Layeb, Abdesslem
A Quantum Inspired Particle Swarm Algorithm for Solving the Maximum Satisfiability Problem
International Journal of Combinatorial Optimization Problems and Informatics
Morelos, México

Available in: http://www.redalyc.org/articulo.oa?id=265219741003
A Quantum Inspired Particle Swarm Algorithm for Solving the Maximum Satisfiability Problem

Abdesslem Layeb
MISC Lab., Computer Science Department - Mentouri University Constantine, Algeria.
Layeb.univ@gmail.com

Abstract. In this paper we investigate the use of quantum particle swarm optimization (QPSO) principles to resolve the satisfiability problem. We describe QPSOSAT, a new iterative approach for solving the well known Maximum Satisfiability problem (MAX-SAT). This latter has been shown to be NP-hard if the number of variables per clause is greater than 3. The basic idea is to harness the optimization capabilities of QPSO algorithm to achieve good quality solutions for Max Sat problem. To enhance the efficiency of the QPSO algorithm, a local search has been used. The obtained results are very promising and show the feasibility and effectiveness of the proposed hybrid approach.

Keywords: Maximum Satisfiability problem, Quantum Particle Swarm Optimization, Local search.

1. Introduction

The Boolean Satisfiability problem (SAT) is a well known decision problem. It is defined as the task of determining the satisfiability of a given Boolean formula by looking for the variable assignment that makes this formula evaluating to true. The Maximum Satisfiability (Max-SAT) problem is an optimization variant of SAT problem. The problem consists to find an assignment of truth values that maximizes the number of satisfied clauses of a Boolean formula in Conjunctive Normal Form (CNF). In 1971, Stephen Cook [1] had demonstrated that the Max-SAT problem is NP-hard. This complex problem has several applications in different areas such as model checking, graph colouring and task planning to cite just few.

Recently, Max-SAT solvers have deeply improved the techniques and algorithms to find optimal solutions. In practice there are two broad classes of algorithms for solving instances of SAT: Complete and Incomplete methods. Complete algorithms are able to verify the satisfiability or unsatisfiability of the SAT problem. They usually have an exponential complexity [2]. The most popular algorithms of this class are based on the Davis-Putnam-Loveland algorithm (DPLL) [3]; for example, a branch and bound algorithm based on DPLL is one of the most competitive exact algorithms for Max-SAT [4]. On the other hand, incomplete methods are principally based on local search and evolutionary algorithms. Incomplete methods find good quality solutions in reasonable time. Therefore, they don’t guarantee optimality. This class of methods encompasses Evolutionary Algorithms (EA) [5], Stochastic Local Search (SLS) methods [6, 7] and hybrid methods [8].

Evolutionary computation has been proven to be an effective way to solve complex engineering problems. It presents many interesting features such as adaptation, emergence and learning. Artificial neural networks, genetic algorithms and swarm intelligence are examples of bio-inspired systems used to this end. In recent years, optimizing by swarm intelligence has become a research interest to many research scientists of evolutionary computation fields. There are many algorithms based...
swarm intelligence like Ant Colony optimization, eco-systems optimization, etc. The main algorithm for swarm intelligence is Particle Swarm Optimization (PSO) [9-10], which is inspired by the paradigm of birds grouping. PSO was used successfully in various hard optimization problems. The simplicity of implementation and use are the main features of PSO compared to other evolutionary computing algorithms. In fact, the updating mechanism in the algorithm relies only on two simple PSO self-updating equations, so CPU-time cost of one generation in PSO algorithm is less than reproduction mechanism using mutation or crossover operations in typical EA.

