The Bi-Objective Quadratic Multiple Knapsack Problem: Model and Heuristics

Yuning Chen [a], Jin-Kao Hao [a,b,*]

[a] *LERIA, Université d'Angers, 2 Bd Lavoisier, 49045 Angers, Cedex 01, France*

[b] *Institut Universitaire de France, Paris, France*

Knowledge-Based Systems, doi:10.1016/j.knosys.2016.01.014

Abstract

The single objective quadratic multiple knapsack problem (QMKP) is a useful model to formulate a number of practical problems. However, it is not suitable for situations where more than one objective needs to be considered. In this paper, we extend the single objective QMKP to the bi-objective case such that we simultaneously maximize the total profit of the items packed into the knapsacks and the 'makespan' (the gain of the least profit knapsack). Given the imposing computational challenge, we propose a hybrid two-stage (HTS) algorithm to approximate the Pareto front of the bi-objective QMKP. HTS combines two different and complementary search methods - scalarizing memetic search (first stage) and Pareto local search (second stage). Experimental assessments on a set of 60 problem instances show that HTS dominates a standard multi-objective evolutionary algorithm (NSGA II), and two simplified variants of HTS. We also present a comparison with two state-of-the-art algorithms for the single objective QMKP to assess the quality of the extreme solutions of the approximated Pareto front.

Keywords: Quadratic multiple knapsack; bi-objective optimization; constrained optimization; memetic and hybrid search; local search and Pareto local search.

1 Introduction

Given a set of weight capacity-constrained knapsacks and a set of objects (or items), each object is associated with a weight, an individual profit and a paired profit with any other object. The quadratic multiple knapsack problem

* Corresponding author.

Email addresses: yuning@info.univ-angers.fr (Yuning Chen), hao@info.univ-angers.fr (Jin-Kao Hao).

(QMKP) aims to determine a maximum profit assignment (packing plan) of objects to the knapsacks subject to their capacity constraints [14]. The profit of a pair is accumulated in the sum only if the two corresponding objects are allocated to the same knapsack.

The QMKP generalizes two well-known knapsack problems, i.e., the quadratic knapsack problem (QKP) [28] and the multiple knapsack problem (MKP) [26]. The QMKP is also related to the so-called discounted 0-1 knapsack problem (DKP) [30]. The DKP is to select items from a set of groups where each group includes three items and at most one of the three items can be selected. For each group, the profit of the third item is a discounted profit which is defined by the interaction of the first two items. This problem remains linear and finds applications in investment. The QMKP has important applications where resources with different levels of interaction have to be distributed among different tasks [14, 31]. A first example involves allocating staff in a company to a set of groups where group member contributions are calculated both individually and in pairs; another example concerns an investment scenario where the knapsacks represent the bounded budgets and the paired values indicate the impact on expected financial performances when different investment options are chosen together.

The QMKP is computational challenging since it generalizes the NP-hard QKP [28]. To solve such problems, exact and heuristic algorithms constitute two main and complementary solution methods in the literature. Exact algorithms have the theoretical advantage of guaranteeing the optimality of the solutions found. However given the intrinsic difficulty of NP-hard problems, the computing time needed to find the optimal solution by an exact algorithm may become prohibitive for large instances. For instance, for the QKP, the most powerful exact algorithm can only deal with instances with no more than 1500 items [29] and typically requires considerable computing time. For the more complicated QMKP, no exact algorithm has been published in the literature to the best of our knowledge. On the other hand, heuristic algorithms aim to find satisfactory sub-optimal solutions (to large problem instances) in acceptable computing time, but without provable quality guarantee of the attained solutions. This approach is particularly useful when it is difficult or impossible to obtain an optimal solution. Within the context of approximating the difficult QMKP, several heuristic algorithms have been reported in the literature. Representative heuristic algorithms include population-based algorithms, such as genetic algorithms [14, 31], memetic algorithm [32], artificial bee colony algorithm [33] and evolutionary path relinking algorithm [8]. Besides, neighborhood search and constructive/destructive search approaches represent another class of effective tools for the QMKP; typical examples include hill-climbing [14], tabu-enhanced iterated greedy search [13], strategic oscillation [12] and iterated responsive threshold search (IRTS) [7]. Finally, note that exact and heuristic approaches complement each other and are use-

ful for problem solving in different settings. They can even be combined to create powerful hybrid algorithms.

The QMKP is a useful model to formulate a number of practical problems [14, 31]. Still it is not suitable for situations where more than one objective needs to be simultaneously considered. Consider the following staff assignment problem for instance. When company managers allocate staff to form a set of groups responsible for different products, they may not only consider the total strength of all groups, but also the balance among the groups for the sake of fairness and sustainable development. For instance, when several groups work on different products destined to different types of customers, in order to pursue a long-term profit, company managers may want to balance the strength of each group when allocating the group members, such that each group has enough capability to ensure a high-quality of its products for the purpose of well satisfying its customers in the long term. In portfolio investments, an investor may be interested not only by maximizing the total return of the invested asset mix, but also by ensuring an expected return of the least profit asset. In these settings, one faces a bi-objective version of the QMKP, in which both the total profit of the packing plan and the gain of the least profit knapsack (corresponding to the makespan in scheduling theory) are to be maximized simultaneously (see Section 2 for the formal definition). To conveniently formulate this type of problems, this work extends the single objective QMKP to the bi-objective QMKP (BO-QMKP). Apart from the aforementioned application scenarios, the BO-QMKP model could find in the future additional applications in other settings where it can be used to formulate either a whole problem or a subproblem. One notices that some well-known knapsack problems and more generally other optimization problems have already a bi-objective or multi-objective counterpart, like the bi-objective 0-1 knapsack problem [10], the multi-objective multidimensional knapsack problem [4], the bi-objective unconstrained binary quadratic programming problem [21], the bi-objective flow-shop scheduling problems [17], the bi-objective traveling salesman problem [20], the multi-objective set covering problem [15] and the bi-objective capacity planning problem [37]. The BO-QMKP introduced in this work enriches these multi-objective modeling tools and enlarges the class of practical problems that can be formulated.

On the other hand, solving the BO-QMKP model represents an imposing computational challenge in the general case since it generalizes the computationally difficult QMKP model. For this reason, we focuses on elaborating a heuristic algorithm to approximate the Pareto front of the BO-QMKP. The proposed hybrid two-stage (HTS) algorithm is based on the general two-stage approach combining two fundamentally different and complementary search strategies, namely the scalarizing approach (first stage) and the Pareto-based approach (second stage). Such a hybrid framework has been successfully applied to solve a number of challenging multi-objective problems such as the

bi-objective flow-shop scheduling problems [17], the multi-objective traveling salesman problem [20] and the bi-objective unconstrained binary quadratic programming problem [22]. In this work, we adapt this general two-stage approach to solve our BO-QMKP model and develop dedicated search procedures for each stage of the proposed algorithm. In particular, we devise a population-based scalarizing memetic search method to effectively solve the scalarizing subproblems in the first stage and a double-neighborhood Pareto local search procedure to further the approximation set in the second stage. By combining complementary search strategies in the two search stages and using dedicated techniques in each stage, the HTS algorithm aims to push the approximation set towards the Pareto front on the one hand and ensure a well-distributed approximation set on the other hand. Thanks to these desirable features, the proposed HTS algorithm proves to be able to attain high quality approximations as shown in Section 4.

The main contributions of this work can be summarized as follows.

- We introduce for the first time the bi-objective quadratic multiple knapsack model whose formal definition is provided in Section 2.
- We provide a detailed description of our hybrid two-stage approach which aims to provide high quality approximation of the Pareto front for the proposed BO-QMKP model (Section 3). We show how HTS makes an original combination of an elitist evolutionary multi-objective optimization algorithm with a state-of-the-art single-objective responsive threshold search procedure for its first stage while adopting an effective Pareto-based local search procedure for the second stage.
- We show experimental studies on a set of 60 benchmark instances to assess the effectiveness of the proposed HTS algorithm (Section 4). In particular, our experiments demonstrate that HTS dominates a conventional non-dominated sorting genetic algorithm (NSGA II) and two simplified variants of the HTS algorithm.

2 The bi-objective quadratic multiple knapsack problem

In this section, we introduce formally the BO-QMKP model. Given a set of objects (or items) $N = \{1, 2, ..., n\}$ and a set of capacity-constrained knapsacks $M = \{1, 2, ..., m\}$. Each object i ($i \in N$) is associated with a profit p_i and a weight w_i. Each pair of objects i and j ($1 \leq i \neq j \leq n$) is associated with a joint profit p_{ij}. Each knapsack k ($k \in M$) has a weight capacity C_k. The BO-QMKP aims to assign the n objects to the m knapsacks (some objects can remain unassigned) such that both the overall profit of the assigned objects and the makespan (the gain of the least profit knapsack) are maximized subject to the following two constraints:

- Each object i ($i \in N$) can be allocated to at most one knapsack;
- The total weight of the objects assigned to each knapsack k ($k \in M$) cannot exceed its capacity C_k.

Given the above notations, a BO-QMKP solution can be represented as a set of groups $S = \{I_0, I_1, ..., I_m\}$ where group $I_k \subset N$ ($k \in M$) represents the set of objects assigned to knapsack k and group I_0 contains all unassigned objects. Then the BO-QMKP can be stated mathematically as follows:

$$max \quad f_1(S) = \sum_{k \in M} \sum_{i \in I_k} p_i + \sum_{k \in M} \sum_{i,j \in I_k, i \neq j} p_{ij} \tag{1}$$

$$max \quad f_2(S) = min_{k \in M} \{ \sum_{i \in I_k} p_i + \sum_{i,j \in I_k, i \neq j} p_{ij} \} \tag{2}$$

subject to:

$$\sum_{i \in I_k} w_i \leq C_k, \forall k \in M \tag{3}$$

$$S \in \{0, ..., m\}^n \tag{4}$$

Equation (1) aims to maximize the total profit of all assigned objects while Equation (2) aims to maximize the gain of the least profit knapsack (or makespan). Constraints (3) guarantees that the total weight of the objects assigned to each knapsack does not exceed its capacity. Constraint (4) requires that each object is allocated to at most one knapsack. Notice that the BO-QMKP could also be conveniently formulated as a bi-objective 0-1 quadratic program (omitted here since this model is not used in this work).

