Vol.14 No.1 J. Comput. Sci. & Technol. Jan. 1999

On Optimizing the Satisfiability (SAT) Problem*

GU Jun (顾 钧), GU Qianping† and DU Dingzhu (堵丁柱)‡

Department of Electrical and Computer Engineering, University of Calgary Calgary, Canada T2N 1N4

Department of Computer Science, Hong Kong University of Science and Technology Clear Water Bay, Kowloon, Hong Kong

†*Department of Computer Software, The University of Aizu, Aizu-Wakamatsu Fukushima, 965-80 Japan*

‡*Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, U.S.A.*

E-mail: j.gu@computer.org

Received October 8, 1998.

Abstract The satisfiability (SAT) problem is a basic problem in computing theory. Presently, an active area of research on SAT problem is to design efficient optimization algorithms for finding a solution for a satisfiable *CNF* formula. A new formulation, the *Universal SAT* problem model, which transforms the SAT problem on Boolean space into an optimization problem on real space has been developed. Many optimization techniques, such as the steepest descent method, Newton's method, and the coordinate descent method, can be used to solve the *Universal SAT* problem. In this paper, we prove that, when the initial solution is sufficiently close to the optimal solution, the steepest descent method has a linear convergence ratio $\beta < 1$, Newton's method has a convergence ratio of order two, and the convergence ratio of the coordinate descent method is approximately $(1-\beta/m)$ for the *Universal SAT* problem with m variables. An algorithm based on the coordinate descent method for the *Universal SAT* problem is also presented in this paper.

Keywords satisfiability problem, optimization algorithm, nonlinear programming, convergence ratio, time complexity

1 Introduction

The satisfiability (SAT) problem is to determine whether there exists an assignment of values in $\{0,1\}$ to a set of Boolean variables $\{x_1,\ldots,x_m\}$ that makes a *conjunctive normal form* (CNF) formula *true*. The satisfiability problem of a CNF formula with at most l literals in each clause is called the l-SAT problem.

Theoretically, for $l \geq 3$, the l-SAT problem is a well-known NP-complete problem. And thus, there exists no polynomial time algorithm for the SAT problem on the assumption that $P \neq NP$. On the other hand, the SAT problem is fundamental in solving many practical problems in logic programming, inference, machine learning, and constraint satisfaction. Many practical algorithms and approaches have been developed to solve the SAT problem[1−6].

Among many algorithms and techniques proposed, the Davis-Putnam algorithm[7], in essence a resolution procedure, has been a major practical method for solving the SAT problem. The Davis-Putnam algorithm is able to determine satisfiability as well as unsatisfiability. However it is not efficient enough to handle a large size problem. If the SAT

This work was supported in part by NSERC Strategic Grant MEF0045793 and NSERC Research Grant OGP0046423.

*A part of an early version of this paper was published in IEEE Transactions on Computers, Vol.45, No.2, 1996.

problem is restricted to the case of finding a solution for a satisfiable formula, the problem can be solved more efficiently in practice. Based on a local search strategy, previously several families of simple local search algorithms have been developed for finding solutions of satisfiable CNF formulas[8]. These algorithms have a polynomial average run time for $l \geq 3$ and $n/m = O(2^l/l)$ for the randomly generated CNF formulas with n clauses, m variables, and l literals in each clause[1]. It has been shown that the $SAT1$ algorithms are more efficient than the Davis-Putnam algorithm in finding solutions of satisfiable CNF formulas[9,10]. Presently, to design an efficient algorithm for finding solutions of satisfiable CNF formulas has become an active research area[8,9,11].

Most algorithms for the SAT problem developed so far solve the problem on the Boolean space. Recently many *Universal SAT* problem models that transform the SAT problem into an optimization problem on the real space have been developed[8,9,12]. Many optimization techniques, such as the steepest descent method, Newton's method, and the coordinate descent method can be used to solve the *UniSAT7* problem. In this paper, the convergence ratios of three basic optimization methods for the *UniSAT7* problem are given. We prove that, when the initial solution is sufficiently close to the optimal solution, the steepest descent method has a linear convergence ratio $\beta < 1$, Newton's method has an order two convergence ratio, and the convergence ratio of the coordinate descent method is approximately $(1-\beta/m)$ for the *UniSAT7* problem with m variables.

Many optimization algorithms for the *UniSAT7* problem were developed[8,9,12]. In this paper, based on a coordinate descent method, we describe a formal version of the $SAT14.7$ algorithm for the *UniSAT7* problem. The experimental results show that the $SAT14.7$ algorithm is much more efficient than the Davis-Putnam algorithm[13,14].

The *UniSAT7* problem model that transfers the SAT problem from Boolean space into a space of real numbers gives a new approach to the SAT problem. It is expected to have numerous practical applications.

The rest of this paper is organized as follows. In the next section, we will briefly overview the previous work in the area. Section 3 describes the *UniSAT7* problem model. In Section 4, we analyze the convergence ratios of the steepest descent method, Newton's method, and the coordinate descent method for the *UniSAT7* problem. The $SAT14.7$ algorithm is described in Section 5. Finally, Section 6 concludes this paper.

2 Previous Work

The existing SAT algorithms can be grouped into the following several classes[14]. Most existing SAT algorithms can be grouped into these categories.

- *Discrete, constrained algorithms.* Algorithms in this category treat an SAT formula as an instance of a constrained decision problem, applying discrete search and inference procedures to determine a solution. One straightforward way to solve an instance of SAT is to enumerate all possible truth assignments and check to see if one satisfies the formula. Many improved techniques, such as consistency algorithms[15], backtracking algorithms[16–20] term-rewriting[1,2], production system[21], multi-valued logic[3], Binary Decision Diagrams[22,23], chip and conquer[24], resolution and regular resolution[5,6,25,31,57–59], independent set algorithm[60], and matrix inequality system[38] have been proposed.

Other specific algorithms using these principles include simplified DP algorithms[61–63], and a simplified DP algorithm with strict ordering of variables[64]. The DP algorithm improved in certain aspects over Gilmore's proof method[65]. Analyses of SAT algorithms often concentrate on algorithms that are simple because it is difficult to do a correct analysis of the best algorithms. Under

[1]In this paper, a quantity $f(n)$ is said to be $O(g(n))$ if $\lim_{n\to\infty} f(n)/g(n) \geq 0$. A quantity $h(n)$ is said to be $o(g(n))$ if $\lim_{n\to\infty} f(n)/g(n) = 0$.

those conditions where simple algorithms are fast, related practical algorithms are also fast. (It is difficult to tell whether a practical algorithm is slow under conditions that make the corresponding simplified algorithm slow.)

- Discrete
  - Constrained
    - 1960: Davis-Putnam (DP) algorithm[7]
    - 1965: Resolution[25]
    - 1971: Consistency algorithms[26–29]
    - 1978: Loveland's Davis-Putnam (DPL)[30,31]
    - 1986: Parallel consistency chips[27,32,33]
    - 1986: Binary decision diagrams DD)[22,23]
    - 1988: Chip and conquer[24]
    - 1990: DPL plus heuristic (DPLH)[34]
    - 1989: Local search & backtracking[35]
    - 1993: Backtracking and probing[36]
    - 1994: Parallel DP algorithm[37]
    - 1994: Matrix inequality system[38]
    - 1996: *CSAT*[39]
  - Unconstrained
    - 1987: Randomized local search (SAT1)[8,35]
    - 1987: Parallel local search (SAT1.6)[8,35]
    - 1988: Local search for n-queen[27,40,41]
    - 1990: *Unison* algorithm and hardware[42,43]
    - 1991: Local search complexity[10,44]
    - 1991: Local search for 2-SAT[45]
    - 1992: Local search with traps (SAT1.5)[9,35]
    - 1992: Greedy local search — GSAT[46]
- Continuous
  - Constrained
    - 1986: Branch-and-bound (APEX)[47]
    - 1988: Programming models[47,48]
    - 1988: Cutting plane[49,50]
    - 1989: Branch-and-cut[51]
    - 1989: Interior point method[52,53]
  - Unconstrained
    - 1987: *UniSAT* models[8,13,54]
    - 1987: Nonlinear optimization (SAT6)[8,13]
    - 1989: Neural net models[55,56]
    - 1990: Nonlinear optimization & backtracking[13]
    - 1991: SAT14 algorithms[13]

Fig.1. Some typical algorithms for the SAT problem.

A number of special SAT problems, such as 2-satisfiability and Horn clauses, are *solvable* in polynomial time[5,66,67]. There are several linear time algorithms[68,69] and polynomial time algorithms[45,70] existing.

• *Discrete, unconstrained algorithms*. In this approach, the number of unsatisfiable *CNF* (or satisfiable *DNF*) clauses is formulated as the value of the objective function, transforming the SAT formula into a discrete, unconstrained minimization problem to the objective function. Local search is a major class of discrete, unconstrained search methods[9,35,44,46]. It can be used to solve the transformed formula.

