Available online at www.sciencedirect.com
ScienceDirect
Journal of Computer and System Sciences 73 (2007) 123–136
JOURNAL OF COMPUTER AND SYSTEM SCIENCES
www.elsevier.com/locate/jcss

Quantum cooperative search algorithm for 3-SAT

Sheng-Tzong Cheng *, Ming-Hung Tao

Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, ROC

Received 22 March 2006; received in revised form 11 September 2006

Available online 19 October 2006

Abstract

Grover's search algorithm, one of the most popular quantum algorithms, provides a good solution to solve NP complexity problems, but requires a large number of quantum bits (qubits) for its functionality. In this paper, a novel algorithm called quantum cooperative search is proposed to make Grover's search algorithm work on 3-SAT problems with a small number of qubits. The proposed algorithm replaces some qubits with classical bits and finds assignments to these classical bits using the traditional 3-SAT algorithms including evolutionary algorithms and heuristic local search algorithms. In addition, the optimal configuration of the proposed algorithm is suggested by mathematical analysis. The experimental results show that the quantum cooperative search algorithm composed by Grover's search and heuristic local search performs better than other pure traditional 3-SAT algorithms in most cases. The mathematical analysis of the appropriate number of qubits is also verified by the experiments.

Keywords: Grover's search algorithm; NP complexity; 3-SAT problems; Evolutionary algorithm; Heuristic local search algorithm

1. Introduction

The satisfiability problem (SAT) is one of the most widely studied NP-complete problems. Its statement is deceptively simple, and yet it remains one of the most challenging problems in research. The real challenge of these kinds of problems for classical computers is to develop algorithms and techniques that can solve realistically sized problems within a reasonable amount of computation time.

Local search is often surprisingly effective for the satisfiability problem. The famous local search algorithm, GenSAT [1], a greedy random hill-climbing procedure is the representative work. Gent and Walsh [2] investigated two variants of GenSAT, called GSAT and HSAT, and showed that the two variants exhibit surprisingly good performance on hard random problems. Frank [3] introduced another variant of GSAT called WGSAT with an adaptive weighting mechanism. Another procedure for handling satisfiability problems is the evolutionary algorithm (EAs), which has been used since 1989 [4]. It was a great challenge to apply EAs to the satisfiability problem because the presence of constraints made finding solutions difficult for an EA. T. Bäck [5] exhibited the superior EA for 3-SAT and suggested that the adaptive EA is not only a good solver for satisfiability problems, but for constraint satisfaction problems in general.

* Corresponding author.
E-mail addresses: stcheng@mail.ncku.edu.tw (S.-T. Cheng), minghung@csie.ncku.edu.tw (M.-H. Tao).

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2006.09.003

Recently an entirely new approach has been developed with potentially enormous consequences. This new approach is called quantum computing, and it relies on the principles of quantum mechanics to obtain solutions for satisfiability problems. Quantum computers were discussed in the early 1980s [6–8]. Since then, a great deal of research has been focused on the topic. Remarkable progress has been made on the development of secure key distribution [9], polynomial time prime factorization [10], quantum communication [11,12], and fast database search algorithm [13]. These results have recently made quantum information science the most rapidly expanding research field. The fast database search is an important technique for solving the NP-complete problems including satisfiability problems. In classical computers, the traditional brute-force search requires an average of $N/2$ comparisons in a database of N elements. However, in quantum computers, due to the bit representation (quantum computer systems represent a single bit of information as a qubit, which is a unit vector in a complex Hilbert space) and the linear superposition of states, Grover's algorithm can identify the target in an unordered database in only $O(\sqrt{N})$. Thus $O(\sqrt{2^n})$ steps are required when applying Grover's algorithm to solve a 3-CNF formula with n variables.

Although the computation power of the quantum computing system is excellent for solving the satisfiability problem, the quantum computer by current techniques can be equipped with only a few qubits. This causes the constraints when Grover's algorithm is performed on the satisfiability problem. Therefore, a novel search algorithm called quantum cooperative search (QCS) is proposed in this paper to make Grover's algorithm work with a small number of qubits. The proposed algorithm replaces some qubits with classical bits and finds assignments to these classical bits using the traditional 3-SAT algorithms including evolutionary algorithms and heuristic local search algorithms. The QCS algorithm can be easily extended to find the solutions for other NP-complete problems. Thus the cooperation between quantum and classical systems can be achieved to solve the complexity problem eventually.

The rest of the paper is organized as follows. In Section 2, we introduce the satisfiability problem and the algorithms (for both classical system and quantum system) for it. In Section 3, the quantum cooperative search algorithm is proposed and the optimal configure for the proposed algorithm is suggested by mathematical analysis. Experimental results are exhibited in Section 4 to demonstrate the performance of the proposed algorithm. Section 5 concludes the paper.

2. Satisfiability problems

One important advance on the P versus NP question came in the early 1970s with the work of Stephen Cook and Leonid Levin [14,15]. They discovered certain problems in NP whose individual complexity is related to that of the entire class. If a polynomial time algorithm exists for any of these problems, all problems in NP would be polynomial-time solvable. These problems are called NP-complete. The phenomenon of NP-completeness is important for both theoretical and practical reasons. The satisfiability problem discussed in this paper is one of the NP-complete problems. Grover's search algorithm and several traditional algorithms on solving the satisfiability problem are described in the following subsections.

2.1. Grover's search algorithm

Grover's search algorithm is based on quantum mechanisms. In a two-state quantum system, each bit (qubit) can be represented using a basis consisting of two eigenstates, denoted by $|0\rangle$ and $|1\rangle$, respectively. These states can be either spin states of a particle or energy levels in an atom. A qubit can be any linear combination of these two states, so we have the state $|\psi\rangle$ of a qubit as $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$, where c_0, c_1 are complex numbers and $|c_0|^2 + |c_1|^2 = 1$. The state shown above exhibits the phenomenon called superposition in quantum mechanisms. The states of qubits are operated by the quantum gates which can be represented in the form of matrix operations. The operation of a quantum gate is composed of rotation operations and phase shift operations. A set of quantum gates which can be used to implement any unitary operation to any desired degree of accuracy is called a universal set [16]. The single qubit and CNOT gates are universal for quantum computation.

