Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

A Novel Quantum Evolutionary Algorithm for Quadratic Knapsack Problem

Apurva Narayan
Schlumberger Oilfield Services
Pune, India
apurvanarayan@ieee.org

C. Patvardhan
Professor
Deptt. of Electrical Engineering
Dayalbagh Educational Institute
Agra, India
cpatvardhan@hotmail.com

***Abstract*—The Quadratic Knapsack Problem (QKP) deals with maximizing a quadratic objective function subject to given constraints on the capacity of the Knapsack. We assume all coefficients to be non-negative and all variables to be binary. Solution to QKP generalizes the problem of finding whether a graph contains a clique of given size. We propose in this paper a Novel Quantum Evolutionary Algorithm (NQEA) for QKPs. These algorithms are general enough and can be used for similar subsection of problems. We report in this paper solutions which lie in less than 1% of the optimal solutions. We also show that our algorithm is scalable to much larger problem sizes and is capable of exploiting the search space to its maximum.**

***Index Terms*—Quantum, Evolutinary Algorithms, Quadratic Knapsack, Clique.**

I. Introduction

The classical knapsack problem (KP) is defined as follows: Given a set of n items, each item j having an integer profit p_j, and an integer weight w_j, the problem is to choose a subset of items such that their overall profit is maximized, while the overall weight does not exceed a given capacity c. Quadratic Knapsack Problem (QKP) is an extension of the classical KP. In the QKPs we are given n items, the j-th having a positive integer weight w_j, a positive integer knapsack capacity c and $n\ X\ n$ nonnegative integer matrix P = (P_{ij}), where p_{jj} is the profit achieved if item j is selected, and, for $j > i$, P_{ij}+P_{ji} is the profit achieved if both items i and j is selected. The QKP calls for selecting an item subset whose overall weight does not exceed the knapsack capacity, so as to maximize the overall profit. The problem may be formulated mathematically as follows

$$maximize\ z(QKP)\ \sum_{i\in N}\sum_{j\in N} p_{ij}x_ix_j \tag{1}$$

$$subject\ to\ \sum_{j\in N} w_jx_j \le c,\ x_j \in 0,1, j \in N \tag{2}$$

QKP is a generalization of the $Knapsack\ Problem$ (KP), which arises when p_{ij} = 0 for all $i \neq j$. Moreover, QKP has the following immediate graph-theoretic interpretation. Given a complete undirected graph of node set N, where each node j has a profit q_j and weight w_j and each edge(i, j) has profit p_{ij}+p_{ji}, select a node subset $S \subseteq N$ whose overall weight does not exceed c so as to maximize the overall profit, given by the sum of the profits of the nodes in S and of the edges with both endpoints in S. It is then easy to see that QKP is also a generaization of the $Clique$ problem.

This $Clique$ problem, in its recognition version, calls for checking whether, for a given positive integer k, a given undirected graph G=(V, E) contains a complete subgraph on k nodes. A possible optimization version of the Clique is given by the so called $Dense\ Subgraph\ Problem$, in which one wants to select a node subset $K \subseteq V$ of cardinality $|K|$=k such that the subgraph of G induced by K contains as many edges as possible. Clearly, the solution to Clique is positive iff the QKP solution has value $k(k-1)$. The most famous version of Clique, called Max Clique, calls for an induced complete subgraph with a maximum number of nodes. This can be solved through a QKP algorithm with some modifications. Max Clique, besides being (strongly) NP-hard, is one of the hardest combinatorial optimization problems studied in the literature, both from a theoretical approximability and from a practical solvability point of view. The same properties apply there to QKP as well, which is consequently much more difficult than the classical KP.

Various algorithms have been proposed in the past for QKP Gallo, Hammer and Simeone [1], who proposed the exact algorithm where upper bounds are computed by using $upper\ planes$, which are linear functions of binary variables. Billionnet and Calmels [2] follow a branch-and-cut approach to the problem, using a classical ILP formulation with $O(n^2)$ variables and constraints. It has been observed in the recent past that QKPs were not studied much but apparently they have attracted great interest. Various Lagrangian relaxation approaches have been described by Billionnet, Faye and Soutif [3], Michelon and Veilleux [4], Hammer and Radar [5]. A more general study of such problems where P is considered negative has been studied by Helmberg, Rendl and Weismantel [6] and propose a combined approach which uses cutting planes and semidefinite programming, and allows for computing in very tight upper bounds. Another version known as $Integer$ QKP, where variables may take any integer value between a lower and an upper bound, is considered by Bretthauer, Shetty and Syam [7], however restricted to $diagonal$ profit matrices P, such that p_{ij}=0 for $i \neq j$.

