A hybrid quantum particle swarm optimization for the Multidimensional Knapsack Problem

Boukthir Haddar, Mahdi Khemakhem, Said Hanafi, Christophe Wilbaut

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

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

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

A B S T R A C T

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).

DOI : 10.1016/j.engappai.2016.05.006

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):

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

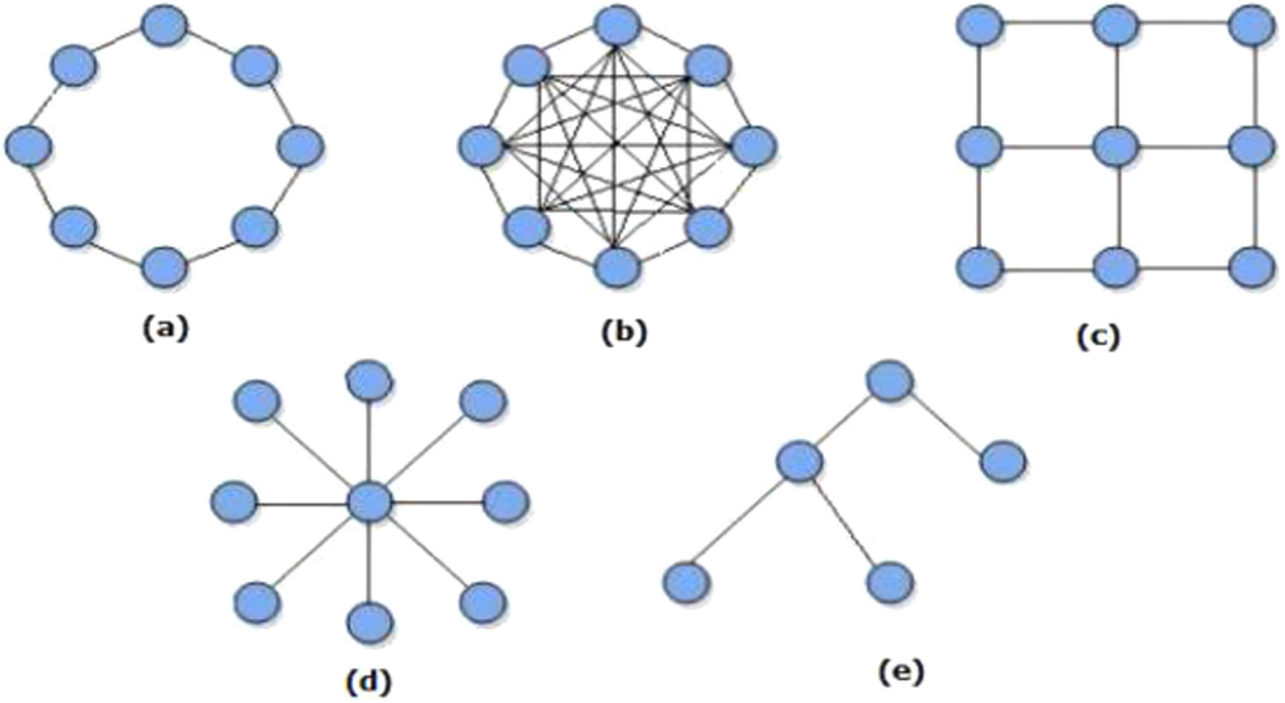

Fig. 1. (a) Ring topology (b) Fully connected topology (c) Mesh topology (d) Star topology (e) Tree topology.

Table 1
Performance comparison on restricted set problems.

Prob.		**QPSO**		**QPSO+RO**		QPSO*	
Inst.	*Opt./best − known*	*Best*	*t(s)*	*Best*	*t(s)*	*Best*	*t(s)*
5-100-00	**24,381**	24,281	2.022	24,314	2.655	**24,381**	3.198
5-100-01	**24,274**	24,190	2.485	24,244	2.903	**24,274**	3.478
5-100-02	**23,551**	23,390	2.897	**23,551**	3.186	**23,551**	3.635
10-100-00	**23,064**	22,975	3.974	23,050	4.255	**23,064**	5.273
10-100-01	**22,801**	22,750	5.177	**22,801**	5.877	**22,801**	6.662
10-100-02	**22,131**	21,972	3.233	22,081	3.946	**22,131**	4.820
30-100-00	**21,946**	21,829	4.861	21,852	5.367	**21,946**	6.333
30-100-01	**21,716**	21,497	5.449	**21,716**	5.893	**21,716**	6.396
30-100-02	**20,754**	20,513	3.972	20,581	4.224	**20,754**	5.944
5-250-00	**59,312**	59,210	12.679	59,243	20.567	**59,312**	22.232
5-250-01	**61,472**	61,344	15.572	61,449	24.945	**61,472**	27.129
5-250-02	**62,130**	61,974	20794	62,033	29.842	**62,130**	31.055
10-250-00	**59,187**	59,069	14.548	59,139	25.679	59,182	28.714
10-250-01	**58,781**	58,577	33.841	58,636	42.972	**58,781**	46.601
10-250-02	**58,097**	57,960	12.337	58,056	21.496	**58,097**	23.522
30-250-00	**56,842**	56,619	34.669	56,754	46.894	56,796	49.612
30-250-01	**58,520**	58,138	20413	58,250	29.848	58,302	31.295
30-250-02	**56,614**	56,403	27.886	56,505	38.662	**56,614**	41.207
5-500-00	**120,148**	119,818	72.501	120,079	88.569	120,130	94.396
5-500-01	**117,879**	117,582	88.963	117,835	103.453	117,844	108.094
5-500-02	**121,131**	120,778	79.066	121,062	93.337	**121,131**	98.998
10-500-00	**117,821**	117,190	88.711	117,660	104.441	117744	109.935
10-500-01	**119249**	118804	87.851	119,067	101.668	119,177	106.955
10-500-02	**119,215**	118,595	84.729	119,071	97772	**119,215**	103.476
30-500-00	**116,056**	115,434	105.683	115,840	119.886	115,991	125.253
30-500-01	**114810**	113,975	93.774	114,568	128.823	114,684	134.910
30-500-02	**116,741**	115,977	107.542	116,509	119.044	116,712	125.746
AVG.	**66,616**	66,328	38.357	66,517	47.267	66590	50.180

$$y^{*s}(k) = \alpha \times x^{*s}(k) + \beta \times (e - x^{*s}(k)),\ \forall\ s \in S \tag{7}$$

$$y^{s}(k+1) = \epsilon_1 \times y^{s}(k) + \epsilon_2 \times y^{*s}(k) + \epsilon_3 \times y^{\#}(k), \quad \forall\ s \in S \tag{8}$$

where e is a vector of ones, $y^s(k)$ refers to the particle swarm $s \in S$ at the current iteration k and $y^s(k+1)$ to its update at the next iteration. Notations $x^{\#}$ and x^{*s} refer to the global best and the local best discrete particle s, respectively, as previously. In these equations, α and β are called control parameters and satisfy $0 \leq \alpha, \beta \leq 1$ with $\alpha + \beta = 1$. Based on similar notations as previously, Eqs. (6) and (7) correspond to the global best quantum particle and the local best quantum particle, respectively.

Eq. (8) describes the evolution of the quantum particle swarm Y, where $0 \leq \epsilon_1, \epsilon_2, \epsilon_3 \leq 1$ with $\epsilon_1 + \epsilon_2 + \epsilon_3 = 1$. The three latter coefficients represent the degree of belief in oneself, the local best, and the global best particle, respectively.

Inside the swarm a topology is defined. It is a set of links between particles, saying who informs whom. The set of particles that informs a particle is called its neighborhood.

Various types of neighborhood topologies are investigated and presented in the literature for PSO (Kennedy, 1999): Ring, Fully connected, Mesh, Star, and Tree topology as shown in the Fig. 1. Each topology has its own way to spread information through the swarm.

The choice for neighborhood topology determines which particle to use for $x^{\#}$. In our implementation, we consider the star (or wheel) topology, also known as G^*, which is a fully-connected neighborhood relation. In the star topology, one particle is selected as a hub, which is connected to all other particles in the swarm. All the other particles are, however, connected only to the hub. The star topology, effectively isolates individuals from each other, since information has to be communicated through the hub node. This hub node compares the performance of every individual in the population and adjusts its own trajectory toward the best of them.

The star topology is static in that their neighbor connections are set at initialization and do not change throughout the search, even if the particles change position. Static topologies have minimal computational overhead since they do not require re-computation and only a single linear pass is needed to update neighborhood bests. Thus, using the G^* model, the propagation becomes very fast (i.e. all the particles in the swarm will be affected by the best solution found during the search process).

This QPSO approach is initialized with a swarm S of p particles. The initial solutions are evaluated, and y^{*s} and $y^{\#}$ are then initialized. During the search process, each particle is moved according to Eqs. (6), (7) and (8). When the particles' positions are updated, they are evaluated, and an MKP-specific Drop/Add repair operator is applied to amend the corresponding solutions (if they are infeasible), or to improve it (when it is possible). If a better position for a particle or for the whole swarm is obtained, then y^{*s} and/or $y^{\#}$ are updated. Then, a local search method is applied to each new best position so far obtained during the search process. This process is repeated until some convergence criteria are satisfied. In this study, the search is stopped once a maximum number of iterations is reached. Algorithm 1 summarizes the QPSO procedure. In this algorithm, $Y(k)$ and $X(k)$ represent the quantum particle swarm and the discrete particle swarm at iteration k, respectively.

Algorithm 1. The QPSO algorithm.

