Intl. Trans. in Op. Res. xx (2024) 1–27

Adaptive feasible and infeasible evolutionary search for the knapsack problem with forfeits

Qing Zhou[a], Jin-Kao Hao[b], Zhong-Zhong Jiang [a,*], and Qinghua Wu[c]

[a]*School of Business Administration, Northeastern University, 195 Chuangxin Road, Shenyang 110169, China*
[b]*LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers Cedex 01, France*
[c]*School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan 430074, China*
E-mail: zhouqing@mail.neu.edu.cn; jin-kao.hao@univ-angers.fr;
zzjiang@mail.neu.edu.cn; qinghuawu1005@gmail.com

Received 08-11-2023; received in revised form 24-06-2024; accepted 24-06-2024

International Transactions In Operational Research, 2024, https://doi.org/10.1111/itor.13512

Abstract

The knapsack problem with forfeits is a generalized knapsack problem that aims to select some items, among a set of candidate items, to maximize a profit function without exceeding the knapsack capacity. Moreover, a forfeit cost is incurred and deducted from the profit function when both incompatible items are placed in the knapsack. This problem is a relevant model for a number of applications and is however computationally challenging. We present a hybrid heuristic method for tackling this problem that combines the evolutionary search with adaptive feasible and infeasible search to find high-quality solutions. A streamlining technique is designed to accelerate the evaluation of candidate solutions, which increases significantly the computational efficiency of the algorithm. We assess the algorithm on 120 test instances and demonstrate its dominance over the best performing approaches in the literature. Particularly, we show 94 improved lower bounds. We investigate the essential algorithmic components to understand their roles.

Keywords: Knapsack; Forfeit; Conflict graph; Evolutionary framework; Heuristic

1. Introduction

The popular knapsack problem (KP) (Martello and Toth, 1990) is the following subset selection problem. Given a knapsack with a predefined capacity c, and a set of n items $V = \{1, 2, ..., n\}$ where each item $i \in V$ has a profit $p_i > 0$ and a weight $w_i > 0$, KP involves in choosing a number of items from V to maximize the total profit without surpassing the capacity c.

*Author to whom all correspondence should be addressed (e-mail: zzjiang@mail.neu.edu.cn).

The knapsack problem with forfeits (KPF) (Cerulli et al., 2020) is a generalized KP that considers forfeits for incompatible items. In KPF, in addition to the capacity c and set $V = \{1, 2, ..., n\}$ of items, we are given a set E of l pairs of incompatible items $E = \{E_k \in V \times V : k = 1, ..., l\}$, where each $E_k = \{i, j\} \in E$ indicates that a given forfeit $d_k > 0$ is induced if items i and j are selected simultaneously. Then KPF aims to select a number of items in V to maximize the total profit minus the total forfeit induced, while satisfying the capacity constraint. According to (Cerulli et al., 2020), KPF can be formally expressed by

$$\textbf{Maximize} \quad f = \sum_{i=1}^{n} p_i x_i - \sum_{k=1}^{l} d_k v_k \tag{1}$$

$$\textbf{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq c \tag{2}$$

$$x_i + x_j - v_k \leq 1, \quad \forall E_k = \{i, j\}, k \in \{1, ..., l\} \tag{3}$$

$$x_i \in \{0, 1\}, \quad \forall i \in \{1, ..., n\} \tag{4}$$

$$v_k \in \{0, 1\}, \quad \forall k \in \{1, ..., l\} \tag{5}$$

where the binary variable x_i is set to 1 if item i is selected, and 0 otherwise, while the binary variable v_k is equal to 1 if the forfeit cost d_k associated with $E_k \in E$ is to be paid, and 0 otherwise.

The objective function (1) is to maximize the overall collected profit minus the total forfeit cost. Constraint (2) imposes that the knapsack capacity constraint is not violated. Constraint (3) bundles each x variable with v variables to guarantee that v_k must be set to 1 if both items i and j are selected, with $E_k = \{i, j\}$. The binary values for variables x_i and v_k are imposed in constraints (4) and (5).

Obviously, when $E = \emptyset$, KPF becomes KP. As a result, KPF is at least as difficult as KP, which is known to be NP-hard (Cerulli et al., 2020). KPF can be used to formulate real-world applications where KP is no more suitable due to the presence of incompatible items. As an example, we consider the following decision problem to optimally load objects in a truck (Capobianco et al., 2022). In this application, each object has an associated weight and profit, and the truck has a limited capacity. Moreover, some pairs of objects may cause forfeit costs (i.e., decontamination costs) if both objects of such a pair are transported together. We want to load some objects among the candidate objects in the truck to maximize the total profit of the loaded objects minus the total forfeit induced while respecting the capacity of the truck. KPF has other relevant applications in areas such as drugs management for patients (Capobianco et al., 2022) and investments decisions (Cerulli et al., 2020), where simultaneous uses of conflicting resources would cause extra costs.

Due to the relevance of KPF, several heuristic algorithms have been studied. The literature review of Section 2 indicates that progresses have been made since the introduction of KPF in 2020 (Cerulli et al., 2020). Nevertheless, considering that there are still large gaps between the results reported by the best KPF methods and the lower bounds given by the commercial CPLEX solver, more powerful methods are necessary to alleviate the limits of existing approaches. Indeed, current KPF algorithms only explore the greedy approach and hybrid genetic-greedy paradigm.

In this work, we present a hybrid evolutionary search method (HESM) dedicated to KPF, which combines a meaningful crossover operator and an effective local optimization. On the one hand, we observe, through many sampled solutions, that high-quality solutions typically have a large number of common items. Based on this observation, we adopt the uniform crossover operator (denoted by UX), which is capable of saving common items shared by parent solutions. Thanks to UX, the algorithm can obtain meaningful offspring solutions from high-quality solutions. On the other hand, the local optimization procedure adopts a mixed search strategy to explore both feasible and infeasible solutions. By dynamically controlling the oscillation between feasible and infeasible searches, the algorithm has more chances to reach high-quality local optimal solutions that are difficult to discover when the search is limited to feasible regions.

We assess the HESM algorithm on the three sets of 120 KPF benchmark instances in the literature and show that our algorithm is extremely effective with respect to the best performing KPF approaches. Specifically, HESM finds 94 new lower bounds and matches 15 other best-known bounds.

Section 2 reviews existing studies related to KPF. Section 3 describes the HESM algorithm. Section 4 provides computational assessments and comparisons with the best performing methods. Section 5 investigates the key algorithmic components, followed by conclusions and perspectives.

2. Related work

Cerulli et al. (2020) first considered KPF. As solution methods, they proposed an integer linear program (ILP) and two greedy heuristics. The first heuristic is a conventional greedy algorithm (denoted by Greedy) and the second heuristic uses the carousel greedy paradigm (Cerrone et al., 2017) (denoted as CG) which is a generalized framework to improve the constructive greedy approach. The two greedy methods obtained solutions of acceptable quality in very short runtime, and CG method dominated Greedy on the test graphs.

Capobianco et al. (2022) developed a hybrid metaheuristic method (denoted as GA-CG) that aims to take advantage of the genetic algorithm and carousel greedy paradigm. They showed that their method was able to produce significantly better results than the two greedy algorithms presented in (Cerulli et al., 2020). Capobianco et al. (2022) also used the CPLEX solver to solve the ILP formulations (Eqs. (1)-(5)) and showed that CPLEX under a cutoff time of 3 hours dominated GA-CG on almost all the test graphs.

D'Ambrosio et al. (2023) introduced a generalization of KPF (i.e., the knapsack problem with forfeit sets, KPFS), and proposed a metaheuristic (denoted as MA) combining memetic approach with the carousel greedy method. To generate an offspring solution, MA implemented a randomized crossover operator working in two steps. The first step adds each item belonging to both parents to the offspring with a probability of p_1, while the second step includes each item existing in one of two parents to the offspring with a probability of p_2 complying with all the constraints of KPFS, where p_1 and p_2 are two parameters of MA. The authors also included computational results on KPF and showed the superiority of MA comparing over the best KPF methods in the literature.

KPF is related to the disjunctively constrained knapsack problem (DCKP) also known as the knapsack problem with conflict graph (Cacchiani et al., 2022a; Yamada et al., 2002). The difference between these two problems is that for DCKP, it is forbidden to select simultaneously two conflicting items in a feasible solution, while for KPF, two incompatible items can be simultaneously selected, but such a selection leads to a forfeit in the objective cost. Compared to KPF, DCKP has received much more research effort in the last 20 years, and various algorithms have been proposed, including exact methods

(Bettinelli et al., 2017; Coniglio et al., 2021; Gurski and Rehs, 2019), approximation methods (Pferschy and Schauer, 2017) and heuristic methods (Ben Salem et al., 2017; Wei and Hao, 2021). Several other canonical optimization problems also have variants that consider conflict constraints such as the bin packing problem with conflicts (Ekici, 2021; Elhedhli et al., 2011) and the minimum spanning tree problem (Lu et al., 2022).