To find valid assignments for a 3-CNF formula with n variables by Grover's algorithm, search through a space of $N = 2^n$ elements is required, thus the index needs to be stored in n bits. The search problem is assumed to have exactly M $(1 \leqslant M \leqslant N)$ solutions. A particular instance of the search problem can be conveniently represented by a function f, which takes x as an input integer, in the range 0 to $N-1$. And $f(x) = 1$ if x is a solution to the search problem, and $f(x) = 0$ if x is not a solution.

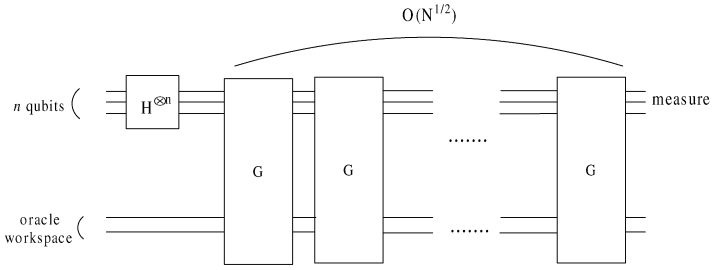

Fig. 1. Schematic circuit of Grover's algorithm.

Grover's Search algorithm uses a quantum oracle (i.e. a black box) to recognize solutions to a particular search problem. This recognition is signalized by making use of an oracle qubit. More precisely, the oracle is a unitary operator, O, defined by its action on the computational basis:

$$|x\rangle|q\rangle \to |x\rangle\big|q \oplus f(x)\big\rangle, \tag{1}$$

where $|x\rangle$ is the index register, $\oplus$ denotes addition modulo 2, and the oracle qubit $|q\rangle$ is a single qubit which is flipped if $f(x) = 1$, and is unchanged otherwise. Integer x is checked to see whether it is a solution to our search problem by preparing $|x\rangle|0\rangle$, applying the oracle, and checking to see if the oracle qubit is flipped to $|1\rangle$.

Grover's algorithm begins with the computer of n qubits in the state $|0\rangle^{\otimes n}$ (each qubit is in $|0\rangle$ state). The Hadamard transform is used to put the computer in the equal superposition state,

$$|\psi\rangle = \frac{1}{N^{1/2}} \sum_{x=0}^{N-1} |x\rangle. \tag{2}$$

The Grover algorithm then consists of repeated application of a quantum subroutine, known as the Grover iteration or Grover operator, which we denote G. The Grover iteration, whose quantum circuit is illustrated in Fig. 1, may be broken up into four steps:

- Apply the Oracle.
- Apply the Hadamard transform $H^{\otimes n}$ ($H = \frac{1}{\sqrt{2}}\left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$).
- Perform a conditional phase shift on the computer, with every computational basis state except $|0\rangle$ receiving a phase shift of -1,

$$|x\rangle \to -(-1)^{\langle x|0\rangle}|x\rangle. \tag{3}$$

- Apply the Hadamard transform $H^{\otimes n}$.

The Grover iteration requires only a single oracle call. The combined effect of steps (2), (3), (4) is

$$H^{\otimes n}\big(2|0\rangle\langle 0| - I\big)H^{\otimes n} = 2|\psi\rangle\langle\psi| - I, \tag{4}$$

in which $|\psi\rangle$ is the equally weighted superposition of states presented in (2). Thus the Grover iteration, G, may be written $G = (2|\psi\rangle\langle\psi| - I)O$. Applying the operation $2|\psi\rangle\langle\psi| - I$ to a general state $\sum_k \alpha_k|k\rangle$ produces

$$\sum_k \big[-\alpha_k + 2\langle\alpha\rangle\big]|k\rangle, \tag{5}$$

in which $\langle\alpha\rangle \equiv \sum_k \alpha_k/N$ is the mean value of α_k. For this phenomenon, $2|\psi\rangle\langle\psi| - I$ is sometimes referred to as the inversion about mean operation and has the effect of amplitude amplification. Repeating the operation with the oracle call O maximizes the amplitudes of the desired states. The number of Grover iterations R required for precisely obtaining a solution for a search problem was proven by M. Boyer et al. [17] to satisfy:

$$R \approx \left\lceil \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rceil. \tag{6}$$

```
procedure GenSAT(Π, Max-tries, Max-flips, p)
  for i := 1 to Max-tries
    T := initial(Π); generate an initial truth assignment
      for j := 1 to Max-flips
        if T satisfies Π then return T
        else Poss-flips := select(Π, T, p); select set of vars to pick from
          V := pick(Post-flips); pick one
          T := T with V's truth assignment flipped
      end
  end
  return "no satisfying assignment found"
function select(Π, T, p)
  if Random[0, 1] < p
    then all variables in unsatisfied clauses
else hclimb(Π, T); compute "best" local neighbors
```

Fig. 2. The GenSAT procedure.

2.2. Traditional algorithms on satisfiability

Two kinds of traditional approaches to Boolean satisfiability problems are introduced in this section. One is the local search algorithm of GenSAT family, and the other is the evolutionary algorithm. GenSAT, a framework of a generalized hill-climbing procedure, is given in Fig. 2 [1]. GenSAT has four parameters: Π, Max-tries, Max-flips and p. Π is the formula to satisfy; Max-tries is the maximum number of restarts and is usually set as large as possible; Max-flips is the maximum number of flips before a restart; and p is the probability of performing a random walk. The function *hclimb* implements the hill-climbing of GenSAT algorithm and evaluates the best local neighbors. GSAT is a particular instance of GenSAT in which there is no random walk, and is very good at solving hard random problems. Given a formula in conjunctive normal form, GSAT computes a randomly generated truth assignment, and hill-climbs by repeatedly flipping the variable assignment which most increases the number of clauses satisfied. If there is a choice between several equally good flips, GSAT picks one at random. If there are no upward flips, GSAT makes a sideways flip. Although there are several different variants of GSAT including CSAT, TSAT, DSAT and HSAT [1], and those with random walk CRSAT, TRSAT, DRSAT and HRSAT [2], we focus on the original GSAT to construct the QCS algorithm.

Evolutionary algorithms are principally probabilistic search techniques and optimization methods based on the principles of natural biological evolution. Compared to the traditional optimization methods, EAs are robust, global and can be generally applied without recourse to domain-specific heuristic. All evolutionary algorithms have the same basic structure: iterations of competitive selection and random variation. Although there are several varieties of EAs, they all follow the format [18] as in Fig. 3. The evaluation function for an individual returns a numeric value representing the quality of the solution described by that individual. This numeric value is often called the fitness of the individual while the evaluation function is called the fitness function. High fitness means the associated individual represents a good solution to the given problem.

