Engineering Applications of Artificial Intelligence 55 (2016) 1–13

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

A hybrid quantum particle swarm optimization for the Multidimensional Knapsack Problem

Boukthir Haddar [a,*], Mahdi Khemakhem [b], Saïd Hanafi [c], Christophe Wilbaut [c]

[a] LOGIQ ISGI, University of Sfax, Sfax, Tunisia
[b] Prince Sattam Bin Abdulaziz University, Al-Kharj, Riyadh, Saudi Arabia
[c] LAMIH UMR CNRS 8201, University of Valenciennes, Valenciennes, France

ARTICLE INFO

Article history:
Received 14 January 2016
Received in revised form
5 April 2016
Accepted 14 May 2016

Keywords:
Combinatorial optimization
Hybrid heuristic
Multidimensional Knapsack Problem
Particle swarm optimization

ABSTRACT

In this paper we propose a new hybrid heuristic approach that combines the Quantum Particle Swarm Optimization technique with a local search method to solve the Multidimensional Knapsack Problem. The approach also incorporates a heuristic repair operator that uses problem-specific knowledge instead of the penalty function technique commonly used for constrained problems. Experimental results obtained on a wide set of benchmark problems clearly demonstrate the competitiveness of the proposed method compared to the state-of-the-art heuristic methods.

1. Introduction

In this paper we are dealing with the NP-hard 0–1 Multidimensional Knapsack Problem (MKP), which seeks to find a subset of items that maximizes a linear objective function while satisfying a set of linear capacity constraints. This problem can be formulated as follows:

$$(\mathbf{MKP}) \begin{cases} \max & \sum_{j=1}^{n} c_j x_j \\ \text{subject to:} & \sum_{j=1}^{n} a_{ij} x_j \le b_i, \forall\, i \in M = \{1,\ldots,m\} \\ & x_j \in \{0,1\}, \forall\, j \in N = \{1,\ldots,n\} \end{cases}$$

where $N = \{1, \ldots, n\}$ is the set of items, and $M = \{1, \ldots, m\}$ is the set of knapsack constraints with capacities b_i ($i \in M$). Each item $j \in N$ yields c_j units of profit and consumes a given amount of resource a_{ij} for each knapsack $i \in M$. The MKP coefficients are all non-negative integer values ($c \in \mathbb{N}^n$, $a \in \mathbb{N}^{m \times n}$, $b \in \mathbb{N}^m$) and there are usually few constraints compared to the number of variables (i.e., $m \ll n$).

Many practical engineering design problems can be formulated as the 0–1 MKP, such as, cutting stock (Gilmore and Gomory, 1966), project selection (Petersen, 1967), cargo loading problems (Shih, 1979), capital budgeting (Weingartner, 1966), databases and processor allocation in distributed systems (Gavish et al., 1982) or the daily management of a satellite (Vasquez and Hao, 2001). Given the practical and the theoretical importance of the 0–1 MKP, this problem has been widely studied and solved by many exact as well as heuristic methods. The reader is referred to Freville (2004), Puchinger et al. (2010) and Varnamkhasti (2012) for a comprehensive and recent annotated bibliography.

Exact methods include dynamic programming (Gilmore and Gomory, 1966; Green, 1967; Weingartner and Ness, 1967), hybrid dynamic programming methods (Bertsimas and Demir, 2002; Balev et al., 2008; Wilbaut et al., 2006), branch and bound algorithms (Fayard and Plateau, 1982; Gavish and Pirkul, 1985; Vimont et al., 2008; Mansini and Speranza, 2012) and hybrid approaches combining constraint programming and integer linear programming (Oliva et al., 2001; Boussier et al., 2010). The major drawback of these methods remains the temporal complexity when dealing with large instances. Therefore, many researchers focus on heuristic and meta-heuristic search methods which can produce solutions of good qualities in a reasonable amount of time. Relevant methods include tabu search (Vasquez and Hao, 2001; Dammeyer and Voss, 1993; Glover and Kochenberger, 1996; Hanafi and Freville, 1998; Vasquez and Vimont, 2005), genetic algorithm (Chu and Beasley, 1998; Berberler et al., 2013; Martins et al., 2014), simulated annealing (Leung et al., 2012; Rezoug et al., 2015), ant colony optimization (Parra-Hernandez and Dimopoulos, 2003; Kong et al., 2008; Ke et al., 2010; Fingler et al., 2014), filter-and-fan algorithm (Khemakhem et al., 2012), particle swarm optimization (Kong et al., 2006; Wan and Nolle, 2009; Chen et al., 2010; Ktari and Chabchoub, 2013; Tisna, 2013; Beheshti et al., 2013; Chih, 2015) and so on.