KPF is also tightly related to the classic quadratic knapsack problem (QKP) (Cacchiani et al., 2022b), and the only difference between KPF and QKP is that if both items i and j are selected, a non-negative quadratic profit p_{ij} is earned for QKP, while a non-positive extra profit (forfeit cost) p_{ij} is incurred for KPF. Various QKP approaches have been developed, including exact algorithms (Caprara et al., 1999; Pisinger et al., 2007; Rodrigues et al., 2012; Fampa et al., 2020; Fomeni et al., 2022), approximation methods (Pferschy and Schauer, 2016; Taylor, 2016; Wu et al., 2020), and heuristics (Yang et al., 2013; Fomeni and Letchford, 2014; Chen and Hao, 2017). Note that among the QKP heuristics, only IHEA (Chen and Hao, 2017) is allowed to visit infeasible solutions using an enlarged swap neighborhood. The proposed HESM algorithm uses a more elaborated mixed search strategy to dynamically control the oscillation between feasible and infeasible solution spaces.

According to the literature, CG (Cerulli et al., 2020), GA-CG (Capobianco et al., 2022) and MA (D'Ambrosio et al., 2023) are the best heuristic approaches for KPF, and IHEA (Chen and Hao, 2017) is the state-of-the-art heuristic for QKP, while the general CPLEX solver is a suitable solution approach if a large time budget and a large computer memory are available. However, there are still large gaps between the results reported by CG (Cerulli et al., 2020), GA-CG (Capobianco et al., 2022), MA (D'Ambrosio et al., 2023) and the lower bounds obtained by the CPLEX solver with the mathematical model (1)-(5) (Cerulli et al., 2020). This work aims to improve our capacity in tackling KPF, and for this presents a hybrid evolutionary search method (HESM), which is highly effective compared to the current best KPF methods as well as a current best QKP heuristic.

3. Hybrid evolutionary search method for KPF

This section is dedicated to the presentation of the general HESM algorithm and its components.

3.1. General algorithm

Algorithm 1 Hybrid evolutionary search method for KPF

1: **Hybrid evolutionary search method for KPF** *in:* I: a KPF instance, γ: population size, t_m: cutoff time *out:* Best found feasible solution S^*
2: $P = \{S^1, ..., S^\gamma\} \leftarrow pop\_initial(\gamma)$ /* Section 3.3, population initialization */
3: $S^* \leftarrow best(P)$ /* S^* keeps the best encountered solution */
4: **while** t_m is not reached **do**
5: Select two parent solutions S^a, S^b from P at random
6: $S^o \leftarrow cross\_over(S^a, S^b)$ /* Section 3.4, offspring generation */
7: $S^o \leftarrow adaptive\_feasible\_infeasible\_tabu\_search(S^o)$ /* Section 3.5, offspring improvement */
8: **if** $f(S^o) > f(S^*)$ **then**
9: $S^* \leftarrow S^o$ /* Renew the best found solution */
10: **end if**
11: $P \leftarrow pop\_update(P, S^o)$
12: **end while**
13: Return S^*

HESM method (Algorithm 1) follows the memetic search framework (Moscato, 1999) and blends population-based search with neighborhood-based search. There are a number of successful memetic algorithms for difficult optimization problems such as knapsack problems (Lai et al., 2018; Wei and Hao, 2021), job-shop scheduling problems (Constantino and Segura, 2022; Wang and Wang, 2022; Wu and Che, 2019; Zhang et al., 2023), and routing problems (Bravo et al., 2019; Mara et al., 2021; Vidal et al., 2013).

HESM starts by building an initial population P (see Section 3.3) consisting of γ individuals, where γ is a parameter called the population size. Then HESM carries out a series of generations to improve the solutions in the population (lines 4-12, Algorithm 1). To build an offspring solution S^o at each generation, the uniform crossover operator presented in Section 3.4 is applied to two randomly chosen parent solutions from P. The quality of the offspring S^o is further raised by the tabu search procedure presented in Section 3.5. Finally, the best recorded solution and population are updated with S^o. For the population, S^o just substitutes the lowest quality solution in P if S^o has a better objective value (Eq. (1)) and it does not already exist in P; otherwise the population keeps unchanged. This algorithm ends when a termination condition such as a cutoff time t_m is satisfied.

3.2. Search space and penalty-based evaluation function

Let $I = (V, E, c, p, w)$ be a KPF instance. The search space Ω is defined as $\Omega = \{S : S \subseteq V\}$, which includes both feasible and infeasible solutions. For a solution S in Ω, it can be conveniently represented by a binary n-vector $S = (x_1, x_2, \ldots, x_n)$, where $x_i = 1$ if item i is selected, and $x_i = 0$ otherwise.

For several grouping problems (e.g., knapsack problem (Zhou et al., 2022), coloring problem (Sun et al., 2020)), it is observed that exploring intermediary infeasible solutions by relaxing some problem constraints is highly beneficial for finding high-quality solutions. In the case of KPF, we relax the knapsack constraint and employ the following extended evaluation function F to assess the quality of any solution S in Ω.

$$F(S) = f(S) - \beta \times EX(S) \tag{6}$$

where $f(S)$ is the value of the objective function (Eq. (1)), $EX(S) = max\{0, \sum_{i \in S} w_i - c\}$ is the overall weight excess over the capacity limit c, and β is a self-adjusted parameter controlling the degree of infeasibility (called infeasibility control parameter). In general, a larger value of β induces a stronger penalization to infeasible solutions, which has the effect of decreasing the attractiveness of infeasible solutions and encouraging the search to leave infeasible areas. Conversely, a smaller value of β increases the attractiveness of infeasible solutions and encourages the search to explore more infeasible solutions. In Section 3.5, we explain how the β parameter is adaptively adjusted to modify the search trajectory of the algorithm.

3.3. Population initialization

The initial population P contains γ feasible solutions (individuals), where each individual is created in two steps. The first step generates a random feasible solution from an empty solution S by executing a number of adding operations. For each adding operation, an unallocated item is selected at random and inserted into S, so long as its weight is not greater than the residual capacity of the knapsack. This step is repeated until the knapsack capacity is reached. The second step invokes the tabu search procedure (see Section 3.5) to further improve S.

The resulting solution is finally inserted into P if the population does not contain the same solution already. This process stops when the number of solutions in P attains γ.

3.4. Uniform crossover

Usually, an effective crossover should be able to conserve good properties of parent solution (Hao, 2012; Neri and Cotta, 2012). For KPF, our preliminary experiments (see Section 5.3) indicated that some particular items frequently appear in high-quality solutions, which might correspond to the backbone of an optimal solution.

Based on this observation and the general principle for applying crossovers, we adopt the canonical uniform crossover operator (Syswerda et al., 1989) (denoted as UX) for KPF, which builds an offspring solution S^o by inheriting randomly the values of two parent solutions. Let $S^a = (x_1^a, x_2^a, \ldots, x_n^a)$ and $S^b = (x_1^b, x_2^b, \ldots, x_n^b)$ be the given parents, $S^o = (x_1^o, x_2^o, \ldots, x_n^o)$ is created as follows. Each x_i^o ($i = 1, 2, \ldots, n$) takes the value of x_i^a or x_i^b with equal probability. For KPF, UX is quite suitable. Indeed, if an item i appear in both parents, then the item is always retained in the offspring. If an item i does not appear in either parent, this item is not selected in the offspring. Finally, if an item i appears only in one parent, this item has a chance of 50% to be selected in the offspring. As such, UX is able to transmit the common items of the parents, which goes with our experimental observation that high-quality solutions share common items. Meanwhile, by retaining randomly the other non-shared items, UX naturally induces a diversification effect, which prevents the algorithm from a premature convergence.

An offspring solution generated by UX can be feasible or infeasible and is assessed by the extended evaluation function F defined by Eq. (6).

The time complexity of UX is obviously bounded by $O(n)$.

As shown in Section 2, MA (D'Ambrosio et al., 2023) which is one of the best performing methods for KPF, adopts the randomized crossover operator RCX for solution recombination. The main difference between RCX and UX is that RCX uses two probabilities p_1 and p_2 to control the transmission of common items and non-common items from the parents to the offspring, while UX always transmits common items to the offspring (non-common items are inherited with equal probability for both parents).

3.5. Exploring feasible and infeasible solution with tabu search

The proposed HESM method uses an adaptive feasible and infeasible tabu search (AFITS) to explore candidate solutions which may be feasible or infeasible. We describe the neighborhood structures, streamlining evaluation technique, and neighborhood exploration strategy of AFITS.

3.5.1. Neighborhood structures and fast neighborhood evaluation technique

AFITS relies on three neighborhoods: the add neighborhood N_a, drop neighborhood N_d and swap neighborhood N_s, which have been successfully used in a previous study (D'Ambrosio et al., 2023). To ensure a high computation efficiency, AFITS applies an incremental streamlining evaluation technique to assess each neighboring solution. Note that other local search methods based on these neighborhoods can benefit from the streamlining technique.