As it is widely known and shown that QKPs have a wide range of applications. Witzgall [8] has discussed a problem which arises in telecommunications when a number of sites for satellite stations have to be selected, such that the global traffic between these stations is maximized and a budget constraint is respected. Similar models can be formulated when we consider things as location of airports, railway stations or freight handling stations. Johnson, Mehrotra and Nemhauser [9] mention a compiler design problem which may be formulated as a QKP. Dijkhuizen and Faigle [10] consider the weighted maximum b-$clique$ problem. Finally, QKP also appears as a column generation subproblem when solving the graph partitioning problem described in Johnson, Mehrotra and Nemhauser [9].

Heuristics which employ history of better solutions obtained in the search process, viz.GA, EA, have been proven to have better convergence and quality of solution for some difficult optimization problems. But, still, problems of slow/premature convergence remain and have to be tackled with suitable implementation for the particular problem at hand. Quantum Evolutionary Algorithms (QEA) is a recent branch of EAs. In [11], QEAs have proven to be effective for optimization of functions with binary parameters [12].

Although the QEAs have shown to be effective on Difficult Knapsack Problems (DKP) [13] but their performance on more generalized problems as the QKPs have not been investigated so far. Dynamic Programming has been used to find exact solution to such problems where the solution for problems as large as 400 items the solution is found to lie in 1% of the optimum solution. This provides the motivation for an attempt to design better Novel Quantum Evolutionary Algorithm (NQEA) with better capability to solve the QKPs. The computational performance of NQEAs is tested on large instances of QKPs with number of items ranging from 50 to 400. The results obtained are compared with those obtained in [14] and with the greedy heuristic.

The rest of the paper is organized as follows. A brief introduction to the NQEAs and some preliminaries to QEAs are provided in Section II. Section III gives the pseudo code explanation of the NQEA. Experimental results of NQEA are given in Section IV.

II. QEA Preliminaries and NQEA

QEA is a population based probabilistic Evolutionary algorithm that integrates concepts from quantum computing for higher representation and robust search. It is based on the concepts of quantum computing such as a quantum bit and superposition of states. Like the EAs, QEA is are also characterized by the representation of the individual, the evaluation function, and the population dynamics. However, instead of binary, numeric, or symbolic representation, QEA uses a Q-bit as a probabilistic representation, defined as the smallest unit of information. A Q-bit individual is defined by a string of Q-bits. The Q-bit individual has the advantage that it can represent a linear superposition of states (binary solutions) in the search space probabilistically. Thus, the Q-bit representation has a better characteristic of population diversity than other representations. A Q-gate is also defined as a variation operator of QEA to drive the individuals towards better solutions and eventually toward a single state.

Initially, QEA can represent diverse individuals probabilistically because a Q-bit individual represents the linear superposition of all possible states with the same probability. As the probability of each Q-bit approaches either 1 or 0 by the Q-gate, the Q-bit individual converges to a single state and the diversity property disappears gradually. By this inherent mechanism, QEA can treat the balance between exploration and exploitation. It may be noted that although QEA is based on the concepts of Quantum Computing, QEA is not a quantum algorithm, but a novel evolutionary algorithm for classical computer.