3 A hybrid two-stage algorithm for the BO-QMKP

Given the computational challenge of the BO-QMKP model in the general case, we introduce the hybrid two-stage search algorithm which aims to effectively calculate a high-quality approximation set of the Pareto front for a given BO-QMKP instance.

3.1 General principle

The proposed two-stage search algorithm for the BO-QMKP is outlined in Algorithm 1. The first stage uses a population-based scalarizing memetic search method while the second stage employs a Pareto local search method. We describe below the general principle of the two stages of our HTS approach. In Sections 3.2 and 3.3, we present in detail the components and particular features of the first stage and second stage respectively.

Algorithm 1: Pseudo-code of the HTS algorithm for the BO-QMKP.

Data: P: an instance of the BO-QMKP; ps: subpopulation size;
Result: a Pareto set approximation;

STAGE 1: Scalarizing memetic search ;
Initialize the archive A ; /* Archive initialization, Sect. 3.2.1 */
$noNS \leftarrow |A|$; /* Record the number of non-dominated solutions */
$imp \leftarrow true$;
$cnt \leftarrow 1$;
while $(imp = true) \vee (noNS > 1)$ **do**
  $imp \leftarrow false$;
  $step \leftarrow 1/noNS$;
  for $i : 0 \rightarrow noNS - 1$ **do**
    if *cnt is odd* **then**
      $start \leftarrow i * step$; $end \leftarrow (i + 1) * step$;
    else
      $start \leftarrow (noNS - i + 1) * step$; $end \leftarrow (noNS - i) * step$;
    $\lambda 1 \leftarrow rand(start, end)$; $\lambda 2 \leftarrow 1 - \lambda 1$; $\lambda \leftarrow (\lambda 1, \lambda 2)$;
    $SP \leftarrow construct\_subpopulation(\lambda, ps)$;
    $(S_1, S_2) \leftarrow$ randomly select two solutions from SP ;
    $S_0 \leftarrow crossover(S_1, S_2)$; /* Solution recombination, Sect. 3.2.3 */
    $S_0 \leftarrow$RTS(S_0, λ) ; /* Improve S_0, Sect. 3.2.4 */
    $A^{'} \leftarrow nondominate\_filter(A \cup \{S_0\})$;
    if $A^{'} \neq A$ **then**
      $imp \leftarrow true$; $A \leftarrow A^{'}$;
      break ;
  $cnt \leftarrow cnt + 1$;
STAGE 2: Pareto local search ;
$A \leftarrow$DNPLS(A) ; /* Sect. 3.3 */
return A

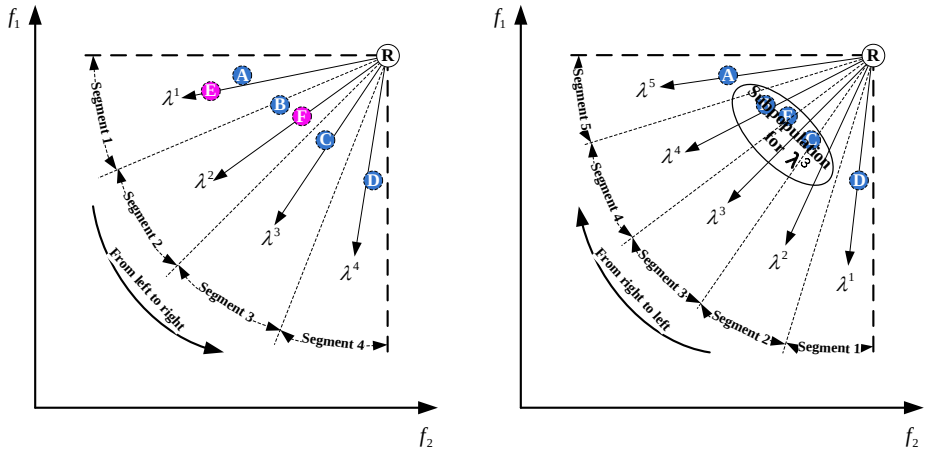
3.1.1 First stage – scalarizing memetic search

The scalarizing memetic search (SMS) method in the first stage is a critical component of our HTS algorithm. SMS follows the population-based memetic framework [16, 25] which combines evolutionary computing and local optimization. To ease the presentation of the principles of SMS, we provide an illustration of the search process and some important notions in Figure 1.

To approximate the Pareto front of the BO-QMKP, SMS generates, by means of different weight vectors, a number of scalarizing subproblems with a single objective. Specifically, given an objective vector $F(S) = \{f_1(S), f_2(S)\}$ and a weight vector $\lambda = \{\lambda 1, \lambda 2\}$, the scalarized objective function of a subproblem is stated as:

$$h(S) = \lambda 1 * f_1(S) + \lambda 2 * f_2(S) \tag{5}$$

Then, solving a scalarizing subproblem amounts to solving a QMKP, whose single objective function is exactly $h(S)$. For instance, in Figure 1(a), the four weight vectors $\lambda^1 - \lambda^4$ lead to four different scalarizing subproblems.



(a) Segment exploration in an odd step

(b) Segment exploration in an even step

Fig. 1. Illustration of the schalarizing memetic search

SMS uses an (unbounded) archive (or population) which is initialized with a set of high-quality non-dominated solutions produced by the dichotomic scalarizing local search method (DSLS, Section 3.2.1). In the main body of SMS, the archive is evolved according to the following three steps.

First, the whole objective space of the BO-QMKP is equally divided into $noNS$ segments, where $noNS$ denotes the current number of non-dominated solutions in the archive. In Figure 1(a), $noNS = 4$ (points A-D) results in four segments while in Figure 1(b) $noNS = 5$ (points A-D and F) results in five segments. Then, SMS launches a round of segment exploration to explore these $noNS$ segments one by one from a direction determined at the beginning of the current round of segment exploration. A subproblem is solved for each segment. The direction of segment exploration changes at each different round. In odd steps where SMS explores segments from left to right (see Figure 1(a) for example), the importance in the scalarized objective function decreases on f_1 and increases on f_2; similarly, in even steps where SMS explores segments from right to left (see Figure 1(b) for example), the importance in the scalarized objective function decreases on f_2 and increases on f_1.

Second, one new solution is generated for each explored segment. Given a segment, SMS first randomly generates a weight vector (all weight vectors shown in Figure 1 are randomly selected from the segment objective space), with which SMS can assign a fitness to each individual in the current archive, and the first ps individuals are copied to the subpopulation SP. For instance, in Figure 1(b), a subpopulation with $ps = 3$ containing three solutions (B, C and F) that are closest to the weigh vectors λ^3 is constructed for the subproblem with respect to λ^3 (Section 3.2.2). From SP, two parent solutions are randomly selected. Then a knapsack-based crossover operator (Section 3.2.3)

is applied to the parents to generate an offspring solution which is further improved by the responsive threshold search (RTS) procedure (Section 3.2.4).

Finally, the improved offspring solution is inserted into the archive if it is not equivalent to, or dominated by any solution from the archive. If the solution is inserted into the archive, SMS breaks the current round of segment exploration and starts a new round from the opposite direction; otherwise SMS explores the next segment by following the same direction. In Figure 1(a), SMS generates an offspring solution E in segment 1 which is dominated by solution A, so it continues to explore the next segment. In segment 2, SMS successfully generates a solution F that is not dominated by any existing solution, so SMS inserts F into the archive, breaks the current round of left-to-right segment exploration, and starts a new right-to-left exploration as shown in Figure 1(b).

The above process repeats until no insertion to the archive has been made for a whole round of segment exploration, or the number of non-dominated solutions is less than two (which rarely happens).

In summary, the first stage with SMS has the following features.

- First, SMS uses a granularity-increasing (GI) strategy to dynamically change the segments to be explored. GI progressively narrows down the segments and increases their granularity. Based on these continuously shrinking segments, diverse and well-distributed weight vectors are defined, which helps SMS to effectively approximate the whole Pareto front. This property makes SMS different from the general-purpose scalarizing methods whose weight vectors are fixed by considering the preference information of decision-makers.
- Second, SMS uses a direction-altering (DA) strategy to switch its search direction between odd and even steps to avoid a bias of the search process towards any of the two objectives, which promotes diversity and uniform distribution of the non-dominated solutions discovered by the search process.
- Last, SMS solves each subproblem with the use of a subpopulation containing the most relevant solutions with respect to its associated weight vector, which accelerates the convergence of the approximation set.

One notices that the GI and DA strategies are very simple and easy to implement. They can be conveniently integrated into any scalarizing method in the context of bi-objective optimization. From their abilities to incorporate and guide other procedures, these two strategies can be considered as meta-strategies for bi-objective optimization problem solving. Also, the idea of constructing subpopulations can be adopted by any population-based scalarizing method to make use of the relevant nearby information and to speed up the convergence of the search process.

To further appreciate the strategies and techniques of the population based SMS algorithm, we provide a detailed presentation of its components in Section 3.2.