Far from SAT problem, Quantum Computing (QC) is a new research field that induced intense researches in the last decade, and that covers investigations on quantum computers and quantum algorithms [11]. QC relies on the principles of quantum mechanics like qubit representation and superposition of states. QC is able of processing huge numbers of quantum states simultaneously in parallel. QC brings new philosophy to optimization due to its underlying concepts. Recently, a growing theoretical and practical interest is devoted to researches on merging evolutionary computation and quantum computing [12, 13]. The aim is to get benefit from quantum computing capabilities to enhance both efficiency and speed of classical evolutionary algorithms. This has led to the design of Quantum inspired Evolutionary Algorithms QEA that have been proven to be better than conventional GAs. Unlike pure quantum computing, QEA doesn’t require the presence of a quantum machine to work. Quantum particle Swarm Optimization (QPSO) is one of the recent methods based on quantum computing principles and PSO algorithm. QPSO has been used to solve successfully many combinatorial optimization problems [14, 15].

Within this issue, we propose in this paper a new framework to cope with the MAX SAT problem. The proposed approach called QPSOSAT is based on a hybrid QPSO algorithm and local search method. The features of the proposed method consist in applying different QPSO principles to govern the dynamics of the population in a way to optimize a defined objective function. To foster the convergence to optimality, a local search has been embedded within the QPSO optimization process. The experiments carried out on QPSOSAT showed the feasibility and the effectiveness of our approach.

The remainder of the paper is organized as follows. In section 2, a formulation of the tackled problem is given. Section 3 presents some basic concepts of quantum computing. Section 4 presents an overview of the Particle Swarm Optimization. In section 5, the proposed method is described. Experimental results are discussed in section 6. Finally, conclusions and future work are drawn.

2. Problem formulation

Given a Boolean formula $F$ expressed in CNF (Conjunctive Normal Form) and having $n$ Boolean variables $x_1, x_2, ..., x_n$, and $m$ clauses. The $k$-SAT problem can be formulated as follows:

- An assignment to those variables is a vector $v = (v_1, v_2, ..., v_n) \in \{0,1\}^n$
- A clause $C_i$ of length $k$ is a disjunction of $k$ literals, $C_i = (x_1 \lor x_2 \lor ... \lor x_k)$
- Each literal is a variable or a negation of a variable
- Each variable can appear multiple times in the expression.

For some constant $k$, the $k$-SAT problem requests a variable assignment that makes a formula $F = C_1 \land C_2 \land ... \land C_m$ evaluate to true.

Max-SAT is the problem of finding the assignment that satisfies the highest possible number of clauses. Therefore, it is categorized as an optimization problem. The Max-SAT problem can be defined by specifying implicitly a pair $(\Omega, SC)$, where $\Omega$ is the set of all potentials solution ($\{0, 1\}^n$) and SC is a mapping $\Omega \rightarrow N$, called score of the assignment, equal to the number of true clauses. Consequently, the problem consists of defining the best binary assignment that maximizes the
number of true clauses in the Boolean formula. Clearly, there are \(2^n\) potential satisfying assignments for this problem, and it has been proven that the k-SAT problem is NP-complete for any \(k \geq 3\). There are other variances of the Max-SAT problem such as Weighted Max-SAT [16] and Partial Max-SAT [17].

In this paper we deal with the Max-3-SAT problem. It is a combinatorial optimization problem. Therefore, it is impossible to obtain exact solutions in polynomial time as the required computation grows exponentially with the size of the problem.

3. An Overview of Quantum Computing

Quantum computing is a new theory which has emerged as a result of merging computer science and quantum mechanics. Its main goal is to investigate all the possibilities a computer could have if it followed the laws of quantum mechanics. The origin of quantum computing goes back to the early 80 when Richard Feynman observed that some quantum mechanical effects cannot be efficiently simulated on a computer. During the last decade, quantum computing has attracted widespread interest and has induced intensive investigations and research since it appears more powerful than its classical counterpart. Indeed, the parallelism that the quantum computing provides reduces obviously the algorithmic complexity. Such an ability of parallel processing can be used to solve combinatorial optimization problems which require the exploration of large solutions spaces. The basic definitions and laws of quantum information theory are beyond the scope of this paper. For in-depth theoretical insights, one can refer to [11].