Evolutionary algorithms have already been applied to 3-SAT [4]. The bit representation is a natural choice for these EAs. Each variable in a 3-CNF formula can be represented by a gene with value 0 or 1. The fitness function of an individual is usually designed to output the number of unsatisfied clauses. However, because the fitness landscape is extremely flat, it is difficult to define fitness in a meaningful way. Michalewicz [19] proposed a new approach for circumventing the problem of defining the fitness function, based on a floating point representation. An example is given to show this approach. Suppose we have the following 3-CNF formula:

$$F(x) = (x_1 \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_2} \vee x_4 \vee \overline{x_5}) \wedge (x_1 \vee \overline{x_2} \vee x_4). \tag{7}$$

Each Boolean variable x_i is transformed into floating point number y_i and the evaluation function is calculated as follows:

$$f(Y) = (y_1 - 1)^2(y_3 - 1)^2(y_4 + 1)^2 + (y_2 + 1)^2(y_4 - 1)^2(y_5 + 1)^2 + (y_1 - 1)^2(y_2 + 1)^2(y_4 - 1)^2. \tag{8}$$

The literal x_i is replaced by $(y_i - 1)^2$, and $\overline{x_i}$ is replaced by $(y_i + 1)^2$. The value assigned to y_i ranges from -1 to 1, which is different from the original 0/1 assignment of x_i. The logical symbol $\vee$ is replaced by the arithmetical

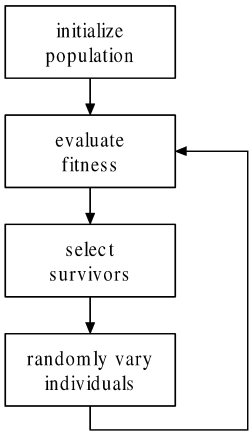

Fig. 3. The canonical EA.

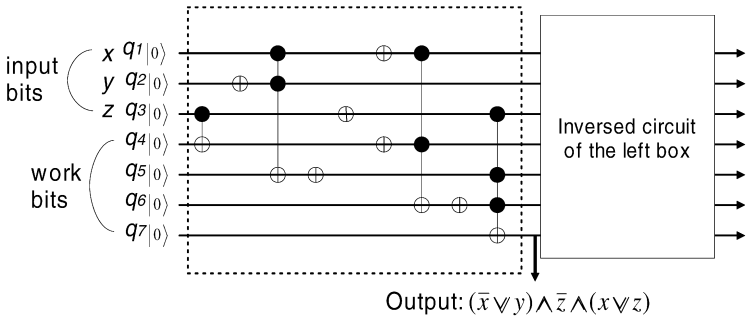

Fig. 4. The oracle circuit for a single-target search.

operator "product" and the symbol $\wedge$ is replaced by the arithmetical operator "plus." The resulting fitness function has a minimum value of 0, when the correct assignment of a formula is found. To check whether a chromosome is a solution to a problem, it is convenient to round positive values to 1 and negative values to -1. By the floating point representation, 3-SAT problems are transformed to become optimization problems, and many efficient evolutionary strategies are well suited for these kinds of problems.

3. Quantum cooperative search on 3-SAT

In reality, since the quantum computer by current techniques can be equipped with few qubits only. The lack of high quantity of stable qubits is the most rigorous challenge to realize the quantum search algorithm. In fact, the number of qubits for quantum search is destined to be limited nowadays due to the following implementation reasons.

- The oracle size: The complexity of an oracle depends on the number of qubits inputted to the oracle, and the performance of Grover's algorithm rests on the number of repeated oracle calls. Thus a large amount of qubits will cause low performance when performing quantum search. Fig. 4 depicts the circuit expressing the formula $(\bar{x} \vee y) \wedge \bar{z} \wedge (x \vee z)$. It also illustrates the possible construction of the oracle and the structure-correlation between the oracle and qubits.
- The reliability of quantum states: The measurement of quantum states must be performed before the quantum states get out of control. However, the cost and complexity on controlling quantum states increase with the number of qubits in a system. Accordingly, a reasonable amount of qubits maintains the reliability of quantum states and leads to the correctness of Grover's search algorithm.

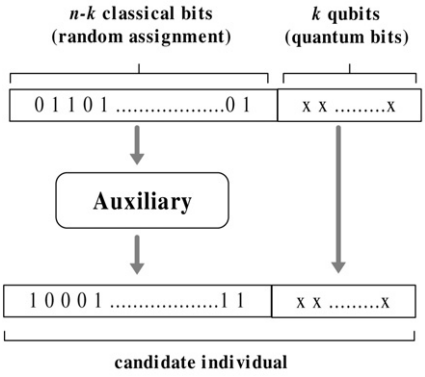

Fig. 5. The illustration of a candidate individual.

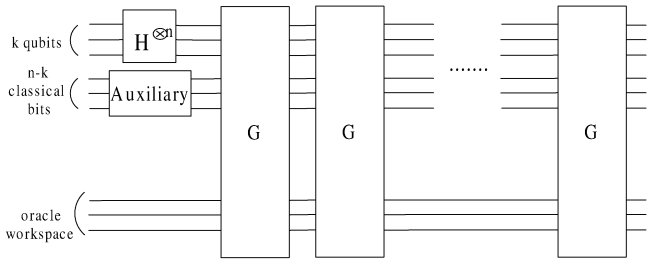

Fig. 6. Circuit of the QCS algorithm.

Due to the lack of qubits, Grover's search algorithm requires some *auxiliaries* to perform search in an enormous database. In the case of 3-SAT problems, we propose the QCS algorithm for quantum systems to solve 3-CNF formulas with small number of qubits by taking advantage of the auxiliaries. Auxiliaries are the materializations of the traditional algorithms on satisfiability described in Section 2, and they prepare some preprocess for Grover's search in order to circumvent the lack of qubits. As shown in Fig. 5, an individual (a bit-assignment) is divided into the *classical-bit* and *quantum-bit* sections during these preprocess. The assignments for the bits on classical-bit section are computed by auxiliaries on classical computers. Then the best assignment to classical-bit section called *candidate individual* is evaluated. Grover's search algorithm searches for the solutions residing in the candidate individual by quantum mechanisms.