3.1.2 Second stage – double-neighborhood Pareto local search

From the non-dominated solution set obtained by SMS in the first stage, HTS applies a double-neighborhood Pareto local search (DNPLS) in its second stage to further improve the Pareto front approximation. In contrast to scalarizing approaches, the selection process in DNPLS is directly guided by the Pareto dominance relation. This stage not only helps to advance the approximation set towards the Pareto front, but also enhances the distribution form and extent of the eventual approximation set obtained by the HTS algorithm. DNPLS is described in details in Section 3.3.

As shown in Algorithm 1, after the above two search stages, the HTS algorithm returns the solutions contained in the archive (set A) as an approximation of the Pareto front. The main components of the HTS algorithm are detailed in the following sections.

3.2 Components of the scalarizing memetic search of the first stage

3.2.1 Archive initialization

In the first stage of HTS, the population based scalarizing memetic search algorithm first creates a set of solutions to form its initial population using the dichotomic scalarizing local search (DSLS) method. Our DSLS method is inspired by the dichotomic search scheme for exact bi-objective optimization [1], and is resulted from incorporating this scheme with local search (the responsive threshold search [7] in our case). Although our DSLS algorithm shares similarities with the dichotomic search methods investigated in [18, 22], it has some particular features as we show below.

The outline of our DSLS algorithm is sketched in Algorithm 2. DSLS starts by identifying a high-quality solution for each individual objective function (Lines 3-6 of Algorithm 2). We call these solutions as *extreme solutions*. Preliminary experiments show that these extreme solutions have an important impact on the performance of the overall algorithm. We thus make a particular computational effort to guarantee the quality of the extreme solutions. For each of the two objectives of the BO-QMKP, we run the iterated version of the RTS (IRTS) algorithm [7] (10 restarts using perturbed starting solutions) in order to get a high-quality solution. The seeding solutions for the IRTS algorithm are generated by a greedy construction method (GCM),

Algorithm 2: Pseudo-code of the DSLS algorithm for the BO-QMKP.

Data: P: an instance of the BO-QMKP;
Result: a set of non-dominated solutions A;

$\lambda^1 = (1, 0)$, $\lambda^2 = (0, 1)$;
$U_I \leftarrow \emptyset$, $U_E \leftarrow \emptyset$;
for $i : 1$ *to* 2 **do**
    $S \leftarrow Greedy\_Construct(\lambda^i)$;
    $S^{'} \leftarrow IRTS(S, \lambda^i)$;
    $U_I \leftarrow U_I \cup \{S^{'}\}$;
while $|U_I| \leq 2$ **do**
    Sort the solutions in U_I in decreasing order of their f_1 value ;
    $(S^1, S^2) \leftarrow$ select the first two solutions from U_I ;
    for $i : 1$ *to* 2 **do**
        $norm^i = |f_i(S^1) - f_i(S^2)|/f_i^{max}$;
    $\lambda \leftarrow (norm^2/(norm^1 + norm^2), norm^1/(norm^1 + norm^2))$;
    $imp \leftarrow false$;
    for $i : 1$ *to* 2 **do**
        $S^0 \leftarrow$RTS(S^i, λ) ; `/* Improve` S^i`, Sect. 3.2.4 */`
        if $(f_1(S^1) > f_1(S^0)) \wedge (f_1(S^0) > f_1(S^2)) \wedge (f_2(S^2) > f_2(S^0)) \wedge (f_2(S^0) > f_2(S^1))$ **then**
            $U_I \leftarrow U_I \cup \{S^0\}$;
            $imp \leftarrow true$;
            break ;
    if $imp = false$ **then**
        $U_E \leftarrow U_E \backslash \{S^1\}$;
$A \leftarrow$ non-dominated solutions from $U_I \cup U_E$;
return A

where at each iteration of the construction process, GCM inserts an unassigned object to a knapsack that achieves the best improvement of the current scalarizing objective function while maintaining the feasibility. The two resulting extreme solutions (one per objective) are included in a set U_I. At each iteration of DSLS, the solutions in U_I are sorted in decreasing order of the first objective value (i.e., f_1 value), and then the first two solutions of U_I are selected to form a pair, say (S^1, S^2). A new weight vector perpendicular to the segment defined by S^1 and S^2 in the objective space is defined as: $\lambda = (norm^2/(norm^1 + norm^2), norm^1/(norm^1 + norm^2))$, where $norm^i = |f_i(S^1) - f_i(S^2)|/f_i^{max}$ $(i = 1, 2)$. f_1^{max} and f_2^{max} can be set to a tight upper bound of the two objectives of the BO-QMKP, but for the sake of simplicity, they are set here as:

$$f_1^{max} = \sum_{i=1}^{n} \sum_{j=i}^{n} p_{ij} \tag{6}$$

$$f_2^{max} = f_1^{max}/m \tag{7}$$

DSLS then runs RTS on the scalarizing subproblem with respect to λ and

the resulting solution S^0 is inserted into U_I if $f_1(S^1) > f_1(S^0) > f_1(S^2)$ and $f_2(S^2) > f_2(S^0) > f_2(S^1)$. S^1 and S^2 are successively used as initial solutions for two independent runs of RTS. Once a S^0 is successfully inserted into U_I, the current iteration finishes; otherwise, S^1 is removed from U_I and put into an external set U_E before starting the next iteration. The above process (Lines 7-21 of Algorithm 2) is iterated until U_I contains less than two solutions, and the non-dominated solutions from $U_I \cup U_E$ are put into A to form the initial archive.

3.2.2 Subpopulation construction

At each iteration of SMS, the algorithm deals with a subproblem that is associated with a weight vector $\lambda = (\lambda 1, \lambda 2)$. A subpopulation of individuals, from which two parent solutions that participate in the subsequent crossover procedure are selected, is temperately constructed. For $\lambda = (\lambda 1, \lambda 2)$, $\lambda 1$ is a real value randomly selected from the interval $[start, end]$ (as to how to obtain *start* and *end*, see Lines 10-13 of Algo. 1), and $\lambda 2 = 1 - \lambda 1$. With λ, each individual in the archive A is assigned a fitness value. The individuals in A are then sorted in decreasing order of their fitness values, and the subpopulation is composed of the first *ps* individuals. One notices that the solutions included in the subpopulation are those obtained by solving the subproblems of previous iterations, each subproblem being associated with a weight vector different from λ. The solutions in the current subpopulation can be considered as the representatives of the *ps* previous subproblems whose associated weight vectors are the *ps* closest ones to λ. According to [38], the optimal solution of the subproblem associated to λ^i should be close to that of the subproblem associated to λ^j if λ^i is close to λ^j. Therefore, we argue that the information of the subproblems whose weight vectors are close to that of the current subproblem should be helpful for solving the current subproblem.

3.2.3 Knapsack-based crossover operator

After subpopulation construction, two different individuals are randomly selected from the subpopulation, which are to be recombined using the knapsack-based crossover (KBX) operator to produce a single offspring solution. KBX works as follows. Given two parent solutions $S^1 = \{I_0^1, I_1^1, ..., I_m^1\}$ and $S^2 = \{I_0^2, I_1^2, ..., I_m^2\}$ (the fitness of S^1 is always better than that of S^2 with respect to the current subproblem), KBX randomly selects two knapsacks of objects I_i^1 ($i \neq 0$) and I_j^2 ($j \neq 0$) from them, respectively. Then, the whole solution of S^1 is copied to the offspring solution $S^0 = \{I_0^0, I_1^0, ..., I_m^0\}$, with only one knapsack of objects I_i^0 (corresponding to I_i^1 of S^1) replaced by I_j^2. Such an offspring solution may be infeasible and two cases may happen: 1) some objects appear twice, 2) some objects are missing in the new solution. For the first

case, KBX removes the duplicated objects from the old knapsacks (other than the newly replaced knapsack I_i^0) of S^0. For the second case, KBX first sorts the missing objects in random order and then for each of them, KBX tries to assign it to a knapsack I_k^0 ($k > 0$) that can accommodate it without violating the capacity constraint. If there are multiple such knapsacks, one knapsack is randomly chosen; if no such knapsack exists, KBX assigns the object to I_0^0.

After each crossover operation, the new offspring solution is improved by a local optimization procedure which is explained in the next section.

3.2.4 Responsive threshold search component

To optimize a single objective subproblem defined with a scalarized objective function, any approach to the conventional QMKP can be applied in general. In our case, we apply the responsive threshold search (RTS) algorithm [7] which is one of the best performing approaches for the QMKP.

RTS basically alternates between a threshold-based exploration phase (Exploration for short) and a descent-based improvement phase (Improvement for short). Exploration is based on three move operators (i.e., $REALLOCATE$, $EXCHANGE$ and $DROP$) and Improvement is based on two operators (i.e., $REALLOCATE$, $EXCHANGE$). In Exploration phase, RTS accepts any encountered neighboring solution that satisfies a responsive quality threshold, while in Improvement phase, RTS continues accepting the first met improving neighboring solutions until no improvement can be found. If the local optimum obtained in the Improvement phase has a better objective value than the recorded best local optimum value, RTS updates this recorded value and restarts a new round of Exploration-Improvement phases. RTS terminates when the recorded best local optimum value has not been improved for a predefined consecutive times of Exploration-Improvement phases. More details about RTS can be found in [7].