Early work in constraint satisfaction and complexity study contributed to the development of local search algorithms for the SAT problem[14]. There were two major approaches in this area: randomized local search (*SAT1*) and greedy local search (*GSAT*). The *SAT1* algorithm was the first local search algorithm developed from the VLSI engineering and scheduling applications. The

GSAT algorithm was derived from the early local search algorithms for the n-queen problem.

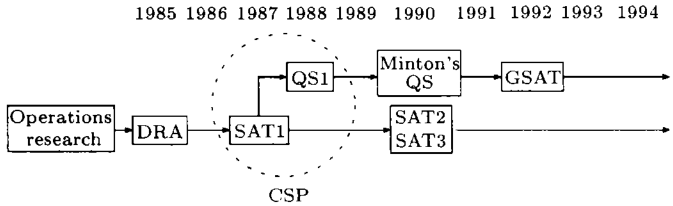

Fig.2. Early development of local search algorithms for SAT problem.

• *Constrained programming algorithms.* Methods in this class were developed based on the fact that *CNF* or *DNF* formulas can be transformed to instances of Integer Programming, and possibly solved using Linear Programming relaxations[47,48,50,52,53,71–73]. Many approaches, including branch-and-bound[47], cutting-plane[49,50], branch-and-cut[51], interior-point[52,53], and improved interior-point[74], have been proposed to solve the integer program representing the inference problem. Researchers found integer programming methods faster than resolution for certain classes of problems, although these methods do not possess a robust convergence property and often fail to solve hard instances of satisfiability[47,48,50,52,53,71].

• *Unconstrained, nonlinear optimization algorithms.* Special models have been formulated to transform a discrete formula on Boolean space $\{0,1\}^n$ (a decision problem) into an unconstrained *UniSAT* problem on real space E^n (an unconstrained nonlinear optimization problem). The transformed formulas can be solved by many existing nonlinear optimization methods[8,9,13,54].

In practice, most sequential SAT algorithms can be mapped onto parallel computer systems, resulting in parallel SAT algorithms[14]. Accordingly, as given in Fig.3, there are *four* classes of parallel algorithms for solving SAT.

- Discrete
  - Constrained
    - 1983: Parallel CLP algorithms[75,76]
    - 1986: Parallel DRA chips[27,32,33]
    - 1987: Parallel DP algorithm[78]
    - 1988: Parallel AC algorithms[77]
    - 1988: Parallel CSP architectures[27,32]
    - 1990: *Unison* algorithm and hardware[42,43]
    - 1992: Vectorized DP algorithm[79]
    - 1994: MIMD DP algorithm[37]
  - Unconstrained
    - 1987: *CNF* local search[8,35]
    - 1987: *DNF* local search[8,13]
    - 1987: Parallel local search[8,35]
    - 1991: Discrete $\alpha\beta$ relaxation[86]
    - 1993: Multiprocessor local search[85]
- Continuous
  - Constrained
    - 1989. Interior point method[52,53]
  - Unconstrained
    - 1987: *UniSAT* models[8,13]
    - 1987: Nonlinear optimization (SAT6)[8,13]
    - 1991: Continuous $\alpha\beta$ relaxation[86]
    - 1991: SAT14 algorithms[13]
    - 1991: Parallel nonlinear optimization[13,54]
    - 1992: Neurocomputing[14]

Fig.3. Some parallel SAT/CSP algorithms.

• *Parallel, discrete, constrained algorithms.* Many discrete, constrained SAT and CSP algorithms have been implemented in parallel algorithms or put on special-purpose, hardware VLSI architectures. These include parallel consistent labeling algorithms[75,76], parallel discrete relaxation (DRA) chips[27,32,33], parallel arc consistency (PAC) algorithms[77], parallel constrained search architectures[27,32], parallel *Unison* algorithms[42], parallel *Unison* architectures[43], parallel DP algorithms[37,78,79], and parallel logical programming languages[80−84].

- *Parallel, discrete, unconstrained algorithms.* A number of discrete local optimization algorithms were implemented on parallel computing machines. These include *CNF* local search[8,35], *DNF* local search[8,13], parallel local search[8,35], and multiprocessor local search[85]. A new $\alpha\beta$ relaxation technique was developed in a parallel and distributed environment[86].
- *Parallel, constrained programming algorithms.* Kamath *et al.* implemented an interior point zero-one integer programming algorithm on a *KORBX(R)* parallel/vector computer[52,53].
- *Parallel, unconstrained, nonlinear optimization algorithms.* Several of these algorithms have been implemented: *UniSAT* models[8,13], parallel, continuous $\alpha\beta$ relaxation[86], and parallel nonlinear optimization algorithms[13,54].

3 *UniSAT7:* A Universal SAT Problem Model

A *CNF* formula F is a logical *and* of n clauses, $C_1 \wedge C_2 \wedge \ldots \wedge C_n$. A clause C_i is a logical *or* of literals, i.e., $Q_1 \vee \ldots \vee Q_l$. A literal Q_j is either a Boolean variable x or the negation of the variable, $\bar{x}$. The satisfiability problem (SAT) is to determine whether there exists an assignment of values in $\{0,1\}$ to a set of Boolean variables $\{x_1,\ldots,x_m\}$ that makes a given *CNF* formula F satisfiable (true)[67,87]. The SAT problem of a *CNF* formula with at most l literals in each clause is called the l-SAT problem.

Let $x_1,\ldots,x_m$ be Boolean variables and $\mathbf{x}$ be the vector $(x_1,\ldots,x_m)$ in $\{0,1\}^m$. Let $y_1,\ldots,y_m$ be real variables and $\mathbf{y}$ be the vector $(y_1,\ldots,y_m)$ in an m-dimensional real space, E^m. We now decribe one formulation for the SAT problem on E^m, the *UniSAT7* problem model[8,9,12,54]. Given a *CNF* formula $F(\mathbf{x})$ from $\{0,1\}^m$ to $\{0,1\}$ with n clauses $C_1,\ldots,C_n$, an objective function $f_1(\mathbf{y})$ from E^m to E is defined as a sum of n *clause functions* $c_i(\mathbf{y})$ $(1 \le i \le n)$:

$$f_1(\mathbf{y}) = \sum_{i=1}^{n} c_i(\mathbf{y}). \tag{1}$$

A clause function $c_i(\mathbf{y})$ is a product of m *literal functions* $q_{ij}(y_j)$ $(1 \le j \le m)$:

$$c_i = \prod_{j=1}^{m} q_{ij}(y_j), \tag{2}$$

where

$$q_{ij}(y_j) = \begin{cases} (y_j-1)^2, & \text{if } x_j \text{ is in clause } C_i \\ (y_j+1)^2, & \text{if } \bar{x_j} \text{ is in clause } C_i \\ 1, & \text{if neither } x_j \text{ nor } \bar{x_j} \text{ is in clause } C_i \end{cases} \tag{3}$$

The correspondence between $\mathbf{x}$ and $\mathbf{y}$ is defined as follows (for $1 \le i \le m$):

$$x_i = \begin{cases} 1, & \text{if } y_i = 1 \\ 0, & \text{if } y_i = -1 \\ undefined, & \text{otherwise} \end{cases}$$

Clearly, $F(\mathbf{x})$ is true if and only if $f_1(\mathbf{y})$ is the global minimum value 0 on the corresponding $\mathbf{y} \in \{-1,1\}^m$.

The *UniSAT7* problem model transforms the SAT problem on the Boolean space into an optimization problem on the real space of E^m. The *UniSAT7* problem is to find a vector $\mathbf{y}$ in E^m such that the corresponding vector $\mathbf{x}$ in $\{0,1\}^m$ satisfies the given *CNF* Boolean formula.

Any numerical optimization method can be used to solve the *UniSAT7* problem. We will analyze in the next section the convergence ratio and the efficiency of three basic optimization methods: the steepest descent method, Newton's method, and the coordinate descent method[88], for the *UniSAT7* problem.

To ensure the theoretical convergence ratios, instead of f_1, the object function

$$f(\mathbf{y}) = f_1(\mathbf{y}) + f_2(\mathbf{y}) \tag{4}$$

will be considered, where

$$f_2(\mathbf{y}) = \sum_{j=1}^{m}(y_j - 1)^2(y_j + 1)^2. \tag{5}$$

For all $\mathbf{y} \in \{-1,1\}^m$, clearly $f_2(\mathbf{y})=0$.

From this, for any $\mathbf{y} \in \{-1,1\}^m$, $f(\mathbf{y})=0$ if and only if $f_1(\mathbf{y})=0$. Thus, $F(\mathbf{x})$ is true if and only if $f(\mathbf{y})=0$ on the corresponding point $\mathbf{y} \in \{-1,1\}^m$. Given a *CNF* formula F, we will call f an object function of F and a vector $\mathbf{y} \in \{-1,1\}^m$ with $f(\mathbf{y})=0$ a solution of f.