In this paper, we propose an efficient hybrid heuristic approach to solve the 0–1 MKP that effectively combines a relatively recent

* Corresponding author.
E-mail addresses: boukthir.haddar@gmail.com (B. Haddar), m.khemakhem@psau.edu.sa (M. Khemakhem), said.hanafi@univ-valenciennes.fr (S. Hanafi), christophe.wilbaut@univ-valenciennes.fr (C. Wilbaut).

http://dx.doi.org/10.1016/j.engappai.2016.05.006
0952-1976/

evolutionary computation technique, the Quantum Particle Swarm Optimization (QPSO), with a local search method. We propose to use QPSO in combination with a heuristic repair operator utilizing problem-specific knowledge, instead of the penalty function technique usually used to avoid the violation of problem constraints. We apply this repair operator to amend infeasible solutions or to improve feasible solutions. In this way, it ensures that the search process will be always guided through a feasible solution space.

The aim of this work is twofold: (i) To investigate the effectiveness of an improved QPSO algorithm when dealing with an NP-hard combinatorial optimization problem such as the 0–1 MKP. (ii) To suggest an efficient hybrid approach that combines QPSO with a local search method in the aim to benefit from the good exploitation (intensification) of the search space offered by a local search method algorithm and the good exploration (diversification) and the fast convergence of the modified QPSO method. Note that the proposed hybrid method remains valid for 0–1 integer programming problems. Special attention should be given to the ways the problem-specific information could be applied into some repair operators.

The remainder of this paper is organized as follows. Section 2 describes the basic features of the classical particle swarm optimization (PSO) technique for continuous optimization and then reviews the fundamental principles of the Binary PSO method (BPSO). Section 3 introduces our QPSO algorithm to solve the 0–1 MKP, whereas Section 4 describes the specific MKP repair operator. Section 5 describes the local search to repair infeasible solutions and to improve feasible solutions. Section 6 presents and discusses the experimental results obtained over a wide set of benchmark problems. Section 7 concludes with a summary of major results and suggestions for future researches.

2. Particle swarm optimization

The Particle Swarm Optimization (PSO) algorithm is a global optimization heuristic method originally introduced by Kennedy and Eberhart in 1995 (Kennedy and Eberhart, 1995). It exploits the concept that the knowledge needed for the search of an optimal solution can be modeled on the basis of observed social behavior.

In the original continuous PSO version, we consider a swarm $S = \{1, \ldots, p\}$ of p particles in a n-dimensional continuous solution space. Those particles are initially placed randomly on the search space and are actively searching for an optimal solution to the problem by updating individual generations. Each particle $s \in S$ of the swarm is associated with a vector x^s that represents a potential solution to the problem and with a velocity vector v^s that gives the rate of change for the position of a given particle at the next iteration. During the search procedure, each particle communicates with its neighbors and tends to move toward the best position (solution) found. The velocity and the position of each particle s are updated according to its best previous solution x^{*s} and to the best solution so far found for the swarm $x^{\#}$. The following equations are used to iteratively update particles' velocities and solutions:

$$v_j^s = \gamma_1 \times v_j^s + \gamma_2 \times r_j \times (x_j^{*s} - x_j^s) + \gamma_3 \times r_j \times (x_j^{\#} - x_j^s), \forall j \in N \quad (1)$$

$$x^s = x^s + v^s \quad (2)$$

Coefficient γ_1 in Eq. (1) refers to the inertia factor, γ_2 and γ_3 refer to the learning factors or accelerated variables and $r = (r_j)$ to a random vector obtained from a uniform distribution in $[0, 1]^n$ for each particle dimension. To avoid divergence, the value of v_j^s is generally limited to a maximum value V_{max} and a minimum value $-V_{max}$, i.e., $v_j^s \in [-V_{max}, V_{max}]$, $\forall j = 1, \ldots, n$.