The current single objective subproblem (QMKP) with respect to a weight vector $\lambda = (\lambda 1, \lambda 2)$, that is to be solved by RTS, takes the following form:

$$max \quad h(S) = \lambda 1 * g_1(S) + \lambda 2 * g_2(S) \tag{8}$$

subject to:

$$\sum_{i \in I_k} w_i \leq C_k, \forall k \in M \tag{9}$$

$$S \in \{0, ..., m\}^n \tag{10}$$

where $g_1(S)$ and $g_2(S)$ are the normalized objective functions which are defined as follows:

$$g_1(S) = f_1(S)/f_1^{max*} \tag{11}$$

$$g_2(S) = f_2(S)/f_2^{max*} \tag{12}$$

In the above equations, f_1^{max*} and f_2^{max*} are the normalized max objective values for f_1 and f_2 which are calculated as follows: let the two extreme solutions of the archive initialization phase (see Section 3.2.1) be S_1^* and S_2^* where $f_1(S_1^*) > f_1(S_2^*)$ and $f_2(S_2^*) > f_2(S_1^*)$, then

$$f_1^{max*} = f_1^{max} \tag{13}$$

$$f_2^{max*} = (f_2(S_2^*) - f_2(S_1^*)) * f_1^{max}/(f_1(S_1^*) - f_1(S_2^*)) \tag{14}$$

The introduction of the normalized objective functions $g_1(S)$ and $g_2(S)$ in Equation (8) is to avoid a possible bias of the scalarized single objective function with respect to any of the two objective functions of the BO-QMKP when the two objective values are of different scales.

3.3 The double-neighborhood Pareto local search of the second stage

At the end of the first stage, HTS obtains a set of high-quality and non-dominated solutions produced by the SMS algorithm. These solutions form an approximation of the Pareto front. The second stage of HTS aims to further improve the quality of the approximation by a Pareto local search method. Pareto local search extends iterative improvement algorithms for single-objective optimization to the multi-objective case [27]. In contrast to the scalarizing approach like SMS, which uses a single-objective acceptance criterion, Pareto local search accepts solutions based on a Pareto dominance relation. Our Pareto local search algorithm (see Algorithm 3) in the second stage of HTS relies on two dedicated neighborhoods N_R and N_E which are induced by two well-known move operators: REALLOCATE and EXCHANGE (see [7] for the definition of these two operators), and the resulting algorithm is called a double-neighborhood Pareto local search (DNPLS) algorithm.

The input of DNPLS is an initial archive (set A) of non-dominated solutions provided by the SMS algorithm. DNPLS uses another set A_F to contain currently unvisited solutions from A. DNPLS starts by marking all solutions in A as unvisited (Lines 1-2 of Algo. 3) and puts all of them to A_F (Line 3 of Algo. 3). At each iteration, DNPLS first randomly selects an unvisited solution S_0 from A_F. Then DNPLS explores the two neighborhoods N_R and N_E of S_0 in a serial way (Line 7-11 of Algo. 3). Any feasible neighboring solution which is not dominated by S is marked unvisited (Line 9 of Algo. 3) and added to the archive A followed by a non-dominated solutions filtering (Line 10 of Algo. 3). S_0 is marked visited after the neighborhood exploration and the unvisited solution set A_F is updated accordingly. DNPLS repeats the above process until A_F becomes empty (i.e., all solutions in the archive A have been visited). At the end of the second stage, the solutions of the archive A form the final result of the whole HTS algorithm.

Algorithm 3: Pseudo-code of DNPLS algorithm for the BO-QMKP.

Data: P: an instance of the BO-QMKP; A: an initial set of non-dominated solutions;
Result: final set of non-dominated solutions A;

```
for each S ∈ A do
    visited(S) ← false ;
A_F ← A ;
while A_F ≠ ∅ do
    S_0 ← randomly extract a solution from A_F ;
    A_F ← A_F \ {S_0} ;
    for each N ∈ {N_R, N_E} do
        for each S ∈ N(S_0) do
            if S is not dominated by S_0 then
                visited(S) ← false ;
                A ← nondominate_filter(A ∪ {S}) ;
    visited(S_0) ← true ;
    A_F ← {S ∈ A | visited(S) = false} ;
return A
```

4 Experimental study

In this section, we present experimental studies of the proposed approach on a set of well-known QMKP instances. After a brief introduction of experimental setup, we divide the experiments into five parts. The first part is dedicated to parameter calibration. The second part is dedicated to computational results of the HTS algorithm and a comparison against two state-of-the-art single-objective algorithms. The purpose of the third part is to compare the performance of HTS against a popular baseline multi-objective algorithm (NSGA-II [9]). The fourth part is to compare the performance of HTS with its two simplified versions of HTS, allowing to appreciate the impact of each component of HTS. Finally, part five shows a graphical study which allows us to visualize the behaviour of HTS and its reference algorithms.

4.1 Experimental setup

Test instances. The experiments were carried out on a set of 60 well-known QMKP benchmark instances which were commonly used in almost all existing QMKP literature. These instances were modified from the quadratic single knapsack instances that were first introduced in [2] and can be downloaded at `http://cedric.cnam.fr/~soutile/QKP/QKP.html`. The QMKP benchmarks are characterized by three factors: 1) the number of objects $n \in \{100, 200\}$; 2) density $d \in \{0.25, 0.75\}$, which means the number of non-zero profit coefficients of the objective function divided by the number of all coefficients; 3) the number of knapsacks $m \in \{3, 5, 10\}$. For each instance, the capacity of

each knapsack is set to 80% of the sum of all object weights divided by the number of knapsacks.

Computational environment. Our HTS algorithm was coded in C++ [1] and compiled by GNU gcc 4.1.2 with the '-O3' flag. HTS as well as the other competing algorithms in the experiments were independently run 30 times on a computer with an AMD Opteron 4184 processor (2.8GHz and 2GB RAM) running Ubuntu 12.04. When solving the DIMACS machine benchmarks [2] without compilation optimization flag, our machine requires 0.40, 2.50 and 9.55 seconds respectively for graphs r300.5, r400.5 and r500.5.

Performance measures. To evaluate the quality of the approximation set obtained by our HTS algorithm and other competing algorithms, we use the well-known epsilon and hypervolume indicators [39].

- Epsilon indicator (I_ϵ): This indicator gives the minimum multiplicative factor by which an approximation set has to be shifted in the objective space in order to slightly dominate the reference set.
- Hypervolume difference indicator (I_H^-): Given an approximation set A, let the hypervolume value of A be $I_H(A)$. The Hypervolume difference indicator of set A is calculated as: $I_H^-(A) = I_H(R) - I_H(A)$, where R is a reference set which is the best-identified approximation complied from the approximation sets of all tested configurations. Intuitively, this indicator measures the portion of the objective space that is dominated by R but not by A. The hypervolume indicator is one of the most commonly used measures for evaluating the multi-objective optimization algorithms, since it is the only unary measure which is consistent with the Pareto dominance relation [5], i.e., it allows to obtain a total order between approximation sets.

In our experiments, these two indicators were computed based on the normalized objective vectors of the non-dominated solutions. For any objective vector, its two objective values are normalized within the interval [1,2] with respect to the $f^{min**} = \{f_1^{min**}, f_2^{min**}\}$ and $f^{max**} = \{f_1^{max**}, f_2^{max**}\}$, where f_i^{min**} (resp. f_i^{max**}, $i = 1, 2$) is the minimal (resp. maximal) value of f_i among all the results obtained over the 30 runs of all compared algorithms. The point (0.9,0.9) was used as the reference point in our experiment. Our computational results reported in the following sections were obtained by the performance assessment software package PISA [19], which implements exactly the above computational procedure. For both indicators, a lower value is always better.

[1] The best non-dominated solution values are available at http://www.info.univ-angers.fr/pub/hao/bi-qmkp.html

[2] dfmax: ftp://dimacs.rutgers.edu/pub/dsj/clique/

4.2 Parameter calibration

The proposed HTS relies on four parameters, among which three parameters from the RTS component were well-tuned in our previous work [7] and their settings in HTS are directly extracted from [7]. The only new parameter introduced in this work is the subpopulation size (ps, see Section 3.2.2). To calibrate ps, we have given four possible values: 4, 8, 12 and 16. With each value of ps, HTS was run 30 times on the 60 QMKP benchmarks. The computational results in terms of I_ϵ and I_H^- are shown in Table 1. From Table 1, we observe that $ps = 8$ is the best setting. Indeed, in terms of both indicators, $ps = 8$ attains the largest number of instances for which it performs the best, the smallest average indicator-value and the smallest standard deviation among the four compared settings. From Table 1, we also learn that the value of ps should be neither too small nor too large. When ps is too small, the solutions included in the subpopulation could be too similar (too close to the current weight vector) which is not helpful for the diversification. When ps is too large, the two parent solutions selected for crossover could be too different (too far away from the current weight vector), leading to a much deteriorated offspring solution, which slows down the convergence of the algorithm.

Table 1
Statistical results of HTS with four different settings of subpopulation size on the 60 QMKP benchmarks. $\#Bests$ counts the number of instances on which the corresponding ps value achieves the best indicator-value among the four possible ps values. For each ps value, $Avg.$ and SD respectively show the average and the standard deviation of the 60 average indicator-values obtained on the 60 test instances. A higher value is better for $\#Bests$ while a smaller value is better for $Avg.$ and SD. Bold values correspond to the best setting for the statistical item and the indicator under consideration.

	I_ϵ				I_H^-			
	ps=4	ps=8	ps=12	ps=16	ps=4	ps=8	ps=12	ps=16
$\#Bests$	11	**32**	26	22	13	**30**	29	21
$Avg.$	0.184035	**0.174198**	0.174910	0.175684	0.138575	**0.131301**	0.131505	0.132318
SD	0.046144	**0.042367**	0.042727	0.043029	0.041283	**0.038827**	0.038850	0.038887

4.3 Computational results of HTS and comparisons with QMKP algorithms

This section reports the computational results of our HTS algorithm on the set of 60 benchmark instances, which are summarized in Table 2. Columns 1 to 5 give the characteristics of the instances, including the number of objects (n), density (d), number of knapsacks (m), instance identity number (I) and capacity of each knapsack (C). Columns 10 to 15 list our results with HTS, which are respectively the two objective values of the leftmost solution (f_1^{left} and f_2^{left}), the two objective values of the rightmost solution (f_1^{right} and f_2^{right}),

the number of non-dominated solutions compiled from the results of 30 runs ($\#NS$) and the average computational time when the program terminates (CPU(s)).