Add operator: For a given incumbent solution $S \subset V$, the Add operator represented by $Add(i)$ transits an item $i \in V \backslash S$ to S. To quickly compute the move value of a candidate move, AFITS uses an efficient incremental evaluation technique (such evaluation techniques have been used in local search algorithms for the quadratic multiple knapsack problem (Zhou et al., 2022) and quadratic assignment problem (Zhou et al., 2020)). The key idea is to hold a n-dimensional vector δ, where element $\delta[i]$ indicates the sum of forfeit costs between item i and all other selected items in solution S, i.e., $\delta[i] = \sum_{j \in S, j \neq i} d_{\{i,j\}}$. The move gain of an $Add(i)$ operation can then be computed in constant time as:

$$\Delta_f(Add(i)) = p_i - \delta[i] \tag{7}$$

After performing an $Add(i)$ move, δ is updated in $O(n)$ time: $\delta[j] = \delta[j] + d_{\{i,j\}}, \forall j \in V, j \neq i$.

The add neighborhood N_a contains all the Add candidate moves, whose size is obviously bounded by $O(|V \backslash S|)$.

Drop operator: The $Drop$ operator, denoted as $Drop(i)$, deletes an item i from the solution. The move value of removing an item i can be expressed as:

$$\Delta_f(Drop(i)) = -p_i + \delta[i] \tag{8}$$

Once a $Drop(i)$ move is executed, δ is updated in $O(n)$ time: $\delta[j] = \delta[j] - d_{\{i,j\}}, \forall j \in V, j \neq i$.

The size of this neighborhood N_d is evidently bounded by $O(|S|)$.

Swap operator: The $Swap$ operator, denoted by $Swap(i, j)$, switches two items $i \in S$ and $j \in V \backslash S$. For a given $Swap(i, j)$ move, its move value can be efficiently calculated by:

$$\Delta_f(Swap(i, j)) = p_j - p_i + \delta[i] - \delta[j] + d_{\{i,j\}} \tag{9}$$

Considering that a $Swap$ move can be treated as two sequential operations of a $Drop$ move, then an Add move (or an Add move, then a $Drop$ move), the vector δ can be successively updated in two times on the basis of the Add move and $Drop$ move in $O(n)$ time.

The swap neighborhood N_s has a size bounded by $O(|S| \times |V \backslash S|)$, and is usually much larger than the add neighborhood N_a and the drop neighborhood N_d.

3.5.2. Solution exploration with tabu search

Algorithm 2 Adaptive feasible and infeasible tabu search

Adaptive feasible and infeasible tabu search *in:* Input solution S, search depth sd *out:* Best feasible solution encountered S^b
if S is feasible **then**
  $S^b \leftarrow S$ /* S^b records the best solution found by AFITS */
else
  $S^b \leftarrow \emptyset$ /* An empty solution is obviously a feasible solution */
end if
$t \leftarrow 0$ /* Iteration counter */
$tl[i] \leftarrow 0$, for each $i \in V$ /* Initialize the tabu list tl */
$\beta \leftarrow 1$ /* Initialize the infeasibility control parameter */
while $t < sd$ **do**
  Choose a best admissible neighbor solution S' from the neighborhood union of N_a, N_d and N_s with respect to F
  $S \leftarrow S'$
  Renew the tabu list tl
  if All previous λ solutions are feasible **then**
    $\beta \leftarrow \beta/\tau$
  else if They are all infeasible solutions **then**
    $\beta \leftarrow \beta \times \tau$
  end if
  if $\beta < 1$ **then**
    $\beta \leftarrow 1$
  end if
  if $f(S) > f(S^b)$ and $EX(S) = 0$ **then**
    $S^b \leftarrow S$
  end if
  $t \leftarrow t + 1$
end while
Return S^b

As described in Algorithm 2, AFITS carries out a series of iterations from an input solution. In each iteration, AFITS inspects the union of the three neighborhoods N_a, N_d and N_s in $O(n^2)$ time, and chooses the best admissible neighbor solution S' on the basis of the extend evaluation function F to substitute the incumbent solution S. To avoid short-term cycling, when an item is discarded from (inserted into) the solution, it is flagged as tabu and cannot join (leave) the solution for the next tt iterations (called the tabu tenure). The tabu condition of a move is ignored only if the move results in a feasible solution whose quality is superior to any visited solution (aspiration criterion). A move is viewed as admissible if it is not flagged as tabu or it satisfies the aspiration criterion.

At the beginning of AFITS, the infeasibility control parameter β used by the function F (see Eq. (6)) is initialized to 1. Then AFITS adaptively adjusts β according to the feasibility of recently examined

Table 1
Reference algorithms for KPF, and scaling factors of the processors utilized by these algorithms, with respect to the processor used in this work.

Algorithm	Reference	Processor type	Base score	Factor
HESM	this work	Intel Xeon Silver 4310	116	1.00
CPLEX	(Cerulli et al., 2020)	Intel Xeon E5-2650 v3	105	0.91
CG	(Cerulli et al., 2020)	Intel Xeon E5-2650 v3	105	0.91
GA-CG	(Capobianco et al., 2022)	Intel Xeon E5-2650 v3	105	0.91
MA	(D'Ambrosio et al., 2023)	Intel Xeon E5-2650 v3	105	0.91

solutions. If all previous λ solutions (λ is a parameter set to 5 by experiment) are infeasible, β is increased by a factor τ (τ is a parameter set to 2 by experiment) to reduce the attractiveness of infeasible solutions and to drive the search to feasible regions. If all previous λ solutions are feasible, β is decreased by the factor τ to the search to move towards infeasible regions. During the search, if the current solution S is feasible (i.e., $EX(S) = 0$) and better than the recorded best solution S^b, S^b is updated. AFITS terminates after sd (search depth) iterations, and returns the best feasible solution S^b.

4. Computational assessment

We present a computational assessment of the HESM method by showing comparative results with the best KPF methods on benchmark instances.

4.1. Benchmark instances

The 120 benchmark instances are grouped into three sets: O, LK and MF, each containing 40 instances. The O set was introduced by Cerulli et al. (2020) and later used in (Capobianco et al., 2022). This set is characterized by the number of items $n \in \{500, 700, 800, 1000\}$, the number of random incompatible pairs $l = 6n$, and the capacity $c = 3n$. The item profits, item weights and forfeit costs are random values in $[5, 25]$, $[3, 20]$ and $[2, 15]$, respectively. For each value of n, 10 instances were randomly generated. The LK and MF instances introduced in (Capobianco et al., 2022) were generated based on the O set instances. Specifically, the LK instances use a larger capacity $c = 5n$, while the MF instances have more incompatible pairs with $l = 8n$ where the forfeit costs are random integers in $[2, 15]$. These benchmark instances are available from the authors of (Capobianco et al., 2022).

4.2. Experimental settings

The HESM was programmed in C++ language (Zhou et al., 2023) and compiled using the g++ 7.3.0 compiler with the "-O3" option. HESM was run on a computer with an Intel Xeon Silver 4310 processor (2.1 GHz) and 1 GB RAM running the Linux operating system. To assess HESM, we make comparisons

Table 2
Values of parameters tuned by the 'irace' software.

Parameter	Section	Description	Value range	Final value
γ	3.1, 3.3	population size	$\{10, 30, 70, 100, 150\}$	30
tt	3.5.2	tabu tenure	$\{5, 15, 30, 50, 100\}$	15
sd	3.5.2	search depth of tabu search	$\{3000, 5000, 7000, 10000, 15000\}$	7000

with two representative KPF heuristics shown in Table 1 as well as the results obtained using the CPLEX solver to solve the mathematical model (Eqs. (1)-(5)). The numerical results of these methods are directly compiled from the latest references on KPF (Capobianco et al., 2022; D'Ambrosio et al., 2023). To conduct a meaningful comparison of running time, we scale the running times reported on different computers into equivalent runtime required on our computer using the base score as the main indicator which is assessed by the Standard Performance Evaluation Corporation (SPEC, via www.spec.org). For the scaling purpose, Table 1 indicates the processor type used by each algorithm, relevant base score from SPEC, and corresponding scaling factors in relation to the processor used in this work serving as a basis. The processor used by HESM is thus a little faster than that utilized by the reference algorithms. Note that the time conversion is for indicative purposes only, due to the fact that the running time required by each compared method is influenced by additional factors such as data structures, programming languages and compiler options.

Given that HESM is a stochastic algorithm, we ran the algorithm independently 10 times per instance. The cutoff time t_m for each run was set to 1800 seconds.

4.3. Parameter tuning

We tune HESM's three parameters (population size γ, tabu tenure tt and search depth sd for tabu search) by an automatic tuning software named 'irace' (López-Ibáñez et al., 2016) designed for off-line parameter configuration. In the tuning experiment, we selected randomly 5 instances from the three sets of benchmarks as the training instances, and set the tuning budget to 200 runs of HESM. Table 2 shows the value range used by 'irace', and the best parameter values recommended by 'irace'.