As we describe the QEA formulation, initially we give the basics of Quantum Computing. The smallest unit of information stored in a two-state quantum computer is called a quantum bit or a qubit [15]. A qubit may be in the state 1 or 0, or in any superposition of the two. The state of a qubit can be represented as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|0\rangle \qquad (3)$$

where α and β are complex numbers that specify the probability amplitudes of the corresponding states. $|\alpha|^2$ gives the probability that the qubit will be found in the state 0 and $|\beta|^2$ gives the probability that the qubit will be found in the the state 1. Normalization of the state to unity guarantees

$$\alpha^2 + \beta^2 = 1 \qquad (4)$$

The state of a qubit can be changed by the operation with a quantum gate. A quantum gate is a reversible gate and can be represented as a unitary operator U acting on the qubit basis states satisfying $U'U = UU'$, where U' is the hermition adjoint of U. Ther are several quantum gates, such as the NOT gate, controlled NOT gate, rotation gate, Hadamard gate, etc [15]. If there is a system of m qubits, the system can represent 2^m states at the same time. However in the act of observing a quantum state, it collapses to a single state.

Inspired by the concept of quantum computing, QEA is designed with a novel Q-bit representation, a Q-gate as a variation operator, and an observation process.

A. Representation of Q-bits

A number of different representations can be used to encode the solutions onto individuals in evolutionary computation. The representations can have various classifications. QEA used a new representation, called a Q-bit, for the probabilistic representation that is based on the concept of qubit, and a Q-bit individual as a string of Q-bits. In this case Q-bit is defined as the smallest unit of information in NQEA, which is defined with a pair of numbers (α, β) as

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \qquad (5)$$

where $|\alpha|^2$+$|\beta|^2$=1. $|\alpha|^2$ gives the probability that the Q-bit will be in state 0 and $|\beta|^2$ gives the probability that the Q-bit will be in state 1. A Q-bit may be in the state 0-1 or in the linear superposition of the two states.

B. *Observation of Q-bits*

The process of generating binary strings from the qubit string, Q, is called Observation. Let us have a binary solution P now by observing the states of Q, where $P = x_1, x_2, ..., x_n$ at any generation in the binary string of length n, which is formed by selecting either 0 or 1 for each bit using the probability, either $|\alpha|^2$ or $|\beta|^2$ of Q_i. In a quantum computer, in the act of observing a quantum state, it collapses to a single state. However, collapsing into a single state does not occur in NQEA, since NQEA is working on a classical computer, not a quantum computer.

To observe a Q-bit string Q, a string consisting of same number of random numbers between 0 and 1 (R) is generated. the element P_i is set to 0 if R_i less than square of Q_i and 1 otherwise. Table 1 represent the observation process.

TABLE I
OBSERVATION OF QUBIT STRING

i	1	2	3	4	5	.....	N$_g$
Q	0.17	0.78	0.72	0.41	0.89	.....	0.36
R	0.24	0.07	0.68	0.92	0.15	.....	0.79
P	1	0	0	1	0	.....	1

C. *Updating the Q-bit String*

In each of the iterations, several solution strings are generated from Q by observation as given above and their fitness values are computed. The solution with the best fitness is identified. The updating process moves the elements of Q towards the best solution slightly such that there is a higher probability of generation of solution strings, which are similar to the best solution, in subsequent iterations. A Quantum gate is utilized for this purpose so that qubits retain their properties. One such gate is rotation gate, which updates the qubits as

$$\begin{bmatrix} \alpha_i^{t+1} \\ \beta_i^{t+1} \end{bmatrix} = \begin{bmatrix} cos(\Delta\Theta_i) & -sin(\Delta\Theta_i) \\ -sin(\Delta\Theta_i) & cos(\Delta\Theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i^{t} \\ \beta_i^{t} \end{bmatrix} \tag{6}$$

where, α_i^{t+1} and β_i^{t+1} denote probabilities for i^{th} qubit in $(t+1)^{th}$ iteration and $\Delta\Theta_i$ is equivalent to the step size in typical iterative algorithms in the sense that it defines the rate of movement towards the currently perceived optimum.

The above description outlines the basic elements of QEA. The qubit string, Q, represents probabilistically the search space. Observing a qubit string n times yields n different solutions because of probabilities involved. Fitness of these is computed and the qubit string, Q, is updated towards higher probability of producing strings similar to the one with highest fitness. The sequence of steps continues. The above ideas can be easily generalized to work with multiple qubit strings. Genetic operators like crossover and mutation can then be invoked to enhance the search power further.

III. NOVEL QUANTUM EVOLUTIONARY ALGORITHM (NQEA)

The algorithm is explained succinctly in the form of a pseudo-code below.