From the objective values displayed in Table 2, we learn that the leftmost and rightmost solutions obtained by HTS are two mutually non-dominated solutions. Obviously, the leftmost solution is much better than the rightmost solution in the first objective value, while much worse in the second one. These two solutions represent the two extreme points of the approximate Pareto front obtained by HTS. Looking at the number of non-dominated solutions, a clear trend is observed: when the density of the instance remains the same, the number of non-dominated solutions obtained by HTS increases with the number of items; when the number of items remains the same, it increases with the density of the instance. As to the computational efficiency, it is observed that the time required by HTS generally increases when the number of non-dominated solutions identified is larger.

Now we turn to our first computational study which aims to assess the quality of the leftmost extreme point (i.e., the best solution in terms of f_1) of the approximate Pareto set reached by the HTS algorithm. As indicated in similar studies presented in [15, 23, 34], this assessment is interesting since even if the leftmost extreme point alone cannot fully characterize the whole approximation set, it does convey useful information about the quality of the approximation when it is used together with other assessment indicators. For this purpose, we apply some state-of-the-art single objective QMKP algorithms to optimize the first objective f_1 (i.e., the total profit of the packing plan, corresponding to the objective of the QMKP) and use the output best solutions, which are supposed to be of high quality in terms of f_1 value, as references for our assessment. In our case, we consider two recent best performing single objective QMKP algorithms: the tabu-enhanced iterated greedy (TIG) algorithm [13] and the iterated responsive threshold search algorithm (IRTS) [7].

For this study, we run these two algorithms 30 times on the same computer where HTS was run. For both IRTS and TIG, we set a specific time limit for each instance, which is exactly the computational time consumed by HTS indicated in column CPU(s) of Table 2. The outcome of the IRTS and TIG algorithms is a single best solution, based on which we can calculate its two corresponding objective values (f_1 and f_2). The final results of IRTS and TIG are displayed in columns 6-7 and 8-9 of Table 2 respectively. Since the QMKP algorithms consider only the total profit of the packing plan (i.e., f_1) during the optimization process, we focus on comparing their best solutions to the leftmost solution of HTS (the non-dominated solution with the highest total profit). For each instance, the non-dominated solution(s) among the three solutions in comparison (i.e., the best solutions of IRTS and TIG, as well as the leftmost solution of HTS) are highlighted in bold. If the non-dominated

solution is unique, it is also starred.

Table 2
Computational results of HTS and comparisons with two state-of-the-art single objective QMKP algorithms. For each instance, the non-dominated solution(s) among the three solutions in comparison, namely the two best solutions of IRTS and TIG as well as the leftmost solution of HTS, are indicated in boldface. If the non-dominated solution is unique, it is also marked with an asterisk.

Instance					IRTS [7]		TIG [13]		HTS					
n	d	m	I	C	f_1	f_2	f_1	f_2	f_1^{left}	f_2^{left}	f_1^{right}	f_2^{right}	$\#NS$	CPU(s)
100	25	3	1	688	**29286**	**6293**	29138	6374	**29286**	**6293**	28894	9624	11	8.2
100	25	3	2	738	**28491**	**7446**	**28491**	**7446**	**28491**	**7446**	27860	9276	13	9.54
100	25	3	3	663	**27179**	**8640**	27039	6241	**27179**	**8640**	26908	8936	9	5.92
100	25	3	4	804	28593	7551	28593	7551	**28593***	**9302***	28410	9426	3	7.63
100	25	3	5	723	**27892**	**8303**	**27892**	**8303**	**27892**	**8303**	27553	9169	10	8.38
100	25	5	1	413	**22581**	**2896**	22379	2679	**22581**	**2896**	21995	4377	16	12.6
100	25	5	2	442	**21704***	**3621***	21623	2587	21678	2744	21523	4250	6	7.77
100	25	5	3	398	**21239**	**3256**	21166	3070	**21239**	**3256**	20823	4113	13	10.5
100	25	5	4	482	**22181**	**3379**	**22181**	**3379**	**22181**	**3379**	21761	4310	13	10.1
100	25	5	5	434	**21669**	**2904**	**21669**	**2904**	**21669**	**2904**	21210	4213	12	8.74
100	25	10	1	206	**16221**	**942**	16139	654	**16221**	**942**	15173	1483	23	22.08
100	25	10	2	221	**15700**	**714**	15561	744	**15700**	**714**	15259	1464	15	15.61
100	25	10	3	199	14893	698	14860	816	**14927***	**838***	13993	1368	18	17.98
100	25	10	4	241	**16181**	**1000**	16180	1000	**16181**	**1000**	15290	1491	16	16.77
100	25	10	5	217	**15326**	**701**	15227	701	**15326**	**701**	14609	1436	11	18.78
200	25	3	1	1381	101442	20236	100349	15894	101465	18416	100127	33362	33	103.08
200	25	3	2	1246	**107958**	**12870**	107898	12988	**107958**	**12870**	101547	33780	80	54.08
200	25	3	3	1335	**104589***	**16510***	104488	16214	104567	15811	99745	33177	85	113.06
200	25	3	4	1413	**100098**	**18553**	98824	17497	**100098**	**18553**	98677	32875	23	39.18
200	25	3	5	1358	**102311**	**18096**	101973	15559	**102311**	**18096**	99203	32995	35	47.97
200	25	5	1	828	75623	8435	74367	7417	75567	9131	73741	14687	29	70.16
200	25	5	2	747	**80033**	**6230**	79480	6367	**80033**	**6230**	74558	14838	46	75.06
200	25	5	3	801	**78043**	**7293**	77695	7196	**78043**	**7293**	73038	14564	45	63.82
200	25	5	4	848	**74111***	**8843***	73132	8403	74073	8688	72829	14487	22	47.7
200	25	5	5	815	**76610**	**7811**	76118	7762	**76610**	**7811**	73258	14596	46	65.14
200	25	10	1	414	**52259**	**2277**	51362	2604	**52259**	**2277**	49825	4946	32	82.51
200	25	10	2	373	**54830**	**2169**	54199	2153	**54830**	**2169**	50695	5030	37	181.74
200	25	10	3	400	**53605***	**2650***	52989	2310	53586	2575	50071	4910	42	144.73
200	25	10	4	424	51151	2281	50591	2480	51135	2458	49568	4868	22	85.72
200	25	10	5	407	53621	2383	52981	2504	53598	2839	49599	4909	40	116.39
100	75	3	1	669	**69977**	**7823**	69935	8006	**69977**	**7823**	64030	21310	40	9.28
100	75	3	2	714	**69504**	**8970**	**69504**	**8970**	**69504**	**8970**	63633	21188	69	13.68
100	75	3	3	686	**68832**	**10614**	68811	8986	**68832**	**10614**	64706	21512	42	9.08
100	75	3	4	666	**70028**	**8499**	**70028**	**8499**	**70028**	**8499**	62783	20909	78	11.73
100	75	3	5	668	**69692**	**9373**	**69692**	**9373**	**69692**	**9373**	65498	21737	46	11.45
100	75	5	1	401	**49421**	**3582**	49397	3554	**49421**	**3582**	43565	8693	32	15.25
100	75	5	2	428	49400	3452	49350	3499	49365	3858	42850	8537	52	18.71
100	75	5	3	411	48495	3788	48495	3788	**48495***	**3816***	44177	8776	47	25.44
100	75	5	4	400	**50246**	**3692**	**50246**	**3692**	**50246**	**3692**	42982	8570	61	12.28
100	75	5	5	400	**48753**	**4170**	48752	4368	**48753**	**4170**	44543	8857	32	12.62
100	75	10	1	200	**30296**	**1052**	30111	842	**30296**	**1052**	26155	2501	53	153.55
100	75	10	2	214	31101	1179	31032	941	**31129***	**1184***	25491	2458	37	63.45
100	75	10	3	205	**29908**	**926**	29829	987	**29908**	**926**	26125	2536	40	76.28
100	75	10	4	200	**31762**	**1057**	31657	1191	**31762**	**1057**	25546	2514	57	58.52
100	75	10	5	200	**30507***	**1049***	30279	1049	30465	958	27030	2637	48	28.84
200	75	3	1	1311	**270718**	**29712**	**270718**	**29712**	**270718**	**29712**	238288	79407	231	121.16
200	75	3	2	1414	**257288**	**38726**	**257288**	**38726**	257156	38420	237784	79231	151	50.09
200	75	3	3	1342	**270069**	**31536**	**270069**	**31536**	**270069**	**31536**	241151	80310	204	59.14
200	75	3	4	1565	246993	38734	246746	36900	246961	38794	231224	77011	107	61.44
200	75	3	5	1336	**279598**	**31892**	**279598**	**31892**	**279598**	**31892**	242174	80715	207	74.4
200	75	5	1	786	**185097***	**13690***	184917	12557	185076	13671	158498	31624	127	78.82
200	75	5	2	848	174812	14404	174739	14626	174836	14220	158785	31706	83	66.28
200	75	5	3	805	**186767***	**13705***	186670	12904	186745	12999	161095	32184	119	72.9
200	75	5	4	939	**166874***	**15174***	166611	16125	166815	14126	153731	30662	66	68.29
200	75	5	5	801	**193310***	**13852***	193100	12712	193240	13558	161412	32217	92	90.47
200	75	10	1	393	112987	3951	112892	3772	**113140***	**4598***	93527	9246	79	836.65
200	75	10	2	424	105846	3829	105353	4315	105597	4849	93269	9218	72	537.29
200	75	10	3	402	114561	3651	114216	3442	114551	4079	93884	9305	89	209.18
200	75	10	4	469	99307	4643	98355	4686	99017	5490	90491	8988	54	230.15
200	75	10	5	400	117141	3209	116640	3994	117026	3415	94586	9369	93	528.8