4.4. Results and comparisons

Tables 3-5 show the computational results of HESM as well as the comparisons instance-by-instance with three reference methods including the CPLEX solver (Cerulli et al., 2020), CG (Cerulli et al., 2020) and GA-CG (Capobianco et al., 2022) on the three sets of benchmarks. Note that MA (D'Ambrosio et al., 2023) does not report detailed numerical results on each instance. Column 'Ins.' gives the instance name and 'f_{bk}' indicates the best-known solution reported in the literature. Columns 'f_{best}' and '$AvgT(s)$' show respectively the best objective value and the average running time in seconds to find the final solutions across several independent runs. Column '$T(s)$' presents the running time for CG and GA-CG for one execution. The running times of CPLEX are not provided in the corresponding literature,

Table 3
Comparison between HESM and the reference methods on the O set benchmarks (best results in bold).

		CPLEX	CG		GA-CG		HESM			
Ins.	f_{bk}	f_{best}	f_{best}	$T(s)$	f_{best}	$T(s)$	f_{best}	f_{avg}	σ	$AvgT(s)$
O500_01	2626	2626	2510	1.35	2568	165.32	**2629**	2627.20	0.60	127.32
O500_02	**2660**	**2660**	2556	1.31	2621	159.89	**2660**	2658.40	0.80	314.76
O500_03	**2516**	**2516**	2400	1.28	2478	162.82	**2516**	2516.00	0.00	125.14
O500_04	**2556**	**2556**	2441	1.29	2515	167.60	**2556**	2556.00	0.00	141.10
O500_05	2625	2625	2502	1.33	2582	166.19	**2633**	2633.00	0.00	526.39
O500_06	**2615**	**2615**	2500	1.26	2557	155.08	**2615**	2614.20	2.40	342.14
O500_07	2627	2627	2470	1.31	2602	163.76	**2632**	2632.00	0.00	233.44
O500_08	**2556**	**2556**	2471	1.27	2522	162.94	**2556**	2556.00	0.00	134.27
O500_09	**2613**	**2613**	2524	1.33	2572	169.97	**2613**	2613.00	0.00	273.92
O500_10	**2558**	**2558**	2439	1.28	2537	165.40	**2558**	2558.00	0.00	131.50
O700_01	**3589**	**3589**	3448	3.56	3511	517.19	3588	3588.00	0.00	639.18
O700_02	3422	3422	3253	3.51	3359	510.37	**3424**	3423.60	0.66	1149.59
O700_03	**3679**	**3679**	3449	3.47	3634	507.37	3671	3669.20	1.17	674.05
O700_04	**3664**	**3664**	3512	3.56	3605	495.49	**3664**	3663.10	0.54	1021.74
O700_05	**3647**	**3647**	3457	3.65	3619	494.02	**3647**	3644.90	2.59	962.37
O700_06	3596	3596	3447	3.53	3553	498.85	**3598**	3598.00	0.00	1294.95
O700_07	**3542**	**3542**	3319	3.61	3446	513.84	3541	3541.00	0.00	471.32
O700_08	**3619**	**3619**	3389	3.53	3545	501.05	3613	3607.50	3.64	1112.01
O700_09	**3553**	**3553**	3363	3.80	3487	508.08	**3553**	3546.30	3.23	1045.86
O700_10	**3652**	**3652**	3462	3.68	3594	515.93	3650	3648.50	0.67	812.94
O800_01	4184	4184	4024	5.53	4125	789.00	**4187**	4183.70	2.69	1192.04
O800_02	4065	4065	3827	5.30	4006	793.28	**4067**	4066.40	0.49	827.99
O800_03	4102	4102	3886	5.58	4018	796.44	**4109**	4106.70	2.24	1317.70
O800_04	4051	4051	3850	5.60	3960	828.26	**4057**	4051.40	3.90	1204.98
O800_05	4085	4085	3900	5.53	4041	803.99	**4090**	4088.60	1.36	1253.60
O800_06	4249	4249	4084	5.56	4184	792.54	**4250**	4246.40	2.50	1357.24
O800_07	4121	4121	3897	5.71	4021	803.40	**4128**	4121.80	5.56	1320.69
O800_08	**4063**	**4063**	3859	5.63	4019	782.48	4061	4057.60	5.12	1376.91
O800_09	4080	4080	3853	5.54	4017	792.30	**4082**	4079.00	3.38	1089.25
O800_10	4124	4124	4050	5.60	4074	800.30	**4130**	4128.90	1.81	976.61
01000_01	4927	4927	4655	12.49	4834	1597.80	**4935**	4930.60	2.73	1408.30
01000_02	4966	4966	4756	12.47	4893	1564.61	**4982**	4976.40	4.61	1211.55
01000_03	5171	5171	4897	12.60	5070	1645.26	**5177**	5170.60	4.92	1575.59
01000_04	**5141**	**5141**	4916	12.36	5065	1553.59	5138	5131.40	5.10	1453.31
01000_05	5134	5134	4935	12.48	5049	1526.05	**5138**	5136.60	0.80	1218.50
01000_06	**5082**	**5082**	4858	12.27	4951	1589.67	5079	5075.60	2.20	1559.52
01000_07	5100	5100	4876	12.40	5033	1585.42	**5119**	5116.10	1.92	1382.29
01000_08	5178	5178	4916	12.47	5071	1572.44	**5193**	5188.80	2.93	1471.44
01000_09	**5108**	**5108**	4890	12.38	5011	1640.44	5104	5099.20	2.40	1531.58
01000_10	5178	5178	4998	12.56	5080	1590.57	**5184**	5176.60	3.85	1409.39
#Best	19	19	0		0		**31**			
#Improve		0	0		0		21	16		
#Match		40	0		0		10	5		
Average	3850.60	3850.60	3670.98	5.72	3785.73	763.73	3853.18	3850.66	1.92	941.81

Table 4
Comparison between HESM and the reference methods on the LK set benchmarks (best results in bold).

		CPLEX	CG		GA-CG		HESM			
Ins.	f_{bk}	f_{best}	f_{best}	$T(s)$	f_{best}	$T(s)$	f_{best}	f_{avg}	σ	$AvgT(s)$
LK500_01	2712	2712	2528	1.36	2649	238.95	**2727**	2727.00	0.00	41.48
LK500_02	2729	2729	2580	1.31	2666	222.13	**2740**	2740.00	0.00	144.43
LK500_03	**2639**	**2639**	2436	1.30	2587	209.55	**2639**	2639.00	0.00	95.05
LK500_04	**2665**	**2665**	2474	1.31	2600	222.91	**2665**	2665.00	0.00	165.36
LK500_05	2686	2686	2509	1.38	2615	218.82	**2695**	2694.90	0.30	268.79
LK500_06	2746	2746	2542	1.37	2707	224.40	**2755**	2754.60	0.49	157.43
LK500_07	2689	2689	2484	1.31	2659	226.16	**2708**	2706.40	0.80	184.50
LK500_08	**2681**	**2681**	2574	1.31	2644	219.99	**2681**	2680.00	1.10	229.34
LK500_09	2652	2652	2524	1.31	2615	223.69	**2654**	2654.00	0.00	82.65
LK500_10	2665	2665	2524	1.32	2619	227.75	**2675**	2675.00	0.00	174.84
LK700_01	3757	3757	3487	3.82	3678	682.19	**3761**	3758.10	2.21	412.18
LK700_02	3611	3611	3288	3.58	3535	695.62	**3613**	3610.00	4.56	590.98
LK700_03	3824	3824	3510	3.60	3799	691.79	**3835**	3829.20	2.40	948.28
LK700_04	3835	3835	3550	3.67	3749	688.94	**3844**	3840.80	1.94	842.19
LK700_05	3823	3823	3585	3.62	3750	639.98	**3851**	3844.40	3.10	423.23
LK700_06	3707	3707	3526	3.55	3651	673.10	**3719**	3714.70	3.35	713.70
LK700_07	3676	3676	3326	3.55	3624	704.14	**3685**	3682.00	3.55	488.41
LK700_08	3762	3762	3538	3.72	3735	667.34	**3800**	3797.40	2.69	810.64
LK700_09	**3651**	**3651**	3393	3.62	3603	702.65	3648	3646.70	1.35	419.62
LK700_10	**3832**	**3832**	3499	3.65	3736	697.56	3825	3819.30	3.69	618.18
LK800_01	4298	4298	4084	5.66	4213	1101.85	**4314**	4313.00	1.34	831.33
LK800_02	4201	4201	3875	5.54	4129	1021.15	**4225**	4219.90	3.11	660.07
LK800_03	4251	4251	3912	5.59	4128	1115.71	**4266**	4257.30	4.78	822.43
LK800_04	4209	4209	3924	5.64	4146	1116.08	**4223**	4221.80	2.32	852.91
LK800_05	4176	4176	3971	5.66	4145	1060.66	**4239**	4236.50	1.63	651.49
LK800_06	4417	4417	4137	5.57	4343	1077.12	**4428**	4425.70	1.19	612.14
LK800_07	4284	4284	3920	5.63	4175	1118.76	**4302**	4292.10	5.77	1014.68
LK800_08	4144	4144	3913	5.48	4120	1138.76	**4162**	4154.90	2.66	1002.61
LK800_09	4271	4271	3976	5.64	4225	1087.30	**4310**	4307.50	2.16	1001.06
LK800_10	4277	4277	4078	5.68	4192	1097.25	**4288**	4287.00	0.63	790.81
LK1000_01	5147	5147	4690	12.31	5037	2214.65	**5183**	5168.00	6.12	1171.42
LK1000_02	5156	5156	4841	12.59	5025	2122.69	**5161**	5157.20	4.53	1444.96
LK1000_03	5340	5340	4940	12.79	5281	2125.16	**5377**	5360.50	6.89	1275.02
LK1000_04	5421	5421	5099	12.80	5323	2087.53	**5435**	5429.00	3.71	1063.41
LK1000_05	5254	5254	5018	12.82	5209	2138.83	**5297**	5284.30	7.38	1452.03
LK1000_06	5345	5345	4964	12.79	5234	2029.42	**5352**	5345.90	5.97	1179.19
LK1000_07	5244	5244	4971	12.59	5139	2159.79	**5276**	5260.70	5.88	1380.15
LK1000_08	5323	5323	5020	12.59	5245	2005.32	**5393**	5388.30	3.49	1106.53
LK1000_09	5295	5295	5031	12.65	5225	2042.03	**5351**	5333.70	10.48	1421.95
LK1000_10	5362	5362	5119	12.72	5318	2060.29	**5404**	5394.50	7.68	1142.97
#Best	5	5	0		0		**38**			
#Improve		0	0		0		35	34		
#Match		40	0		0		3	2		
Average	3993.93	3993.93	3734.00	5.81	3926.83	1024.95	4012.65	4007.91	2.98	717.21