Data: An instance of the MKP.
Result: A feasible solution of the MKP.

```
k = 0 ;
Initialize Y(k) and X(k) ;
Evaluate Y(k);
Store the best solution among X(k);
while not termination-condition do
    k = k + 1;
    Evolve Y(k) using Equations (6), (7) and (8);
    Get X(k) from Y(k) using Equation (5);
    Apply the Drop/Add repair operator;
    Evaluate X(k);
    Apply the Local Search algorithm if a new best position is found;
    Update the best solution among X(k), if necessary ;
end
Return the best solution of the swarm;
```

4. The MKP-specific drop/add repair operator.

We can observe that a potential infeasible solution can be generated for the 0–1 MKP during the process described in Algorithm 1. A particle representing an infeasible solution is a particle that violates, at least, one of the knapsack constraints.

Since the solution representation described above does not guarantee the feasibility of the solutions generated throughout the search process, we have incorporated an MKP-specific Drop/Add repair operator to ensure that the search process will always be guided through the feasible solution space. This idea comes from Chu and Beasley (1998). In recent years, several works using this technique have been proposed for the MKP such as (Kong et al., 2008; Khemakhem et al., 2012; Kong et al., 2006; Ktari and Chabchoub, 2013; Bonyadi and Li, 2012), among others.

The general idea behind this method is described as follows. The repair operator is based on the notion of the pseudo-utility ratio σ_j derived from a given multiplier $u \in \mathbb{R}_+^m$. Hence, the pseudo-utility is defined as follows:

$$\sigma_j = \frac{c_j}{\sum_{i \in M} u_i a_{ij}}, \quad \forall\ j \in N \tag{9}$$

where $u = (u_1, \ldots, u_m) \in \mathbb{R}_+^m$. The multiplier u can be derived, for instance, from the dual variables of the LP–relaxation, or Lagrangian or surrogate relaxation. In this paper, we use the surrogate relaxation (for more details see Pirkul (1987)). The Surrogate Relaxation problem of the 0–1 MKP (denoted SR-MKP) is defined as follows:

$$(\mathbf{SR-MKP}) \begin{cases} \max & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} & \sum_{j=1}^{n} (\sum_{i=1}^{m} u_i a_{ij}) x_j \leq u_i b_i, \forall\ i \in M \\ & x_j \in \{0, 1\}, \forall\ j \in N \end{cases}$$

To obtain reasonably good surrogate weights, we solve the Linear Programming (LP) relaxation of the original MKP and we use the values of the dual variables as the weights. In our implementation, u_i is set to the shadow price of the ith constraint in the LP relaxation of the MKP.

Based on the σ_j values associated with all items, the repair operator consists of two phases: a Drop phase to restore the feasibility of the solution (if necessary) and an Add phase to improve the solution (if possible).

In the drop phase items are considered in the ascending order of the pseudo-utility ratio σ_j. A single item among those already put in the knapsack is removed from the current solution, and this process is repeated as long as the solution is not feasible.

In the add phase items are examined in the descending order of the pseudo-utility ratio σ_j. Only one item among those not already included in the knapsack is added (if possible), and this process is repeated as long as no resource constraint is violated.

We assume that a preprocessing routine is applied to each instance of the problem that sorts and renumbers variables according to the decreasing order of of the pseudo-utility ratio σ_j. The pseudo-code for the repair operator is given in Algorithm 2.

Algorithm 2. Repair operator for the MKP.

Data: A solution x to the MKP (a feasible or an infeasible one).
Result: An updated solution to the MKP (An improved or a feasible one).

Let: $R_i = \sum_{j=1}^{n} a_{ij}x_j, \forall i \in M$;
/* Drop phase */
for $j = n$ *to* 1 **do**
  if $x_j = 1$ *and* $R_i > b_i$, *for some* $i \in M$ **then**
    set $x_j \leftarrow 0$;
    set $R_i \leftarrow R_i - a_{ij}, \forall i \in M$;
  end
end
/* Add phase */
for $j = 1$ *to* n **do**
  if $x_j = 0$ *and* $R_i + a_{ij} \leq b_i, \forall i \in M$ **then**
    set $x_j \leftarrow 1$;
    set $R_i \leftarrow R_i + a_{ij}, \forall i \in M$;
  end
end

This repair operator has a significant role in the quick search for good quality solutions to the 0–1 MKP. In fact, when applied alone, this repair operator can be considered as a problem-specific greedy search approach that provides only solutions of poor quality. When used in combination with the QPSO approach, the repair operator acts as a local search to the solutions found by the QPSO algorithm, which greatly improves the solution quality. This repair operator process can also ensure that the QPSO process be always guided through the feasible solution space by transferring each infeasible solution to the feasible solution domain, thus making the algorithm search around the promising area.

5. A local search method for the 0–1 MKP

In order to improve the quality of the solutions produced by our new QPSO algorithm and to accelerate its convergence to optimal or near-optimal solutions, we propose to use it in combination with a local search method which is launched as an intensification strategy applied to each new best position obtained during the search process. This local search is a classical descent method in which we use a simple neighborhood structure called Drop/Add moves. Starting from an initial solution, the principle of the local search is as follows. We consider every item (from the first to the last one). If the item is already included in the current solution then we drop it and we try to improve the resulting solution. In the case of the item is not included in the current solution then we add it and we try to repair the resulting solution (if necessary).

As add moves are launched to improve the current solution, items are examined in the descending order of the pseudo-utility ratio described above. On the contrary, drop moves are used to repair the infeasible solution, so items are examined in the ascending order of pseudo-utility ratio, and as long as the solution is infeasible. This process is repeated as long as the current solution can be improved. The pseudo-code of this Local Search method is given in Algorithm 3.

Algorithm 3. A Local search method for the MKP.

Data: A feasible solution x for the MKP.
Result: The best neighbor solution x^* of x.
Set $v^* \leftarrow c^T x$, $x^* \leftarrow x$, $Improve \leftarrow True$;
while $Improve$=$True$ **do**
  $Improve \leftarrow False$;
  for $j = 1$ *to* n **do**
    $x' \leftarrow x$;
    if $x'_j = 1$ **then**
      $x'_j \leftarrow 0$;
      Try to improve x' by using the Add phase of Algorithm 2
    end
    else
      $x'_j \leftarrow 1$;
      Amend x' if it is infeasible by using the Drop phase of Algorithm 2;
    end
    $v' \leftarrow c^T x'$;
    if $(v' > v^*)$ **then**
      $v^* \leftarrow v'$;
      $x^* \leftarrow x'$;
      $Improve \leftarrow True$;
    end
  end
  if $Improve = True$ **then**
    $x \leftarrow x^*$;
  end
end

6. Computational results

Our approach is tested on two sets of MKP instances. The first set is available in the OR-Library (Beasley, 1990). It is a collection of 270 correlated, and thus difficult, instances generated using the procedure proposed in Fréville and Gérard (1994). These instances were generated by varying the number of constraints ($m \in \{5, 10, 30\}$) and the number of variables ($n \in \{100, 250, 500\}$). Thirty instances were generated for each $n - m$ size, and each set of 30 instances is divided into 3 series associated the capacities b_i where $b_i = t \times \sum_{j \in N} a_{ij}$ with a tightness ratio $t \in \{0.25, 0.5, 0.75\}$, $\forall\, i \in M$. The second set contains 18 instances proposed by Glover and Kochenberger in Glover and Kochenberger (1996). In this set of instances, the number of items varies between 100 and 2500, while the number of constraints varies between 15 and 100. These instances are very large and they are known to be very hard to solve

Table 2
Performance comparison on benchmark instances with $m=5$ and $n=100$.