For the object function f and the *UniSAT7* problem, we have the following theorem.

Theorem 1. *Let f be an object function of a CNF formula F. For any $\mathbf{y} \in E^m$ with $f(\mathbf{y}) < 1$, we can find a vector $\mathbf{y}^* \in \{-1,1\}^m$ such that $f(\mathbf{y}^*) = 0$, i.e., we can find a solution of the formula F.*

Proof. Let $\mathbf{y}$ be a vector in E^m such that $f(\mathbf{y}) < 1$. Then from $f = f_1 + f_2$, we have $f_1(\mathbf{y}) < 1$ and $f_2(\mathbf{y}) < 1$. Then we have for each *clause function* c_i $(1 \le i \le n)$ of f_1 defined in (3), $c_i(\mathbf{y}) < 1$. Therefore, for each clause function c_i $(1 \le i \le n)$, there exists a *literal function* q_{ij} in c_i defined in (3) such that $q_{ij}(\mathbf{y}) < 1$. Define a *round-off operation* as follows:

$$y_j^* = \begin{cases} 1, & \text{if } y_j \ge 0 \\ -1, & \text{if } y_j < 0 \end{cases}$$

Let $\mathbf{y}^*$ be the vector obtained from the round-off operation on $\mathbf{y}$. Then clearly, the literal function $q_{ij}(\mathbf{y}^*) = 0$. This implies that the clause function $c_i(\mathbf{y}^*) = 0$ and thus $f_1(\mathbf{y}^*) = 0$.

For each clause function $(y_j - 1)^2(y_j + 1)^2$ $(1 \le j \le m)$ in f_2, $f_2 < 1$ implies that either $(y_j - 1)^2 < 1$ or $(y_j + 1)^2 < 1$. Therefore, for each clause function $(y_j - 1)^2(y_j + 1)^2$ $(1 \le j \le m)$ in f_2, by the round-off operation, we have $(y_j^* - 1)^2(y_j^* + 1)^2 = 0$. Thus, we have $f_2(\mathbf{y}^*) = 0$. Combining this and $f_1(\mathbf{y}^*) = 0$, $\mathbf{y}^*$ is a solution of f. □

From the above theorem, the optimization process for solving the *UniSAT7* problem can be stopped when a vector $\mathbf{y} \in E^m$ with $f(\mathbf{y}) < 1$ is found.

The following definitions will be used in deriving the convergence ratios of the steepest descent method, Newton's method, and the coordinate descent method for the *UniSAT7* problem.

For an object function $f(\mathbf{y})=f(y_1, \ldots, y_m)$, we define the *gradient* of f to be the vector

$$\nabla f(\mathbf{y}) = \Big(\frac{\partial f(\mathbf{y})}{\partial y_1}, \ldots, \frac{\partial f(\mathbf{y})}{\partial y_m}\Big).$$

In matrix calculations the *gradient* is considered to be a row vector.

For $f(\mathbf{y})$, we define the *Hessian* of f at $\mathbf{y}$ be the $m \times m$ matrix denoted $\mathbf{H}(\mathbf{y})$ as

$$\mathbf{H}(\mathbf{y}) = \Big[\frac{\partial^2 f(\mathbf{y})}{\partial y_i \partial y_j}\Big].$$

For function f defined in (4) and (5), clearly, f has the continuous first and second derivatives, and

$$\frac{\partial^2 f}{\partial y_i \partial y_j} = \frac{\partial^2 f}{\partial y_j \partial y_i}.$$

Therefore, the *Hessian* of f is a real symmetric matrix.

In the following of the paper, $\mathbf{y}$ will denote a row vector and $\mathbf{y}^T$ will denote a column vector.

4 Convergence Ratios

The *UniSAT7* problem model transforms the SAT problem into an unconstrained optimization problem on the real space of E^m. Many nonlinear programming techniques can be used to optimize the object function f. In this section, we analyze the convergence ratio and efficiency of three basic methods: the steepest descent method, Newton's method, and the coordinate descent method, for the object function f defined in (4) and (5). We do not describe these methods here. They can be found in most nonlinear programming text books[88].

The main result of this section is that for any Boolean CNF formula F, if $\mathbf{y}^*$ is a solution point of the object function f defined in (4) and (5), then the *Hessian* matrix $\mathbf{H}(\mathbf{y}^*)$ of f at $\mathbf{y}^*$ is positive definite. From this result, the convergence ratios of the three optimization methods can be derived[89].

Definition 2[90]**.** *An $m \times m$ real symmetric matrix $\mathbf{H}$ is positive definite if and only if for all nonzero vector $\mathbf{d}$ in E^m, $\mathbf{d}{\cdot}\mathbf{H}{\cdot}\mathbf{d}^T > 0$. Or equivalently, $\mathbf{H}$ is positive definite if and only if all the eigenvalues of $\mathbf{H}$ are larger than zero.*

Theorem 3. *Let $\mathbf{y}^* \in \{-1,1\}^m$ be a solution point of f. Then the Hessian matrix $\mathbf{H}(\mathbf{y}^*)$ of f is positive definite.*

Proof. Let $\mathbf{y}^* \in \{-1,1\}^m$ be a solution of f. Since the *Hessian* matrix $\mathbf{H}(\mathbf{y}^*)$ of f is a real symmetric matrix, by Definition 2, $\mathbf{H}(\mathbf{y}^*)$ is positive definite if and only if $\mathbf{d}{\cdot}\mathbf{H}(\mathbf{y}^*){\cdot}\mathbf{d}^T > 0$ for any non-zero vector $\mathbf{d} = (d_1, d_2, \ldots, d_m)$ in E^m.

Let $\mathbf{d} = (d_1, \ldots, d_m)$ be an arbitrary non-zero vector in E^m,

$$\mathbf{y}(\alpha) = \mathbf{y}^* + \alpha\mathbf{d} = (y_1^* + \alpha d_1, \ldots, y_m^* + \alpha d_m),$$

and

$$g(\alpha) = f(\mathbf{y}(\alpha)) = f(\mathbf{y}^* + \alpha\mathbf{d}) = f(y_1^* + \alpha d_1, \ldots, y_m^* + \alpha d_m).$$

By Taylor's theorem, we have

$$g(\alpha) = g(0) + \frac{dg(\alpha)}{d\alpha}\Big|_{\alpha=0} \times \alpha + \frac{1}{2}\frac{d^2 g(\alpha)}{d\alpha^2}\Big|_{\alpha=0} \times \alpha^2 + o(\alpha^2).$$

From this, we have

$$g(\alpha) = g(0) + g'(0)\alpha + \frac{1}{2}g''(0)\alpha^2 + o(\alpha^2), \tag{6}$$

$$\begin{aligned} g'(0) = &\frac{dg(\alpha)}{d\alpha}\Big|_{\alpha=0} = \Big(\frac{\partial f}{\partial y_1}\frac{\partial y_1}{\partial \alpha} + \ldots + \frac{\partial f}{\partial y_m}\frac{\partial y_m}{\partial \alpha}\Big)_{\alpha=0} \\ =&\Big(\frac{\partial f}{\partial y_1}d_1 + \ldots + \frac{\partial f}{\partial y_m}d_m\Big)_{(y_1,\ldots,y_m)=(y_1^*,\ldots,y_m^*)} = \nabla f(\mathbf{y}^*) \cdot \mathbf{d}^T \end{aligned} \tag{7}$$

and

$$g''(0) = \frac{d^2 g(\alpha)}{d\alpha^2}\Big|_{\alpha=0} = \Big(\sum_{i=1}^{m}\sum_{j=1}^{m}\frac{\partial^2 f}{\partial y_i \partial y_j}\frac{\partial y_i}{\partial \alpha}\frac{\partial y_j}{\partial \alpha}\Big)_{\alpha=0}$$

$$= \Big(\sum_{i=1}^{m}\sum_{j=1}^{m}\frac{\partial^2 f}{\partial y_i \partial y_j} d_i d_j\Big)_{(y_1,\ldots,y_m)=(y_1^*,\ldots,y_m^*)} = \mathbf{d}\cdot\mathbf{H}(\mathbf{y}^*)\cdot\mathbf{d}^T. \tag{8}$$

Since f has the global minimum value 0 at the solution point $\mathbf{y}^*$,

$$\nabla f(\mathbf{y}^*) = \Big(\frac{\partial f}{\partial y_1},\ldots,\frac{\partial f}{\partial y_m}\Big)_{(y_1,\ldots,y_m)=(y_1^*,\ldots,y_m^*)} = (0,\ldots,0).$$

From this and (7), we have $g'(0) = 0$ for any $\mathbf{d}\in E^m$. Therefore, from (6), (8), and $g(0) = f(\mathbf{y}^*)=0$, we have

$$g(\alpha) = \frac{1}{2}g''(0)\alpha^2 + o(\alpha^2) = \frac{1}{2}\mathbf{d}\cdot\mathbf{H}(\mathbf{y}^*)\cdot\mathbf{d}^T\alpha^2 + o(\alpha^2). \tag{9}$$