Table 5
Comparison between HESM and the reference methods on the MF set benchmarks (best results in bold).

		CPLEX	CG		GA-CG		HESM			
Ins.	f_{bk}	f_{best}	f_{best}	$T(s)$	f_{best}	$T(s)$	f_{best}	f_{avg}	σ	$AvgT(s)$
MF500_01	**2368**	**2368**	2223	1.70	2305	253.76	**2368**	2368.00	0.00	502.39
MF500_02	2310	2310	2027	1.57	2300	256.48	**2319**	2318.80	0.60	1209.61
MF500_03	**2284**	**2284**	2108	1.58	2229	254.26	**2284**	2283.90	0.30	559.90
MF500_04	2259	2259	2098	1.61	2228	247.65	**2273**	2273.00	0.00	763.32
MF500_05	2321	2321	2199	1.62	2272	263.74	**2327**	2327.00	0.00	38.13
MF500_06	2316	2316	2230	1.64	2283	254.87	**2327**	2327.00	0.00	1189.01
MF500_07	2288	2288	2129	1.59	2207	250.01	**2294**	2294.00	0.00	54.29
MF500_08	2201	2201	2052	1.61	2161	248.32	**2215**	2215.00	0.00	16.19
MF500_09	2259	2259	2034	1.59	2219	256.47	**2272**	2272.00	0.00	33.11
MF500_10	2305	2305	2069	1.57	2285	254.06	**2319**	2317.20	0.60	153.08
MF700_01	3127	3127	2915	4.91	3050	752.68	**3130**	3127.80	3.03	1182.09
MF700_02	3038	3038	2775	4.79	2966	735.33	**3059**	3057.80	0.98	818.12
MF700_03	3197	3197	3000	5.01	3162	758.40	**3224**	3223.70	0.46	616.33
MF700_04	3233	3233	2994	4.92	3176	753.76	**3247**	3244.90	1.37	762.56
MF700_05	3238	3238	3035	4.78	3134	723.80	**3246**	3246.00	0.00	167.56
MF700_06	3129	3129	2901	4.81	3095	733.72	**3133**	3132.70	0.46	599.53
MF700_07	3015	3015	2668	4.70	2948	745.79	**3052**	3050.30	2.45	913.92
MF700_08	3166	3166	2924	4.78	3096	724.56	**3177**	3175.80	1.47	965.92
MF700_09	3186	3186	3017	4.96	3146	753.78	**3219**	3218.60	0.49	1037.33
MF700_10	3203	3203	2940	4.88	3154	726.75	**3216**	3213.70	2.33	724.19
MF800_01	3691	3691	3428	7.87	3639	1132.03	**3702**	3701.30	0.64	901.79
MF800_02	3711	3711	3489	7.73	3647	1114.68	**3735**	3734.20	1.17	962.19
MF800_03	3605	3605	3398	7.65	3566	1195.34	**3684**	3682.60	1.80	841.44
MF800_04	3490	3490	3203	7.63	3391	1097.15	**3529**	3526.30	1.79	1410.84
MF800_05	3741	3741	3483	7.82	3704	1097.60	**3754**	3754.00	0.00	396.85
MF800_06	3772	3772	3502	7.77	3697	1106.53	**3782**	3782.00	0.00	415.73
MF800_07	3683	3683	3442	7.93	3611	1100.25	**3688**	3686.70	0.90	947.20
MF800_08	3575	3575	3334	7.51	3524	1118.76	**3612**	3606.20	5.96	1007.44
MF800_09	3593	3593	3340	7.57	3521	1082.35	**3608**	3604.90	2.02	791.96
MF800_10	3633	3633	3336	7.54	3531	1114.64	**3643**	3641.00	2.28	741.42
MF1000_01	4450	4450	4112	15.25	4393	2200.10	**4470**	4466.20	2.68	816.85
MF1000_02	4408	4408	4133	15.28	4337	2053.81	**4471**	4464.90	3.91	1514.09
MF1000_03	4577	4577	4268	15.26	4572	2102.17	**4645**	4639.50	2.80	1332.18
MF1000_04	4564	4564	4122	15.60	4468	2107.51	**4592**	4589.20	1.89	1324.51
MF1000_05	4468	4468	4210	15.29	4399	2106.83	**4531**	4528.40	4.00	1329.76
MF1000_06	4569	4569	4258	15.63	4512	2073.79	**4604**	4603.20	1.17	1326.59
MF1000_07	4578	4578	4321	15.54	4485	2162.87	**4612**	4609.50	1.75	1344.36
MF1000_08	4486	4486	4167	15.63	4421	2123.27	**4572**	4559.60	4.50	1518.82
MF1000_09	4631	4631	4430	15.55	4570	2036.23	**4643**	4636.80	2.52	1005.37
MF1000_10	4660	4660	4345	15.58	4559	2096.12	**4678**	4674.70	2.97	1388.05
#Best	2	2	0		0		**40**			
#Improve		0	0		0		38	38		
#Match		40	0		0		2	1		
Average	3408.20	3408.20	3166.48	7.41	3349.08	1054.26	3431.40	3429.46	1.48	840.60

Table 6
Summarized results between HESM and four reference algorithms on the O, LK and MF sets of instances. The best results are marked in bold.

Inst.group		CPLEX	CG		GA-CG		MA		HESM	
Type	n	sol	sol	$time(s)$	sol	$time(s)$	sol	$time(s)$	sol	$time(s)$
O	500	2595.20	2481.30	1.30	2555.40	163.90	2579.10	14.38	**2596.80**	235.00
	700	**3596.30**	3409.90	3.59	3535.30	506.22	3563.80	56.05	3594.90	918.40
	800	4112.40	3923.00	5.56	4046.50	798.20	4075.10	111.30	**4116.10**	1191.70
	1000	5098.50	4869.70	12.45	5005.70	1586.59	5057.30	312.24	**5104.90**	1422.15
LK	500	2686.40	2517.50	1.33	2636.10	223.44	2676.40	33.64	**2693.90**	154.39
	700	3747.80	3470.20	3.64	3686.00	684.33	3731.50	139.39	**3758.10**	626.74
	800	4252.80	3979.00	5.61	4181.60	1093.46	4241.30	318.64	**4275.70**	823.95
	1000	5288.70	4969.30	12.67	5203.60	2098.57	5279.40	719.10	**5322.90**	1263.76
MF	500	2291.10	2116.90	1.61	2248.90	253.96	2279.40	40.20	**2299.80**	451.90
	700	3153.20	2916.90	4.85	3092.70	740.86	3142.40	171.98	**3170.30**	778.76
	800	3649.40	3395.50	7.70	3583.10	1115.93	3638.80	236.00	**3673.70**	841.69
	1000	4539.10	4236.60	15.46	4471.60	2106.27	4547.50	551.51	**4581.80**	1290.06

Table 7
Results of the Wilcoxon signed-rank test for HESM and the reference methods on the three set of instances, with a significance level of 0.05.