PSO has been originally developed for continuous nonlinear optimization where velocity and position are represented as real values (see Abraham et al., 2006; Kennedy, 2000). It is, therefore, not able to deal with a binary combinatorial optimization problem, such as the MKP. Accordingly, in Kennedy and Eberhart (1997) proposed a binary version of PSO, termed Binary PSO, to tackle problems with binary variables. This version uses the concept of velocity as a probability that a bit (position) takes on a value of "0" or "1". A sigmoid function is then used to transform all real valued velocities to the range [0.0, 1.0]. The velocity updating formula remains unchanged as defined in Eq. (1), with x^{*s} and $x^{\#}$ being integers in $\{0, 1\}^n$ in binary case.

The variable updating rule is, however, re-defined by the following equation:

$$x_j^s = \begin{cases} 1 & \text{if } r_j < \dfrac{1}{1 + exp(-v_j^s)} \\ 0 & \text{otherwise} \end{cases} \forall j = 1, \ldots, n. \quad (3)$$

As the optimization ability of the standard BPSO is not ideal (Nezamabadi-pour et al., 2008; Engelbrecht, 2005; Pampara et al., 2005; Khanesar et al., 2007), several enhanced versions of this approach have been proposed during the last few decades. Some of these proposals have studied other neighborhood topologies (Kennedy, 2000; Clerc, 2006; Mohais et al., 2005) and (Parrott and Li, 2006) while others have tried to introduce different techniques to simulate particle flights by direct sampling using a random number generator with a certain probability distribution (Pampara et al., 2005; Langeveld and Engelbrecht, 2012; Kennedy, 2003; Sun et al., 2004) and (Sun et al., 2004). The Quantum PSO represents one of the most efficient versions due to its effective global search ability and out-performance on several optimization problems as demonstrated in Krohling and dos Santos Coelho (2006).

The following section will describe the basic components of the discrete PSO algorithm which we use to solve the 0–1 MKP. Some of the concepts applied derive from the quantum PSO algorithm proposed in 2004 by Yang et al. (2004), which has been slightly modified and extended to fit the 0–1 MKP. The next section will then discuss the advantages of incorporating a heuristic repair operator that uses problem-specific knowledge into the modified algorithm to amend the potential generation of infeasible solutions.

3. Quantum particle swarm optimization for the 0–1 MKP

In the QPSO algorithm, a particle is probabilistically represented as a quantum vector in which a value of a given single bit (*qubit*) could be in the "1" or "0" state, or in any superposition of both states (see Hey, 1999).

A quantum particle swarm Y at iteration k is defined as:

$$Y(k) = \{y^1(k), y^2(k), \ldots, y^p(k)\} \text{ with } y^s(k) \in [0, 1]^n \ \forall s \in S. \quad (4)$$

The value $y_j^s(k)$ in Eq. (4) denotes the probability of the jth bit of the sth particle to be in the "0" state. Then, a quantum particle vector is transformed into a discrete particle vector $X(k) = \{x^1(k), x^2(k), \ldots, x^p(k)\}$ with $x^s(k) \in \{0, 1\}^n$, $\forall s \in S$, based on the following rule:

$$x_j^s(k) = \begin{cases} 1 & \text{if } r_j > y_j^s(k) \\ 0 & \text{otherwise} \end{cases} \forall j = 1, \ldots, n. \quad (5)$$

where $r_j \in [0, 1]$ is a random number. Once the rule to obtain a discrete particle swarm from a quantum one is known, the evolution of the quantum particle can be defined according to the propositions described in Yang et al. (2004):

$$y^{\#}(k) = \alpha \times x^{\#}(k) + \beta \times (e - x^{\#}(k)) \quad (6)$$