Swarm and Evolutionary Computation 26 (2016) 175–190

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Regular Paper

Parallel improved quantum inspired evolutionary algorithm to solve large size Quadratic Knapsack Problems

C. Patvardhan [a], Sulabh Bansal [a,*,1], A. Srivastav [b]

[a] Faculty of Engineering, Dayalbagh Educational Institute, Agra 282005, India
[b] Institut für Informatik, Kiel University, 24098 Kiel, Germany

ARTICLE INFO

Article history:
Received 11 May 2015
Received in revised form
28 August 2015
Accepted 23 September 2015
Available online 3 October 2015

Keywords:
Combinatorial Optimization
Large Size Quadratic Knapsack Problem
Parallel Algorithm
Quantum Inspired Evolutionary Algorithm

ABSTRACT

Quadratic Knapsack Problem (QKP), an extension of the canonical simple Knapsack Problem, is NP Hard in the stronger sense. No pseudo-polynomial time algorithm is known to exist which can solve QKP instances. QKP has been studied intensively due to its simple structure yet challenging difficulty and numerous applications. A few attempts have been made to solve large size instances of QKP due to its complexity. Quantum Inspired Evolutionary Algorithm (QIEA) provides a generic framework that has often been carefully tailored for a given problem to obtain an effective implementation. In this work, an improved and parallelized QIEA, dubbed IQIEA-P is presented. Several additional features make it more balanced in exploration and exploitation and thus have better applicability. Computational experiments are presented on large QKP instances of 1000 and 2000 items. The improvements are inherently parallelizable and, therefore, good speedups are obtained on a multi-core machine. No parallel algorithm is available for QKP. The solutions provided by QIEA-P are competitive with those obtained from the state of the art algorithm.

1. Introduction

The 0/1 Quadratic Knapsack Problem (QKP) is a generalization of the 0/1 Knapsack Problem (KP) introduced by Gallo et al. [1]. Given n items to be filled in a knapsack where w_j is the positive integer weight of jth item, c is a positive integer knapsack capacity and an $n \times n$ nonnegative integer matrix $P=(p_{ij})$ is given, where p_{jj} is a profit achieved if item j is selected, and, for $j>i$, $p_{ij}+p_{ji}$ is the additional profit achieved if both items i and j are selected. Without loss of generality matrix P is considered symmetric such that $p_{ij}=p_{ji}$ for all i and j. Hence, additional profit achieved if both items i and j are selected is considered as p_{ij} rather than $p_{ij}+p_{ji}$, for $j>i$. QKP is to find a subset of items whose total weight is not more than the knapsack capacity c such that the overall profit is maximized. If x_j is binary variable which is equal to 1 if jth item is selected and 0 otherwise, the problem is formulated as follows.

$$\begin{aligned}&\text{Maximize}: \textstyle\sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}x_i x_j\\ &\text{Subject to}: \textstyle\sum_{j=1}^{n} w_j x_j \le c\\ &x_j \in \{0,1\},\quad j\in\{1,\dots,n\}\end{aligned} \tag{1}$$

The KP is a particular case of QKP which arises when $p_{ij}=0$ for all $i \ne j$. The *Clique* problem, is another particular case of QKP, which requires checking whether, for a given integer k, a given undirected graph $G=(V, E)$ contains a complete subgraph on k nodes. The popular optimization version of *Clique*, called *Max Clique*, calls for an induced complete subgraph with a maximum number of nodes.

The *Max Clique*, can be solved using a QKP algorithm by using binary search. *Max Clique* is not only NP-hard in strong sense but is one of the hardest combinatorial optimization problems. Same properties apply to QKP as well. Pseudo polynomial time algorithms exist for KP but no such algorithm exists for QKP. QKP is thus considered much more difficult than the simple KP [2,3].

QKP is thus a challenging problem. Nevertheless, it has been studied widely due to its generality and wide applicability in several areas like facility location problems [4,5], compiler design [6], finance [7], VLSI design [8] and weighted maximum b-clique problem [9,10].

* Corresponding author.
E-mail addresses: cpatvardhan@gmail.com (C. Patvardhan), sulabh.bansal.78@gmail.com (S. Bansal), asr@informatik.uni-kiel.de (A. Srivastav).
[1] Permanent Residential Address: 1/25, Hazuri Bhawan, Pipal Mandi, Agra 282003, India.

http://dx.doi.org/10.1016/j.swevo.2015.09.005
2210-6502/

Gallo et al. [1] introduced QKP and presented a method to derive upper bounds using upper planes. Several attempts have been made to solve QKP [3]. Some recent ones are as follows. Pisinger et al. [11] presented an algorithm based on Lagrangian relaxation\decomposition and aggressive reduction. It has been shown to solve some large-sized instances with 1500 binary variables. Le´tocart et al. [12] presented another algorithm based on a re-optimization technique to accelerate the resolution of each independent sequence of 0–1 linear knapsack problems induced by the Lagrangian relaxation\decomposition. Computational results for randomly generated instances of 600 binary variables were presented. Large-sized benchmark instances of Pisinger et al. [11] (1500 binary variables) and Le´tocart et al. [12] (600 binary variables) were randomly generated and tested. They have not been recorded by the authors [13].