The qubit is the smallest unit of information stored in a two-state quantum computer. Contrary to classical bit which has two possible values, either 0 or 1, a qubit will be in the superposition of those two values. The state of a qubit can be represented by using the braket notation:

\[
|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle
\]

(1)

Where \(|\Psi\rangle\) denotes more than a vector \(\vec{q}\) in some vector space. \(|0\rangle\) and \(|1\rangle\) represent respectively the classical bit values 0 and 1. \(\alpha\) and \(\beta\) are complex number that specify the probability amplitudes of the corresponding states. When we measure the qubit’s state we may have ‘0’ with a probability \(|\alpha|^2\) and we may have ‘1’ with a probability \(|\beta|^2\). A system of m-qubits can represent \(2^m\) states at the same time. Quantum computers can perform computations on all these values at the same time. It is this exponential growth of the state space with the number of particles that suggests exponential speed-up of computation on quantum computers over classical computers. Each quantum operation will deal with all the states present within the superposition in parallel. When observing a quantum state, it collapses to a single state among those states.

Quantum Algorithms consist in applying successively a series of quantum operations on a quantum system. Quantum operations are performed using quantum gates and quantum circuits. It should be noted that designing quantum algorithms is not easy at all. Yet, there is not a powerful quantum machine able to execute the developed quantum algorithms. Therefore, some researchers have tried to adapt some properties of quantum computing in the classical algorithms. Since the late 1990s, merging quantum computation and evolutionary computation has been proven to be a productive issue when probing complex problems [12,13].

4. Classical and Quantum Particle Swarm Optimization

The Particle Swarm Optimization algorithms (PSO) were introduced in 1995 by Kennedy and Eberhart [10] as an alternative to the standard genetic algorithms. These algorithms are inspired by the swarms of insects (or of the fish benches or the clouds of birds) and their coordinated movements. Indeed, the algorithms based on PSO principles seek solutions to an optimization problem just like the group moves of animals to look for food or to avoid the predatory animals. The individuals of the algorithm are called particles and the population is called swarm. PSO uses a population (swarm) of particles to explore the search space. Each particle represents a candidate solution of an optimization problem and has an associated position, velocity, and memory vector.
In this algorithm, a particle decides on its next movement according to its own experience, which is in this case the memory of the best position than it met, and according to its best neighbor. This vicinity can be defined spatially by taking for example the Euclidean distance between the positions of two particles or socio-metrically (position in the particle swarm). The new speeds and direction of the particle will be defined according to following three tendencies: propensity to go on its own way, its tendency to return towards its best position reached and its tendency to go towards its best neighbor. The algorithms with swarm of particles can apply as well to discrete optimization problems as to continuous problems. The main part of the PSO algorithm can be described formally as follows [9, 10]:

Suppose a swarm of S particles. Each particle consists of three elements:
- The first one, is its position in the search space \( x_i \),
- The second element \( v_i \), expresses the velocity,
- The third element \( P_i \), is its memory, used to store the elite particles of the best global particle \( P_g \) found, as well as the best solutions as found by each individual particle \( P_i \) so far. This vector is often referred to as personal best in the PSO. The number of \( P_i \) particles stored in the memory of the algorithm is equivalent to the number of particles in the swarm.

Unlike replacement methods in conventional genetic algorithms, it’s not required to put always in future populations any elite individual found, although each particle in the population tries to be near to the \( P_g \) and \( P_i \) solutions in the memory by using an updating process governed by PSO weight update equations. Moreover, there is no selection scheme is incorporated in PSO, since the updating of the particles is done by the self-updating equations, and the elite particles \( P_i \) are stored in the memory, to prevent the degeneration of the overall fitness of the particle swarm. The self-updating equations of PSO are as follows:

\[
\begin{align*}
\dot{v}_{id} &= w \cdot v_{id} + C_1 \cdot f_1 \cdot (P_{id} - x_{id}) + C_2 \cdot f_2 \cdot (P_g - x_{id}) \\
\dot{x}_{id} &= x_{id} + v_{id}
\end{align*}
\]

where:
- \( v_{id} \): Represent the velocity of the \( i^{th} \) particle in iteration \( t \).
- \( x_{id} \): Represent the position of the \( i^{th} \) particle in iteration \( t \).
- \( C_1, C_2 \): Learning factors
- \( P_i \): Best position achieved so long by particle \( i \).
- \( P_g \): Best position found by the neighbors of particle \( i \).
- \( f_1, f_2 \): Random factors in the \([0, 1]\) interval.
- \( w \): Inertia weight.