On the other hand,

$$g(\alpha) = f(\mathbf{y}^* + \alpha\mathbf{d}) = f_1(\mathbf{y}^* + \alpha\mathbf{d}) + f_2(\mathbf{y}^* + \alpha\mathbf{d}). \tag{10}$$

Clearly, from (1) and (3), we have $f_1(\mathbf{y}^* + \alpha\mathbf{d}) \geq 0$. Now we calculate $f_2(\mathbf{y}^* + \alpha\mathbf{d})$. For $y_j^* = 1$,

$$(y_j^* - 1 + \alpha d_j)^2(y_j^* + 1 + \alpha d_j)^2 = (\alpha d_j)^2(2 + \alpha d_j)^2 = (2\alpha d_j)^2 + o(\alpha^2),$$

and for $y_j^* = -1$,

$$(y_j^* - 1 + \alpha d_j)^2(y_j^* + 1 + \alpha d_j)^2 = (-2 + \alpha d_j)^2(\alpha d_j)^2 = (2\alpha d_j)^2 + o(\alpha^2).$$

Therefore,

$$f_2(\mathbf{y}^* + \alpha\mathbf{d}) = \sum_{j=1}^{m}(y_j^* - 1 + \alpha d_j)^2(y_j^* + 1 + \alpha d_j)^2 = \sum_{j=1}^{m}(2\alpha d_j)^2 + o(\alpha^2)$$

$$= 4(d_1^2 + \ldots + d_m^2)\alpha^2 + o(\alpha^2). \tag{11}$$

From (10) and (11), we have

$$g(\alpha) = f(\mathbf{y}^* + \alpha\mathbf{d}) = f_1(\mathbf{y}^* + \alpha\mathbf{d}) + 4(d_1^2 + d_2^2 + \ldots + d_m^2)\alpha^2 + o(\alpha^2). \tag{12}$$

From (9) and (12), we have

$$f_1(\mathbf{y}^* + \alpha\mathbf{d}) + 4(d_1^2 + \ldots + d_m^2)\alpha^2 + o(\alpha^2) = \frac{1}{2}\mathbf{d}\cdot\mathbf{H}(\mathbf{y}^*)\cdot\mathbf{d}^T\alpha^2 + o(\alpha^2).$$

Since $f_1(\mathbf{y}^* + \alpha\mathbf{d})\geq 0$, α can be arbitrarily small, and for any non-zero vector $\mathbf{d}$, $(d_1^2 + \ldots + d_m^2) > 0$, the above equation holds if and only if $\mathbf{d}\cdot\mathbf{H}(\mathbf{y}^*)\cdot\mathbf{d}^T > 0$. Thus, from Definition 2, at any solution point $\mathbf{y}^*$, $\mathbf{H}(\mathbf{y}^*)$ is positive definite. □

Now we give the convergence ratios of the steepest descent method, Newton's method, and the coordinate descent method for the *UniSAT7* problem.

Definition 4[88]. *Let the sequence $\{r_k\}$ converge to r. The order of convergence of $\{r_k\}$ is defined as the supremum of the nonnegative numbers p satisfying*

$$0 \le \overline{\lim_{k\to\infty}} \frac{|r_{k+1}-r|}{|r_k-r|^p} < \infty.$$

Definition 5[88]. *If a sequence $\{r_k\}$ converges to r in such a way that*

$$\lim_{k\to\infty} \frac{|r_{k+1}-r|}{|r_k-r|} = \beta < 1,$$

the sequence $\{r_k\}$ is said to converge linearly to r with convergence ratio β.

Proposition 6[88]. *Suppose f has second partial derivatives which are continuous on E^m. Suppose further that at the local minimum point $\mathbf{y}^*$ the Hessian matrix of f, $\mathbf{H}(\mathbf{y}^*)$, is positive definite. If $\{\mathbf{y}_k\}$ is a sequence generated by the steepest descent method that converges to $\mathbf{y}^*$, then the sequence of objective values $\{f(\mathbf{y}_k)\}$ converges to $f(\mathbf{y}^*)$ linearly with a convergence ratio no greater than $[(A-a)/(A+a)]^2$, where $A \ge a > 0$ are the largest and smallest eigenvalues of the Hessian matrix $\mathbf{H}(\mathbf{y}^*)$, respectively.*

Proposition 7[88]. *Suppose f has third partial derivatives which are continuous on E^m. Suppose further that at the local minimum point $\mathbf{y}^*$ the Hessian matrix of f, $\mathbf{H}(\mathbf{y}^*)$, is positive definite. Then if started sufficiently close to $\mathbf{y}^*$, the points generated by Newton's method converge to $\mathbf{y}^*$. The order of convergence is at least two.*

Lemma 8. *Suppose f has second partial derivatives which are continuous on E^m. Suppose further that at the local minimum point $\mathbf{y}^*$ the Hessian matrix of f, $\mathbf{H}(\mathbf{y}^*)$, is positive definite. If started sufficiently close to $\mathbf{y}^*$ and $\{\mathbf{y}_k\}$ is a sequence generated by the coordinate descent method where at each stage the coordinate corresponding to the largest (in absolute value) component of the gradient vector is selected (the Gauss-Southwell Method*[88]*) that converges to $\mathbf{y}^*$, then the sequence of objective values $\{f(\mathbf{y}_k)\}$ converges to $f(\mathbf{y}^*)$ linearly with a convergence ratio no greater than $1-\frac{a}{A(m-1)}$, where $A \ge a > 0$ are the largest and smallest eigenvalues of the Hessian matrix $\mathbf{H}(\mathbf{y}^*)$, respectively.*

Proof. See Appendix. □

Theorem 9. *Let f be the function defined in (4) and (5). If $\{\mathbf{y}_k\}$ is a sequence of vectors generated by the steepest descent method that converges to a solution $\mathbf{y}^*$ of f, then the sequence of the objective values $\{f(\mathbf{y}_k)\}$ converges to $f(\mathbf{y}^*)$ linearly with a convergence ratio $[(A-a)/(A+a)]^2 < 1$, where $A \ge a > 0$ are the largest and smallest eigenvalues of the Hessian matrix $\mathbf{H}(\mathbf{y}^*)$ of f, respectively.*

Proof. Clearly, f has second partial derivatives which are continuous on E^m. Therefore, the theorem follows from Theorem 3 and Proposition 6. □

Theorem 10. *Let f be the function defined in (4) and (5). If started sufficiently close to a solution point $\mathbf{y}^*$, the sequence $\{\mathbf{y}_k\}$ generated by Newton's method converge to $\mathbf{y}^*$. The order of convergence is at least two.*

Proof. The theorem follows from Theorem 3 and Proposition 7. □

Theorem 11. *Let f be the function defined in (4) and (5). If started sufficiently close to $\mathbf{y}^*$ and $\{\mathbf{y}_k\}$ is a sequence generated by the coordinate descent method where at each stage the coordinate corresponding to the largest (in absolute value) component of the gradient vector is selected (the Gauss-Southwell Method*[88]*) that converges to a solution $\mathbf{y}^*$ of f, then the sequence of the objective values $\{f(\mathbf{y}_k)\}$ converges to $f(\mathbf{y}^*)$ linearly with a convergence ratio $\left(1-\frac{a}{A(m-1)}\right) < 1$, where $A \ge a > 0$ are the largest and smallest eigenvalues of the Hessian matrix $\mathbf{H}(\mathbf{y})$, respectively.*

Proof. The theorem follows from Theorem 3 and Lemma 8. □

From the convergence properties given above, we can roughly estimate the efficiency of the steepest descent method and the coordinate descent method for solving the *UniSAT7* problem.

Let $\{\mathbf{y}_k\}$ be a sequence of vectors generated by the steepest descent method that converge to a solution point $\mathbf{y}^*$ and let the initial value of the vector $\mathbf{y}_0$ to be $(0,\ldots,0)$. Then $f(\mathbf{y}_0) = n+m$.

From Theorem 9, we have that the convergence ratio of the steepest descent method for f is

$$\left(\frac{A-a}{A+a}\right)^2,$$

where $A \geq a > 0$ are the largest and smallest *eigenvalues* of the *Hessiah* matrix $\mathbf{H}(\mathbf{y}^*)$ of f, respectively.

From this, we have

$$f(\mathbf{y}_{k+1}) \leq \left(\frac{A-a}{A+a}\right)^2 f(\mathbf{y}_k), \tag{13}$$

for sufficiently large k.

From $A \geq a > 0$, clearly there exists a constant $\beta < 1$ such that

$$\left(\frac{A-a}{A+a}\right)^2 \leq \beta.$$

Therefore, if (13) holds for every $k \geq 1$, then for $k > -\log(m+n)/\log\beta$, we have

$$f(\mathbf{y}_k) \leq \beta^k(n+m) < \frac{1}{n+m}(n+m) = 1.$$

Thus, from Theorem 1, the *UniSAT7* problem can be solved in $O(\log(n+m))$ iterations by the steepest descent method on the assumption that (13) holds for every $k \geq 1$.