Existing standard deterministic approaches like CPLEX can not solve large instances of QKP. Several studies on heuristic and meta-heuristic methods have also been made in the literature. These provide satisfactory solutions for QKP within reasonable time. Some effective heuristic and meta-heuristic methods applied in last few decades to solve QKP are given in Table 1. From the table, it is clear that state of the art method applied on large QKP instances is GRASP and Tabu Search proposed recently by Yang et al. [13]. A GRASP and tabu search method [13] solves larger instances of size 1000 and 2000 variables. No parallel algorithm exists which solves large size QKP.

Evolutionary Algorithms (EA), inspired by natural selection, mimic iterative evolutionary processes with a set of solutions encoded in a population. The population evolves based on the rule of "survival of the fittest" [14]. The computational challenges are faced due to problem difficulty and size, the complexity of fitness function, and distribution characteristics of solution space, and also on runtime efficiency of stochastic search [15]. EA's are considered inherently parallelisable [16].

QIEA refers to a subclass of EA where representation and evolution is implemented based on concept of Quantum computing. Similar to EA, QIEA exhibit the property of inherent parallelism embedded in the evolutionary process. Some attempts have been made in literature to utilize parallel implementations of QIEA for simple KP [17,18].

In this work, an improved and parallelized QIEA, dubbed IQIEA-P is presented with several additional features to make it more balanced in exploration and exploitation and also have better applicability to different types of combinatorial optimization problems. The improvements are inherently parallelizable and, therefore, good speedups are obtained on a multi-core machine. This is the first attempt to parallelise the QIEA for QKP. This attempt in fact presents the first parallel algorithm to solve large instances of QKP. Computational experiments are presented on large QKP instances used by Yang et al. [13] (1000 and 2000 binary variables) which have been obtained on request. Quality of solutions provided by IQIEA-P is competitive to best known results. Parallelization provides good speedup.

The rest of the paper is organized as follows. The basic concept of QIEA is explained in Section 2. In Section 3 the proposed IQIEA-P is presented. The primary differences of the strategy used to improve QIEA in present work as compared to earlier QIEAs are discussed. A comparison of IQIEA-P with sequential version (IQIEA) is done in Section 4. Conclusions and future work are discussed in Section 5.

2. Quantum inspired evolutionary algorithm (QIEA)

The QIEA introduced in [28] is population-based stochastic evolutionary algorithm. It uses the qubit, a vector, to represent the probabilistic state of individual. Each qubit is represented as $q_i = \begin{bmatrix}\alpha_i \\ \beta_i\end{bmatrix}$, where α_i, β_i are complex numbers so that $|\alpha_i|^2$ is the probability of state being 0 and $|\beta_i|^2$ is the probability of state being 1 such that $|\alpha_i|^2+|\beta_i|^2=1$. For the purpose of QIEAs, α_i and β_i are assumed to be real. Thus, a qubit string Q, represents a superposition of 2^n binary states and provides an extremely compact representation of entire space.

The process of generating binary strings from the qubit string, Q, is known as observation. To observe the qubit string Q, a string P is generated randomly, the ith bit P_i being 1 with probability Q_i^2 independent of other bits. 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 best fitness is identified. The updating process moves the qubits of Q towards the best solution slightly such that there is a higher probability of generation of solution strings, which are similar to best solution, in subsequent iterations. A quantum gate is utilized for this purpose so that qubits retain their properties [28].

One such gate used in this work is the Rotation Gate, which updates the qubits as follows:

$$\begin{bmatrix}\alpha_i^{t+1} \\ \beta_i^{t+1}\end{bmatrix} = \begin{bmatrix}\cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i)\end{bmatrix}\begin{bmatrix}\alpha_i^{t} \\ \beta_i^{t}\end{bmatrix} \quad (2)$$

where, α_i^{t+1} and β_i^{t+1} denote probabilities for ith 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 best solution. The value for $\Delta\theta_i$ is chosen to be 0.01 when observed solution is not better than best solution found till the time of observation.

The above description outlines the basic elements of QIEA. Observing a qubit string n times yields n different solutions because of the probabilities involved. The 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. This sequence of steps continues; these ideas can be easily generalized to work with multiple qubit strings.

QIEA of [25] (dubbed here QIEA-QKP) could give near optimal solutions for QKP benchmark instances of size up to 200 binary variables in reasonable time. Following improvements have been made in the original QIEA to obtain QIEA-QKP.

- To update the qubit individuals the rotation gate used is slightly different, which may assign different rotation angle to different qubits depending on the bits of observed current solution and best solution.
- A rudimentary local search technique is used which generate n solutions in neighborhood of solutions (provided by observation of qubit individuals) and keep the best.
- Migration Operator is removed.

A host of QIEA-based attempts have been reported in the literature that utilizes QIEAs for the solution of a wide variety of problems [31,32]. QIEAs, of course, are not a "one-size fits all" solution. The No Free lunch Theorem prohibits that. However, a particular QIEA has to be designed for the problem at hand to achieve high performance with respect to the state-of-art algorithms for the problem. The primary strengths and weaknesses of the QIEAs are briefly discussed in Table 2. Many of the weaknesses are shared with other EAs as well.