The Quantum inspired Particle Swarm Optimization (QPSO) is one of the recent optimization methods based on quantum mechanics. Like any other evolutionary algorithm, a quantum inspired particle swarm algorithm relies on the representation of the individual, the evaluation function and the population dynamics. The particularity of quantum particle swarm algorithm stems from the quantum representation it adopts which allows representing the superposition of all potential solutions for a given problem. Moreover, the position of a particle depends on the probability amplitudes \( \hat{a} \) and \( \hat{b} \) of the wave function \( \Psi \) (eq.1). QPSO also stems from the quantum operators it uses to evolve the entire population through generations. QPSO constitutes a powerful strategy to diversify the QPSO population and enhance the QPSO’s performance in avoiding premature convergence to local minima [14, 15].
The development of the suggested approach called QPSOSAT is based mainly on a quantum representation of the search space associated with the problem and a QPSO dynamic enhanced by local search procedure used to explore this space by operating on the quantum representation by using quantum operations. In order to show how quantum computing concepts have been tailored to the problem at hand, we need first to derive a representation scheme which includes the definition of an appropriate quantum representation of potential MAX SAT solutions and the definition of quantum operators. Then, we describe how these defined concepts have been integrated in PSO algorithm.

5.1 Quantum representation of MAX 3-SAT

To successfully apply quantum principles on Max 3-SAT problem, we have needed to map potential solutions into a quantum representation that could be easily manipulated by quantum operators. The Boolean assignment is represented as binary vector of size N, where N is the number of Boolean variables in formula F. In terms of quantum computing, each solution is represented as a quantum register of length N as shown in figure 1. Each column represents a single qubit and corresponds to the binary digit 1 or 0. The probability amplitudes \( a_i \) and \( b_i \) are real values satisfying \( \left| a_i \right|^2 + \left| b_i \right|^2 = 1 \). For each qubit, a binary value is computed according to its probabilities \( \left| a_i \right|^2 \) and \( \left| b_i \right|^2 \), which can be interpreted as the probabilities to have respectively 0 or 1. Consequently, all potential solution can be represented by a Quantum Vector QV (fig 1) that contains the superposition of all possible solutions. This quantum vector can be viewed as a probabilistic representation of all the MAX 3-SAT solutions. It plays the role of a particle in the PSO algorithm. A quantum representation offers a powerful way to represent the solution space and reduces consequently the required number of particles. Only one particle is needed to represent the entire population.

\[
\begin{pmatrix}
    a_1 \\
    a_2 \\
    \vdots \\
    a_n \\
    b_1 \\
    b_2 \\
    \vdots \\
    b_n
\end{pmatrix}
\]

Fig.1 Quantum representation of the SAT solution

5.2 Quantum operators

The quantum operations used in our approach are as follows:

5.2.1. The quantum interference: The operation of interference is useful to intensify research around the best solution. This operation amplifies the amplitude of the best solution and decreases the amplitudes of the bad ones. It primarily consists in moving the state of each qubit in the direction of the corresponding bit value in the best solution in progress. This operation can be accomplished by using a unitary quantum operator which achieves a rotation whose angle is a function of the amplitudes \( a_i \), \( b_i \) and of the value of the corresponding bit in the solution reference (fig 2) [8]. The values of the rotation angle \( \delta \theta \) is chosen so that to avoid premature convergence. A big value of the rotation angle can lead to premature convergence or divergence; however a small value to this parameter can increase the convergence time. Consequently, the angle is set experimentally and its direction is determined as a function of the values of \( a_i \), \( b_i \) and the corresponding element’s value in the binary vector (table1). In our algorithm we have set the rotation angle \( \delta \theta = \pi/20 \).
5.2.2. Quantum Flight: this operator allows moving from the current solution to one of its neighbors, in order to enhance the capacity of space exploration. QPSOSAT uses a heuristic flip which consists in flipping the qubits in the quantum particle (fig. 3). The flip is accepted if there is improvement in the number of satisfied clauses. The process is repeated until there is no improvement. Our approach is flexible, so we can use other stochastic local search algorithms.