Let M be the number of solutions to a formula with an input candidate individual. To find a solution for the formula, Grover's search algorithm needs $O(\sqrt{N/M})$ oracle calls even if M is unknown [17]. In the case of $M = 0$, a time-out condition is applied in Grover's search algorithm to let the algorithm run with $O(\sqrt{N})$ Grover iterations. Figure 6 shows the circuit implementing the QCS algorithm. This circuit is similar to the circuit of Grover's search algorithm in Fig. 1. The difference is that the circuit of the QCS algorithm has classical input bits generated by auxiliaries to provide sufficient quantity of bits for the search space. Therefore, the oracle function in Grover's iteration has to recognize the input data mixed by quantum and classical bits as shown in Fig. 7.

The whole procedure of the QCS algorithm on 3-SAT is shown in Fig. 8. The loop in the figure is regarded as a try in the QCS algorithm. The termination condition in the flowchart is a time bound of a run (a run may consist of many tries). If the termination condition is reached during a run, this run fails to find a solution. This time bound in our algorithm is the maximal number of queries performing on oracle calls and fitness functions.

Two different auxiliaries are proposed in this paper. One is a GenSAT-based auxiliary and the other is an EA-based auxiliary. For an example in which a 3-CNF formula with n literals (variables) is evaluated in a k-bits quantum computer and the GenSAT auxiliary is employed as the auxiliary. GenSAT generates a truth assignment of the first $n - k$ variables randomly, repeatedly flips the variable assignment, and hill-climbs the assignment which most increases the number of clauses satisfied. A variant of GenSAT is proposed in this paper to improve the performance of GenSAT on assisting Grover's algorithm. This variant is modified from the original GenSAT to hill-climb the assignment which most decreases the number of clauses unsatisfied. These unsatisfied clauses do not include the clauses in which the

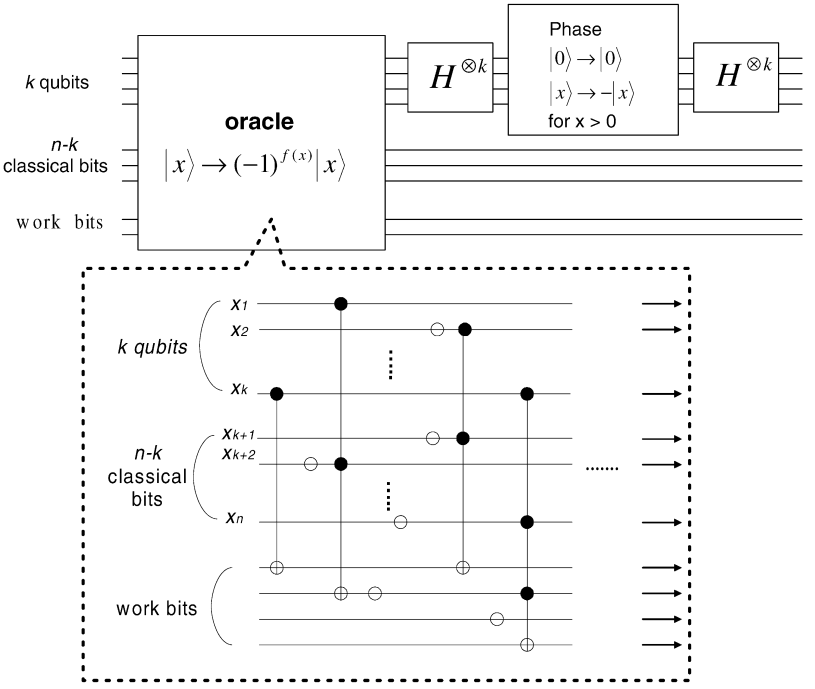

Fig. 7. Circuit of the Grover iteration in the QCS algorithm.

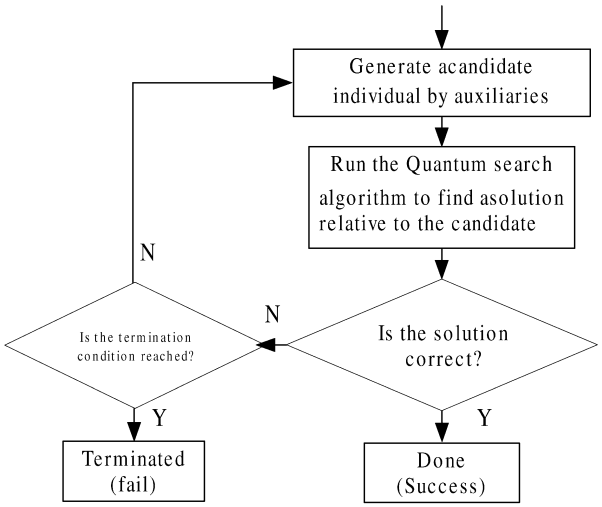

Fig. 8. Flowchart of the QCS algorithm.

qubit-variables are involved. An example is given in Table 1 to illustrate the difference between the modified GenSAT and GenSAT. The performance of the QCS algorithm equipped with the modified GenSAT is evaluated in Section 4 by conducting simulations.

On the other hand, the EA-based auxiliary works on the formula transformed from the original 3-CNF formula by Michalewicz's method. Let us consider an example for illustration; $F(X) = (x_1 \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_2} \vee x_4 \vee \overline{x_5}) \wedge (x_1 \vee \overline{x_2} \vee x_4)$ is a 3-CNF formula and the variables x_4 and x_5 are qubit-variables. According to Michalewicz's method, the evaluation function of $F(X)$ is $f(Y) = (y_1 - 1)^2(y_3 - 1)^2 + (y_2 + 1)^2 + (y_1 - 1)^2(y_2 + 1)^2$, where the value assigned to y_i changes from the range of 0 to 1 (for x_i) to the range of -1 to 1. The aim of the EA-based auxiliary is to seek the minimal fitness value to the transformed formula. Moreover, we make some modifications to Michalewicz's

Table 1
An example illustrates the difference between the modified GSAT and original GSAT