Let $\{\mathbf{y}_k\}$ be a sequence of vectors generated by the coordinate descent method where at each stage the coordinate corresponding to the largest (in absolute value) component of the gradient vector is selected (the Gauss-Southwell Method[88]) that converges to a solution point $\mathbf{y}^*$. Then by Theorem 11, we have

$$f(\mathbf{y}_{k+1}) \leq \left(1 - \frac{a}{A(m-1)}\right) f(\mathbf{y}_k), \tag{14}$$

for sufficiently large k. Since $A \geq a > 0$, clearly there exists a $\beta < 1$ such that,

$$\left(1 - \frac{a}{A(m-1)}\right)^m \leq \beta.$$

Therefore, if (14) holds for all $k \geq 1$, then initially choosing $\mathbf{y}_0 = (0,\ldots,0)$ and for $k > -m\log(m+n)/\log\beta$, we have

$$\begin{aligned} f(\mathbf{y}_k) \leq& \left(1 - \frac{a}{A(m-1)}\right)^k f(\mathbf{y}_0) < \left(1 - \frac{a}{A(m-1)}\right)^{m(-\log(m+n)/\log\beta)}(m+n) \\ \leq& \beta^{-\log(m+n)/\log\beta}(m+n) = \frac{1}{m+n}(m+n) = 1. \end{aligned}$$

From this and Theorem 1, the *UniSAT7* problem can be solved in $O(m\log(n+m)/\log\beta)$ iterations by the coordinate descent method on the assumption that (14) holds for all $k \geq 1$.

5 An Algorithm for the *UniSAT7* Problem

Many optimization algorithms for the *UniSAT7* problem were developed[8,9,12,13]. Based on a coordinate descent method[88], we now describe a formal version of the $SAT14.7$ algorithm for the *UniSAT7* problem on E^m (Fig.4). The kernel of the $SAT14.7$ algorithm is the *minimizer* which minimizes the object function by the coordinate descent method.

Since the computation of the *gradient* of f_1 is much easier than that of f, to optimize the function f_1 is more efficient than to optimize the function f for the *UniSAT7* problem in practice, though optimization on f may have better convergence ratio. Given the object function f_1 on E^m, the $SAT14.7$ algorithm initially chooses $\mathbf{y}$ from E^m and then the function f_1 is minimized with respect to each variables y_i $(1 \leq i \leq m)$ in the *minimizer* until $f_1 < 1$. Since for each clause function c_i $(1 \leq i \leq n)$ in f_1, each variable y_j $(1 \leq j \leq m)$ appears in c_i at most once, f_1 is a quadratic function with respect to y_j. Thus, minimizing f_1 with respect to one variable can be done in $O(nl)$ time. When $f_1 < 1$ then a *round_off* operation defined in Section 3 is performed to find the solution.

In practice, before $f_1 < 1$, the algorithm could be stuck at a local minimum point. To overcome this problem, a *local handler* is added in the $SAT14.7$ algorithm. In the local handler, a new initial vector $\mathbf{y}$ is generated.

```
Procedure SAT14.7 ()
begin
    /* initialization */
    y:=initial_vector();
    local := 0; limit :=poly(m);
    /* search */
    while (f1(y)≥ 1 and local ≤ limit) do
    begin
        old_f := f1(y);
        /* minimizer */
        for i := 1 to m do
            minimize f1(y) with respect to yi;
        /* local handler */
        if f1(y)≥ old_f then
            begin y:=initial_vector(); local := local + 1 end;
    end;
    if f1(y) < 1 then y*:=round_off(y) else y*:=enumerate();
end.
```

Fig.4. **SAT14.7:** An optimization algorithm for the SAT problem on the real space E^m.

The run time of the $SAT14.7$ algorithm can be estimated as follows. The initial portion and the computation of $f_1(\mathbf{y})$ take $O(ln)$ time. In one iteration of the *while* loop, minimizing $f_1(\mathbf{y})$ with respect to one variable can be computed in $O(ln)$ time, and thus, the *minimizer* takes $O(lmn)$ time. Clearly, one execution of the local handler takes $O(m)$ time. Summarizing the above, the run time of the $SAT14.7$ algorithm is $O(klmn)$, where k is the iteration times of the *while* loop. The experimental results show that the iteration times of the *while* loop for optimizing f_1 is less than that for f. Using the results of the iteration times in the last section, the value of k is expected to be $O(\log(m+n)/\log\beta)$, where $\beta = (1 - a/(A(m-1)))^m$, $A \geq a > 0$ are the largest and the smallest eigenvalues of the Hessian matrix $\mathbf{H}(\mathbf{y}^*)$, and the average time complexity of the $SAT14.7$ algorithm is expected to be $O(lmn\log(m+n)/\log\beta)$ if the $SAT14.7$ algorithm is not stuck at a local minimum point.

6 Conclusion

Recently, to design an efficient optimization algorithm for finding a solution of a satisfiable *CNF* Boolean formula has become an active research. A new formulation, the *UniSAT7* problem model, which transforms the SAT problem into an optimization problem on real space, has been developed[8,12,54].

In this paper, we prove that, when the initial solution is sufficiently close to the optimal solution, the steepest descent method has a linear convergence ratio $\beta < 1$, Newton's method has an order two convergence ratio, and the coordinate descent method has a convergence ratio of $(1 - \beta/m)$ for the *UniSAT7* problem with m variables.

Acknowledgments We thank for Ranan Banerji, Roberto Battiti, Max Böhm, Endre Boros, Guoliang Chen, Xiaotie Deng, Bin Du, Peter Hammer, Pierre Hansen, Peter Heusch, Jieh Hsiang, Frank Hsu, Kazuo Iwama, Philip Jackson, Bob Johnson, David Johnson, Lewis Johnson, Dennis Kibler, Scott Kirkpatrick, Oliver Kullmann, Vipin Kumar, R.C.T. Lee, Theo Lettmann, Guojie Li, Ming Li, Wei Li, Hans Maaren, M.V. Marathe, John Mitchell, Henri Morel, Christos Papadimitriou, Panos Pardalos, Vaughan Pratt, D. Pretolani, Marco Protasi, Paul Purdom, Ruchir Puri, Mauricio Resende, Daniel Rosenkrantz, Andy Sage, Sandeep Shukla, Rok Sosič, Ewald Speckenmeyer, Peter Vanbekbergen, Anthony Vannelli, Stephen Vavasis, Ben Wah, Jinchang Wang, Wei Wang, David Yau, for their insightful comments on various early versions of this paper.

We are grateful to our colleagues and students for various discussion of the *UniSAT* problem models and their optimization algorithms during our lectures in Chinese Academy of Sciences, University of Science and Technology of China, Tsinghua University and Nanjing University in 1992.

References

[1] Dershowitz N, Hsiang J, Josephson N, Plaisted D. Associative-commutative rewriting. In *Proceedings of IJCAI,* 1983, pp.940–944.

[2] Hsiang J. Refutational theorem proving using term-rewriting systems. *Artificial Intelligence,* 1985, 16: 255–300.

[3] Shensa M J. A computational structure for the propositional calculus. In *Proceedings of IJCAI,* 1989, pp.384–388.

[4] P C Jackson Jr. Heuristic search algorithms for the satisfiability problem. Submitted to the third IEEE TAI Conference, Jul. 1991.

[5] Nilsson N J. Principles of Artificial Intelligence. Tioga Publishing Company, Palo Alto, California, 1980.

[6] Winston P H. Artificial Intelligence. Addison-Wesley, Reading, 1984.

[7] Davis M, Putnam H. A computing procedure for quantification theory. *J. ACM,* 1960, 7: 201–215.

[8] Gu J. How to solve very large-scale satisfiability problems. Technical Report UUCS-TR-88-032. 1988, and UCECE-TR-90-002, 1990.

[9] Gu J. Efficient local search for very large-scale satisfiability problem. *SIGART Bulletin,* Jan. 1992, 3(1): 8–12, ACM Press.

[10] Gu J, Gu Q P. Average time complexities of several local search algorithms for the satisfiability problem. Technical Report UCECE-TR-91-004, 1991. In *Lecture Notes in Computer Science,* 1994, 834: 146–154, and to appear in *IEEE Trans. Knowledge and Data Engineering.*

[11] Koutsoupias E, Papadimitriou C H. On the greedy algorithm for satisfiability. *Information Processing Letters,* 10 Aug. 1992, 43: 53–55.

[12] Gu J. On Optimizing a Search Problem. In *Advanced Series on Artificial Intelligence,* Jan. 1992, 1 (Chapter 2): 63–105.