Prob.		GA	F&F	S-CLPSO	BAPSA	TEPSOq	SACRO-BPSO(1)	SACRO-BPSO(2)	QPSO*	
Inst.	*Opt.*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Avg.*
5-100-00	**24,381**	**24,381**	**24,381**	**24,381**	**24,381**	24,381	24,343	24,343	**24,381**	24,381.0
5-100-01	**24,274**	**24,274**	**24,274**	**24,274**	**24,274**	**24,274**	**24,274**	**24,274**	**24,274**	24,274.0
5-100-02	**23,551**	**23,551**	**23,551**	**23,551**	23,523	**23,551**	23,538	23,538	**23,551**	23,551.0
5-100-03	**23,534**	**23,534**	**23,534**	23,527	23,486	**23,534**	23,527	23,527	**23,534**	23,534.0
5-100-04	**23,991**	**23,991**	**23,991**	**23,991**	23,966	**23,991**	**23,991**	23,966	**23,991**	23,991.0
5-100-05	**24,613**	**24,613**	**24,613**	**24,613**	*	**24,613**	24,601	24,601	**24,613**	24,613.0
5-100-06	**25,591**	**25,591**	**25,591**	**25,591**	*	**25,591**	**25,591**	**25,591**	**25,591**	25,591.0
5-100-07	**23,410**	**23,410**	**23,410**	**23,410**	*	**23,410**	**23,410**	**23,410**	**23,410**	23,410.0
5-100-08	**24,216**	**24,216**	**24,216**	**24,216**	*	24,195	24,204	**24,216**	**24,216**	24,216.0
5-100-09	**24,411**	**24,411**	**24,411**	**24,411**	*	24,375	24,399	**24,411**	**24,411**	24,411.0
Avg.	**24,197**	**24,197**	**24,197**	24,194	*	24,192	24,188	24,191	**24,197**	24,197.0
5-100-10	**42,757**	**42,757**	**42,757**	*	*	*	42,705	42,705	**42,757**	42,757.0
5-100-11	**42,545**	**42,545**	**42,545**	*	*	*	42,494	42,471	**42,545**	42,545.0
5-100-12	**41,968**	**41,968**	**41,968**	*	*	*	41,959	41,959	**41,968**	41,968.0
5-100-13	**45,090**	**45,090**	**45,090**	*	*	*	**45,090**	**45,090**	**45,090**	45,090.0
5-100-14	**42,218**	**42,218**	**42,218**	*	*	*	**42,218**	**42,218**	**42,218**	42,218.0
5-100-15	**42,927**	**42,927**	**42,927**	*	*	*	**42,927**	**42,927**	**42,927**	42,927.0
5-100-16	**42,009**	**42,009**	**42,009**	*	*	*	**42,009**	**42,009**	**42,009**	42,009.0
5-100-17	**45,020**	**45,020**	**45,020**	*	*	*	45,010	**45,020**	**45,020**	45,020.0
5-100-18	**43,441**	**43,441**	**43,441**	*	*	*	**43,441**	43,381	**43,441**	43,441.0
5-100-19	**44,554**	**44,554**	**44,554**	*	*	*	**44,554**	44,529	**44,554**	44,554.0
Avg.	**43,253**	**43,253**	**43,253**	*	*	*	43,241	43,231	**43,253**	43,253.0
5-100-20	**59,822**	**59,822**	**59,822**	*	*	*	**59,822**	**59,822**	**59,822**	59,822.0
5-100-21	**62,081**	**62,081**	**62,081**	*	*	*	**62,081**	**62,081**	**62,081**	62,081.0
5-100-22	**59,802**	**59,802**	**59,802**	*	*	*	**59,802**	59,754	**59,802**	59,802.0
5-100-23	**60,479**	**60,479**	**60,479**	*	*	*	60,478	60,478	**60,479**	60,479.0
5-100-24	**61,091**	**61,091**	**61,091**	*	*	*	61,055	61,079	**61,091**	61,091.0
5-100-25	**58,959**	**58,959**	**58,959**	*	*	*	**58,959**	58,937	**58,959**	58,959.0
5-100-26	**61,538**	**61,538**	**61,538**	*	*	*	**61,538**	**61,538**	**61,538**	61,538.0
5-100-27	**61,520**	**61,520**	**61,520**	*	*	*	61,489	**61,520**	**61,520**	61,520.0
5-100-28	**59,453**	**59,453**	**59,453**	*	*	*	**59,453**	**59,453**	**59,453**	59,453.0
5-100-29	**59,965**	**59,965**	**59,965**	*	*	*	59,960	59,960	**59,965**	59,965.0
Avg.	**60,471**	**60,471**	**60,471**	*	*	*	60,464	60,462	**60,471**	60,471.0

using branch-and-bound methods.

The algorithms described above were coded in C language, and experimental tests were performed on a Personal Computer with a 2.2 GHz Core 2 Duo processor and 3 GB RAM. We performed some preliminary experiments to find the "*good*" values of the QPSO parameters. We noted that our algorithm displayed better performance for $\alpha < 0.3$. High values implied slow convergence and low values implied convergence to non-optimal performance values. Thus, we chose $\alpha = 0.1$ and $\beta = 0.9$. We also observed that the weight of best global position factor (ϵ_3) was more important than ϵ_1 and ϵ_2. Low values for ϵ_3 implied slow convergence and high values result in a lack of diversity. A good trade-off was obtained for the values $\epsilon_1 = 0.4$, $\epsilon_2 = 0.2$ and $\epsilon_3 = 0.4$. Our QPSO algorithm starts with an initial population containing 20 particles generated randomly in an n-dimensional discrete space. In order to balance between the running time and the quality of the final solution, the maximum number of iterations was set at 500 iterations. The experimental tests are based on 30 runs.

6.1. Evaluation of the contribution of each component of our approach

The proposed QPSO approach is a complex method combining different features. In order to better state the contribution provided by its components, we consider three variants of our method. The first variant, termed as baptized QPSO, is composed by the QPSO algorithm alone. In this variant, if a particle flies outside the feasible space, its fitness is not evaluated and the corresponding solution is discarded (i.e., we do not use the repair operator in this variant). The next variant, namely QPSO+RO, incorporates the MKP-specific Drop/Add repair operator described in Section 4 within the QPSO approach to ensure that the search process is always guided through the feasible space. Finally, the last version, called QPSO*, incorporates the repair operator and the Local Search method described in Section 5.

Table 1 provides a summary of the results obtained by the three variants of our approach over a reduced set of 27 instances selected among the 270 benchmarks of OR-Library (Beasley, 1990), taking into account only three instances for each $n - m$ size (the first three instances for each $n - m$ size). The first two columns present respectively, the instance name and the corresponding optimal or best known solution value. The remaining columns provide, for each variant, the best solution value found and the corresponding computation time in seconds. The best results in this Table are marked in bold characters.

Table 1 shows that the first variant of our approach obtains solution values of poor quality compared with the two other variants but reports a significant reduced amount of computational effort.

The QPSO algorithm, with the MKP-specific Drop/Add repair operator, needs larger computational effort than QPSO alone but provides solution of high quality. More concretely, the average solution values increases from 66,328 with QPSO alone to 66,517 with QPSO+RO and the number of the optimal or the best known solutions visited increases from 0 to 2.

By incorporating a local search strategy within the QPSO algorithm, the results are logically enhanced. Indeed, the average solution values increases from 66,517 with QPSO+RO to 66,590 with QPSO* and the number of the optimal or the best known solutions visited increases from 2 to 17. In terms of computation time, we can notice that the local search does not lead to an important extra time since the average computation time reported for QPSO* is close to the one of QPSO+RO.

These results demonstrate that the incorporation of both the repair operator and the local search leads to an important improvement of the performance of our approach. According to this observation, we consider the QPSO* variant in the following.

6.2. A comparison with population-based algorithms over the 270 benchmarks of or-library

Tables 2–10 compare the solutions obtained by our QPSO* approach over the 270 0–1 MKP benchmarks of the OR-Library with those obtained by different population-based algorithms including the Filter-and-Fan algorithm (F&F) proposed in Khemakhem et al. (2012), the Genetic Algorithm (GA) introduced in Chu and Beasley (1998), the Novel Set-Based Particle Swarm Optimization Method (S-CLPSO) proposed in Chen et al. (2010), the Binary Accelerated Particle Swarm Algorithm (BAPSA) presented in Beheshti et al. (2013), the Essential Particle Swarm Optimization queen with Tabu Search (TEPSOq) proposed in Ktari and Chabchoub (2013) and the two strategies of the Self-adaptive Check and Repair Operator-based Particle Swarm Optimization, SACRO-BPSO(1) and SACRO-BPSO(2), proposed in Chih (2015).

In these tables, the first two columns present respectively, the instance name and the optimal or the corresponding best known solution value. The last two columns present respectively, the best and the average solution value found by our QPSO* approach for each instance over 30 runs. The remaining columns provide, for each heuristic, the corresponding best solution. The character "*" in these tables means that the value is not available. If all values are not availables for an $n-m$ size then the algorithm is not presented in the corresponding table. The best results are marked in bold characters.

Tables 2–4 show that our algorithm finds an optimal solution for 89 out of 90 instances, leading to an interesting performance. The corresponding instances have (only) 100 variables, but the values reported in these tables show that other PSO-based approaches are not able to provide as far as optimal solutions. More generally, the performance of our hybrid QPSO* approach over these instances is similar to the one of the GA and the F&F.

In addition, Tables 6 and 10 clearly show that our approach provides results that are better than those produced by TEPSOq, whereas Table 8 confirms the superiority of our QPSO* algorithm compared to SACRO-BPSO(1) and SACRO-BPSO(2).

Globally, we can observe from these 9 tables that our algorithm provides better solutions than the GA, in particular for the largest instances (with $m=30$ or $n=500$).

Finally, the results show that our approach is competitive with the F&F; none of the approaches has an advantage over the other. Indeed, QPSO* finds 172 best known solutions over 270 problems while F&F finds only 147 best known solutions over 270.

Table 3
Performance comparison on benchmark instances with $m=10$ and $n=100$.