	A 3-SAT formula $F(X) = (x_1 \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_2} \vee x_4 \vee \overline{x_5}) \wedge (x_1 \vee \overline{x_2} \vee x_4) \wedge (x_3 \vee x_4 \vee x_6)$, x_1, x_2, x_3 and x_4 are classical-bit variables; x_5 and x_6 are quantum-bit variables.
Original GSAT	If the assignment is $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0$: $(x_1 \vee x_3 \vee \overline{x_4}) = 1$, $(\overline{x_2} \vee x_4 \vee \overline{x_5}) = 0$, $(x_1 \vee \overline{x_2} \vee x_4) = 1$, $(x_3 \vee x_4 \vee x_6) = 0$, thus the fitness value (the number of satisfied clauses) $= 2$.
Modified GSAT	If the assignment is $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0$: $(x_1 \vee x_3 \vee \overline{x_4}) = 1$, $(\overline{x_2} \vee x_4 \vee \overline{x_5}) = 0$, $(x_1 \vee \overline{x_2} \vee x_4) = 1$, $(x_3 \vee x_4 \vee x_6) = 0$, thus the fitness value (the number of unsatisfied clauses that have no qubit-variable) $= 0$.

```
procedure Cooperative(Π, Max-queries, Num-Generations)
  for queries := 0 to Max-queries
    Candidate := initial(Π); generate an initial truth assignment in the classical-bit section
    for i := 1 to Num-Generations
      Candidate := Gen_SAT(Π, Candidate); select the best candidate by GenSAT
      queries := queries + num_children; num_children is the number of children generated by a
parent
    end
    R := QCSearch(Π, Candidate)
    queries := queries + O(2^num_q/2); num_q is the number of qubits
    if R satisfies Π then return R
  end
  return "no satisfying assignment found"
function Gen_SAT(Π, Candidate); auxiliary function
return hclimb(Π, Candidate); compute "best" local neighbors with the defined fitness function
```

Fig. 9. The procedure of the QCS algorithm based on GenSAT.

method: all variables except qubit-variables are assigned with randomly floating point values and evolve into high-fitness assignments. The values of qubit-variables are assigned to be zero during the whole evolution process. We conduct simulations to evaluate the performance of the QCS algorithm equipped with an asexual EA in Section 4.

The pseudo code of the QCS algorithm with the GenSAT-based auxiliary is shown in Fig. 9. There are three input parameters in the code: Π, Max-queries and Num-Generations. Π is the formula to be satisfied; Max-queries is the maximum number of queries performing on fitness functions and oracle calls; Num-Generations is the maximum number of flips performed for evolving a candidate individual. The function QCSearch performs Grover's search on the formula with a candidate individual. Since each auxiliary decides its owned auxiliary function, if we replace GenSAT auxiliary with other auxiliaries in the pseudo code, only the auxiliary function needs to be redesigned.

3.1. The selection on qubits

In this subsection, we present how to select qubits-variables among the variables in a 3-CNF formula to optimize the QCS algorithm.

Definition 1. The *crash assignment* to a variable is the assignment of other variables which lets a formula unsatisfied no matter what value is assigned to this variable.

For example, here is a formula: $f(x) = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$. If x_2 is assigned with 0 and x_3 is assigned with 1, then either the assignment $(x_1 = 0, x_2 = 0, x_3 = 1)$ or the assignment $(x_1 = 1, x_2 = 0, x_3 = 1)$ lets $f(x)$ unsatisfied. Therefore, we say that $(x_2 = 0, x_3 = 1)$ is a crash assignment to variable x_1 for formula $f(x)$.

Definition 2. The *crashing probability* of a variable is the probability that an assignment excluded this variable is a crash assignment.

The crashing probability of variable x_1 in the above example is $1/4$ since among the four assignments: $(x_2 = 0, x_3 = 0)$, $(x_2 = 0, x_3 = 1)$, $(x_2 = 1, x_3 = 0)$, $(x_2 = 1, x_3 = 1)$, only the assignment $(x_2 = 0, x_3 = 1)$ is a crash assignment.

Definition 3. The *positive appearance* of variable x_i is the number of times that x_i appears in a 3-CNF formula while the *negative appearance* is the number of times that $\overline{x_i}$ appears in the formula. The *appearance* of a variable is the sum of its positive appearance and negative appearance.

In the example of $f(x) = (x_2 \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$, the appearance of variable x_1 and x_4 is both 1 while the appearance of variable x_2 and x_3 is both 2. Variable x_3 has positive appearance equal to 1 and negative appearance equal to 1.

Theorem 1. *When applying QCS algorithm to* 3*-SAT, selecting the variables having smaller appearance as qubit-variables has higher success probability on finding a solution than selecting the variables having larger appearance. If two variables are equal in appearance, selecting the one which has larger value of |positive appearance–negative appearance| as qubit-variable will have higher success probability on finding a solution.*

Selecting the variables with less crashing probability as qubit-variables increases the probability of successfully searching solutions by Grover's algorithm. Thus we are going to prove that a variable which has smaller appearance in a formula has less crashing probability.

Proof. When evaluating the crashing probability of the variable x_i in a 3-CNF formula, we assume the disjunction of the other two variables in the clause where x_i or $\overline{x_i}$ is involved is equal to 1 or 0 with equal probability. Let p_i denote the positive appearance of x_i, g_i denote the negative appearance of x_i, and $S_i = p_i + g_i$. If an assignment lets the disjunction of the other two variables in one of the clauses where x_i is involved be 0 and lets the disjunction of the other two variables in one of the clauses where $\overline{x_i}$ is involved be 0, this assignment consequently is a crashing assignment to variable x_i. Thus we obtain the crashing probability of x_i as follows according to the above assumption:

$$P_{c_i} = \Pr\{\textit{Crashing probability of } x_i\} = \left[1 - \left(\frac{1}{2}\right)^{p_i}\right] \cdot \left[1 - \left(\frac{1}{2}\right)^{g_i}\right] = 1 - \frac{2^{p_i} + 2^{g_i} - 1}{2^{p_i + g_i}}. \tag{9}$$

We fix the value of p_i ($2^{p_i} = c$, c is a constant) and observe the influence of g_i on the crashing probability of x_i. The following equation is obtained from (9):

$$P_{c_i} = 1 - \frac{c + 2^{g_i} - 1}{c \cdot 2^{g_i}} = 1 - \left(\frac{c - 1}{c \cdot 2^{g_i}} + \frac{1}{c}\right). \tag{10}$$

When $c > 1$, it is obvious that increasing the value of g_i increases the crashing probability of x_i. On the other hand, if we fix the value of g_i and observe the influence of p_i on P_{c_i}, we also conclude that a smaller p_i will have a smaller P_{c_i}. Thus the proof that a variable having smaller appearance in a formula has less crashing probability is completed.