Table 2 discloses that HTS easily outperforms TIG. Indeed, HTS always finds a solution that is equal to or dominates TIG for *all* test instances. Compared

to the best performing IRTS algorithm, HTS remains very competitive. Specifically, HTS is able to identify 34 non-dominated solutions that match those found by IRTS. More importantly, HTS discovers 5 solutions that are not reached by IRTS. In terms of the total number of non-dominated solutions found, HTS also achieves a result that is comparable to IRTS (39 v.s. 44). Among the 36 non-dominated solutions of HTS that have the same f_1 value as IRTS, there are 34 of them having exactly the same value in terms of f_2, with only two exceptions (Instance 100_25_3_4 and 100_75_5_3). This is due to the quadratic nature of the BO-QMKP problem where one item is associated with all other items of the same knapsack with a crossproduct value. Thus, a change of the assignment of a single item can lead to a large difference of the f_1 value. This implies that a f_1 value is less likely to correspond to multiple different solutions and explains why HTS and IRTS find exactly the same solutions in 34 instances. From this study, we learn that though the computational resources of HTS are evenly allocated to two objectives, the resulting best solutions in terms of the first objective (f_1) are quite good even compared to their state-of-the-art results obtained by the best performing single objective algorithms, which devote all their efforts on f_1. This study demonstrates clearly that the leftmost part of the (approximate) Pareto set discovered by HTS is of high quality. To further assess the quality of the *whole* approximation set, additional comparative studies are carried out in the following sections.

4.4 Comparative results of HTS with NSGA-II

In this section, we turn to a comparison of the proposed HTS algorithm against NSGA-II [9], which is a well-known representative of the traditional multi-objective evolutionary algorithms without local search. In this experiment for solving the BO-QMKP, NSGA-II employs the same solution representation and crossover operator as HTS. The mutation is a random implementation of the exchange operator. That is, two objects from two different knapsacks are randomly selected and exchanged. The parameter settings of NSGA-II are displayed in Table 3. NSGA-II was run 30 times under the same computational environment as HTS.

Table 3
Parameter settings for the NSGA-II developed for the BO-QMKP.

Parameter description	Value
Population size	100
Crossover rate	1.0
Mutation rate	0.1
Max generation	$100*n$

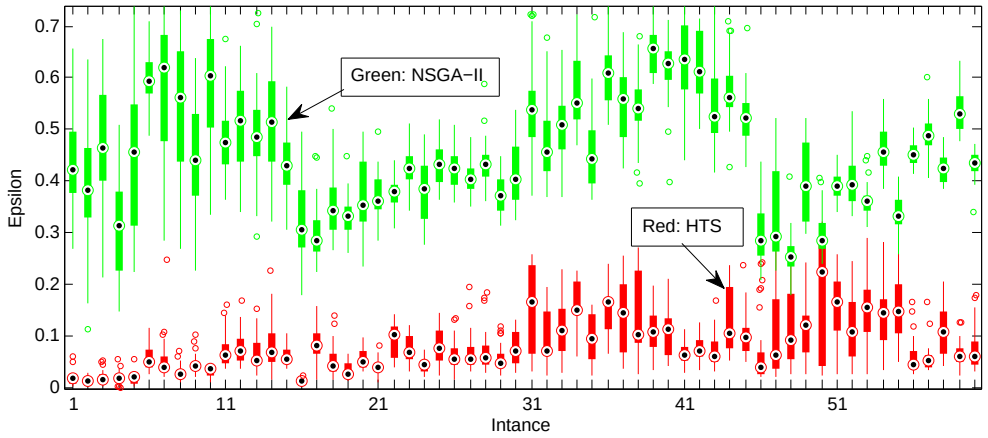
The comparative results in terms of the epsilon indicator (I_ϵ) and the hyper-

volumn indicator (I_H^-) are summarized in the box-and-whisker plot of Figure 2. Here, each instance is indicated by a number, ranging from 1 to 60, according to the order it appears in Table 2. The x-axis of the plot indicates the instances while the y-axis shows the indicator values. Recall that, for each instance, each algorithm was run 30 times, resulting in 30 approximation sets, and thus 30 indicator values. The box-and-whisker inside the plot provides a number of statistical information including the minimum and maximum of the indicator values (the ends of the whiskers), the first and third quartiles (the bottom and top of the box) and the median (the point inside the box). Concerning the I_ϵ indicator, Figure 2(a) shows that HTS performs much better than NSGA-II across the whole instance set. Indeed, the box-and-whisker plots of HTS are typically below those of the NSGA-II. HTS always achieves a smaller median I_ϵ value compared to NSGA-II for *all* instances. For 55 out of 60 instances, even the maximum I_ϵ value of HTS is smaller than the minimum I_ϵ value of NSGA-II. A similar observation can be made for the I_H^- indicator, and the superiority of HTS appears to be even more obvious. Indeed, from Figure 2(b), it can be observed that the box-and-whiskers of HTS are totally below those of NSGA-II, without any overlap at any instance. Moreover, the boxes of HTS are typically very small, indicating a small variation of the results and a strong robustness of the proposed algorithm. We applied a Mann-Whitney test to compare the two groups of approximation sets of each instance. All the resulting p-values are much smaller than 0.05, indicating that HTS is significantly better than NSGA-II for *all* the test instances in terms of both I_ϵ and I_H^-. Furthermore, HTS is much more efficient than NSGA-II in terms of the average CPU time in seconds across the whole instance set (84.6 v.s. 543.17).

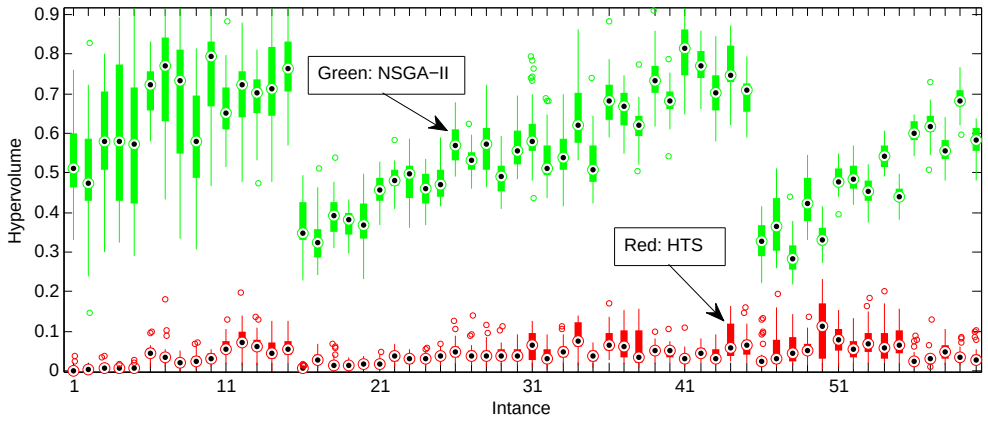
4.5 Comparative results of HTS with two simplified algorithm variants

This section is dedicated to a performance comparison of HTS with its two simplified variants: SMS and DSLS. The SMS version was obtained by removing the Pareto local search component (the second stage) from HTS; while the DSLS version was obtained by removing the memetic search component from SMS, merely keeping the archive initialization component. These experiments allow us to appreciate the impact of the removed components over the performance of the proposed HTS algorithm.

All algorithms (HTS, SMS and DSLS) were run 30 times on the set of 60 benchmark instances. The computational results in terms of the average indicator values (for both I_ϵ and I_H^-) are summarized in Figure 3. For the sake of better readability, we do not provide the detailed box-and-whisker plots which embody numerous overlaps. Complementary to Figure 3, we also provide in Table 4, a summary of the computational results in terms of the average in-



(a) Epsilon indicator

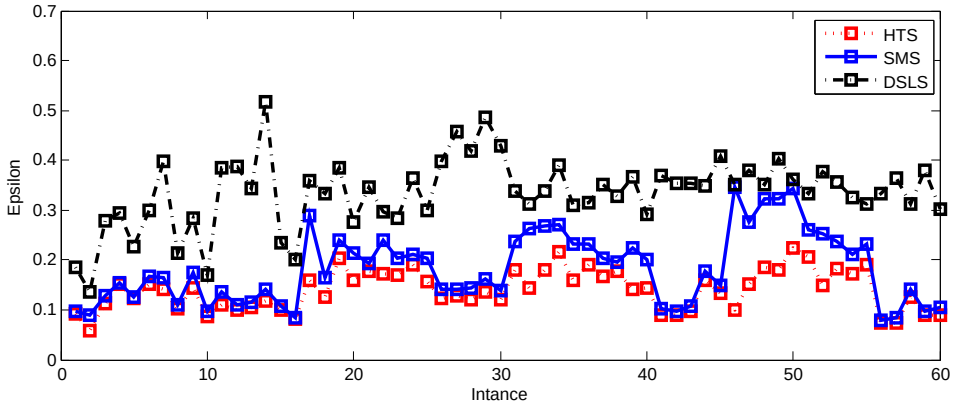


(b) Hypervolume difference indicator

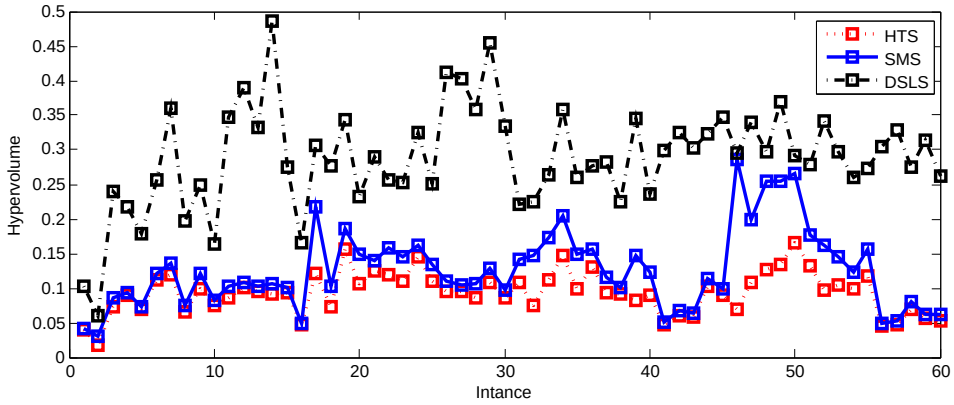
Fig. 2. Comparison of HTS with NSGA-II with respect to the epsilon indicator and the hypervolume difference indicator on 60 benchmark instances. For each instance and each indicator, the box-and-whisker summarizes the 30 values obtained by each compared algorithm. A lower indicator-value is better.