Prob.		**GA**	**F&F**	**S-CLPSO**	**BAPSA**	**TEPSOq**	**SACRO-BPSO(1)**	**SACRO-BPSO(2)**	**QPSO***	
Inst.	*Opt.*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Best*	*Avg.*
10-100-00	**23,064**	**23,064**	**23,064**	23,057	23,055	**23,064**	**23,064**	**23,064**	**23,064**	23,064.0
10-100-01	**22,801**	**22,801**	**22,801**	**22,801**	22,753	**22,801**	22,739	22,750	**22,801**	22,801.0
10-100-02	**22,131**	**22,131**	**22,131**	**22,131**	22,081	**22,131**	**22,131**	**22,131**	**22,131**	22,131.0
10-100-03	**22,772**	**22,772**	**22,772**	**22,772**	22,643	**22,772**	**22,772**	22,717	**22,772**	22,772.0
10-100-04	**22,751**	**22,751**	**22,751**	22,697	**22,751**	**22,751**	**22,751**	**22,751**	**22,751**	22,751.0
10-100-05	**22,777**	**22,777**	22,739	22,703	*	22,716	22,725	22,716	**22,777**	22,758.0
10-100-06	**21,875**	**21,875**	**21,875**	21,821	*	21,821	**21,875**	**21,875**	**21,875**	21,875.0
10-100-07	**22,635**	**22,635**	**22,635**	**22,635**	*	22,573	22,551	22,542	**22,635**	22,635.0
10-100-08	**22,511**	**22,511**	**22,511**	22,422	*	**22,511**	**22,511**	22,438	**22,511**	22,511.0
10-100-09	**22,702**	**22,702**	**22,702**	**22,702**	*	**22,702**	**22,702**	**22,702**	**22,702**	22,702.0
Avg.	**22,602**	**22,602**	22,598	22,574	*	22,584	22,582	20,521	**22,602**	22,600.0
10-100-10	**41,395**	**41,395**	**41,395**	*	*	*	**41,395**	41,388	**41,395**	41,395.0
10-100-11	**42,344**	**42,344**	**42,344**	*	*	*	**42,344**	**42,344**	**42,344**	42,344.0
10-100-12	**42,401**	**42,401**	**42,401**	*	*	*	42,350	42,350	**42,401**	42,401.0
10-100-13	**45,624**	**45,624**	**45,624**	*	*	*	45,585	45,511	**45,624**	45,611.0
10-100-14	**41,884**	**41,884**	**41,884**	*	*	*	41,799	41,833	**41,884**	41,884.0
10-100-15	**42,995**	**42,995**	**42,995**	*	*	*	**42,995**	**42,995**	**42,995**	42,995.0
10-100-16	**43,574**	43,559	**43,574**	*	*	*	43,497	43,517	43,553	43,553.0
10-100-17	**42,970**	**42,970**	**42,970**	*	*	*	**42,970**	**42,970**	**42,970**	42,970.0
10-100-18	**42,212**	**42,212**	**42,212**	*	*	*	**42,212**	**42,212**	**42,212**	42,212.0
10-100-19	**41,207**	**41,207**	**41,207**	*	*	*	41,123	41,134	**41,207**	41,207.0
Avg.	**42,661**	42,659	**42,661**	*	*	*	34,948	34,947	42,659	42,657.2
10-100-20	**57,375**	**57,375**	**57,375**	*	*	*	**57,375**	**57,375**	**57,375**	57,375.0
10-100-21	**58,978**	**58,978**	**58,978**	*	*	*	58,922	**58,978**	**58,978**	58,978.0
10-100-22	**58,391**	**58,391**	**58,391**	*	*	*	**58,391**	**58,391**	**58,391**	58,391.0
10-100-23	**61,966**	**61,966**	**61,966**	*	*	*	**61,966**	**61,966**	**61,966**	61,966.0
10-100-24	**60,803**	**60,803**	**60,803**	*	*	*	**60,803**	**60,803**	**60,803**	60,803.0
10-100-25	**61,437**	**61,437**	**61,437**	*	*	*	61,368	61,368	**61,437**	61,437.0
10-100-26	**56,377**	**56,377**	**56,377**	*	*	*	**56,377**	**56,377**	**56,377**	56,377.0
10-100-27	**59,391**	**59,391**	**59,391**	*	*	*	59,332	**59,391**	**59,391**	59,391.0
10-100-28	**60,205**	**60,205**	**60,205**	*	*	*	**60,205**	**60,205**	**60,205**	60,205.0
10-100-29	**60,633**	**60,633**	**60,633**	*	*	*	60,629	60,629	**60,633**	60,633.0
Avg.	**59,556**	**59,556**	**59,556**	*	*	*	59,537	59,548	**59,556**	59,555.6

Table 4
Performance comparison on benchmark instances with $m=30$ and $n=100$.

Prob.		GA	F&F	QPSO*	
Inst.	*Opt.*	*Best*	*Best*	*Best*	*Avg.*
30-100-00	**21,946**	**21,946**	**21,946**	**21,946**	21,946.0
30-100-01	**21,716**	**21,716**	**21,716**	**21,716**	21,716.0
30-100-02	**20,754**	**20,754**	**20,754**	**20,754**	20,754.0
30-100-03	**21,464**	**21,464**	**21,464**	**21,464**	21,448.0
30-100-04	**21,844**	21,814	**21,844**	**21,844**	21,828.5
30-100-05	**22,176**	**22,176**	**22,176**	**22,176**	22,176.0
30-100-06	**21,799**	**21,799**	**21,799**	21,772	21,772.0
30-100-07	**21,397**	**21,397**	**21,397**	**21,397**	21,361.5
30-100-08	**22,525**	22,493	22,493	**22,525**	22,503.5
30-100-09	**20,983**	**20,983**	**20,983**	**20,983**	20,983.0
Avg.	**21,660**	21,654	21,657	21,658	21,648.9
30-100-10	**40,767**	**40,767**	**40,767**	**40,767**	40,728.5
30-100-11	**41,308**	41,304	41,304	**41,308**	41,306.0
30-100-12	**41,630**	41,560	**41,630**	**41,630**	41,606.0
30-100-13	**41,041**	**41,041**	**41,041**	**41,041**	41,041.0
30-100-14	**40,889**	40,872	**40,889**	40,872	40,872.0
30-100-15	**41,058**	**41,058**	**41,058**	**41,058**	41,058.0
30-100-16	**41,062**	**41,062**	**41,062**	**41,062**	41,062.0
30-100-17	**42,719**	**42,719**	**42,719**	**42,719**	42,719.0
30-100-18	**42,230**	**42,230**	**42,230**	**42,230**	42,230.0
30-100-19	**41,700**	**41,700**	**41,700**	**41,700**	41,700.0
Avg.	**41,440**	41,431	**41,440**	41,439	41,432.3
30-100-20	**57,494**	**57,494**	**57,494**	**57,494**	57,494.0
30-100-21	**60,027**	**60,027**	**60,027**	**60,027**	60,027.0
30-100-22	**58,052**	58,025	**58,052**	**58,052**	58,052.0
30-100-23	**60,776**	**60,776**	**60,776**	**60,776**	60,776.0
30-100-24	**58,884**	**58,884**	**58,884**	**58,884**	58,884.0
30-100-25	**60,011**	**60,011**	**60,011**	**60,011**	60,011.0
30-100-26	**58,132**	**58,132**	58,104	**58,132**	58,118.0
30-100-27	**59,064**	**59,064**	**59,064**	**59,064**	59,064.0
30-100-28	**58,975**	**58,975**	**58,975**	**58,975**	58,975.0
30-100-29	**60,603**	**60,603**	**60,603**	60,593	60,593.0
Avg.	**59,202**	59,199	59,199	59,201	59,199.4

Table 5
Performance comparison on benchmark instances with $m=5$ and $n=250$.

Prob.		GA	F&F	QPSO*	
Inst.	*Opt.*	*Best*	*Best*	*Best*	*Avg.*
5-250-00	**59,312**	**59,312**	**59,312**	**59,312**	59,312.0
5-250-01	**61,472**	**61,472**	61,468	**61,472**	61,470.0
5-250-02	**62,130**	**62,130**	**62,130**	**62,130**	62,130.0
5-250-03	**59,463**	59,446	59,436	59,427	59,427.0
5-250-04	**58,951**	**58,951**	**58,951**	**58,951**	58,951.0
5-250-05	**60,077**	60,056	60,062	**60,077**	60,056.0
5-250-06	**60,414**	**60,414**	**60,414**	**60,414**	60,414.0
5-250-07	**61,472**	**61,472**	61,454	**61,472**	61,460.5
5-250-08	**61,885**	**61,885**	**61,885**	**61,885**	61,885.0
5-250-09	**58,959**	**58,959**	**58,959**	**58,959**	58,925.5
Avg.	**60,414**	60,410	60,407	60,410	60,403.1
5-250-10	**109,109**	**109,109**	**109,109**	109,066	109,058.5
5-250-11	**109,841**	**109,841**	**109,841**	**109,841**	109,841.0
5-250-12	**108,508**	108,489	**108,508**	**108,508**	108,508.0
5-250-13	**109,383**	**109,383**	**109,383**	109,356	109,347.5
5-250-14	**110,720**	**110,720**	**110,720**	**110,720**	110,710.0
5-250-15	**110,256**	**110,256**	**110,256**	**110,256**	110,256.0
5-250-16	**109,040**	109,016	**109,040**	**109,040**	109,022.5
5-250-17	**109,042**	109,037	109,016	**109,042**	109,018.5
5-250-18	**109,971**	109,957	109,957	**109,971**	109,955.0
5-250-19	**107,058**	107,038	107,058	**107,058**	107,048.0
Avg.	**109,293**	109,285	109,289	109,286	109,276.5
5-250-20	**149,665**	149,659	149,659	**149,665**	149,650.5
5-250-21	**155,944**	155,940	**155,944**	**155,944**	155,942.0
5-250-22	**149,334**	149,316	**149,334**	**149,334**	149,334.0
5-250-23	**152,130**	**152,130**	**152,130**	152,130	152,130.0
5-250-24	**150,353**	**150,353**	**150,353**	**150,353**	150,353.0
5-250-25	**150,045**	**150,045**	**150,045**	**150,045**	150,045.0
5-250-26	**148,607**	**148,607**	**148,607**	**148,607**	148,607.0
5-250-27	**149,782**	149,772	**149,782**	149,772	149,762.5
5-250-28	**155,075**	**155,075**	**155,075**	155,057	155,045.0
5-250-29	**154,668**	154,662	**154,668**	**154,668**	154,668.0
Avg.	**151,560**	151,556	**151,560**	151,558	151,553.7

6.3. *A comparison with other state-of-the-art heuristic methods over the 90 largest or-library problems*

In order to properly evaluate the performance of our approach in terms of computation time and quality of solutions, we consider the most relevant state-of-the-art heuristic methods and the 90 most difficult instances in the OR-Library (those with $n=500$).