Instance set	Comparison	$\mathrm{R}^{+}_{\mathrm{best}}$	$\mathrm{R}^{-}_{\mathrm{best}}$	p-value
O Set (40 instances)				
	HESM vs. CPLEX	21	9	6.52e-3
	HESM vs. CG	40	0	3.56e-8
	HESM vs. GA-CG	40	0	3.56e-8
LK Set (40 instances)				
	HESM vs. CPLEX	35	2	2.45e-7
	HESM vs. CG	40	0	3.57e-8
	HESM vs. GA-CG	40	0	3.56e-8
MF Set (40 instances)				
	HESM vs. CPLEX	38	0	7.67e-8
	HESM vs. CG	40	0	3.57e-8
	HESM vs. GA-CG	40	0	3.56e-8

while the CPLEX solver was run under the time limit of 3 hours for each execution. Note that for each instance, the reference algorithms CPLEX, CG, GA-CG and MA were performed only once, while we ran HESM 10 independent times given its stochastic nature. Therefore, in addition to the best objective value and the running time, we also report the average objective value (‘f_{avg}’) and the standard deviation (‘σ’). Row ‘#Best’ records the number of cases for which a corresponding method yields the best results among all the compared approaches. Row ‘#Improve’ (‘#Match’) summarizes the number of instances that an approach improves (matches) the best-known solution from the literature. Row ‘*Average*’ gives the average values of each indicator across each set of test graphs.

Table 6 shows the average results of each instance group. For each algorithm, we report the average objective values ('sol') and the average running time in seconds ('$time(s)$') across each instance group.

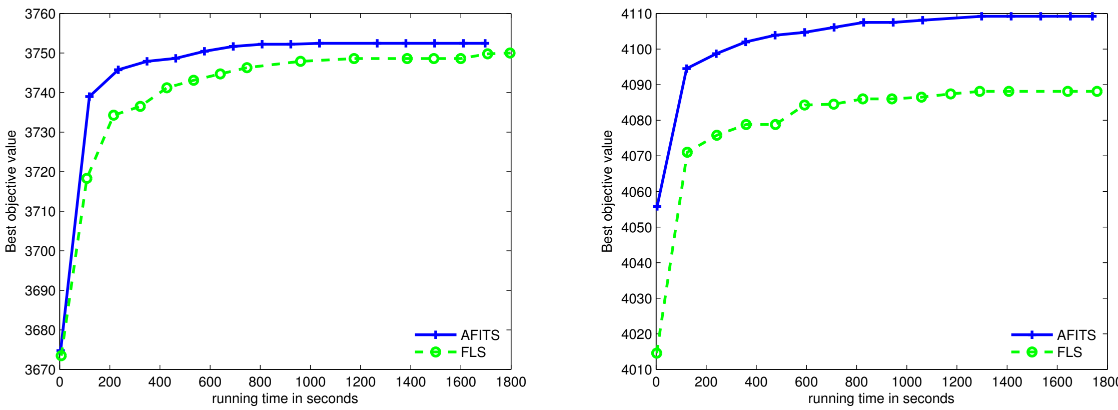
One can observe from Tables 3-5 that with respect to the best objective value, HESM produces the best result for 31 (38, 40) cases out of the 40 O instances (LK instances, MF instances), while CPLEX, CG and GA-CG obtain the best result for 19 (5, 2), 0 (0, 0) and 0 (0, 0) cases, respectively. In particular, HESM is able to find 94 (21, 35 and 38 cases for the O set, LK set and MF set respectively) improved best-known solutions out of the 120 benchmark instances, while missing the best-known solution for only 11 instances (9 O cases and 2 LK cases). One also notices that the average objective values obtained by HESM are better than the best objective values of CPLEX, CG and GA-CG on most instances, especially better than CG and GA-CG on all the instances and better than CPLEX on 88 instances over all the 120 benchmarks. Moreover, HESM yields small standard deviations on all test graphs, which discloses its robustness.

Concerning the runtime required by each compared algorithm to reach its best result, the greedy method CG is by far the fastest, but its solutions are much worse compared to those of HESM and GA-CG. For HESM and GA-CG, HESM typically needs comparable or less time to find solutions of better quality. We can also observe from Table 6 that HESM reports the best results on the average solution values for all the instance groups except the case O700 group.

To verify whether there are statistical differences between HESM and each compared algorithm, Table 7 presents the results of the popular Wilcoxon signed-rank test with a significance level of 0.05. Column $\mathrm{R}^{+}_{\mathrm{best}}$ indicates the sum of ranks for the cases where HESM outperforms the compared algorithm with respect to the best objective value, while $\mathrm{R}^{-}_{\mathrm{best}}$ records the sum of ranks for the opposite cases. Table 7 shows that HESM performs statistically significant better than the compared methods including CPLEX, CG and GA-CG on the three sets of benchmarks with p-values < 0.05. These observations reveal the advantages of HESM in solution quality and computational efficiency, compared with the current best-performing KPF approaches.

Considering that KPF is a special case of KPFS, recently introduced in (D'Ambrosio et al., 2023), we slightly adapted the proposed HESM algorithm for tackling KPFS by only modifying the incremental streamlining evaluation method described in Section 3.5.1, and kept other HESM ingredients unchanged. Specifically, for each neighborhood, once a neighboring solution satisfying that the number of items allowed for a forfeit set s_i ($s_i = \{0, 1, ..., |n_i|\}$) is violated where $|n_i|$ is the number of forfeit sets of item i belonging to, a forfeit cost fc_i is added to (or removed from) the objective value of the neighboring solution if an item i is added to (or removed from) the knapsack. We report the comparative results between HESM and the compared methods including the CPLEX solver and MA (D'Ambrosio et al., 2023) in Table 10 of the Appendix. We observe from Table 10 that HESM shows competitive performance compared to the reference algorithms.

The studied KPF problem is also related to the classic quadratic knapsack problem (QKP). Indeed, if two items i and j are chosen simultaneously, an extra non-negative profit p_{ij} is earned for QKP, while a non-positive profit (forfeit cost) p_{ij} is incurred for KPF. We ran a state-of-the-art QKP heuristic method, i.e., IHEA (Chen and Hao, 2017) with the source code shared by the authors under the experimental environment shown in Section 4.2, to evaluate its performance on the KPF benchmarks. In addition, we ran the IBM ILOG CPLEX 12.8.0 solver to solve the mathematical model (1)-(5) under a long cutoff time of 8 hours and 16 GB RAM, to get some insights about the difficulty for solving the KPF instances. Tables 11-13 of the Appendix present the comparative results between HESM and the compared methods including CPLEX and IHEA (Chen and Hao, 2017) by mainly providing the objective values and



(a) MF800_05 (b) O800_07

Fig. 1. Convergence curves of AFITS and its variant FLS on two graphs MF800_05 and O800_07.

percentage gaps from the obtained objective value to the upper bound obtained by CPLEX. We can observe that the O set benchmarks seem easier to solve than LK and MF benchmarks by providing much smaller percentage gap from the solution to the upper bound. In addition, CPLEX finds the optimal solutions for 4 instances of the O set, while there is no optimal solution found for the LK and MF instances. The relatively high average percentage gaps (approximately 10% for LK and MF benchmarks, and 2.5% for O set instances) from the solution obtained by the compared methods to the upper bound, also show to some extent that the instances are not easy to solve, especially for LK and MF instances. The proposed HESM algorithm and IHEA (Chen and Hao, 2017) show competitive performance if we compare the best objective value and the solution value returned by CPLEX. When comparing HESM and IHEA, we can find that HESM outperforms IHEA for most instances of the three benchmark sets in terms of the best and average performance, although HESM uses a little more running time to reach the final solution.

5. Discussions

This section presents additional experiments to understand some key important components of the HESM algorithm: joint exploration of feasible and infeasible solutions and population-based framework. These experiments were carried out on 20 randomly selected instances from the three benchmark sets.

5.1. Advantage of exploring both feasible and infeasible regions

As depicted in Section 3.5, HESM employs the AFITS procedure for local optimization, which examines both feasible and infeasible regions. To assess the effect of this mixed search strategy, we conducted an experiment to compare AFITS with a variant (denoted as FLS) that visits only feasible solutions. To

Table 8
Comparison between AFITS and its a variant FLS that limits to explore feasible regions on the 20 randomly chosen instances. The best results are indicated in bold.