5.2.3. Measurement: This is very important operation because it allows us to obtain binary particles from quantum particles. This operation transforms by projection the quantum vector into a binary vector (fig. 4). Therefore, there will be a solution among all the solutions present in the superposition. But contrary to the pure quantum theory, this measurement does not destroy the superposition. That has the advantage of preserving the superposition for the following iterations knowing that we operate on traditional machines. The binary values for a qubit are computed according to its probabilities $|a|^2$ and $|b|^2$ [8]. In addition, the measurement operation plays also another role that of a diversification operator. Indeed, two successive measurements do not give necessarily the same solution which increases the diversification capacity of our approach.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>Reference bit value</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0</td>
<td>&gt; 0</td>
<td>1</td>
<td>+δθ</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>&gt; 0</td>
<td>0</td>
<td>-δθ</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>&lt; 0</td>
<td>1</td>
<td>-δθ</td>
</tr>
<tr>
<td>&gt; 0</td>
<td>&lt; 0</td>
<td>0</td>
<td>+δθ</td>
</tr>
<tr>
<td>&lt; 0</td>
<td>&gt; 0</td>
<td>1</td>
<td>-δθ</td>
</tr>
<tr>
<td>&lt; 0</td>
<td>&gt; 0</td>
<td>0</td>
<td>+δθ</td>
</tr>
<tr>
<td>&lt; 0</td>
<td>&lt; 0</td>
<td>1</td>
<td>+δθ</td>
</tr>
<tr>
<td>&lt; 0</td>
<td>&lt; 0</td>
<td>0</td>
<td>-δθ</td>
</tr>
</tbody>
</table>
19

**Input**: a Quantum Particle (QP)

```plaintext
improvement = 1;
repeat
  improvement = 0;
  For i = 1 to nVar
    flip the i-th qubit of QP;
    compute the gain of flip;
    If (gain >= 0)
      accept the flip;
      improvement = improvement + gain;
    End If
  End For
Until (improvement = 0)
```

Fig. 3. Flip heuristic procedure

\[
\left( \begin{array}{cccc}
  a_1 & a_2 & \ldots & a_s \\
  b_1 & b_2 & \ldots & b_n \\
\end{array} \right) \xrightarrow{\text{Measure}} (0,1,\ldots,1)
\]

Fig. 4. Measure operation

6. **The Proposed Approach**

Now, we describe how the representation scheme including quantum representation and quantum operators has been embedded within a QPSO algorithm and resulted in a hybrid stochastic algorithm performing variable truth assignment search. In more details, the proposed QPSOSAT can be described as in figure 5.

**Input**: Boolean formula in CNF form A

1) Initialize the population, positions and velocities
2) Evaluate the fitness of the each particle (Pi)
3) Apply randomly the interference operation on each particle
4) Save the individuals highest fitness (Pg)
5) Modify velocities based on Pi and Pg position
6) Update the particles position using quantum flight
7) Update particle best
8) Update global best
9) Terminate if the condition is met
10) Go to Step 2

**Output**: best variable assignment for A

Fig. 5. QPSOSAT scheme
Given a set $S$ of MAX SAT variables to be assigned, first, a population of quantum particles is created at random positions to represent all possible Boolean assignment. The algorithm progresses through a number of generations according to a quantum PSO based dynamics. During each iteration, the following main tasks are performed. The first operation is the evaluation of the current population of quantum particles. For that, we apply the measure operation in order to get a binary solution which represents the assignment variables. In the Max 3-Sat problem, the objective function to maximize is the number of satisfied clauses. After this step, we apply the interference operation on some particles. Thirdly, we modify the particle’s velocity according to its best fitness and the global best fitness of the swarm. This step is followed by the application of the quantum flight operation in order of improve the quality of quantum particles. Finally, the global best solution is then updated if a better one is found and the whole process is repeated until reaching a stopping criterion.