Table 11 extends Tables 5 and 6 presented in Vasquez and Vimont (2005) and Khemakhem et al. (2012), respectively, where each row summarizes 10 instances. This table compares the results in terms of objective values and computation times in hours obtained by our QPSO* approach with those reported by F&F, GA, the surrogate constraints and cutting plane based approach (Fix+Cuts) proposed in Osorio et al. (2002) (see also Hanafi and Glover, 2007), the hybrid approach that combines Linear Programming and Tabu Search (LP+TS) proposed in Vasquez and Hao (2001), the hybrid approach that combines variable fixing, Linear Programming and Tabu Search (Fix+LP+TS) presented in Vasquez and Vimont (2005) and finally the hybrid algorithm of Nested Partition, Binary Ant System and Linear Programming (NP+BAS+LP) proposed in Al-Shihabi and ilafsson (2010). The best results in this Table are marked in bold characters.

Since the Personal Computer used to evaluate the performance of QPSO* is the same one used to evaluate the performance of the F&F algorithm proposed in Khemakhem et al. (2012), the computation times provided in this paper are equal to the original ones presented in Khemakhem et al. (2012). Note that the computation times provided in Al-Shihabi and ilafsson (2010) (Xeon 2.5 GHz) and in Khemakhem et al. (2012) (Core 2 Duo 2.2 GHz) are half of the original ones provided in Vasquez and Vimont (2005) (P4 2 GHz).

In terms of solutions quality, Table 11 confirms that our QPSO* approach provides in average better solutions than GA. The results also show that our method rivals with Fix+Cuts in the most of problem sets. Finally, the results confirm that our approach is competitive with F&F.

The values reported in Table 11 show that LP+TS and NP+BAS+LP provide solutions that are slightly better than those obtained by our QPSO* approach, but these methods require a greater computational effort. Finally, we should notice that the solutions produced by Fix+LP+TS are the best and they clearly demonstrate that this approach dominates all the other approaches. However, high computation time is needed to reach these quality results.

In terms of processing times, a notable observation concerns the weakness of the Fix+Cuts, LP+TS and FIX+LP+TS with respect to the computation times needed to find their best solutions. Indeed, the computation times of QPSO* vary between 0.03 and 0.05 hours, while the computation times of LP+TS and Fix+LP+TS vary between 2 and 6 hours and 4 and 16 hours, respectively. The computation times of Fix+Cuts was fixed to 1.5 hour.

6.4. *QPSO* results for the 18 instances of Glover and Kochenberger*

To more evaluate the performance of our approach in terms of computation time and quality of solutions, we provide our results on the 18 instances proposed by Glover and Kochenberger in

Table 6
Performance comparison on benchmark instances with $m=10$ and $n=250$.

Prob.		GA	F&F	TEPSOq	QPSO*	
Inst.	*Opt.*	*Best*	*Best*	*Best*	*Best*	*Avg.*
10-250-00	**59,187**	**59,187**	59,164	**59,187**	59,182	59,173.0
10-250-01	**58,781**	58,662	58,693	**58,781**	**58,781**	58,733.0
10-250-02	**58,097**	58,094	58,094	**58,097**	**58,097**	58,095.5
10-250-03	**61,000**	**61,000**	60,972	60,662	**61,000**	60,986.0
10-250-04	**58,092**	**58,092**	**58,092**	**58,092**	**58,092**	58,092.0
10-250-05	**58,824**	58,803	**58,824**	58,549	**58,824**	58,824.0
10-250-06	**58,704**	58,607	58,632	58,350	58,606	58,596.5
10-250-07	**58,936**	58,917	58,917	57,902	58,902	58,889.5
10-250-08	**59,387**	59,384	59,381	**59,387**	59,372	59,357.5
10-250-09	**59,208**	59,193	**59,208**	**59,208**	**59,208**	59,208.0
Avg.	**59,022**	58,994	58,998	58,822	59,006	58,995.5
10-250-10	**110,913**	110,863	110,889	**110,913**	110,857	110,843.0
10-250-11	**108,717**	108,659	108,702	108,713	108,687	108,687.0
10-250-12	**108,932**	**108,932**	108,922	108,491	108,891	108,889.0
10-250-13	**110,086**	110,037	110,059	**110,086**	**110,086**	110,060.5
10-250-14	**108,485**	108,423	**108,485**	108,225	**108,485**	108,459.5
10-250-15	**110,845**	110,841	110,841	110,257	**110,845**	110,843.0
10-250-16	**106,077**	106,075	106,075	**106,077**	106,047	106,036.0
10-250-17	**106,686**	**106,686**	106,685	106,455	**106,686**	106,681.5
10-250-18	**109,829**	109,825	109,822	109,225	109,788	109,755.0
10-250-19	**106,723**	**106,723**	**106,723**	**106,723**	**106,723**	106,723.0
Avg.	**108,729**	108,706	108,720	108,517	108,710	108,697.8
10-250-20	**151,809**	151,790	151,790	151,194	151,779	151,769.0
10-250-21	**148,772**	**148,772**	**148,772**	**148,772**	**148,772**	148,772.0
10-250-22	**151,909**	151,900	**151,909**	151,858	**151,909**	151,909.0
10-250-23	**151,324**	151,275	151,281	**151,324**	151,281	151,281.0
10-250-24	**151,966**	151,948	**151,966**	151,372	**151,966**	151,938.0
10-250-25	**152,109**	**152,109**	**152,109**	152,007	**152,109**	152,109.0
10-250-26	**153,131**	**153,131**	**153,131**	153,046	**153,131**	153,131.0
10-250-27	**153,578**	153,520	153,533	**153,578**	153,529	153,529.0
10-250-28	**149,160**	149,155	**149,160**	**149,160**	**149,160**	149,145.0
10-250-29	**149,704**	**149,704**	149,688	149,637	149,646	149,637.0
Avg.	**151,346**	151,330	151,334	151,195	151,328	151,322.0

Table 7
Performance comparison on benchmark instances with $m=30$ and $n=250$.

Prob.		GA	F&F	QPSO*	
Inst.	*Opt./ best − known*	*Best*	*Best*	*Best*	*Avg.*
30-250-00	**56,842**	56,693	56,796	56,796	56,745.5
30-250-01	**58,520**	58,318	58,333	58,302	58,302.0
30-250-02	**56,614**	56,553	56,553	**56,614**	56,570.5
30-250-03	**56,930**	56,863	**56,930**	**56,930**	56,892.0
30-250-04	**56,629**	**56,629**	**56,629**	**56,629**	56,629.0
30-250-05	**57,205**	57,119	57,149	57,146	57,115.5
30-250-06	**56,348**	56,292	56,263	56,303	56,246.5
30-250-07	**56,457**	56,403	**56,457**	56,392	56,374.5
30-250-08	**57,447**	57,442	57,373	**57,447**	57,407.5
30-250-09	**56,447**	**56,447**	**56,447**	**56,447**	56,447.0
Avg.	**56,944**	56,876	56,893	56,901	56,873.0
30-250-10	**107,770**	107,689	107,735	107,703	107,696.0
30-250-11	**108,392**	108,338	**108,338**	**108,338**	108,336.5
30-250-12	**106,442**	106,385	106,415	**106,442**	106,413.5
30-250-13	**106,876**	106,796	106,832	106,851	106,828.0
30-250-14	**107,414**	107,396	**107,414**	107,382	107,382.0
30-250-15	**107,271**	107,246	**107,271**	**107,271**	107,236.5
30-250-16	**106,372**	106,308	106,277	106,248	106,242.0
30-250-17	**104,032**	103,993	104,003	103,988	103,988.0
30-250-18	**106,856**	106,835	106,835	**106,856**	106,845.5
30-250-19	**105,780**	105,751	105,742	105,751	105,740.0
Avg.	**106,721**	106,674	106,686	106,683	106,670.8
30-250-20	**150,163**	150,083	150,138	150,096	150,052.0
30-250-21	**149,958**	149,907	**149,958**	**149,958**	149,932.5
30-250-22	**153,007**	152,993	**153,007**	**153,007**	153,007.0
30-250-23	**153,234**	153,169	153,182	**153,234**	153,200.0
30-250-24	**150,287**	**150,287**	**150,287**	**150,287**	150,287.0
30-250-25	**148,574**	148,544	148,549	148,544	148,528.5
30-250-26	**147,477**	147,471	147,455	147,471	147,463.0
30-250-27	**152,912**	152,841	152,841	152,835	152,835.0
30-250-28	**149,570**	149,568	**149,570**	**149,570**	149,541.0
30-250-29	**149,668**	149,572	149,587	**149,668**	149,620.0
Avg.	**150,485**	150,444	150,457	150,467	150,446.6

Glover and Kochenberger (1996). These instances are divided into two subsets where the first one is constituted of 7 instances and the second one is constituted of 11 instances. For the first subset, the number of items varies between 100 and 500, while the number of constraints varies between 15 and 25. For the second subset, the number of items varies between 100 and 2500, while the number of constraints varies between 15 and 100. These two subsets of instances are known to be very hard to solve using branch-and-bound methods.

Table 12 compares the results obtained by our QPSO* approach (columns 6 and 7) with those reported by the tabu search (TS_HF) approach proposed in Hanafi and Freville (1998) (column 4) and the Linear Programming and Tabu Search (LP+TS) approach proposed in Vasquez and Hao (2001) (column 5) on the first subset of instances. The best results in this Table are marked in bold characters.

The results presented in Table 12 show that our QPSO* approach find the same values found by LP+TS in a reduced amount of computation time. Indeed, Vasquez and Vimont cited in Vasquez and Hao (2001) that an average of 380 seconds for the test problems GK18…GK22, 600 seconds for GK23 and 1100 seconds for GK24 were needed to get the reported values whereas the total execution time reported by QPSO* for all the 7 problems was less than 214 seconds. The results also show that our approach provides better solutions than TS_HF.