	FLS			AFITS		
Ins.	f_{best}	f_{avg}	$Avg\ T(s)$	f_{best}	f_{avg}	$Avg\ T(s)$
O500_02	**2658**	2656.50	600.78	**2658**	**2657.30**	265.41
O500_04	2534	2529.60	928.65	**2556**	**2555.10**	736.89
O500_07	2626	2617.70	963.25	**2627**	**2625.50**	925.21
O700_04	3646	3638.70	822.20	**3657**	**3654.80**	1153.80
O700_10	3632	3625.30	903.80	**3645**	**3643.00**	999.16
O800_07	4097	4091.30	758.21	**4112**	**4104.90**	844.90
O800_08	4044	4040.20	846.49	**4050**	**4047.90**	679.46
O1000_03	5131	5115.20	1281.20	**5152**	**5138.40**	1402.51
O1000_05	5092	5085.30	784.75	**5116**	**5112.30**	960.05
LK500_01	**2727**	**2727.00**	56.30	**2727**	**2727.00**	286.15
LK500_04	**2663**	**2663.00**	76.60	**2663**	**2663.00**	419.40
LK700_01	3756	**3751.80**	695.89	**3757**	3751.30	1160.21
LK800_07	4284	4275.90	858.79	**4285**	**4277.20**	1057.23
LK1000_05	**5265**	**5258.90**	789.35	5264	5255.20	984.57
MF500_01	**2367**	**2367.00**	368.35	**2367**	**2367.00**	178.68
MF500_05	**2327**	**2327.00**	23.44	**2327**	**2327.00**	69.71
MF700_06	**3132**	**3132.00**	484.51	**3132**	**3132.00**	577.08
MF800_02	3730	3725.00	808.57	**3731**	**3728.00**	965.66
MF800_05	**3754**	3751.90	649.52	3753	**3752.10**	719.01
MF1000_08	**4547**	**4537.80**	1085.95	4539	4532.60	1020.63
#Best	9	8		**17**	**17**	
p-value	0.02	0.01				
Average	3600.60	3595.86	689.33	**3605.9**	**3602.58**	770.29

escape from local optima, FLS and AFITS were performed in a multi-restart manner until the cutoff time was reached. Both AFITS and FLS were performed under the experimental conditions given in Section 4.2.

Table 8 shows the experimental results. One observes from Table 8 that AFITS outperforms FLS both in terms of the best and average results (17 and 17 cases for AFITS against 9 and 8 cases for FLS). Moreover, AFITS and FLS exhibit similar average running times to find their best solutions across the 20 test instances. The statistically significant differences between FLS and AFITS for the best and average performances are confirmed by the small p-values (< 0.05).

To show the evolution of the best solution found along the time, we provide the convergence curves of AFITS and FLS on two randomly selected instances MF800_05 and O800_07 in Fig. 1, where the X-axis records the running time in seconds and the Y-axis represents the best objective value. We can observe from Fig. 1 that AFITS finds always better solutions than FLS while requiring less time. Similar results are observed in other instances. This experiment clearly shows the advantage of mixing feasible and the infeasible searches.

5.2. Effect of population framework

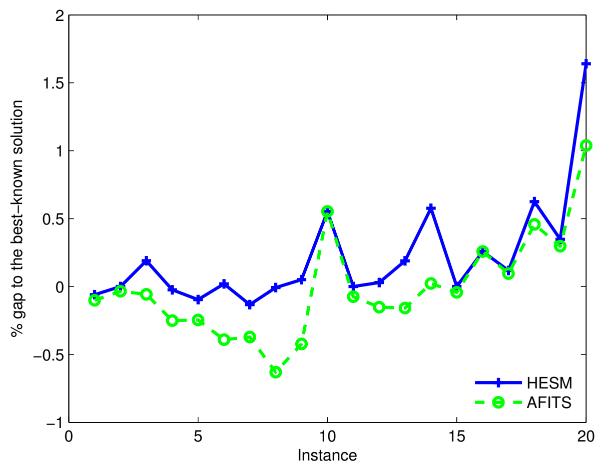
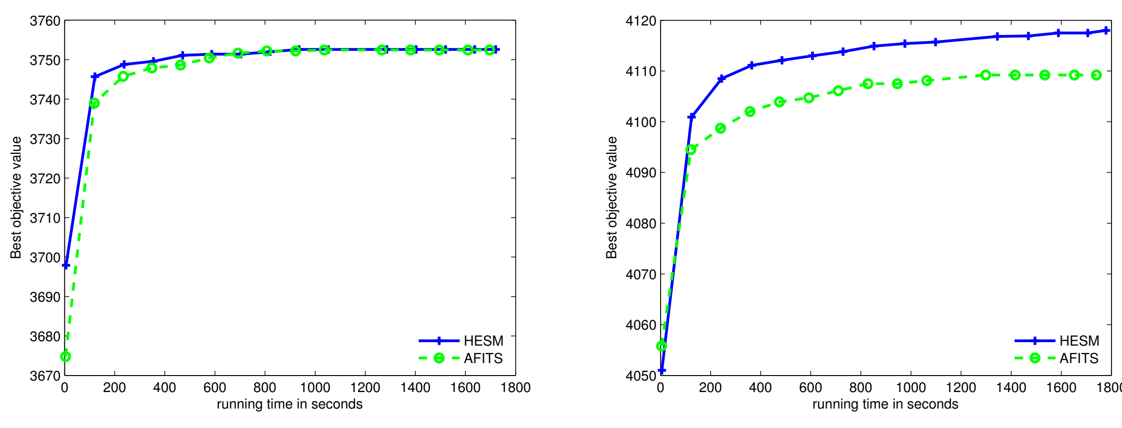


Fig. 2. Comparison between HESM and its underlying local search algorithm AFITS on 20 randomly selected instances.

To assess the effect of the population framework, we compared HESM with its underlying local search method without the crossover operator, i.e., the AFITS method. Both HESM and AFITS were run independently 10 times on each of the 20 randomly selected instances as shown in Table 8. To be fair, AFITS was executed in a multi-restart manner until it reached the cutoff time.

We plotted the average objective values of the two compared methods in Fig. 2, where the Y-axis



(a) MF800_05

(b) O800_07

Fig. 3. Convergence curves of HESM and AFITS on the graphs MF800_05 and O800_07.

Table 9
Similarity between 100 high quality solutions.

Ins.	n	sel_{max}	sel_{avg}	sel_{min}	sim_{max}	sim_{avg}	sim_{min}
O500_02	500	154	149.50	146	150	130.86	101
O500_04	500	151	146.23	140	145	125.38	89
O500_07	500	156	150.92	146	150	117.93	87
O700_04	700	211	207.18	202	205	173.96	137
O700_10	700	209	203.75	200	201	177.20	144
O800_07	800	246	237.17	230	231	182.26	137
O800_08	800	246	240.20	234	236	195.30	153
O1000_03	1000	308	300.14	290	297	239.69	184
O1000_05	1000	302	297.48	294	289	246.08	199
LK500_01	500	171	166.33	160	167	145.88	118
LK500_04	500	174	167.43	162	167	135.82	90
LK700_01	700	243	237.08	230	237	197.55	143
LK800_07	800	278	271.32	264	266	212.68	163
LK1000_05	1000	352	339.70	331	333	273.48	223
MF500_01	500	143	138.58	134	139	108.30	72
MF500_05	500	142	138.46	135	137	111.06	79
MF700_06	700	197	191.96	187	192	170.37	131
MF800_02	800	221	214.82	208	210	159.39	120
MF800_05	800	227	220.63	216	221	191.38	151
MF1000_08	1000	276	268.63	261	265	203.03	139

represents the percentage gap between the average results acquired by each approach and the best-known solutions in the literature. These results disclose the importance of the population-based framework to HESM's overall performance. The convergence curves from Fig. 3 further suggest that HESM consumes less running time to obtain better solutions than AFITS for both instances.

5.3. Motivation behind the adopted crossover

To further understand why the adopted uniform crossover operator is useful for the algorithm, we analyze the structural similarity of high quality solutions. For this, we select 20 instances and run HESM to find (and record) 100 different high quality solutions for each instance. Then we calculate the pairwise similarity of the 100 solutions as follows. Given two solutions S^a and S^b, their similarity is measured as the number of commonly shared items: $sim(S^a, S^b) = |S^a \cap S^b|$.

Table 9 shows the structural information found among the 100 solutions for each instance, where sel_{max}, sel_{min} and sel_{avg} indicate the maximal, average, and minimal number of selected items among 100 solutions, and sim_{max}, sim_{min} and sim_{avg} are the maximal, average, and minimal similarity between those solutions. We observe that, the average similarity between these high quality solutions is very high, which suggests that a large number of shared items might form the kernel of an optimal solution. By inheriting the shared items of parent solution to offspring, the adopted uniform crossover naturally conserves the kernel information of high quality, favoring the discovery of still better or even optimal solutions.

6. Conclusions

The knapsack problem with forfeits is a suitable model for a number of real-world applications that cannot be formulated by the conventional knapsack problem due to the presence of incompatible items. In this work, we presented the hybrid evolutionary search method HESM for KPF that integrates the uniform crossover and a tabu search procedure exploring both feasible and infeasible regions. In addition, a fast evaluation technique is proposed to accelerate the examination of three neighborhoods.