Moreover, to prove that selecting the variable having larger value of |*positive appearance–negative appearance*| as qubit-variable is appropriate when two or more variables are equal in appearance, let D_i denote the value of $|g_i - p_i|$. We obtain another expression for P_{c_i} as follows:

$$P_{c_i} = 1 - \frac{2^{\frac{S_i + D_i}{2}} + 2^{\frac{S_i - D_i}{2}} - 1}{2^{S_i}} = 1 - \frac{2^{\frac{S_i}{2}} \cdot (2^{\frac{D_i}{2}} + 2^{-\frac{D_i}{2}}) - 1}{2^{S_i}}. \tag{11}$$

Since $D_i \geqslant 0$, a larger D_i has a larger value of $2^{\frac{D_i}{2}} + 2^{-\frac{D_i}{2}}$. Thus a smaller P_{c_i} is obtained by a larger D_i when the value of S_i is fixed. It is proved that selecting the variable having larger D_i as qubit-variable is appropriate when two or more variables are equal in appearance. The proof of Theorem 1 is completed. □

3.2. The best number of qubits

The performance of evolutionary algorithms and heuristic search algorithms is always evaluated by the number of queries to fitness functions while the performance of quantum algorithms is evaluated by the number of oracle

calls. Thus the performance of the QCS algorithm can be evaluated by the sum of the number of queries to fitness functions and the number of oracle calls. In this subsection, we discuss how many qubits are required to achieve the best performance for the QCS algorithm.

Theorem 2. *Assume that the maximal number of generations (flips) performed on auxiliaries in a try is proportional to the number of classical bits, i.e., the number of classical bits* $\times\ r =$ *the maximal number of generations (flips), where* r *is a constant. The QCS algorithm has the minimal number of queries when the number of qubits* n_q *is equal to*

$$\frac{n\ln 2 - 2W(\frac{2^{-5+n/2}\pi(\ln 2)^2}{r})}{\ln 2},$$

where $W(z)$ *is a* ProductLog *function which gives the principal solution for* w *in* $z = we^w$.

Proof. Let $S_Q(n_q)$ denote the number of queries (to fitness functions and oracle calls) in a try for different values of n_q. $S_Q(n_q)$ is obtained as follows:

$$S_Q(n_q) = (n - n_q)\cdot r(n - n_q) + \frac{\pi}{4}\cdot 2^{\frac{n_q}{2}}. \tag{12}$$

$S_Q(n_q)$ has a minimal value when its derivative at a certain n_q is equal to 0. Let $S'_Q(n_q)$ denote the derivative of $S_Q(n_q)$, then the following equation is obtained:

$$S'_Q(n_q) = 2r(n_q - n) + \frac{\pi}{4}\cdot \ln 2\cdot 2^{\frac{n_q}{2}-1}. \tag{13}$$

We solve the equation $S'_Q(n_q) = 0$ and obtain n_q as follows:

$$n_q = \frac{n\ln 2 - 2W(\frac{2^{-5+n/2}\pi(\ln 2)^2}{r})}{\ln 2}. \tag{14}$$

The proof of Theorem 2 is complete. □

Let us consider a 3-CNF formula with 50 variables for example. By (14), if r is equal to 1, the best number of qubits is equal to 16 (The *ProductLog* function can be calculated by some mathematical applications such as *Mathematica*). Thus the number of queries in a try is equal to 1412. If we solve this formula by GenSAT algorithms, the maximal number of queries in a try is equal to 2500. The QCS algorithm cuts down almost 1100 queries when comparing to GenSAT algorithms.

4. Experimental results

Several experiments are conducted in this section to evaluate the performance of the QCS algorithm. These experiments are demonstrated and described in two subsections. Section 4.1 compares the performance of the QCS algorithm with that of EA and heuristic search algorithms. Section 4.2 discusses the impact of the number of qubits on the QCS algorithm.

4.1. The performance of the QCS and traditional 3-SAT algorithms

To evaluate the performance of the QCS algorithm based on different auxiliaries, three kinds of auxiliaries based on different traditional algorithms are considered in the experiments. One auxiliary is based on the exhaustive search algorithm which exhaustedly prepares the candidate individuals, while quantum search algorithm handles the rest (the quantum-bit section) for it. This kind of auxiliary is called *striving* auxiliary. The other two auxiliaries are based on the GenSAT algorithm and EA, respectively. Rather than hill-climbing to the neighbor which most increases the number of clauses satisfied as original GenSAT does, the GenSAT-based auxiliary hill-climbs to the neighbor which most decreases the number of clauses unsatisfied. The Max-flips of the GenSAT-based auxiliary is equal to the number of classical-variables. The EA-based auxiliary is an asexual $(1, 10)$-EA without combination processes. It is based on the *self-adaptation of standard deviation* mutation mechanism with standard deviation upper-bounded by 3.0. The

Table 2
3-SAT problem instances

Test case	Clause length	n	l	Random seed
No. 1–30	3	25	108	Random
No. 31–60	3	30	129	Random
No. 61–90	3	35	151	Random
No. 91–120	3	40	172	Random
No. 121–150	3	50	215	Random
No. 151–180	3	80	344	Random

Table 3
Success rate (SR) for different test cases

Test case	QCS with striving	GenSAT	QCS with GenSAT	Asexual EA-(1, 10)	QCS with asexual EA-(1, 10)
	Avg. SR	Avg. SR	Avg. SR	Avg. SR	Avg. SR
No. 1–30	1.0	1.0	**1.0**	1.0	**1.0**
No. 31–60	1.0	1.0	**1.0**	1.0	**1.0**
No. 61–90	0.0	1.0	**1.0**	0.83	**1.0**
No. 91–120	0.0	1.0	**0.94**	0.86	**0.79**
No. 121–150	0.0	0.96	**0.89**	0.77	**0.76**
No. 151–180	0.0	0.62	**0.68**	0.26	**0.33**

maximal number of generations before a restart in the EA-based auxiliary is equal to the number of classical-variables. In addition, the original GenSAT algorithm and EA algorithm are simulated as pure traditional algorithms on solving 3-SAT for comparison. The configurations for the pure GenSAT and EA algorithms are the same as the GenSAT-based and EA-based auxiliaries.