Table 13 compares the results obtained by our QPSO* approach (columns 8 and 9) with those reported by the tabu search (TS_GK) approach proposed in Glover and Kochenberger (1996) (column 4), by the F&F algorithm proposed in Khemakhem et al. (2012) (columns 5 and 6), the Linear Programming and Tabu Search (LP+TS) approach proposed in Vasquez and Hao (2001) (column 7) and the most recent and relevant exact algorithm, called CORAL (for CORe ALgorithm), proposed by Mansini and Speranza in Mansini and Speranza (2012) (columns 10 and 11) on the 11 most difficult instances of Glover and Kochenberger (1996). Note that in Mansini and Speranza (2012), the authors cited that a time limit of five hours for each instance was assigned to CORAL. The character * in this Table means that the value is not available. The best results are marked in bold characters.

In terms of solutions quality, the results presented in Table 13 confirm that our QPSO* approach provides better solutions than TS_GK, CORAL and F&F. The results also show that our method rivals with LP+TS in the most of instances.

In terms of processing times, the results clearly show that our QPSO* approach provides its best solutions in a short amount of computation time compared to F&F, CORAL and TS_GK. Indeed, Vasquez and Vimont cited in Vasquez and Hao (2001) that up to three days were needed to get the reported values, whereas the total execution time reported by F&F was about three hours and the total execution time reported by QPSO* was less than one hour. Furthermore, CORAL needs more than one day to solve the test problems MK_GK01…MK_GK07, whereas the total execution time reported by QPSO* for these problems was less than 245 seconds. Even taking into account the relative speed of the processors involved, the difference in computation time is too large to be ignored.

These results demonstrate the robustness of our QPSO* approach compared to other state-of-the-art methods for the MKP. The results also demonstrate that the proposed method is highly

Table 8
Performance comparison on benchmark instances with $m=5$ and $n=500$.

Prob.		GA	F&F	SACRO-BPSO(1)	SACRO-BPSO(2)	QPSO*	
Inst.	*Opt.*	*Best*	*Best*	*Best*	*Best*	*Best*	*Avg.*
5-500-00	**120,148**	120,130	120,134	119,867	120,009	120,130	120,105.7
5-500-01	**117,879**	117,837	117,864	117,681	117,699	117,844	117,834.3
5-500-02	**121,131**	121,109	**121,131**	120,951	120,923	121,131	121,092.0
5-500-03	**120,804**	120,798	120,794	12,045	120,563	120,752	120,740.3
5-500-04	**122,319**	**122,319**	**122,319**	122,037	122,054	**122,319**	122,300.7
5-500-05	**122,024**	122,007	**122,024**	121,918	121,901	**122,024**	121,981.7
5-500-06	**119,127**	119,113	119,109	118,771	118,846	119,094	119,075.0
5-500-07	**120,568**	**120,568**	**120,568**	120,364	120,376	120,536	120,513.3
5-500-08	**121,586**	121,575	121,575	121,201	121,185	**121,586**	121,527.3
5-500-09	**120,717**	120,699	120,707	120,471	120,453	120,685	120,662.3
Avg.	**120,630**	120,616	120,623	109,531	120,401	120,610	120,583.3
5-500-10	**218,428**	218,422	**218,428**	218,291	218,269	**218,428**	218,394.7
5-500-11	**221,202**	221,191	**221,202**	221,025	221,007	**221,202**	221,152.3
5-500-12	**217,542**	217,534	217,534	217,337	217,398	217,528	217,513.0
5-500-13	**223,560**	223,558	223,558	223,429	223,450	**223,560**	223,537.7
5-500-14	**218,966**	218,962	**218,966**	*	*	218,965	218,964.3
5-500-15	**220,530**	220,514	**220,530**	220,337	220,428	220,527	220,498.7
5-500-16	**219,989**	219,987	**219,989**	219,686	219,734	219,943	219,931.3
5-500-17	**218,215**	218,194	**218,215**	218,094	218,096	**218,215**	218,185.0
5-500-18	**216,976**	**216,976**	**216,976**	216,785	216,851	**216,976**	216,955.3
5-500-19	**219,719**	219,693	**219,719**	219,561	219,549	**219,719**	219,698.0
Avg.	**219,513**	219,503	219,512	*	*	219,506	219,483.0
5-500-20	**295,828**	**295,828**	**295,828**	295,346	295,309	**295,828**	295,797.7
5-500-21	**308,086**	308,077	308,079	307,666	307,808	**308,086**	308,064.0
5-500-22	**299,796**	**299,796**	**299,796**	299,292	299,393	299,788	299,778.0
5-500-23	**306,480**	306,476	306,476	305,915	305,992	**306,480**	306,466.3
5-500-24	**300,342**	**300,342**	**300,342**	29,981	299,947	**300,342**	300,310.0
5-500-25	**302,571**	302,560	**302,571**	302,132	302,156	302,560	302,547.0
5-500-26	**301,339**	301,322	301,329	300,905	300,854	301,322	301,317.3
5-500-27	**306,454**	306,430	306,430	306,132	306,069	306,422	306,409.0
5-500-28	302,828	302,814	302,814	302,436	302,447	302,828	302,808.7
5-500-29	**299,910**	299,904	299,904	299,456	299,558	**299,910**	299,885.3
Avg.	**302,363**	302,355	302,357	274,926	301,953	302,357	302,338.3

Table 9
Performance comparison on benchmark instances with $m=10$ and $n=500$.

Prob.		GA	F&F	TEPSOq	QPSO*	
Inst.	*Opt./best − known*	*Best*	*Best*	*Best*	*Best*	*Avg.*
10-500-00	**117,821**	117,726	117,734	117,811	117,744	117,733.5
10-500-01	**119,249**	119,139	119,181	119,232	119,177	119,148.5
10-500-02	**119,215**	119,159	119,194	118,997	**119,215**	119,146.5
10-500-03	**118,829**	118,802	118,784	117,999	118,775	118,747.5
10-500-04	**116,530**	116,434	116,471	115,828	116,502	116,449.5
10-500-05	**119,504**	119,454	119,442	119,410	119,402	119,391.5
10-500-06	**119,827**	119,749	119,764	119,063	**119,827**	119,784.0
10-500-07	**118,344**	118,288	118,309	118,329	118,309	118,282.5
10-500-08	**117,815**	117,779	117,781	117,025	117,721	117,710.0
10-500-09	**119,251**	119,125	119,183	117,815	**119,251**	119,200.5
Avg.	118,639	118,566	118,584	118,151	118,592	118,559.4
10-500-10	**217,377**	217,318	217,318	**217,377**	217,308	217,289.5
10-500-11	**219,077**	219,022	219,036	219,068	**219,077**	219,049.5
10-500-12	**217,847**	217,772	217,797	**217,847**	217,797	217,772.0
10-500-13	**216,868**	216,802	216,843	216,257	**216,868**	216,826.0
10-500-14	**213,873**	213,809	213,811	213,796	13,795	213,783.0
10-500-15	**215,086**	215,013	215,021	**215,086**	**215,086**	215,053.5
10-500-16	**217,940**	217,896	217,880	217,825	217,868	217,853.0
10-500-17	**219,990**	219,949	219,969	219,825	219,949	219,919.5
10-500-18	**214,382**	214,332	214,346	214,368	**214,382**	214,364.0
10-500-19	**220,899**	220,833	220,849	220,168	220,827	220,814.5
Avg.	**217,334**	217,275	217,287	217,162	217,296	217,272.5
10-500-20	**304,387**	304,344	304,344	**304,387**	304,344	304,329.5
10-500-21	**302,379**	302,332	302,345	302,196	302,341	302,341.0

Table 9 (*continued*)

Prob.		GA	F&F	TEPSOq	QPSO*	
Inst.	*Opt./best − known*	*Best*	*Best*	*Best*	*Best*	*Avg.*
10-500-22	**302,417**	302,354	302,408	302,416	**302,417**	302,386.5
10-500-23	**300,784**	300,743	300,743	300,645	**300,784**	300,763.5
10-500-24	**304,374**	304,344	304,357	304,001	304,340	304,328.5
10-500-25	**301,836**	301,730	301,742	299,774	**301,836**	301,787.5
10-500-26	**304,952**	304,949	304,911	304,841	**304,952**	304,924.5
10-500-27	**296,478**	296,437	296,447	295,875	296,437	296,432.0
10-500-28	**301,359**	301,313	301,331	300,964	301,293	301,284.0
10-500-29	**307,089**	307,014	307,078	306,010	307,002	306,963.5
Avg.	**302,606**	302,556	302,571	302,111	302,575	302,554.1

Table 10
Performance comparison on benchmark instances with $m=30$ and $n=500$.