Notations

- Max = number of items in QKP
- Cap = Capacity of Knapsack
- Profit = 2D Array of profit
- Weight = 1D Array of weight
- Gsol = QKP solution by Greedy Heuristic taking into consideration the diagonal profits
- GProfit = Profit of items selected in GSol
- P, ep, rp : Quantum bit strings used in the search
- NO1 = Operator used to evolve ep towards best solution found so far
- NO2 = Operator used to evove rp randomly
- NO3 = Rotation operator used to evolve p
- bestcep = String produced by operator NO1 giving best result on observation
- bestcrp = Quantum string produced by NO2 giving best result on observation
- Best = Best profit found by NQEA of solution maxsol

A. *Pseudo Code of the Algorithm*

TABLE II
PSEUDO CODE OF THE ALGORITHM

1	Initialize iteration, t, cap, profit, weight, max
2	Sort items in descending order of (diagonal profit)/weight
3	Find greedy solution G with profit GCost
4	Best = GCost, MaxSol = G
5	Initialize for every k If $G[k]$==1, $p[k]$=$ep[k]$=$rp[k]$=0.8 else $p[k]$=$ep[k]$=$rp[k]$=0.2
6	BestCap = ep, BestCap=rp /*Initialization*/
7	Observe $p[k]$ to get solution cost as tcost; if(tcost>Best)Best=tcost; Maxsol=sol;
8	while (termination_criterion != TRUE) Do Steps 9-15

9	Apply NO1 to p to generate the current rp i.e crp;
	Iterate : Observe crp to obain solution rsol with cost = rcost;
	if(rcost>Best)
	Crossover rsol with Maxsol to obtain tsol with cost = tcost;
	if(rcost>Best)rsol = tsol; rcost = tcost;
	bestcrp = crp;
	If(rcost>Best)Best = rcost; maxsol=rsol

10	Repeat Step 9 with NO2 on p to obtain bestcep, ecost and esol;
11	Apply NO3 on p

12	Iterate
	qmax[i]=findmax(bestcep[i],bestcrp[i], p[i]);
	qmin[i] = findmix(bestcep[i], bestcrp[i],p[i]);
	if(p[i] > 0.5)
	if((maxsol[i]==1)and (bestcep[i]==1)and (bestcrp[i]==1))
	p[i]=qmax[i];
	else if((maxsol[i]==0)and (bestcep[i]==0)and (bestcrp[i]==0))
	p[i]=qmin[i];

13	Settled = number of p[i]s with α_i>0.98 or α_i<0.02
14	t = t+1

15	If((t > iter_count) or (settled > 0.98 * max))
	termination_criterion=TRUE

Algorithm NQEA starts with the greedy solution and initializes the qubit strings in accordance with the greedy solution as in step 4. Observe operation in step 5 is a modified form of observation described in section 3. In this, the solution string resulting from the observation is checked for violation of capacity constraint and repaired, if necessary, using a greedy approach i.e. selected items are deselected in increasing order of profit/weight till capacity constraint is satisfied. The crossover operator is a simple two-point crossover with greedy repair of constraint violations.

B. Evolving Qubits

In NQEA, the qubit is the evolved state of the qubit, which is a superposition of state 0 and 1, is shifted to a new superposition state. This change in probability mangnitudes $|\alpha|^2$ and $|\beta|^2$ with change in state is transformed into real valued parameters in the problem space by two neighborhood operators.

C. Neighborhood Operator 1(NO1)

It generates a new qubit array crp from the rp. An array R is created with max elements generated at random such that every element in R is either +1 or -1. Let ρ_k be the k^{th} element in R. Then Θ_k^t is given by

$$\Theta_k^t = \Theta_k^{t-1} + \rho_k * \delta \tag{7}$$

where, δ is alteration in angle and Θ_k^t is the rotated angle given by arctan(β_k^t/α_k^t). δ is randomly chosen in the range [0, Θ_k^{t-1}] if ρ_k=-1 and in the range [Θ_k^{t-1}, Π/2] if ρ_k = +1.

