Quantum-Inspired Tabu Search Algorithm for Solving 0/1 Knapsack Problems

Chia-Hui Chiu
National Chi-Nan University
No. 1, University Rd.
Puli 54561, Taiwan
s98321525@ncnu.edu.tw

Yi-Jyuan Yang
National Chi-Nan University
No. 1, University Rd.
Puli 54561, Taiwan
s96321044@ncnu.edu.tw

Yao-Hsin Chou *
National Chi-Nan University
No. 1, University Rd.
Puli 54561, Taiwan
yhchou@ncnu.edu.tw

ABSTRACT

In this paper, we propose a novel quantum-inspired evolutionary algorithm, called quantum-inspired Tabu search (QTS). QTS is based on the classical Tabu search and the characteristic of quantum computation such as superposition. We will present how we implement QTS to solve 0/1 knapsack problem. Furthermore, the results of experiments are also compared with other quantum-inspired evolutionary algorithm and other heuristic algorithms' experimental results. The final outcomes show that QTS performs much better than the other heuristic algorithms on 0/1 knapsack problem, without premature convergence and more efficiency.

Track Name: Combinatorial Optimization and Metaheuristics

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Experimentation, Theory

Keywords

combinatorial optimization, knapsack problem, quantum computing, quantum-inspired evolutionary algorithm, tabu search

1. INTRODUCTION

Quantum-inspired evolutionary algorithm (QEA) is a special issue of estimation of distribution algorithm (EDA), where quantum probability amplitudes are used with probabilistic models to describe the promising areas of decision space and to guide the exploration of the global optimal. The most successful case of using Quantum-inspired evolutionary computing is by Han and Kim in 2002 [3]. We had an inspiration that combining the concept of QEA [1]-[3] with the traditional methods Tabu Search called Quantum-inspired Tabu Search (QTS) to improve the performance of evolutionary algorithms.

*This is the corresponding author.

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

2. QTS

The QTS evolutionary algorithm is based on TS. Unlike traditional simple TS, QTS increases two important components which are the diversification and the intensification strategies spring from QEA. And the operation of the new move mechanism called $move - gate$ which uses two-dimensional quantum rotation gate to intensifed search toward attractive regions more thoroughly.

Algorithm 1 : QTS

1. $i \leftarrow 0$
2. initialize quantum population $Q(0)$
3. initialize solution s^I by measuring $Q(0)$
4. initialize best fitness b by repair s^I then evaluating $f(s^I)$
5. **while** (**not** termination-condition) **do**
6. $\quad i \leftarrow i + 1$
7. $\quad$ produce neighborhood Set N by multiple measurements of $Q(i-1)$
8. $\quad$ repair $s \in N$ then evaluate $f(s)$
9. $\quad$ update b
10. $\quad$ find the best solution s^b and the worst solution s^w
11. $\quad$ renew tabu list T
12. $\quad$ update quantum individual $Q(i)$ by applying $U(\triangle\ \theta_1)$ based on s^I and s^b in part 1
13. $\quad$ update individual s^I by measuring $Q(i)$
14. $\quad$ update quantum individual $Q(i)$ by applying $U(\triangle\ \theta_2)$ based on s^I and s^w in part 2
15. $\quad$ update individual s^I by measuring $Q(i)$
16. **end while**

1) The quantum population $Q(0) = \{q_1, q_2, \ldots, q_n\}$, where n is number of *qubit*, α and β of all $q_k, k = 1, 2, \ldots, n$, are initialized with $1/\sqrt{2}$ as in [3].

2) Initialize solution by measuring corresponding *qubit* individual state of $Q(0)$ forms a binary string $s^I = \{x_1, x_2, \ldots, x_n\}$, each bit $x_k, k = 1, 2, \ldots, n$, using the probability of $|\alpha_k|^2$ or $|\beta_k|^2$ to compare with a random number $R \in [0, 1]$, when $R > |\beta_k|^2$, set $x_k = 1$, otherwise $x_k = 0$.

3) The best fitness b is used to record best solution achieved through the evolutionary algorithm. After repairing s^I, it's initialized by evaluating objective function $f(s^I)$.

4) Neighborhood solutions N (line 7 of Algorithm 1) is generated by multiple measuring $Q(i-1)$ m times. We may not get the same results from the same measurement which performs repeatedly on the same *qubit*, but if the gap between α and β was smaller, the more divergent neighborhood solutions will we get.

Table 1: The move-gate lookup table

part 1				part 2			
s_k^I	s_k^b	$q_k \in T$	$\triangle\ \theta_1$	s_k^I	s_k^w	$q_k \in T$	$\triangle\ \theta_2$
0	0	true	0	0	0	true	0
0	1	false	$+\theta$	0	1	false	$-\theta$
1	0	false	$-\theta$	1	0	false	$+\theta$
1	1	true	0	1	1	true	0

*if q_k locate in first or third quadrant, $\triangle\ \theta$ is same as the table
*if q_k locate in second or fourth quadrant, $\triangle\ \theta$ multiplies negative

5) Repair $s \in N$ so that can conform to feasible domain, then evaluate every profit of N with the objective function $f(s)$ (line 8 of Algorithm 1).