Prob.		GA	F&F	TEPSOq	QPSO*	
Inst.	*Opt./best − known*	*Best*	*Best*	*Best*	*Best*	*Avg.*
30-500-00	**116,056**	115,868	115,903	116,055	115,991	115,906.0
30-500-01	**114,810**	114,667	114,718	**114,810**	114,684	114,661.0
30-500-02	**116,741**	116,661	116,583	115,998	116,712	116,642.5
30-500-03	**115,354**	115,237	115,198	115,268	**115,354**	115,062.5
30-500-04	**116,525**	116,353	116,474	**116,525**	116,435	116,378.5
30-500-05	**115,741**	115,604	115,734	115,626	115,594	115,583.5
30-500-06	**114,181**	113,952	113,996	114,122	113,987	113,936.5
30-500-07	**114,348**	114,199	114,266	114,305	114,184	114,135.5
30-500-08	**115,419**	115,247	**115,419**	115,287	**115,419**	115,271.0
30-500-09	**117,116**	116,947	117,011	117,101	116,909	116,909.0
Avg.	**115,629**	115,474	115,530	115,510	115,527	115,448.6
30-500-10	**218,104**	217,995	218,068	218,073	218,068	218,068.0
30-500-11	**214,648**	214,534	214,626	214,645	214,626	214,546.5
30-500-12	**215,978**	215,854	215,836	215,918	215,839	215,839.0
30-500-13	**217,910**	217,836	217,862	217,836	217,816	217,816.0
30-500-14	**215,689**	215,596	215,592	213,625	215,544	215,544.0
30-500-15	**215,919**	215,762	215,784	215,086	215,753	215,753.0
30-500-16	**215,907**	215,772	215,824	214,999	215,789	215,784.5
30-500-17	**216,542**	216,336	216,418	216,425	216,387	216,387.0
30-500-18	**217,340**	217,290	217,225	216,368	217,217	217,211.0
30-500-19	**214,739**	214,624	214,663	214,168	**214,739**	214,686.5
Avg.	**216,278**	216,160	216,190	215,714	216,178	216,163.6
30-500-20	**301,675**	301,627	301,643	301,601	301,643	301,635.0
30-500-21	**300,055**	299,985	299,982	300,002	299,965	299,963.5
30-500-22	**305,087**	304,995	305,062	304,416	305,038	305,038.0
30-500-23	**302,032**	301,935	301,982	301,645	301,982	301,982.0
30-500-24	**304,462**	304,404	304,413	304,001	304,346	304,346.0
30-500-25	**297,012**	296,894	296,918	296,774	296,892	296,892.0
30-500-26	**303,364**	303,233	303,320	303,329	303,287	303,287.0
30-500-27	**307,007**	306,944	306,908	306,940	306,915	306,915.0
30-500-28	**303,199**	303,057	303,109	303,158	303,169	303,169.0
30-500-29	**300,572**	300,460	300,471	300,129	300,449	300,449.0
Avg.	**302,447**	302,353	302,381	302,200	302,369	302,367.7

effective for providing optimal and near-optimal solutions for the MKP in a reasonable computation time.

7. Conclusion

In this paper, we proposed a new hybrid heuristic that combines Quantum Particle Swarm Optimization (QPSO) with a local search method to solve the Multidimensional Knapsack Problem (MKP). We also incorporated an MKP-specific Drop/Add repair operator within the proposed approach to guarantee the feasibility of the generated solutions and to improve their quality (if possible).

The proposed hybrid approach is tested on a wide set of MKP benchmark problems from the literature and the results are compared with those produced by several population-based algorithms and state-of-the-art heuristic methods. The results demonstrated that our algorithm can produce solutions of good quality (optimal and near-optimal solutions) in a short and reasonable amount of computation time.

Table 11
A comparison between QPSO and the known state-of-the-art algorithms over the 90 largest OR-Library problems ($n=500$).

Prob.		QPSO*		F&F		GA		Fix+Cuts		LP+TS		Fix+LP+TS		NP+BAS+LP	
m	*α*	*v*	*t**	*v*	*t**	Best	*t**	*v*	*t**	Best	*t**	Best	*t**	Best	*t**
5	0.25	120,610	**0.03**	120,621	0.04	120,616	0.05	120,610	1.5	120,623	2.5	**120,628**	4.25	**120,628**	0.15
	0.5	219,506	0.04	219,509	**0.039**	219,503	0.05	219,504	1.5	219,507	2.5	**219,512**	4.25	**219,512**	0.15
	0.75	302,357	**0.03**	302,357	0.041	302,325	0.05	302,361	1.5	302,360	2.5	**302,363**	4.25	302,362	0.15
10	0.25	118,592	**0.03**	118,584	0.058	118,566	0.1	118,584	1.5	118,600	4.5	**118,629**	3.8	118,610	0.24
	0.5	217,296	**0.04**	217,287	0.055	217,275	0.1	217,297	1.5	217,298	4.5	**217,326**	3.8	217,319	0.26
	0.75	302,575	**0.03**	302,571	0.037	302,556	0.1	302,562	1.5	302,575	4.5	**302,603**	3.8	302,585	0.25
30	0.25	115,527	**0.04**	115,530	0.068	115,474	0.2	115,520	1.5	115,547	6	**115,624**	16.5	115,514	0.33
	0.5	216,178	**0.05**	216,190	0.064	216,157	0.2	216,180	1.5	216,211	6	**216,275**	16.5	216,183	0.33
	0.75	302,369	**0.04**	302,381	0.055	302,353	0.2	302,373	1.5	302,404	6	**302,447**	16.5	302,386	0.34

Table 12
Results for the 7 GK instances of Glover and Kochenberger (1996).

Problem			TS_HF	LP+TS	QPSO*	
GK	*n*	*m*	*Best*	*Best*	*Best*	*t(s)*
18	100	25	4524	**4528**	**4528**	17.31
19	100	25	3866	**3869**	**3869**	15.72
20	100	25	5177	**5180**	**5180**	18.47
21	100	25	3195	**3200**	**3200**	21.94
22	100	25	2521	**2523**	**2523**	16.87
23	200	15	9231	**9235**	**9235**	29.89
24	500	25	9062	**9070**	**9070**	92.45

Table 13
Results for the 11 most difficult instances of Glover and Kochenberger (1996).

Problem			TS_GK	F&F		LP+TS	QPSO*		CORAL	
MK_GK	*n*	*m*	*Best*	*Best*	*t(s)*	*Best*	*Best*	*t(s)*	*Best*	*t(s)*
01	100	15	**3766**	**3766**	21.091	**3766**	**3766**	12.162	**3766**	942.81
02	100	25	**3958**	**3958**	23.618	**3958**	**3958**	16.398	**3958**	18,000
03	150	25	5650	5650	35.912	**5656**	**5656**	22.123	5655	18,000
04	150	50	5764	5764	83.959	**5767**	**5767**	29.283	**5767**	18,000
05	200	25	7557	7557	51.823	**7560**	**7560**	34.704	7559	18,000
06	200	50	7672	7671	59.951	**7677**	**7677**	26.801	7672	18,000
07	500	25	19,215	19,217	143.562	**19,220**	**19,220**	95.122	19,214	18,000
08	500	50	18,801	18,802	167.303	**18,806**	**18,806**	113.677	*	*
09	1500	25	58,085	58,085	1649.078	**58,087**	**58,087**	748.218	*	*
10	1500	50	57,292	57,292	1167.772	**57,295**	57,292	779.414	*	*
11	2500	100	95,231	95,234	7995.115	**95,237**	95,234	1320.517	*	*

The aim of future works is to investigate the use of the proposed hybrid approach to solve other NP-hard and combinatorial optimization problems. Other relevant studies will also focus on fine-tuning the QPSO parameters by using some kind of adaptive strategy. Indeed, in this study no serious attempt was done to optimize the running parameters for QPSO whereas, that could improve the performance of the proposed approach.

Acknowledgment

The present research work has been supported by International Campus on Safety and Intermodality in Transportation (CISIT) and ELSAT-2020, the Nord-Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the Ministry of Higher Education and Research, and the National Center for Scientific Research (CNRS). The authors gratefully acknowledge the support of these institutions. We also would like to thank the referees for their valuable suggestions in improving this paper.

References

Abraham, A., Guo, H., Liu, H., 2006. Swarm intelligence: Foundations, Perspectives and Applications. Springer.
Al-Shihabi, S., ílafsson, S., 2010. A hybrid of nested partition, binary ant system, and linear programming for the multidimensional knapsack problem. Comput. Oper. Res. 37 (2), 247–255.
Balev, S., Yanev, N., Fréville, A., Andonov, R., 2008. A dynamic programming based reduction procedure for the multidimensional 0–1 knapsack problem. Eur. J. Oper. Res. 186 (1), 63–76.
Beasley, J.E., 1990. OR-library: distributing test problems by electronic mail. J. Oper. Res. Soc. 41 (11), 1069–1072.