dicator values, the number of non-dominated solutions and the CPU times in seconds for 12 instance classes, each of which (named as n_d_m) includes 5 instances with the same number of items (n), the same density (d) and the same number of knapsacks (m). The values reported in Table 4 are the average of the 5 results of each instance class.

Figure 3 shows that compared to the simplest DSLS algorithm, the SMS algorithm performs much better in terms of both I_ϵ and I_H^- values across the whole instance set. Indeed, the blue lines indicating the indicator values of SMS are consistently below the black lines representing the indicator values of the DSLS. Also, the average number of non-dominated solutions identified by SMS are higher than those of DSLS for 11 out of 12 instance classes (except for 100_25_3). Due to the inclusion of an additional memetic search component, SMS consumes on average more computing time than DSLS. However, the



(a) Epsilon indicator



(b) Hypervolume difference indicator

Fig. 3. Comparison of HTS with two simplified algorithm variants with respect to the epsilon indicator and hypervolume difference indicator on 60 benchmark instances. For each instance and each indicator, the value plotted is the average of the 30 values obtained by each compared algorithm. A lower indicator-value is better.

Table 4
Comparative results of HTS with its two simplified algorithm variants. 60 benchmarks are divided into 12 instance classes, each containing 5 instances. The values in the table are the average of the 5 results for each instance class. A smaller value is better for both I_ϵ and I_H^- indicators.

	DSLS				SMS				HTS			
Instance	I_ϵ	I_H^-	NoNS	CPU(s)	I_ϵ	I_H^-	NoNS	CPU(s)	I_ϵ	I_H^-	NoNS	CPU(s)
100_25_3	0.28	0.23	8.80	5.59	0.12	0.08	8.60	7.87	0.10	0.07	9.20	7.93
100_25_5	0.35	0.32	10.00	6.42	0.13	0.10	11.40	9.87	0.12	0.09	12.00	9.94
100_25_10	0.24	0.22	9.20	8.03	0.13	0.10	14.60	18.13	0.12	0.09	16.60	18.24
200_25_3	0.33	0.27	19.20	23.12	0.19	0.12	35.80	69.54	0.14	0.08	51.20	71.47
200_25_5	0.34	0.27	23.00	30.79	0.18	0.11	26.20	63.83	0.15	0.09	37.60	64.38
200_25_10	0.35	0.31	22.40	36.72	0.21	0.14	25.00	121.89	0.15	0.10	34.60	122.22
100_75_3	0.35	0.31	13.20	4.77	0.17	0.12	26.00	10.68	0.13	0.10	55.00	11.04
100_75_5	0.36	0.32	17.00	5.81	0.19	0.13	31.80	16.63	0.15	0.10	44.80	16.86
100_75_10	0.35	0.30	21.80	7.88	0.20	0.15	41.40	75.90	0.17	0.12	47.00	76.13
200_75_3	0.35	0.31	12.20	18.99	0.21	0.15	22.80	33.49	0.12	0.08	180.00	73.25
200_75_5	0.36	0.31	24.20	25.23	0.21	0.14	37.20	73.05	0.15	0.09	97.40	75.35
200_75_10	0.34	0.29	26.40	35.42	0.25	0.18	58.00	467.46	0.17	0.11	77.40	468.41

average time of 80.69 seconds (less than 1.35 min) of SMS across the whole instance set is quite acceptable in practice. The above observation confirms the effectiveness of the memetic search component of the SMS algorithm.

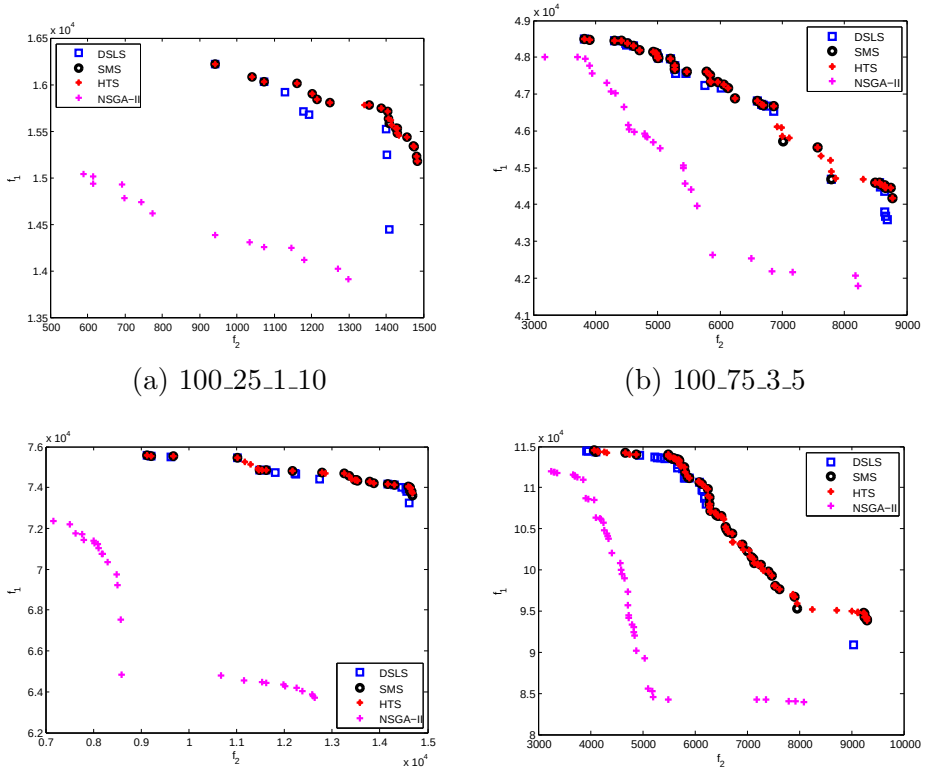
HTS, which combines SMS and the Pareto local search component, further improves on SMS in terms of approximation quality. From Table 4, it can be seen that HTS is always slightly better than SMS in terms of I_ϵ and I_H^- for all instance classes. Moreover, compared to SMS, HTS substantially enlarges the number of non-dominated solutions found, with a very small increase of the average computing time. This observation implies that HTS can marginally advance the (approximate) Pareto fronts identified by SMS. In fact, these approximate Pareto fronts are already quite good which are difficult to be further improved. However, compared to SMS, HTS is able to enrich the non-dominated solutions contained in the approximation set using a small amount of additional time, which helps to discover more non-dominated solutions and promotes a more uniform distribution of these solutions.

4.6 Graphical study

Graphical representations are well suited to visualize the behaviour of bi-objective optimization methods. In this section, we provide a set of plots of the non-dominated solutions obtained by the compared algorithms on selected instances. To this end, we selected one instance from each combination of n_d, leading to four representative instances: 100_25_10_1, 100_75_5_3, 200_25_5_1 and 200_75_10_3 (instance names are in the form of $n\_d\_m\_I$). Figure 4 illustrates the non-dominated solutions obtained on the four representative instances by HTS, SMS, DSLS and NSGA-II. For each algorithm and each instance, the non-dominated solutions are compiled from the 30 approximation sets obtained in 30 runs.

From Figure 4, it can be observed that NSGA-II attains the worst convergence among the four compared algorithms. It is even much worse than DSLS, the most simplified variant of the proposed HTS algorithm, not to mention the others. This observation is true for all four selected instances. The disappointing performance of NSGA-II could be due to the fact that it lacks an effective intensification-oriented search procedure (contrary to the RTS procedure used in HTS and its variants) and consequently it is unable to ensure a suitable balance between diversification and intensification of its search process.

By comparing DSLS with SMS, the latter always achieves more efficient and better distributed solutions than the former. Indeed, the black diamonds representing SMS are typically closer to the upright corner (where the true Pareto front locates) than the blue squares of DSLS. The most obvious example is



(a) 100_25_1_10 (b) 100_75_3_5

(c) 200_25_1_5 (d) 200_75_3_10

Fig. 4. Non-dominated solutions obtained by 30 runs of the four compared algorithms for four representative instances.

