conference on AI*, pages 459–465. American Association for Artificial Intelligence, 1992.

[76] P. Morris. The Breakout method for escaping from local minima. In *Proceedings of the 11th National Conference on AI*. American Association for Artificial Intelligence, 1993.

[77] P. Pandurang Nayak and B.C. Williams. Fast context switching in real-time propositional reasoning. In *Proceedings of the 14th National Conference on AI*. American Association for Artificial Intelligence, 1997.

[78] Jens Otten. On the Advantage of a Non-Clausal Davis-Putnam Procedure. Technical Report AIDA–97–01, FG Intellektik, FB Informatik, TH Darmstadt, January 1997.

[79] C.H. Papadimitriou. On selecting a satisfying truth assigment. In *Proceedings of the Conference on the Foundations of Computer Science*, pages 163–169, 1991.

[80] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[81] A.J. Parkes and J.P. Walser. Tuning local search for satisfiability testing. In *Proceedings of AAAI-96*, pages 356–362, 1996.

[82] R. Reiter and A. Mackworth. A logical framework for depiction and image interpretation. *Artificial Intelligence*, 42(2):125–155, 1989.

[83] R. Rodosek. A new approach on solving 3-satisfiability. In *Proceedings of the 3rd International Conference on Artificial Intelligence and Symbolic Mathematical Computation*, pages 197–212. Springer, LNCS 1138, 1996.

[84] B. Selman and H. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *Proceedings of the 13th IJCAI*. International Joint Conference on Artificial Intelligence, 1993.

[85] B. Selman and H. Kautz. An empirical study of greedy local search for satisfiability testing. In *Proceedings of the 11th National Conference on AI*, pages 46–51. American Association for Artificial Intelligence, 1993.

[86] 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.

[87] B. Selman and S. Kirkpatrick. Critical behavior in the computational cost of satisfiability testing. *Artificial Intelligence*, 81:273–295, 1996.

[88] B. Selman, H. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In *Proceedings of the 10th National Conference on AI*, pages 440–446. American Association for Artificial Intelligence, 1992.

[89] J. Slaney, M. Fujita, and M. Stickel. Automated reasoning and exhaustive search: quasigroup existence problems. *Computers and Mathematics with Apllications*, 29:115–132, 1995.

[90] Raymond M Smullyan. *First Order Logic.* Springer-Verlag, Berlin, 1968.

[91] W.M. Spears. A nn algorithm for boolean satisfiability problems. In *Proceedings of the 1996 International Conference on Neural Networks*, pages 1121–1126, 1996.

[92] W.M. Spears. Simulated annealing for hard satisfiability problems. In D.S. Johnson and M.A. Trick, editors, *In Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, pages 533–558. 1996. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 26, American Mathematical Society.

[93] T.E. Uribe and M.E. Stickel. Ordered binary decision diagrams and the davis-putnam procedure. In *Proceedings of the First International Conference on Constraints in Computational Logics, Munich, Germany*, pages 34–49, 1994.

[94] A. van Gelder and Y.K. Tsuji. Satisfiability testing with more reasoning and less guessing. In D.S. Johnson and M.A. Trick, editors, *In Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge.* 1996. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 26, American Mathematical Society.

[95] J.P. Walser. Solving linear pseudo-boolean constraints with local search. In *Proceedings of the 14th National Conference on AI*, pages 269–274. American Association for Artificial Intelligence, 1997.

[96] B.C. Willians and P. Pandurang Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of the 13th National Conference on AI.* American Association for Artificial Intelligence, 1996.

[97] N. Yugami. Theoretical analysis of Davis-Putnam procedure and propositional satisfiability. In *Proceedings of IJCAI-95*, pages 282–288, 1995.

[98] H. Zhang. Sato: An efficient propositional prover. In *Proceedings of 14th Conference on Automated Deduction*, pages 272–275, 1997.

[99] H. Zhang. Specifying Latin squares in propositional logic. In R. Veroff, editor, *Automated Reasoning and Its Applications, Essays in honor of Larry Wos*, chapter 6. MIT Press, 1997.

[100] H. Zhang, M.P. Bonacina, and J. Hsiang. Psato: A distributed propositional prover and its application to quasigroup problems. *Journal of Symbolic Computation*, 11:1–18, 1996.

[101] H. Zhang and Stickel M. Implementing the Davis-Putnam Algorithm by Tries. Technical report, University of Iowa, 1994.

[102] H. Zhang and M.E. Stickel. An efficient algorithm for unit propagation. In *Working Notes of the Fourth International Symposium on Artificial Intelligence and Mathematics, Fort Lauderdale, Florida*, 1996.

[103] J. Zhang and H. Zhang. SEM: a System for Enumerating Models. In *Proceedings of IJCAI-95*, volume 1, pages 298–303. International Joint Conference on Artificial Intelligence, 1995.

[104] J. Zhang and H. Zhang. Combining local search and backtracking techniques for constraint satisfaction. In *Proceedings of 13th National Conference on Artificial Intelligence*, pages 369–374. American Association for Artificial Intelligence, AAAI Press/The MIT Press, 1996.