Beheshti, Z., Shamsuddin, S.M., Yuhaniz, S.S., 2013. Binary accelerated particle swarm algorithm (BAPSA) for discrete optimization problems. J. Glob. Optim. 57 (2), 549–573.
Berberler, M.E., Guler, A., Nurıyev, U.G., 2013. A genetic algorithm to solve the multidimensional knapsack problem. Math. Comput. Appl. 18 (3), 486–494.
Bertsimas, D., Demir, R., 2002. An approximate dynamic programming approach to multidimensional knapsack problems. Manag. Sci. 48, 550–565.
Bonyadi, M.R., Li, X., 2012. A new discrete electromagnetism-based meta-heuristic for solving the multidimensional knapsack problem using genetic operators. Oper. Res. 12 (2), 229–252.
Boussier, S., Vasquez, M., Vimont, Y., Hanafi, S., Michelon, P., 2010. A multi-level search strategy for the 0–1 multidimensional knapsack problem. Discrete Appl. Math. 158, 97–109.
Chen, W.-N., Zhang, J., Chung, H.S.H., Zhong, W.-L., Wu, W.-G., Shi, Y.-H., 2010. A novel set-based particle swarm optimization method for discrete optimization problems. IEEE Trans. Evoluti. Comput. - TEC 14 (2), 278–300.
Chih, M., 2015. Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem. Appl. Soft Comput. 26, 378–389.
Chu, P., Beasley, J., 1998. A genetic algorithm for the multidimensional knapsack problem. J. Heuristics 4, 63–86.
Clerc, M., 2006. Particle Swarm Optimization. ISTE Publishing Company, London, UK.
Dammeyer, F., Voß, S., 1993. Dynamic tabu list management using the reverse elimination method. Ann. Oper. Res. 41 (2), 29–46.
Engelbrecht, A.P., 2005. Fundamentals of Computational Swarm Intelligence. Wiley, Hoboken, NJ.
Fayard, D., Plateau, G., 1982. An algorithm for the solution of the 0–1 knapsack problem. Computing 28, 269–287.
Fingler, H., Cáceres, E.N., Mongelli, H., Song, S.W., 2014. A cuda based solution to the multidimensional knapsack problem using the ant colony optimization. Procedia Comput. Sci. 29, 84–94.
Fréville, A., 2004. The multidimensional 0–1 knapsack problem: an overview. Eur. J. Oper. Res. 155 (1), 1–21.
Fréville, A., Gérard, P., 1994. An efficient preprocessing procedure for the multidimensional 0–1 knapsack problem. Discrete Appl. Math. 49, 189–212.
Gavish, B., Pirkul, H., 1985. Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. Math. Program. 31, 78–105.
Gavish, B., Pirkul, H., 1982. Allocation of databases and processors in a distributed data processing. In: In: J. Akola (ed.), Management of Distributed Data Processing, North-Holland, Amsterdam, pp. 215–231.
Gilmore, P.C., Gomory, R.E., 1966. The theory and computation of knapsack functions. Oper. Res. 14 (6), 1045–1074.
Glover, F., Kochenberger, G.A., 1996. Critical Event Tabu Search for Multidimensional Knapsack Problems. In: Meta-Heuristics, Springer, pp. 407–427.
Green, C., 1967. Two Algorithms for Solving the Independent Multidimensional Knapsack Problems, Management Sciences Research Report, Management Sciences Research Group, Carnegie Institute of Technology, GraduateSchool of Industrial Administration.
Hanafi, S., Fréville, A., 1998. An efficient tabu search approach for the 0–1 multidimensional knapsack problem. Eur. J. Oper. Res. 106 (2–3), 659–675.
Hanafi, S., Glover, F., 2007. Exploiting nested inequalities and surrogate constraints. Eur. J. Oper. Res 179 (1), 50–63.
Hey, T., 1999. Quantum computing: an introduction. Comput. Control Eng. J. 10 (3), 105–112.
Ke, L., Feng, Z., Ren, Z., Wei, X., 2010. An ant colony optimization approach for the multidimensional knapsack problem. J. Heuristics 16 (1), 65–83.
Kennedy, J., 1999. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, Vol. 3, pp. –1938 Vol. 3.
Kennedy, J., 2000. Stereotyping: improving particle swarm performance with cluster analysis. In: Proceedings of the 2000 Congress on Evolutionary Computation, Vol. 2, pp. 1507–1512.
Kennedy, J., 2003. Bare bones particle swarms. In: Proceedings of the 2003 Swarm Intelligence Symposium, IEEE, pp. 80–87.
Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Vol. 4, pp. 1942–1948.
Kennedy, J., Eberhart, R., 1997. A discrete binary version of the particle swarm algorithm. In: Proceedings of the 1997 IEEE International Conference on Computational Cybernetics and Simulation, Vol. 5, pp. 4104–4108.
Khanesar, M.A., Teshnehlab, M., Shoorehdeli, M.A., 2007. A novel binary particle swarm optimization. In: Control & Automation, 2007. MED'07. Mediterranean Conference on, IEEE, pp. 1–6.
Khemakhem, M., Haddar, B., Chebil, K., Hanafi, S., 2012. A filter-and-fan meta-heuristic for the 0–1 multidimensional knapsack problem. Int. J. Appl. Meta-heuristic Comput. (IJAMC) 3 (4), 43–63.
Kong, M., Tian, P., Kao, Y., 2008. A new ant colony optimization algorithm for the multidimensional knapsack problem. Comput. Oper. Res. 35 (8), 2672–2683.
Kong, M., Tian, P., 2006. Apply the particle swarm optimization to the multi-dimensional knapsack problem. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L., zurada, J. (Eds.), Artificial Intelligence and Soft Computing–ICAISC 2006, Vol. 4029 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 1140–1149.
Krohling, R.A., dos Santos Coelho, L., 2006. PSO-E: Particle swarm with exponential distribution. In: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, IEEE, pp. 1428–1433.
Ktari, R., Chabchoub, H., 2013. Essential particle swarm optimization queen with tabu search for mkp resolution. Computing 95 (9), 897–921.
Langeveld, J., Engelbrecht, A.P., 2012. Set-based particle swarm optimization applied to the multidimensional knapsack problem. Swarm Intell. 6 (4), 297–342.
Leung, S.C., Zhang, D., Zhou, C., Wu, T., 2012. A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem. Comput. Oper. Res. 39 (1), 64–73.
Mansini, R., Speranza, M.G., 2012. Coral: an exact algorithm for the multidimensional knapsack problem. INFORMS J. Comput. 24 (3), 399–415.
Martins, J.P., Longo, H., Delbem, A.C., 2014. On the effectiveness of genetic algorithms for the multidimensional knapsack problem. In: Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion, ACM, pp. 73–74.
Mohais, A., Mendes, R., Ward, C., Posthoff, C., 2005. Neighborhood restructuring in particle swarm optimization. In: Proceedings of the 2005 Advances in Artificial Intelligence, Vol. 3809, Lecture Notes in Computer Science, Springer, pp. 776–785.
Nezamabadi-pour, H., Rostami Shahrbabaki, M., Maghfoori-Farsangi, M., 2008. Binary particle swarm optimization: challenges and new solutions. CSI J. Comput. Sci. Eng. Persian 6 (1), 21–32.
Oliva, C., Michelon, P., Artigues, C., 2001. Constraint and linear programming: Using reduced costs for solving the zero/one multiple knapsack problem. In: In: International Conference on Constraint Programming, CP 01, Proceedings of the Workshop on Cooperative Solvers in Constraint Programming, CoSolv 01, pp. 87–98.
Osorio, M.A., Glover, F., Hammer, P., 2002. Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions. Ann. Oper. Res. 117 (1–4), 71–93.
Pampara, G., Franken, N., Engelbrecht, A.P., 2005. Combining particle swarm optimisation with angle modulation to solve binary problems. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on, Vol. 1, IEEE, pp. 89–96.
Parra-Hernandez, R., Dimopoulos, N., 2003. On the performance of the ant colony system for solving the multidimensional knapsack problem. In: Communications, Computers and signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference on, Vol. 1, pp. 338–341.
Parrott, D., Li, X., 2006. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans. Evolut. Comput. 10 (4), 440–458.
Petersen, C.C., 1967. Computational experience with variants of the balas algorithm applied to the selection of r&d projects. Manag. Sci. 13 (9), 736–750.
Pirkul, H., 1987. A heuristic solution procedure for the multiconstraint zero-one knapsack problem. Nav. Res. Logist. 34 (1–2), 161–172.
Puchinger, J., Raidl, G.R., Pferschy, U., 2010. The multidimensional knapsack problem: structure and algorithms. INFORMS J. Comput. 22 (2), 250–265.
Rezoug, A., Boughaci, D., Rezoug, A., 2015 Stochastic local search combined with simulated annealing for the 0–1 multidimensional knapsack problem. In: Symposium on Complex Systems and Intelligent Computing (CompSIC).
Shih, W., 1979. A branch and bound method for the multiconstraint zero one knapsack problem. J. Oper. Res. Soc. 30 (4), 369–378.
Sun, J., Feng, B., Xu, W., 2004. Particle swarm optimization with particles having quantum behavior. In: Proceedings of the 2004 Congress on Evolutionary Computation, Vol. 1, pp. 325–331.
Sun, J., Xu, W., Feng, B., 2004. A global search strategy of quantum-behaved particle swarm optimization. In: Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems, Vol. 1, pp. 111–116.
Tisna A., F.D., Abusini, S., Andar, A., 2013. Hybrid greedy–particle swarmoptimization–genetic algorithm and its convergence to solve multidimensionalknapsack problem 0-1. J. Theor. Appl. Inform. Technol. 58 (3), 522–528.
Varnamkhasti, M.J., 2012. Overview of the algorithms for solving the multi-dimensional knapsack problems. Adv. Stud. Biol. 4 (1), 37–47.
Vasquez, M., Hao, J.-K., 2001. Une approche hybride pour le sac à dos multi-dimensionnel en variables 0–1. Oper. Res. 35 (4), 415–438.
Vasquez, M., Vimont, Y., 2005. Improved results on the 0–1 multidimensional knapsack problem. Eur. J. Oper. Res. 165 (1), 70–81.
Vimont, Y., Boussier, S., Vasquez, M., 2008. Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem. J. Comb. Optim. 15 (2), 165–178.
Wan, N.F., Nolle, L., 2009. Solving a multi-dimensional knapsack problem using a hybrid particle swarm optimization algorithm. In: Proceedings of the 23rd European Conference on Modelling and Mimulation, ECMS 2009.
Weingartner, H.M., 1966. Capital budgeting of interrelated projects: survey and synthesis. Manag. Sci. 12 (7), 485–516.
Weingartner, H.M., Ness, D.N., 1967. Methods for the solution of the multi-dimensional 0/1 knapsack problem. Oper. Res. 15, 83–103.
Wilbaut, C., Hanafi, S., Fréville, A., Balev, S., 2006. Tabu search: global intensification using dynamic programming. Control Cybern. 35 (3), 579–598.
Yang, S., Wang, M., Jiao, L., 2004. A quantum particle swarm optimization. In: Proceedings of the 2004 Congress IEEE Conference on Evolutionary Computation, Vol. 1, pp. 320–324.