The new probability amplitudes, β_k^t, α_k^t are calculated using the rotation gate as

$$\begin{bmatrix} \alpha_{ijk}^t \\ \beta_{ijk}^t \end{bmatrix} = \begin{bmatrix} cos(\delta) & -sin(\delta) \\ -sin(\delta) & cos(\delta) \end{bmatrix} \begin{bmatrix} \alpha_{ijk}^{t-1} \\ \beta_{ijk}^{t-1} \end{bmatrix} \tag{8}$$

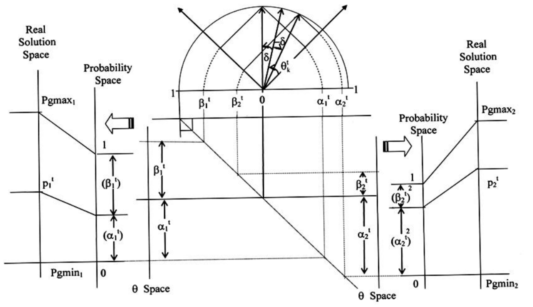

Fig. 1. Neighborhood Operator 1 (NO1)

D. Neighborhood Operator 2 (NO2)

NO2 works just as NO1 except that it generates a point between ep and BEST. It is primarily utilized for exploitation of search space. The rationale for two neighborhood operators is as follows. NO1 has a greater tendency for exploration. NO2 has a greater tendency towards exploitation because, as the algorithm progresses, the values of ep converge towards BEST. Table 3 shows the frequency of use of NO1 and NO2.

The neighborhood operators thus evolve new quantum strings from the existing strings. This approach removes all disadvantages of binary representation of real numbers while, at the same time, balances exploration and exploitation in the sense that it adopts the step-size from large initially to progressively smaller size.

TABLE III
FREQUENCY OF USE OF NO1 AND NO2

Stage of Search	Proportion of NO1 (%)	Proportion of NO1 (%)
First one-fifth iterations	90	10
Second one-fifth iterations	70	30
Third one-fifth iterations	50	50
Fourth one-fifth iterations	30	70
Fifth one-fifth iterations	10	90

Fig. 2. Updating the k^{th} element of qubit string Q

E. Updating Qubit String (NO3)

In the updating process, the individual states of all the qubits in p are modified so that probability of generating a solution string which is similar to the current best solution is increased in each of the subsequent iterations. Amount of change in these probabilities is decided by Learning Rate, $\Delta\theta$, and is taken as 0.001π. Updating process is done as explained in previous section. Table IV presents the choice of $\Delta\theta$ for various conditions of objective function values and i^{th} element of p and BEST in t^{th} iteration. $F(p)$ is the profit of the current solution observed from p. F(BEST) is the profit of maxsol.

TABLE IV
CALCULATION OF $\Delta\Theta$ FOR t^{th} ITERATION

Fitness	Elemental Values	$\Delta\theta$
X	p_i=$BESTi$	0
F(BEST)>F(p)	p_i>$BESTi$	0.001π
F(BEST)>F(p)	p_i<$BESTi$	-0.001π
F(BEST)<F(p)	p_i>$BESTi$	-0.001π
F(BEST)<F(p)	p_i<$BESTi$	0.001π
F(BEST)=F(p)	p_i>$BESTi$	0
F(BEST)=F(p)	p_i<$BESTi$	0

The updating process is illustrated in figure 2 for k^{th} element for t^{th} iteration i.e. changes in state of k^{th} qubit and corresponding change in probability amplitudes. $Findmax$ finds the maximum of the three arguments whereas $findmin$ finds the minimum of the three arguments.

IV. COMPUTATIONAL RESULTS AND CONCLUSIONS

The computational experiments have been performed for problem sized upto 400. The results were compared between greedy heuristic and the algorithm NQEA. It was found that in almost 95% of the cases NQEA gave better results. It was also observed that convergence of the algorithm would vary with varying the ratio of NO1 and NO2.

The problem instances were genrated using the problem generator from Pisinger [16]. It was observed that an optimal ratio of usage of NO1 and NO2 the best soltuion can be obtained. A graph below shows the convergence of the algorithm for one of the problems at various ratios of NO1 and NO2 as mentioned in Table III.

The table below shows a result comparison between the greedy heuristic and NQEA. The alogrithm is powerful enough to search for better and more complex solutions.