7. Implementation and Evaluation

QPSOSAT is implemented in C++ and is tested on a microcomputer with a processor of dual-core 1.7 GHZ and 2 GB of memory. In order to assess the efficiency and accuracy of our approach, we perform experiments on a number of benchmark instances called AIM. The AIM instances are all generated with a particular Random-3-SAT instance generator [18]. The tests are divided into two sets: satisfiable tests and unsatisfiable tests, each test contains four benchmarks. Furthermore, we have compared our result with those of several programs based on different paradigms: two popular local search programs (GSAT and WALKSAT) [6], QGASAT: a MAX 3-SAT solver based on Quantum Genetic Algorithm [19], and an exact algorithm based on DPLL procedure [3]. Furthermore, we have analyzed the performance of QPSOSAT without the local search procedure. Finally, Freidman tests were carried out to test the significance of the difference in the accuracy of each method in this experiment. In all experiments, we have run the QPSOSAT program using the parameters’ settings shown in Table 2. The tables 3 and 4 summarize the obtained results for unsatisfiable and satisfiable tests. The results of the experiments are encouraging and prove the feasibility and the efficiency of our approach. In most cases, QPSOSAT performs as good or better than WALKSAT and QGASAT and even better than GSAT. Concerning the performances of the programs in the unsatisfiable tests, all the programs succeeded in this test (table 3). Indeed, the test of Friedman shows clearly that the results found by the programs QGASAT, QPSOSAT, GSAT, and WalkSat equalize the results found by an exact method. On the other hand, in the satisfiable tests both our method and WALKSAT are reasonably successful as shown by Friedman’s test (figure 6). As we can see also in the figure 6, QPSOSAT provides a performance that is very close to that of the exact method. Finally, the local search plays a very important role in the algorithm developed. Indeed, the performance of QPSOSAT without the local search procedure (QPSOSAT-LS) is rather poor.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Parameter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>20</td>
</tr>
<tr>
<td>Topology</td>
<td>linear</td>
</tr>
<tr>
<td>Total number of generations</td>
<td>100</td>
</tr>
<tr>
<td>Neighborhood Size</td>
<td>5</td>
</tr>
<tr>
<td>Velocity_Init_Min</td>
<td>-1</td>
</tr>
<tr>
<td>Velocity_Init_Max</td>
<td>1</td>
</tr>
<tr>
<td>Velocity_Min</td>
<td>-1</td>
</tr>
<tr>
<td>Velocity_Max</td>
<td>1</td>
</tr>
<tr>
<td>Learning_Factor1 (velocity)</td>
<td>1.7</td>
</tr>
<tr>
<td>Learning_Factor2 (velocity)</td>
<td>2.1</td>
</tr>
<tr>
<td>Inertia_Factor (velocity)</td>
<td>1</td>
</tr>
<tr>
<td>Interference angle</td>
<td>$\pi/20$</td>
</tr>
</tbody>
</table>
Table 3: Results for unsatisfiable tests. (QSOSAT-LS: QSOSAT without local search procedure)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#var</th>
<th>#clause</th>
<th>QSOSAT-LS</th>
<th>QPSOSAT</th>
<th>GSAT</th>
<th>WalkSAT</th>
<th>QGASAT</th>
<th>EXACT</th>
</tr>
</thead>
<tbody>
<tr>
<td>aim-50-1_6-no</td>
<td>50</td>
<td>80</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>aim-50-2_0-no</td>
<td>50</td>
<td>100</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>aim-100-1_6-no</td>
<td>100</td>
<td>160</td>
<td>156.75</td>
<td>159</td>
<td>159</td>
<td>159</td>
<td>159</td>
<td>159</td>
</tr>
<tr>
<td>aim-100-2_0-no</td>
<td>100</td>
<td>200</td>
<td>192.75</td>
<td>199</td>
<td>199</td>
<td>199</td>
<td>199</td>
<td>199</td>
</tr>
<tr>
<td>aim-200-2_0-no</td>
<td>200</td>
<td>400</td>
<td>380.5</td>
<td>399</td>
<td>399</td>
<td>399</td>
<td>399</td>
<td>399</td>
</tr>
</tbody>
</table>