We use the generator mkcnf.c by Allen Van Gelder [20] that is loosely based on mwff.c by Bart Selman to generate the problem instances (180 instances) listed in Table 2, where l presents the number of clauses and n presents the number of variables in a formula. The problem instances we use are forced to be satisfiable to present our experiments. Mitchell et al. [21] reported that the phase transition, where the hardest problem instances were located, was found when $l = 4.3 \cdot n$. For this reason, the QCS algorithm and traditional 3-SAT algorithms are tested on the problem instances with $l = 4.3 \cdot n$. n is assigned with values of 25, 30, 35, 40, 50, 80, respectively. For each problem instance, we use Theorem 2 (Section 3.2) to decide the number of qubits and Theorem 1 (Section 3.1) to select the qubit-variables for the QCS algorithm.

To evaluate the performance the algorithms, we use *success rate* (SR) and *average number of queries* (ANQ) as the measures. We run each algorithm 50 times (one time is called a run) on each problem instance to determine SR. SR is the percentage of all runs where a solution is found. A run is terminated and regarded as a failure run if the number of queries performed on this run exceeds 10^6. ANQ is the average number of queries at termination in those success runs. The number of queries includes the number of queries to fitness functions and the number of oracle calls since we assume a query to the oracle call is equal to a query to the fitness function in terms of computation complexity. When counting the number of oracle calls, we do not really implement the quantum circuit or Grover's algorithm. Instead, we estimate the required oracle calls for solving a formula with a candidate individual by (6).

The experimental results in terms of SR and ANQ are exhibited in Tables 3 and 4, respectively. With respect to SR, the QCS algorithm with the GenSAT-based auxiliary is slightly worse than the pure GenSAT in cases No. 1–150, but is better than the pure GenSAT in cases No. 151–180. Similarly, the same phenomenon can be obtained by comparing the QCS algorithm with the EA-based auxiliary with the pure EA-(1, 10) algorithm. Thus we infer that the QCS algorithm has better SR than pure 3-SAT algorithms have when the problem instance has a large number of variables. As for ANQ, the QCS algorithm with the GenSAT-based auxiliary obviously has a smaller number of queries than the pure GenSAT has. However, the QCS algorithm with the EA-based auxiliary is not superior to the pure EA-(1, 10) in terms of ANQ. Accordingly, we conclude that the GenSAT algorithm is better than the EA as being an auxiliary of the QCS algorithm.

Table 4
Average number of queries to solution (ANQ) for different test cases

Test case	QCS with striving	GenSAT	QCS with GenSAT	Asexual EA-(1, 10)	QCS with asexual EA-(1, 10)
	ANQ	ANQ	ANQ	ANQ	ANQ
No. 1–30	179	11 998	**9389**	31 982	**52 118**
No. 31–60	14 982	15 432	**13 371**	133 077	**131 301**
No. 61–90	–	62 701	**54 711**	108 481	**157 366**
No. 91–120	–	85 011	**64 732**	224 811	**248 910**
No. 121–150	–	473 892	**159 127**	423 150	**405 799**
No. 151–180	–	713 749	**325 420**	827 129	**754 027**

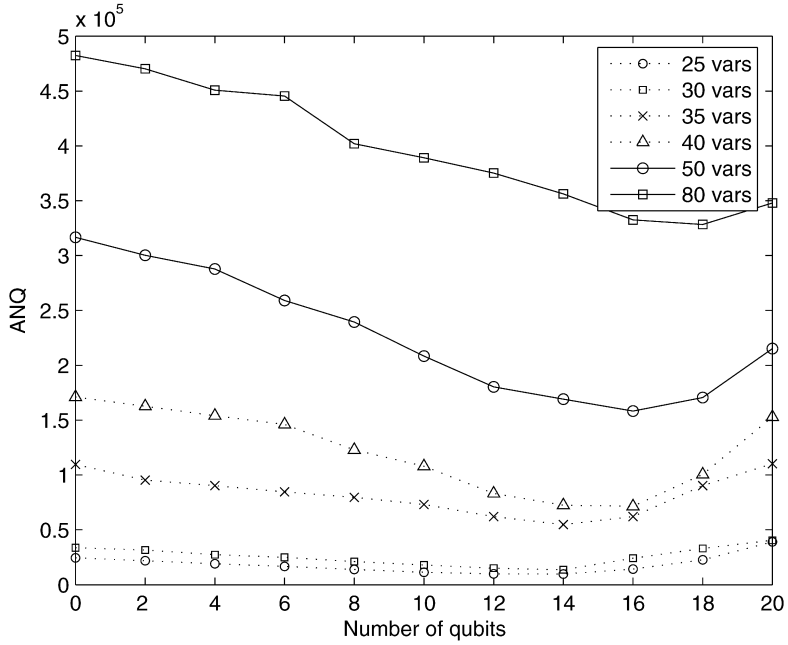

Fig. 10. ANQ of the QCS algorithm on different number of qubits.

4.2. The impact of the quantity of qubits

In Theorem 2, we analyze the best number of qubits for the QCS algorithm to solve a 3-CNF formula. Furthermore, we verify Theorem 2 by conducting simulations in this section. We vary the number of qubits (k) and observe the impact of the number of qubits on the QCS algorithm. Since we conclude that GenSAT algorithm is better than EA algorithm as being an auxiliary in Section 4.1, GenSAT is applied as the auxiliary in this scenario.

As shown in Fig. 10, the smallest ANQ of each test case resides on the qubit quantity that is close to the optimized value derived by Theorem 2 (25 vars = 13 qubits, 30 vars = 14 qubits, 35 vars = 14 qubits, 40 vars = 15 qubits, 50 vars = 16 qubits, 80 vars = 18 qubits). Similarly, the highest SR of each test case shown in Fig. 11 also resides on the qubit quantity derived by Theorem 2. We say that the experimental results match the mathematical analysis in Theorem 2.