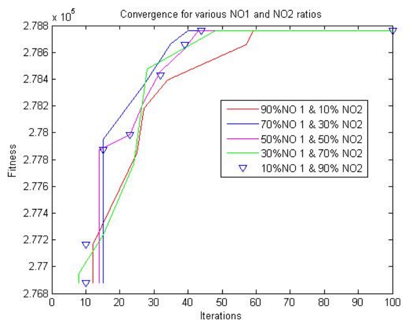

Fig. 3. Convergence of various ratio of NO1 and NO2

TABLE V
COMPUTATION RESULTS COMPARISON OF NQEA, GREEDY AND PISINGER

Number of Items	Greedy Profit	Pisinger Profit	NQEA Profit
20	17002	17072	17069
50	88120	88252	88252
100	220276	220537	220537
150	228701	228731	228731
200	339870	339964	339958
250	560283	560332	560332
350	1394801	1395078	1395078

ACKNOWLEDGMENT

The authors would like to thank Chairman, Advisory Committee on Education, Dayalbagh Educational Institute, Dayalbagh, Agra, UP, India for constant guidance and motivation.

REFERENCES

[1] G. Gallo, P.L.Hammer, and B.Simeone, "Quadratic knapsack problems," *Jour. of Mathematical Programming*, 1980.

[2] A. Billionnet and F. Calmels, "Linear programming for the 0-1 quadratic knapsack problem," *European Journal of Operation Research*, vol. 92, pp. 310–325, 1996.

[3] A. Billionnet, A. Faye, and E. Soutif, "A new upper-bound and an exact algorithm for the 0-1 quadratic knapsack problem," *European Journal of Operation Research*, vol. 92, pp. 310–325, 1996.

[4] P. Michelon and L.Veilleux, "Lagrangean methods for 0-1 quadratic knapsack problem," *European Journal of Operation Research*, vol. 92, pp. 326–341, 1996.

[5] P. L. Hammer and D. J. Rader, "Efficient methods for solving quadratic 0-1 knapsack problems," in *Proc. INFOR*, pp. 170–182, 1997.

[6] C. Helmberg, F. Rendl, and R. Weismantel, "Quadratic knapsack relaxations using cutting planes and semidefinite programming," in *Proc. of the Fifth IPCO Conference, LNCS*, pp. 175–189, Springer-Verlag, 1996.

[7] K. Bretthauer, B. Shetty, and S.Syam, "A branch-and-bound algorithm for integer quadratic knapsack problems," *ORSA Journal on Computing*, vol. 7, pp. 109–116, 1995.

[8] Witzgall, "Mathematical methods of site selection for electronic message systems(ems)," tech. rep., NBS Internal Report, 1975.

[9] E. Johnson, A. Mehrota, and G. Nemhauser, "Min-cut clustering," *Journal of Mathematical Programming*, vol. 62, pp. 133–152, 1993.

[10] G. Dijkhuizen and U. Faigle, "A cutting-plane approach to the edge-weighted maximal cique problem," *European Journal of Operation Research*, vol. 69, pp. 121–130, 1993.

[11] K. Han and J. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," in *IEEE Trans. on Evolutionary Computation*, pp. 580–593, IEEE, 2002.

[12] D. Deutsch, "Quantum theory, the church turing principle and the universal quantum computer," *Proc. Royal Society of London*, vol. A 400, pp. 97–117, 1985.

[13] C. Patvardhan, A. Narayan, and A. Srivastav, "Enhanced quantum evolutionary algorithm for difficult knapsack problem," in *Proc. PReMI 07, LNCS*, pp. 252–260, Springer-Verlag, 2007.

[14] A. Caprara, D. Pisinger, and P. Toth, "Exact solution of the quadratic knapsack problem," tech. rep., 98/21,DIKU,University of Copenhagen, Denmark, 1998.

[15] T. Hey, "Quantum computing: An introduction," *Computing and Control Engineering Journal*, vol. 10, no. 3, pp. 105–112, 1999.

[16] D. Pisinger. DIKU, Univ. Of Copenhagen, Denmark, [Online]. Available: www.diku.dk/ pisinger/codes.html.