Table 4: Results for satisfiable tests. (QSOSAT-LS: QSOSAT without local search procedure)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#var</th>
<th>#clause</th>
<th>QSOSAT-LS</th>
<th>QPSOSAT</th>
<th>GSAT</th>
<th>WalkSAT</th>
<th>QGASAT</th>
<th>EXACT</th>
</tr>
</thead>
<tbody>
<tr>
<td>aim-50-1_6-yes</td>
<td>50</td>
<td>80</td>
<td>78.75</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>80</td>
</tr>
<tr>
<td>aim-50-2_0-yes</td>
<td>50</td>
<td>100</td>
<td>98</td>
<td>100</td>
<td>99.5</td>
<td>99.5</td>
<td>99.5</td>
<td>100</td>
</tr>
<tr>
<td>aim-50-3_4-yes</td>
<td>50</td>
<td>170</td>
<td>165</td>
<td>170</td>
<td>167.75</td>
<td>170</td>
<td>170</td>
<td>170</td>
</tr>
<tr>
<td>aim-50-6_0-yes</td>
<td>50</td>
<td>300</td>
<td>283.25</td>
<td>300</td>
<td>287.5</td>
<td>300</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>aim-100-1_6-yes</td>
<td>100</td>
<td>160</td>
<td>155</td>
<td>159</td>
<td>159</td>
<td>159</td>
<td>159</td>
<td>160</td>
</tr>
<tr>
<td>aim-100-2_0-yes</td>
<td>100</td>
<td>200</td>
<td>192.75</td>
<td>199</td>
<td>199</td>
<td>199</td>
<td>199</td>
<td>200</td>
</tr>
<tr>
<td>aim-100-3_4-yes</td>
<td>100</td>
<td>340</td>
<td>323.25</td>
<td>340</td>
<td>335.75</td>
<td>340</td>
<td>339.75</td>
<td>340</td>
</tr>
<tr>
<td>aim-100-6_0-yes</td>
<td>100</td>
<td>600</td>
<td>559.25</td>
<td>600</td>
<td>600</td>
<td>600</td>
<td>600</td>
<td>600</td>
</tr>
<tr>
<td>aim-200-2_0-yes</td>
<td>200</td>
<td>400</td>
<td>375.25</td>
<td>399</td>
<td>399</td>
<td>399</td>
<td>399</td>
<td>400</td>
</tr>
<tr>
<td>aim-200-6_0-yes</td>
<td>200</td>
<td>1200</td>
<td>1104.75</td>
<td>1200</td>
<td>1177.75</td>
<td>1200</td>
<td>1200</td>
<td>1200</td>
</tr>
</tbody>
</table>

Fig. 6. Friedman test for satisfiable tests.
8. Conclusion

In this paper, we have presented a new approach called QPSOSAT to deal with the problem of MAX 3-SAT. QPSOSAT is based on a hybridizing of quantum computing principles and a PSO algorithm. Our algorithm, compared to PSO algorithms, provides the advantage of giving a greater diversity by using quantum coding of solutions. The quantum representation of the solutions allows the coding of all the potential variable assignments with a certain probability. The optimization process consists of the application of PSO dynamics hybridized by quantum operations such as the interference and measurement operations. In order to enhance the performance of our algorithm, a local search procedure is used. This integration is crucial for the effectiveness of the resulting algorithm. Although more experiments and comparisons are required, the results so far are promising and demonstrate the feasibility of QPSOSAT algorithm to deal with the Max SAT problem; in most cases, our program gives comparable or better solutions than popular MAX 3-SAT solver programs.

References