Figure 4(a), followed by the other three. Also, the black diamonds cover more better the objective space than the blue squares. Indeed, the blue squares do not reach some specific parts of the objective space, such as the left area of Figure 4(b) and the middle-right area of Figure 4(b),(d). These areas are however filled by the black diamonds. Compared to SMS, HTS obtains an even better spread of the solutions and sometimes achieves a better convergence of the approximation sets. Indeed, in the middle-left area of Figure 4(c), middle-right area of Figure 4(b) and right area of 4(d), the red asterisks of HTS appear at the places where the black diamonds of the SMS are absent, making the non-dominated solutions of HTS well spread in the whole objective space. Also, some red-asterisk solutions of HTS in Figure 4(b) dominate their nearby black-diamond solutions of SMS.

5 Conclusion

In this paper, we introduced the bi-objective quadratic multiple knapsack problem (BO-QMKP), that aims to maximize both the total profit of the packing plan and the makespan (the gain of the weakest knapsack). With

an additional objective, the BO-QMKP enriches the single objective QMKP model and opens the way for more practical problems to be conveniently formulated and solved.

Like the single objective case, solving the BO-QMKP represents an imposing computational challenge. To approximate effectively the Pareto front of a BO-QMKP instance, we introduced the hybrid two-stage (HTS) algorithm, which combines respectively the scalarizing strategy and the Pareto-based strategy in its two stages. The first stage of HTS relies on a scalarizing memetic search method, that integrates a state-of-the-art responsive threshold search algorithm for the single objective QMKP with the evolutionary framework. The second stage is a double-neighborhood Pareto local search which is characterized by its two dedicated neighborhoods. By allowing the generation of both supported and unsupported solutions, the proposed HTS approach aims to find a well-covered and well-distributed Pareto set approximation.

Experimental studies on a set of 60 well-known QMKP benchmarks showed that the proposed approach obtains significantly better results than the conventional NSGA-II algorithm without local search, demonstrating the interest of including an intensification-oriented local search method in HTS. The comparative studies of HTS with its two simplified algorithm variants additionally demonstrated the dominance of HTS and highlight the contribution of the memetic search framework and the Pareto local search component used by HTS. Finally, we also showed experimental evidences that the population based HTS approach is very competitive with the state-of-the-art QMKP algorithms when only the first objective is considered.

This study opens the way for some future work. First, it would be meaningful to study the BO-QMKP features through a fitness landscape analysis [36], which would help to improve the proposed HTS algorithm. Second, for the purpose of better solving the BO-QMKP model, it would be interesting to investigate other solution methods, for instance, the well-known epsilon-constraint method [6] which is widely used in multiple objective optimization and other types of multi-objective algorithms like those proposed in [3, 11, 24]. Third, given the success of hybridizing scalarizing approaches with Pareto local search in the context of the BO-QMKP, it would be relevant to apply such a hybridization methodology to other bi-objective problems. Finally, we hope that the BO-QMKP model will attract additional research effort as to new solution methods and practical applications.

Acknowledgment

We are grateful to the anonymous referees for valuable suggestions and comments which helped us to improve the paper. The work was partially supported by the PGMO (2014-0024H) project from the Jacques Hadamard Mathematical Foundation (Paris, France) and the National Natural Science Foundation of China (Grants 61473301, 71201171, 71501179). Support for Yuning Chen from the China Scholarship Council is also acknowledged.

References

[1] Aneja Y.P., Nair KPK. Bicriteria transportation problem. Management Science 1979; 25(1): 73-78.

[2] Billionnet A. and Soutif E. An exact method for the 0-1 quadratic knapsack problem based on lagrangian decomposition. European Journal of Operational Research 2004; 157(3): 565-575.

[3] Amiri B., Hossain L., Crawford J.W., Wigand R.T. Community detection in complex networks: Multiobjective enhanced firefly algorithm. Knowledge-Based Systems 2013; 46: 1–11.

[4] Barichard V., Hao J.K. Genetic tabu search for the multi-objective knapsack problem. Tsinghua Science and Technology 2003; 8(1): 8-13.

[5] Bader J., Zitzler E. HypE: An algorithm for fast hypervolume-based many-objective optimization. Evolutionary Computation 2011; 19(1): 45-76.

[6] Chankong V., Haimes Y.Y. Multiobjective decision making: theory and methodology, North-Holland, New York, 1983.

[7] Chen Y., Hao J.K. Iterated responsive threshold search for the quadratic multiple knapsack problem. Annals of Operations Research 2015; 226(1): 101-131.

[8] Chen Y., Hao J.K., Glover F. An evolutionary path relinking approach for the quadratic multiple knapsack problem. Knowledge-Based Systems 2016; 92:23–34.

[9] Deb K., Pratap A., Agarwal S. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 2002; 6(2): 182-197.

[10] da Silva C.G., Climaco J., Figueira J. A scatter search method for bi-criteria 0, 1-knapsack problems. European Journal of Operational Research 2006; 169(2): 373-391.

[11] Dai C., Wang Y. A new uniform evolutionary algorithm based on decomposition and CDAS for many-objective optimization. Knowledge-Based Systems 2015; 85: 131–142.

[12] García-Martínez C., Glover F., Rodríguez-Díaz F.J., Lozano M, Martí R. Strategic oscillation for the quadratic multiple knapsack problem. Computational Optimization and Applications 2014; 58(1): 161-185.

[13] García-Martínez C., Rodríguez-Díaz F.J., Lozano M. A tabu-enhanced iterated greedy algorithm: a case study in the quadratic multiple knapsack problem. European Journal of Operational Research 2014; 232(3): 454-463.

[14] Hiley A., Julstrom B. The quadratic multiple knapsack problem and three heuristic approaches to it. In Proceedings of the genetic and evolutionary computation conference (GECCO) 2006; pp 547-552.

[15] Jaszkiewicz A. Do multiple objective metaheuristics deliver on their promises? a computational experiment on the set covering problem. IEEE Transactions on Evolutionary Computation 2003; 7(2):133-43.

[16] Joshua K., Corne D. Memetic algorithms for multiobjective optimization: issues, methods and prospects. Recent advances in memetic algorithms. Springer Berlin Heidelberg 2005; 313-352.

[17] Jérémie D.L., López-Ibáñez M., and Stützle T. A hybrid TP+ PLS algorithm for bi-objective flow-shop scheduling problems. Computers & Operations Research 2011; 38(8): 1219-1236.

[18] Jérémie D.L., López-Ibáñez M., and Stützle T. Improving the anytime behavior of two-phase local search. Annals of mathematics and artificial intelligence 2011; 61(2): 125-154.

[19] Knowles J., Thiele L., Zitzler E. A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland (2006).

[20] Lust T., Teghem J. Two-phase Pareto local search for the biobjective traveling salesman problem. Journal of Heuristics 2010; 16(3): 475-510.

[21] Liefooghe A., Verel S., Paquete L., Hao J.K. A hybrid metaheuristic for multiobjective unconstrained binary quadratic programming. Applied Soft Computing 2014; 6:10-19.

[22] Liefooghe A., Verel S., Paquete L., Hao J.K. Experiments on Local Search for Bi-objective Unconstrained Binary Quadratic Programming. In A. Gaspar-Cunha et al. (Eds): EMO 2015; Lecture Notes in Computer Science 9018: 171-186.

[23] Mei Y., Tang K., Yao X. Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem. IEEE Transactions on Evolutionary Computation 2011; 15(2): 151–165.

[24] Mansour I.B., Alaya I. Indicator based ant colony optimization for multi-objective knapsack problem. Procedia Computer Science 2015; 60: 448-457.

[25] Neri, F., Cotta, C., Moscato, P. (Eds.). Handbook of memetic algorithms. Studies in Computational Intelligence 379, Springer, 2012.

[26] Pisinger D. An exact algorithm for large multiple knapsack problems. European Journal of Operational Research 1999; 114(3): 528-541.

[27] Paquete L., Chiarandini M., Stützle T. Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. Metaheuristics for Multiobjective Optimisation. Volume 535 of Lecture Notes in Economics and Mathematical Systems, Springer Berlin Heidelberg 2004; 177-199.

[28] Pisinger D. The quadratic knapsack problema survey. Discrete Applied Mathematics 2007; 155(5): 623-648.

[29] Pisinger D., Rasmussen A.B., Sandvik R. Solution of large quadratic knapsack problems through aggressive reduction. INFORMS Journal on Computing 2007; 19(2): 280-290.

[30] Rong A., Figueira J.R., Klamroth k. Dynamic programming based algorithms for the discounted 01 knapsack problem. Applied Mathematics and Computation 2012; 218(12): 6921-6933.

[31] Saraç T., Sipahioglu A. A genetic algorithm for the quadratic multiple knapsack problem. In: BVAI (LNCS 4729), 2007; pp 490-498.

[32] Singh A., Baghel A. A new grouping genetic algorithm for the quadratic multiple knapsack problem. EvoCOP (LNCS 4446) 2007, pp 97-109.

[33] Sundar S., Singh A. A swarm intelligence approach to the quadratic multiple knapsack problem. In: ICONIP (LNCS 6443), 2010; pp 626-633

[34] Shang R., Wang J., Jiao L., Wang Y. An improved decomposition-based memetic algorithm for multi-objective capacitated arc routing problem. Applied Soft Computing 2014; (19): 343-361.

[35] Vallada E., Rubén R. Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. Omega 2010; 38(1): 57-67.

[36] Verel S., Liefooghe A., Jourdan L., Dhaenens C. On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives. European Journal of Operational Research 2013; 227(2): 331-342.

[37] Xiong J., Yang K., Liu J., Zhao Q., Chen Y. A two-stage preference-based evolutionary multi-objective approach for capability planning problems. Knowledge-Based Systems 2012; 31: 128–139.

[38] Zhang QF, Hui L. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 2007; 11(6): 712-731.

[39] Zitzler E., Thiele L., Laumanns M, Fonseca CM., da Fonseca VG. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation 2003; 7(2): 117-132.