5. Conclusions

It is well known that Grover's algorithm can find a solution in a search space of size N in time $O(\sqrt{N})$. The proposed QCS algorithm provides Grover's search algorithm the capability of solving 3-SAT problems with a small number of qubits. This is done by replacing some qubits with classical bits and finding assignments to these classical

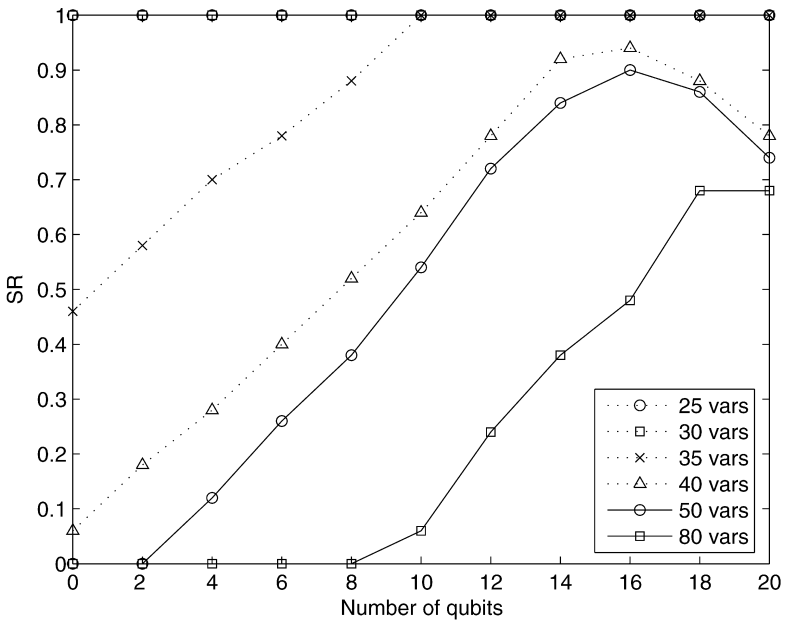

Fig. 11. SR of the QCS algorithm on different number of qubits.

bits using 3-SAT traditional algorithms. The optimal configuration for the QCS algorithm is suggested by mathematical analysis. In the experiments, the QCS algorithm with various kinds of auxiliaries is compared with the traditional 3-SAT algorithms. The experimental results show that the QCS algorithm with GenSAT-based auxiliary has the best performance in terms of query complexity. Moreover, the mathematical analysis on the appropriate number of qubits is verified by the experiments.

Further research will be concerned with the details of constructing the auxiliaries. For example, the adaptation of constraint weights which increases the performance of the EA algorithm will be addressed when constructing the EA-based auxiliaries; the random work of the local search will be adopted when constructing the GenSAT-based auxiliaries. Furthermore, the QCS algorithm will be conducted to handle other combinatorial optimization problems in future work.

References

[1] I. Gent, T. Walsh, Toward an understanding of hill-climbing procedures for SAT, in: Proc. 11th National Conference of AI, AAAI Press, 1993, pp. 28–33.
[2] I. Gent, T. Walsh, Unsatisfied variables in local search, in: J. Hallam (Ed.), Hybrid Problems, Hybrid Solutions, IOS Press, Amsterdam, 1995, pp. 73–85.
[3] J. Frank, Weighting for godot: Learning heuristics for GSAT, in: Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference, AAAI Press, 1996, pp. 338–343.
[4] K.A. de Jong, W.M. Spears, Using genetic algorithms to solve NP-complete problems, in: Third International Conference on Genetic Algorithms, George Mason University, Fairfax, VA, 1989, pp. 124–132.
[5] T. Bäck, A.E. Eiben, M.E. Vink, A superior evolutionary algorithm for 3-SAT, in: Proceedings of the 7th International Conference on Evolutionary Programming VII , Springer-Verlag, March 25–27, 1998, pp. 125–136.
[6] P. Benioff, The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by Turing machines, J. Stat. Phys. 22 (1980) 563–591.
[7] R. Feynman, Simulating physics with computers, Int. J. Theor. Phys. 21 (1982) 467–488.
[8] D. Deutsch, Quantum theory, the Church–Turing principle and the universal quantum computer, in: Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci., vol. 400, 1985, pp. 97–117.
[9] C. Bennett, G. Brassard, Quantum cryptography: Public key distribution and coin tossing, in: Proc. IEEE Int. Conf. Computers Systems and Signal Processing, IEEE, 1984, pp. 175–179.
[10] P. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in: Proc. 35th Annu. IEEE Symp. Foundations of Computer Science, IEEE, 1994, pp. 124–134.
[11] S.T. Cheng, C.Y. Wang, M.H. Tao, Quantum communication for wireless WAN networks, IEEE Journal of Selected Areas in Communication 23 (7) (2005) 1424–1432.

[12] S.T. Cheng, C.Y. Wang, Quantum switching and quantum merge sorting, IEEE Trans. Circuits Syst. 53 (2) (2006) 316–325.
[13] L. Grover, A fast quantum mechanical algorithm for database search, in: Proc. 28th Annu. ACM Symp. Theory of Computing, ACM Press, 1996, pp. 212–219.
[14] S.A. Cook, The complexity of theorem proving procedures, in: Annual ACM Symposium on Theory of Computing, ACM, 1971, pp. 151–158.
[15] A.K. Zvonkin, L.A. Levin, The complexity of finite objects and the algorithmic concepts of information and randomness, Russian Math. Surveys 25 (6) (1970) 83–124.
[16] S. Perdrix, P. Jorrand, Measurement-based quantum Turing machines and their universality, Quantum Physics (2004), quant-ph/0404146.
[17] M. Boyer, G. Brassard, P. Hoeyer, A. Tapp, Tight bounds on quantum searching, Fortschr. Phys. 46 (1998) 493–506.
[18] G.W. Greenwood, Finding solutions to NP problems: Philosophical differences between quantum and evolutionary search algorithms, in: Proc. of the 2001 Congress on Evolutionary Computation, vol. 2, IEEE, May 2001, pp. 815–822.
[19] Z. Michalewicz, Heuristic methods for evolutionary computation techniques, J. Heuristics 1 (2) (1995) 177–206.
[20] ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/contributed/UCSC/instances/.
[21] D. Mitchell, B. Selman, H.J. Levesque, Hard and easy distributions of SAT problems, in: Proc. of the AAAI, San Jose, CA, AAAI Press, 1992, pp. 459–465.