[13] Gu J. Global optimization for satisfiability (SAT) problem. *IEEE Trans. Knowledge and Data Engineering,* Jun. 1994, 6(3): 361–381, Feb. 1995, 7(1):192.

[14] Gu J, Purdom P W, Franco J, Wah B W. Algorithms for satisfiability (SAT) problem: A survey. *DIMACS Volume Series on Discrete Mathematics and Theoretical Computer Science,* The Satisfiability (SAT) Problem, 1997, 35: 19–151, American Mathematical Society.

[15] Mackworth A K. Consistency in networks of relations. *Artificial Intelligence,* 1977, 8: 99–119.

[16] Bitner J R, Reingold E M. Backtrack programming techniques. *Comm. ACM*, Nov. 1975, 18(11): 651–656.
[17] Brown C A, Purdom P W. An average time analysis of backtracking. *SIAM J. Computing*, Aug. 1981, 10(3): 583–593.
[18] Bugrara K, Purdom P. Clause order backtracking. Technical Report 311, 1990.
[19] Larrabee T. Test pattern generation using Boolean satisfiability. *IEEE Trans. Computer-Aided Design*, Jan. 1992, 11(1): 4–15.
[20] Purdom P W. Search rearrangement backtracking and polynomial average time. *Artificial Intelligence*, 1983, 21: 117–133.
[21] Siegel P. Representation et Utilization de la Connaissances en Calcul Propositionel. Ph.D. thesis, University Aix-Marseille II, 1987.
[22] Ashar P, Ghosh A, Devadas S. Boolean satisfiability and equivalence checking using general Binary Decision Diagrams. *Integration, the VLSI journal*, 1992, 13(1): 1–16.
[23] Bryant R E. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Computers*, Aug. 1986, C-35(8): 677–691.
[24] Gelder A V. A satisfiability tester for non-clausal propositional calculus. *Information and Computation*, Oct. 1988, 79(1): 1–21.
[25] Robinson J A. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 1965, 12: 23–41.
[26] Clowes M B. On seeing things. *Artificial Intelligence*, 1971, 2: 79–116.
[27] Gu J. Parallel algorithms and architectures for very fast search. Technical Report UUCS-TR-88-005, Jul. 1988.
[28] Huffman D A. Impossible Objects as Nonsense Sentences. In *Machine Intelligence*, Meltzer B, Michie D (eds.), 1971, pp.295–323. Edinburgh University Press, Edinburgh, Scotland.
[29] Waltz D. Generating semantic descriptions from drawings of scenes with shadows. Technical Report AI271, MIT, Nov. 1972.
[30] Davis M, Logemann G, Loveland D. A machine program for theorem proving. *Communications of the ACM*, 1962, 5: 394–397.
[31] Loveland D W. Automated Theorem Proving: A Logical Basis. North-Holland, 1978.
[32] Gu J, Wang W. A novel discrete relaxation architecture. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Aug. 1992, 14(8): 857–865.
[33] Gu J, Wang W, Henderson T C. A parallel architecture for discrete relaxation algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Nov. 1987, PAMI-9(6): 816–831.
[34] Jeroslow R E, Wang J. Solving propositional satisfiability problems. *Annals of Mathematics and AI*, 1990, 1: 167–187.
[35] Gu J. Local search for satisfiability (SAT) problem. *IEEE Trans. Systems, Man, and Cybernetics*, Jul. 1993, 23(4): 1108–1129, Apr. 1994, 24(4): 709.
[36] Purdom P W, Haven G N. Backtracking and probing. Technical Report No. 387, Dept. of Computer Science, Indiana University, Aug. 1993.
[37] Bohm M, Speckenmeyer E. A fast parallel SAT-solver — Efficient workload balancing. Presented at the 3rd International Symposium on AI & Mathematics, Jan. 1994.
[38] Truemper K. Polynomial algorithms for problems over d-systems. Presented at the 3rd International Symposium on AI & Mathematics, Jan. 1994.
[39] Dubois O, Andre P, Boufkhad Y, Carlier J. SAT versus UNSAT. *DIMACS Series Volume: Clique, Graph Coloring, and Satisfiability — Second DIMACS Implementation Challenge*, Johnson D S, Trick M A (eds.), 26, American Mathematical Society, 1996.
[40] Sosič R, Gu J. How to search for million queens. Technical Report UUCS-TR-88-008, Feb. 1988.
[41] Sosič R, Gu J. Quick n-queen search on VAX and Bobcat machines. CS 547 AI Class Project, Feb. 1988.
[42] Sosič R, Gu J, Johnson R. The Unison algorithm: Fast evaluation of Boolean expressions. *ACM Transactions on Design Automation of Electronic Systems*, Oct. 1996, 1(4): 456–477.
[43] Sosič R, Gu J, Johnson R. A universal Boolean evaluator. *IEEE Trans. Computers*, accepted for publication in 1992.
[44] Papadimitriou C H. Private Communications, 1992.
[45] Papadimitriou C H. On selecting a satisfying truth assignment. In *Proceedings of the 32nd Annual Symposium of the Foundations of Computer Science*, 1991, pp.163–169.
[46] Selman B, Levesque H, Mitchell D. A new method for solving hard satisfiability problems. In *Proceedings of AAAI'92*, Jul. 1992, pp.440–446.
[47] Blair C E, Jeroslow R G, Lowe J K. Some results and experiments in programming techniques for propositional logic. *Computers and Operations Research*, 1986, 5: 633–645.
[48] Jeroslow R G. Computation-oriented reductions of predicate to propositional logic. *Decision Support Systems*, 1988, 4: 183–197.

[49] Hooker J N. Generalized resolution and cutting planes. *Annals of Operations Research,* 1988, 12: 217–239.
[50] Hooker J N. Resolution vs. cutting plane solution of inference problems: Some computational experience. *Operations Research Letter,* 1988, 7(1): 1–7.
[51] Hooker J N, Fedjki C. Branch-and-cut solution of inference problems in propositional logic. Technical Report 77-88-89, GSIA, Carnegie Mellon University, Aug. 1989.
[52] Kamath A P, Karmarkar N K, Ramakrishnan K G, Resende M G C. Computational experience with an interior point algorithm on the satisfiability problem. *Annals of Operations Research,* 1990, 25: 43–58.
[53] Kamath A P, Karmarkar N K, Ramakrishnan K G, Resende M G C. Computational experience with an interior point algorithm on the satisfiability problem. *Mathematical Sciences Research Center, AT&T Bell Laboratories,* Oct. 1989.
[54] Gu J. The *UniSAT* problem models (appendix). *IEEE Trans. Pattern Analysis and Machine Intelligence,* Aug. 1992, 14(8): 865.
[55] Johnson J L. A neural network approach to the 3-satisfiability problem. *J. Parallel and Distributed Computing,* 1989, 6: 435–449.
[56] Chakradhar S, Agrawal V, Bushnell M. Neural net and Boolean satisfiability model of logic circuits. *IEEE Design & Test of Computers,* Oct. 1990, pp.54–57.
[57] Genesereth M R, Nilsson N J. Logical Foundations of Artificial Intelligence. Morgan Kaufmann Publishers, Los Altos, California, 1987.
[58] Tseitin G S. On the Complexity of Derivations in Propositional Calculus. In *Structures in Constructive Mathematics and Mathematical Logic, Part II,* Slisenko A O (ed.), 1968, pp.115–125.
[59] Urquhart A. The complexity of propositional proofs. *The Bulletin of Symbolic Logic,* 1995, 1(4): 425–467.
[60] Iwama K. CNF satisfiability test by counting and polynomial average time. *SIAM J. Computing,* 1989, 18: 385–391.
[61] Galil Z. On the complexity of regular resolution and the Davis-Putnam procedure. *Theoretical Computer Science,* 1977, pp.23–46.
[62] Goldberg A, Purdom P W, Brown C A. Average time analysis of simplified Davis-Putnam procedures. *Information Processing Letters,* Sept. 1982, 15(2): 72–75.
[63] Purdom P W, Brown C A. The pure literal rule and polynomial average time. *SIAM J. Computing,* 1985, 14: 943–953.
[64] Hu T H, Tang C Y, Lee R C T. An average case analysis of a resolution principle algorithm in mechanical theorem proving. *Annals of Mathematics and Artificial Intelligence,* 1992, 6: 235–252.
[65] Gilmore P C. A proof method for quantification theory. *IBM J. Res. Develop.,* 1960, 4: 28–35.
[66] Aho A V, Hopcroft J E, Ullman J D. The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, 1974.
[67] Cook S A. The complexity of theorem-proving procedures. In *Proceedings of the Third ACM Symposium on Theory of Computing,* 1971, pp.151–158.
[68] Aspvall B, Plass M F, Tarjan R E. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters,* Mar. 1979, 8(3): 121–132.
[69] Even S, Itai A, Shamir A. On the complexity of timetable and multi-commodity flow problems. *SIAM J. Computing,* 1976, 5(4): 691–703.
[70] Schaefer T J. The complexity of satisfiability problems. In *Proceedings of the 10th ACM Symposium on Theory of Computing,* 1978, pp.216–226.
[71] Hooker J N. A quantitative approach to logical inference. *Decision Support Systems,* 1988, 4: 45–69.
[72] Papadimitriou C H, Steiglitz K. Combinatorial Optimization: Algorithms and Complexity. Prentice Hall, Englewood Cliffs, 1982.
[73] Williams H P. Linear and integer programming applied to the propositional calculus. *Systems Research and Information Sciences,* 1987, 2: 81–100.
[74] Shi R C -J, Vannelli A, Vlach J. An improvement on Karmarkar's algorithm for integer programming. *COAL Bulletin of the Mathematical Programming Society,* 1992.
[75] Tyler J E. Parallel computer architectures and problem solving strategies for the consistent labeling problem. Master's thesis, Dept. of Elec. Eng., Virginia Polytech. Inst. State Univ., Blacksburg, Virginia, Oct. 1983.
[76] McCall J T, Tront J G, Gray F G, Haralick R M, McCormack W M. Parallel computer architectures and problem solving strategies for the consistent labeling problem. *IEEE Trans. Computers,* Nov. 1985, C-34(11): 973–980.
[77] Samal A, Henderson T C. Parallel consistent labeling algorithms. *International Journal of Parallel Programming,* 1988.
[78] Chen W T, Liu L L. Parallel approach for theorem proving in propositional logic. *Inform. Sci.,* 1987, 41(1): 61–76.
[79] Fang M Y, Chen W T. Vectorization of a generalized procedure for theorem proving in propositional logic on vector computers. *IEEE Trans. Knowledge and Data Engineering,* Oct. 1992, 4(5): 475–486.