6) Update b by line 9 of Algorithm 1. If current iteration has profit better than b, b will be replaced.

7) Renew tabu list T by line 11 of Algorithm 1. Unlike general common tabu list records recent moves, it is used to records *qubits* which are prohibited to change state in QTS, therefore, tabu size is dynamic. It help us avoiding trapping in local optimality. The tabu conditions for the *kth qubit* in $Q(i)$ is shown in table 1.

8) Update Q process is by applying the $move-gate\ U(\triangle\ \theta) = \begin{bmatrix} cos(\triangle\ \theta) & -sin(\triangle\ \theta) \\ sin(\triangle\ \theta) & cos(\triangle\ \theta) \end{bmatrix}$, the $Q(i)$ must be updated twice that reference the tabel 1. In part 1, when s_k^I=0 , s_k^b=1 and q_k, k=1,2,...n, are not in T, we implied corresponding theta to turn the solution closer to the best solution. In part 2, we implied corresponding theta in current solution to keep it away from the worst solution. The $\triangle\ \theta$ should be designed in compliance with the application problem, but it usually set to a small value to prevent prematurely convergence (line 12 to 15 of Algorithm 1).

3. PROBLEM AND RESULTS

The knapsack problem can be described as : Given a set of n items, each kind of item k has a weight w_k and a profit p_k, select a subset of items so that the total weight is less than or equal to the capacity C of the knapsack $\sum_{k=1}^{n} w_k x_k \leq C$, and obtain the maximize total profit by objective function $f(x) = \sum_{k=1}^{n} p_k x_k$. If the kth item is selected for the knapsack, $x_k = 1$, otherwise $x_k = 0$. In our experiments, the average knapsack capacity is used $C = 1/2 \sum_{k=1}^{n} w_k$. In the first case of 0/1 knapsack problem, we set $w_k \in [1, 10]$, and let $p_k = w_k + 5$, as in [3], then we can get Table 2. One can see from this table, the heavier item always gets the less profit per unit weight, but in the real world, things are not always going this way. Hence, we also perform the experiment to test the case II: $p_k = w_k + l_k$, where $l_k \in [0, 5]$, as in [1].

Table 3 and 4 show the experimental results, the average best, of knapsack problems with 250 and 500 items. The maximum number of the iterations is 1,000, and the number of runs is 100. The population size is 10 set in all algorithm. The tabu size of TABU is the same as dynamic as our proposed algorithm, and it is prohibited to change the status of items. The θ both in QTS and QEA are set to 0.01π. For the GA, we set the probabilities of crossover and mutation are respectively fixed to 0.65 and 0.05 as in [2].

Table 2: The unreasonable setting in 0-1 knapsack problem

w_k	1	2	3	4	5	6	7	8	9	10
p_k	6	7	8	9	10	11	12	13	14	15
p_k/w_k	6	...	...	...	2	...	...	...	...	1.5

Table 3: Experimental results of the 0-1 knapsack problem (case I). The total profit and running time.

Items	GA	TABU	QEA	QTS
250	1400.97	1494.03	1532.33	1546.55
t(sec/run)	0.7449	0.5353	0.9266	0.6739
500	2766.55	2981.94	3030.37	3098.40
t(sec/run)	0.9712	0.5654	1.5083	1.0064

Table 4: Experimental results of the 0-1 knapsack problem (case II). The total profit and running time.

Items	GA	TABU	QEA	QTS
250	1068.48	1152.68	1192.02	1207.64
t(sec/run)	0.7376	0.514	0.8835	0.6187
500	2115.68	2309.42	2355.30	2425.06
t(sec/run)	0.9503	0.6754	1.5708	0.9265

4. CONCLUSIONS

In order to improve QTS, we focus more attention on the depth and breadth search of the possible solutions. Generally, the above-mentioned steps can help us get the approximately optimal solution of combinatorial optimization problems. As we see, the results of experiments of 0/1 knapsack problem show that QTS does perform better than other heuristic algorithms, both in time and the optimal solution.

5. REFERENCES

[1] Y. Wanga, X. Y. Fenga, Y. X. Huanga, D. B. Pub, W. G. Zhoua, Y. C. Lianga and C. G. Zhou,"A novel quantum swarm evolutionary algorithm and its applications" in *Neurocomputing,* January, 2007, pp. 633-640.

[2] K. H. Han and J. H. Kim,"Genetic quantum algorithm and its application to combinatorial optimization problem," in *Proc. 2000 Congress on Evolutionary Computation.* Piscataway, NJ, IEEE Press, vol. 2, pp. 1354-1360, July 2000.

[3] K. H. Han and J. H. Kim,"Quantum-inspire evolutionary algorithm for a class of combinatorial," in *IEEE Transactions on Evolutionary Computation,* vol. 6, pp. 580-593, Dec. 2002.

[4] F. Glover,"Tabu Search: Part I," in *ORSA Journal on Computing* , vol. 1, no. 3, pp. 190-206, 1989.

[5] F. Glover,"Tabu Search: Part II," in *ORSA Journal on Computing* , vol. 2, no. 1, pp. 4-32, 1990.