[80] Conery J S. Parallel Execution of Logic Programs. Kluwer Academic Publishers, Boston, 1987.
[81] Lin Y J. A Parallel Implementation of Logic Programs. PhD thesis, The Univ. of Texas at Austin, Dept. of Computer Science, May 1988.
[82] Wah B, Li G J. Computers for Artificial Intelligence Applications. IEEE Computer Society Press, Washington D. C., 1986.
[83] Wah B W (ed.). New computers for artificial intelligence processing. *IEEE Computer,* 1987, 20(1), IEEE Computer Society Press, 1987.
[84] Wah B W, Lowrie M B, Li G J. Computers for symbolic processing. In *Proceedings of the IEEE,* Apr. 1989, 77(4): 509–540.
[85] Sosič R, Gu J. A parallel local search algorithm for satisfiability (SAT) problem. Submitted for publication, 1993.
[86] Gu J. An $\alpha\beta$-relaxation for global optimization. Technical Report UCECE-TR-91-003, Apr. 1991.
[87] Garey M R, Johnson D S. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, San Francisco, 1979.
[88] Luenberger D G. Linear and Nonlinear Programming. Addison-Wesley, Reading, 1984.
[89] Du Dingzhu. Convergence Theory of Feasible Direction Methods. Science Press, 1991.
[90] Strang G. Linear Algebra and Its Applications. Academic Press, New York, 1976.
[91] Chvátal V, Szemerédi E. Many hard examples for resolution. *J. ACM,* 1988, 35: 759–770.
[92] Franco J. Elimination of infrequent variables improves average case performance of satisfiability algorithms. *SIAM J. Computing,* 1991, 20: 1119–1127.

For the biography of **GU Jun**, please refer to p.90 of this issue.

GU Qianping received the B.S. degree from Shandong University, China, M.S. degree from Ibaraki University, Japan, and Ph.D. degree from Tohoku University, Japan, all in computer science, in 1982, 1985, and 1988, respectively. He is currently an Associate Professor in the Department of Computer Software, the University of Aizu, Japan. He was with the Institute of Software, Chinese Academy of Sciences, Beijing, China, and the Department of Electrical and Computer Engineering, the University of Calgary, Canada. His research interests include algorithms, computational complexity, machine learning, parallel processing, and optimization. He is a member of ACM, IEEE Computer Society, and IEICE of Japan.

DU Dingzhu received his M.S. degree in 1982 from the Institute of Applied Mathematics, Chinese Academy of Sciences, and his Ph.D. degree in Computer Science in 1985 from the University of California at Santa Barbara. He visited MSRI in 1985, MIT in 1986, and Princeton University in 1990. Currently he is a tenure faculty at the Department of Computer Science, University of Minnesota, and a research Professor at the Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing. He has published eight books and over 100 journal papers. His research interests are computational complexity, analysis and design of algorithms, combinatorial optimization, computational geometry, communication networks, and linear and nonlinear programming. In 1986, with Xiangsun Zhang, he established the convergence of Rosen's gradient projection method and in 1990, with Frank Huang, he proved the Gilbert-Pollak conjecture on Steiner ratio.

Appendix. The Proof of Lemma 8

Since f has the global minimum value 0 at the solution point $\mathbf{y}^*$, $\nabla f(\mathbf{y}^*) = 0$. Then from Taylor's theorem, we have

$$f(\mathbf{y}_k) - f(\mathbf{y}^*) = (\mathbf{y}_k - \mathbf{y}^*)\mathbf{H}(\mathbf{y}^*)(\mathbf{y}_k - \mathbf{y}^*)^T + o((\mathbf{y}_k - \mathbf{y}^*)^2).$$

From this, we have

$$\begin{aligned} f(\mathbf{y}_k) - f(\mathbf{y}_{k+1}) =& (f(\mathbf{y}_k) - f(\mathbf{y}^*)) - (f(\mathbf{y}_{k+1}) - f(\mathbf{y}^*)) = (\mathbf{y}_k - \mathbf{y}^*)\mathbf{H}(\mathbf{y}^*)(\mathbf{y}_k - \mathbf{y}^*)^T \\ & - (\mathbf{y}_{k+1} - \mathbf{y}^*)\mathbf{H}(\mathbf{y}^*)(\mathbf{y}_{k+1} - \mathbf{y}^*)^T + o((\mathbf{y}_k - \mathbf{y}^*)^2 + (\mathbf{y}_{k+1} - \mathbf{y}^*)^2) \\ =& - 2(\mathbf{y}_{k+1} - \mathbf{y}_k)\mathbf{H}(\mathbf{y}^*)(\mathbf{y}_k - \mathbf{y}^*)^T \\ & - (\mathbf{y}_{k+1} - \mathbf{y}_k)\mathbf{H}(\mathbf{y}^*)(\mathbf{y}_{k+1} - \mathbf{y}_k)^T + o((\mathbf{y}_k - \mathbf{y}^*)^2 + (\mathbf{y}_{k+1} - \mathbf{y}^*)^2) \end{aligned}$$

$$= -2\alpha_k \mathbf{d}_k \mathbf{H}(\mathbf{y}^*)\mathbf{z}_k^T - (\alpha_k)^2 \mathbf{d}_k \mathbf{H}(\mathbf{y}^*)\mathbf{d}_k^T + o((\mathbf{y}_k - \mathbf{y}^*)^2 + (\mathbf{y}_{k+1} - \mathbf{y}^*)^2), \tag{15}$$

where

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \alpha_k \mathbf{d}_k, \quad \mathbf{z}_k = \mathbf{y}_k - \mathbf{y}^*,$$

and α_k satisfies

$$\nabla f(\mathbf{y}_k + \alpha_k \mathbf{d}_k)\mathbf{d}_k^T = 0.$$

From this and

$$\nabla f(\mathbf{y}) - \nabla f(\mathbf{y}_k) = \int_{\mathbf{y}_k}^{\mathbf{y}} \mathbf{H}(\mathbf{y})d\mathbf{y} = (\mathbf{y} - \mathbf{y}_k)\mathbf{H}(\mathbf{y}_k)(1 + o(1)),$$

we get

$$0 = \nabla f(\mathbf{y}_k + \alpha_k \mathbf{d}_k)\mathbf{d}_k^T = (\alpha_k \mathbf{d}_k \mathbf{H}(\mathbf{y}_k))\mathbf{d}_k^T(1 + o(1)) + \nabla f(\mathbf{y}_k)\mathbf{d}_k^T.$$

Therefore,

$$\alpha_k = \frac{-\nabla f(\mathbf{y}_k)\mathbf{d}_k^T}{\mathbf{d}_k \mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T(1 + o(1))} = \frac{-\mathbf{g}_k \mathbf{d}_k^T}{\mathbf{d}_k \mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T(1 + o(1))}, \tag{16}$$

where $\mathbf{g}_k = \nabla f(\mathbf{y}_k)$. In addition, since $\nabla f(\mathbf{y}^*) = 0$,

$$\nabla f(\mathbf{y}_k) = \nabla f(\mathbf{y}_k) - \nabla f(\mathbf{y}^*) = \int_{\mathbf{y}^*}^{\mathbf{y}_k} \mathbf{H}(\mathbf{y}^*)d\mathbf{y} = (\mathbf{y}_k - \mathbf{y}^*)\mathbf{H}(\mathbf{y}^*)(1 + o(1)) = \mathbf{z}_k \mathbf{H}(\mathbf{y}^*)(1 + o(1)).$$

From this,

$$\mathbf{d}_k \mathbf{H}(\mathbf{y}^*)\mathbf{z}_k^T = \mathbf{z}_k \mathbf{H}(\mathbf{y}^*)\mathbf{d}_k^T = \frac{\mathbf{g}_k \mathbf{d}_k^T}{(1 + o(1))}.$$

Therefore, from (15) and (16), we have

$$f(\mathbf{y}_k) - f(\mathbf{y}_{k+1}) = 2\frac{(\mathbf{g}_k \mathbf{d}_k^T)^2}{\mathbf{d}_k \mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T(1 + o(1))^2} - \frac{(\mathbf{g}_k \mathbf{d}_k^T)^2 \mathbf{d}_k \mathbf{H}(\mathbf{y}^*)\mathbf{d}_k^T}{(\mathbf{d}_k \mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T)^2(1 + o(1))^2}. \tag{17}$$

From

$$f(\mathbf{y}_k) - f(\mathbf{y}^*) = \mathbf{z}_k \mathbf{H}(\mathbf{y}^*)\mathbf{z}_k^T(1 + o(1))$$

and (17), we have

$$\begin{aligned}
\frac{f(\mathbf{y}_k) - f(\mathbf{y}_{k+1})}{f(\mathbf{y}_k) - f(\mathbf{y}^*)} &= \frac{2\frac{(\mathbf{g}_k \mathbf{d}_k^T)^2}{\mathbf{d}_k \mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T(1+o(1))^2} - \frac{(\mathbf{g}_k \mathbf{d}_k^T)^2 \mathbf{d}_k \mathbf{H}(\mathbf{y}^*)\mathbf{d}_k^T}{(\mathbf{d}_k \mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T)^2(1+o(1))^2}}{\mathbf{z}_k \mathbf{H}(\mathbf{y}^*)\mathbf{z}_k^T(1 + o(1))} \\
&= \frac{(\mathbf{g}_k \mathbf{d}_k^T)^2}{(\mathbf{d}_k \mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T)(\mathbf{z}_k \mathbf{H}(\mathbf{y}^*)\mathbf{z}_k^T)(1 + o(1))} \\
&\quad \times \left(\frac{2}{(1 + o(1))^2} - \frac{\mathbf{d}_k \mathbf{H}(\mathbf{y}^*)\mathbf{d}_k^T}{\mathbf{d}_k \mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T(1 + o(1))^2}\right) \\
&= \frac{(\mathbf{g}_k \mathbf{d}_k^T)^2}{(\mathbf{d}_k \mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T)(\mathbf{g}_k \mathbf{H}^{-1}(\mathbf{y}^*)\mathbf{g}_k^T)(1 + o(1))} \\
&\quad \times \left(\frac{2}{(1 + o(1))^2} - \frac{\mathbf{d}_k \mathbf{H}(\mathbf{y}^*)\mathbf{d}_k^T}{\mathbf{d}_k \mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T(1 + o(1))^2}\right).
\end{aligned} \tag{18}$$

Let A_k and a_k be the largest and the smallest eigenvalues of $\mathbf{H}(\mathbf{y}_k)$, respectively. Then $A_k \ge a_k > 0$ for $\mathbf{y}_k$ sufficiently close to $\mathbf{y}^*$, since $\mathbf{H}(\mathbf{y}^*)$ is positive definite.

From this,

$$0 < a_k(\mathbf{d}_k \mathbf{d}_k^T) \le \mathbf{d}_k \mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T \le A_k(\mathbf{d}_k \mathbf{d}_k^T). \tag{19}$$

Therefore,

$$\frac{(\mathbf{g}_k\mathbf{d}_k^T)^2}{(\mathbf{d}_k\mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T)(\mathbf{g}_k\mathbf{H}^{-1}(\mathbf{y}^*)\mathbf{g}_k^T)(1+o(1))} > 0.$$

From this and

$$\frac{f(\mathbf{y}_k)-f(\mathbf{y}_{k+1})}{f(\mathbf{y}_k)-f(\mathbf{y}^*)} \geq 0,$$

we have

$$\Big(\frac{2}{(1+o(1))^2} - \frac{\mathbf{d}_k\mathbf{H}(\mathbf{y}^*)\mathbf{d}_k^T}{\mathbf{d}_k\mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T(1+o(1))^2}\Big) \geq 0.$$

Therefore, from (19),

$$\mathbf{g}_k\mathbf{H}^{-1}(\mathbf{y}^*)\mathbf{g}_k^T \leq \frac{1}{a}(\mathbf{g}_k\mathbf{g}_k^T),$$

where a is the smallest eigenvalue of $\mathbf{H}(\mathbf{y}^*)$, and (18),

$$\begin{aligned}\frac{f(\mathbf{y}_k)-f(\mathbf{y}_{k+1})}{f(\mathbf{y}_k)-f(\mathbf{y}^*)} &\geq \frac{(\mathbf{g}_k\mathbf{d}_k^T)^2}{(A_k\mathbf{d}_k\mathbf{d}_k^T)(\frac{1}{a}\mathbf{g}_k\mathbf{g}_k^T)(1+o(1))} \\ &\quad\times\Big(\frac{2}{(1+o(1))^2} - \frac{\mathbf{d}_k\mathbf{H}(\mathbf{y}^*)\mathbf{d}_k^T}{\mathbf{d}_k\mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T(1+o(1))^2}\Big) \\ &= 2\frac{a(\mathbf{g}_k\mathbf{d}_k^T)^2}{A_k(\mathbf{d}_k\mathbf{d}_k^T)(\mathbf{g}_k\mathbf{g}_k^T)(1+o(1))} \\ &\quad - \frac{a(\mathbf{g}_k\mathbf{d}_k^T)^2}{A_k(\mathbf{d}_k\mathbf{d}_k^T)(\mathbf{g}_k\mathbf{g}_k^T)(1+o(1))} \times \frac{\mathbf{d}_k\mathbf{H}(\mathbf{y}^*)\mathbf{d}_k^T}{(\mathbf{d}_k\mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T)} + o(1). \end{aligned} \tag{20}$$

Let

$$\mathbf{e}_i = (\underbrace{0,\ldots,0,1}_{i},0,\ldots,0).$$

Then it is clear that

$$\sum_{i=1}^{m}\cos^2(\mathbf{e}_i,\mathbf{g}_k) = 1. \tag{21}$$

Since one $\mathbf{e}_i$ is in the direction of $\mathbf{d}_{k-1}$, say $\mathbf{e}_m$, $\cos(\mathbf{e}_m,\mathbf{g}_k) = 0$. Therefore, from (21), we have that at least one of the terms $\cos^2(\mathbf{e}_i,\mathbf{g}_k) \geq 1/(m-1)$. Thus, from the fact that the coordinate corresponding to the largest (in absolute value) component of the gradient vector $\mathbf{g}_k$ is selected at each stage,

$$\frac{(\mathbf{g}_k\mathbf{d}_k^T)^2}{(\mathbf{d}_k\mathbf{d}_k^T)(\mathbf{g}_k\mathbf{g}_k^T)} = \max_{i=1,\ldots,m-1}\cos^2(\mathbf{e}_i,\mathbf{g}_k) \geq \frac{1}{m-1}.$$

From this and (20),

$$\begin{aligned}\frac{f(\mathbf{y}_k)-f(\mathbf{y}_{k+1})}{f(\mathbf{y}_k)-f(\mathbf{y}^*)} &\geq 2\frac{a}{A_k}\times\frac{1}{(m-1)(1+o(1))} \\ &\quad - \frac{a}{A_k}\times\frac{1}{(m-1)(1+o(1))}\times\frac{\mathbf{d}_k\mathbf{H}(\mathbf{y}^*)\mathbf{d}_k^T}{(\mathbf{d}_k\mathbf{H}(\mathbf{y}_k)\mathbf{d}_k^T)} + o(1).\end{aligned}$$

As $k\to\infty$, $\mathbf{y}_k\to\mathbf{y}^*$, we obtain

$$\lim_{k\to\infty}\frac{f(\mathbf{y}_k)-f(\mathbf{y}_{k+1})}{f(\mathbf{y}_k)-f(\mathbf{y}^*)} \geq \frac{a}{A(m-1)},$$

where A is the largest eigenvalue of $\mathbf{H}(\mathbf{y}^*)$.

From this, we get

$$\lim_{k\to\infty}\frac{f(\mathbf{y}_{k+1})-f(\mathbf{y}^*)}{f(\mathbf{y}_k)-f(\mathbf{y}^*)} \leq 1-\frac{a}{A(m-1)}. \qquad \square$$