Experimental results on three sets of 120 KPF test graphs in the literature revealed the dominance of the algorithm over the existing state-of-the-art methods. In particular, HESM discovered 94 new lower bounds and matched 15 other best-known results. The effects of the main algorithmic components were also assessed.

Since the algorithm presented in this work is a heuristic approach, the gaps of its solutions to the optimal solutions remain unknown for difficult instances. Therefore, it is worth investigating exact algorithms in future studies to obtain optimal solutions or tight bounds. Moreover, the basic ideas of the proposed algorithm would be useful for developing effective algorithms for other related problems with incompatibility constraints.

Acknowledgments

We thank the reviewers for their insightful comments and suggestions. This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 72201053, 71971052, 72331003, 72122006), Major Program of the National Social Science Foundation of China (Grant No. 23&ZD050), Natural Science Foundation of Liaoning Province (Grant No. 2023MS092), China Postdoctoral Science Foundation (Grant No. 2021M700716), and Fundamental Research Funds for the Central Universities (Grant Nos. N2406012, N2206009, N2406005, N2306008).

References

Ben Salem, M., Hanafi, S., Taktak, R., Ben Abdallah, H., 2017. Probabilistic tabu search with multiple neighborhoods for the disjunctively constrained knapsack problem. *RAIRO-Operations Research-Recherche Opérationnelle* 51, 3, 627–637.

Bettinelli, A., Cacchiani, V., Malaguti, E., 2017. A branch-and-bound algorithm for the knapsack problem with conflict graph. *INFORMS Journal on Computing* 29, 3, 457–473.

Bravo, M., Rojas, L.P., Parada, V., 2019. An evolutionary algorithm for the multi-objective pick-up and delivery pollution-routing problem. *International Transactions in Operational Research* 26, 1, 302–317.

Cacchiani, V., Iori, M., Locatelli, A., Martello, S., 2022a. Knapsack problems-An overview of recent advances. Part I: Single knapsack problems. *Computers & Operations Research* 143, 105692.

Cacchiani, V., Iori, M., Locatelli, A., Martello, S., 2022b. Knapsack problems-An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research* 143, 105693.

Capobianco, G., D'Ambrosio, C., Pavone, L., Raiconi, A., Vitale, G., Sebastiano, F., 2022. A hybrid metaheuristic for the knapsack problem with forfeits. *Soft Computing* 26, 1–14.

Caprara, A., Pisinger, D., Toth, P., 1999. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing* 11, 2, 125–137.

Cerrone, C., Cerulli, R., Golden, B., 2017. Carousel greedy: A generalized greedy algorithm with applications in optimization. *Computers & Operations Research* 85, 97–112.

Cerulli, R., D'Ambrosio, C., Raiconi, A., Vitale, G., 2020. The knapsack problem with forfeits. In *Combinatorial Optimization: 6th International Symposium, ISCO 2020, Montreal, QC, Canada, May 4–6, 2020, Revised Selected Papers 6*, Springer, pp. 263–272.

Chen, Y., Hao, J.K., 2017. An iterated 'hyperplane exploration' approach for the quadratic knapsack problem. *Computers & Operations Research* 77, 226–239.

Coniglio, S., Furini, F., San Segundo, P., 2021. A new combinatorial branch-and-bound algorithm for the knapsack problem with conflicts. *European Journal of Operational Research* 289, 2, 435–455.

Constantino, O.H., Segura, C., 2022. A parallel memetic algorithm with explicit management of diversity for the job shop scheduling problem. *Applied Intelligence* 52, 1, 141–153.

D'Ambrosio, C., Laureana, F., Raiconi, A., Vitale, G., 2023. The knapsack problem with forfeit sets. *Computers & Operations Research* 151, 106093.

Ekici, A., 2021. Bin packing problem with conflicts and item fragmentation. *Computers & Operations Research* 126, 105113.

Elhedhli, S., Li, L., Gzara, M., Naoum-Sawaya, J., 2011. A branch-and-price algorithm for the bin packing problem with conflicts. *INFORMS Journal on Computing* 23, 3, 404–415.

Fampa, M., Lubke, D., Wang, F., Wolkowicz, H., 2020. Parametric convex quadratic relaxation of the quadratic knapsack problem. *European Journal of Operational Research* 281, 1, 36–49.

Fomeni, F.D., Kaparis, K., Letchford, A.N., 2022. A cut-and-branch algorithm for the quadratic knapsack problem. *Discrete Optimization* 44, 100579.

Fomeni, F.D., Letchford, A.N., 2014. A dynamic programming heuristic for the quadratic knapsack problem. *INFORMS Journal on Computing* 26, 1, 173–182.

Gurski, F., Rehs, C., 2019. Solutions for the knapsack problem with conflict and forcing graphs of bounded clique-width. *Mathematical Methods of Operations Research* 89, 411–432.

Hao, J.K., 2012. Memetic algorithms in discrete optimization. In *Handbook of memetic algorithms*. Springer, pp. 73–94.

Lai, X., Hao, J.K., Glover, F., Lü, Z., 2018. A two-phase tabu-evolutionary algorithm for the 0–1 multidimensional knapsack problem. *Information sciences* 436, 282–301.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3, 43–58.

Lu, Y., Benlic, U., Wu, Q., 2022. A hybrid evolutionary algorithm for the capacitated minimum spanning tree problem. *Computers & Operations Research* 144, 105799.

Mara, S.T.W., Kuo, R., Asih, A.M.S., 2021. Location-routing problem: a classification of recent research. *International Transactions in Operational Research* 28, 6, 2941–2983.

Martello, S., Toth, P., 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc.

Moscato, P., 1999. Memetic algorithms: A short introduction. In *New Ideas in Optimization*. McGraw-Hill, New York.

Neri, F., Cotta, C., 2012. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation* 2, 1–14.

Pferschy, U., Schauer, J., 2016. Approximation of the quadratic knapsack problem. *INFORMS Journal on Computing* 28, 2, 308–318.

Pferschy, U., Schauer, J., 2017. Approximation of knapsack problems with conflict and forcing graphs. *Journal of Combinatorial Optimization* 33, 4, 1300–1323.

Pisinger, W.D., Rasmussen, A.B., Sandvik, R., 2007. Solution of large quadratic knapsack problems through aggressive reduction. *INFORMS Journal on Computing* 19, 2, 280–290.

Rodrigues, C.D., Quadri, D., Michelon, P., Gueye, S., 2012. 0-1 quadratic knapsack problems: an exact approach based on a t-linearization. *SIAM Journal on Optimization* 22, 4, 1449–1468.

Sun, W., Hao, J.K., Wang, W., Wu, Q., 2020. Memetic search for the equitable coloring problem. *Knowledge-Based Systems* 188, 105000.

Syswerda, G., et al., 1989. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms, San Francisco, CA, USA, June 1989*, Morgan Kaufmann Publishers Inc., pp. 2–9.

Taylor, R., 2016. Approximation of the quadratic knapsack problem. *Operations Research Letters* 44, 4, 495–497.
Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & operations research* 40, 1, 475–489.
Wang, J.J., Wang, L., 2022. A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling. *Computers & Industrial Engineering* 168, 108126.
Wei, Z., Hao, J.K., 2021. A threshold search based memetic algorithm for the disjunctively constrained knapsack problem. *Computers & Operations Research* 136, 105447.
Wu, X., Che, A., 2019. A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega* 82, 155–165.
Wu, Z., Jiang, B., Karimi, H.R., 2020. A logarithmic descent direction algorithm for the quadratic knapsack problem. *Applied Mathematics and Computation* 369, 124854.
Yamada, T., Kataoka, S., Watanabe, K., 2002. Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Information Processing Society of Japan Journal* 43, 9.
Yang, Z., Wang, G., Chu, F., 2013. An effective grasp and tabu search for the 0–1 quadratic knapsack problem. *Computers & Operations Research* 40, 5, 1176–1185.
Zhang, Z., Zhu, L., Chen, Y., Guan, C., 2023. A multi-objective hybrid evolutionary search algorithm for parallel production line balancing problem including disassembly and assembly tasks. *International Transactions in Operational Research* 30, 6, 3508–3553.
Zhou, Q., Hao, J.K., Jiang, Z.Z., Wu, Q., 2023. KPFgit: The source code of the hybrid evolutionary search method for knapsack problem with forfeits. [EB/OL]. https://github.com/neteasefans/knapsack-problem-with-forfeits-/ Accessed November, 9, 2023.
Zhou, Q., Hao, J.K., Wu, Q., 2022. A hybrid evolutionary search for the generalized quadratic multiple knapsack problem. *European Journal of Operational Research* 296, 3, 788–803.
Zhou, Y., Hao, J.K., Duval, B., 2020. Frequent pattern-based search: A case study on the quadratic assignment problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52, 3, 1503–1515.

Appendix

This appendix provides 1) the comparative results between HESM and the reference algorithms including the CPLEX solver and MA (D'Ambrosio et al., 2023) on the KPFS instances, and 2) comparative results between HESM and the compared algorithms, i.e., the CPLEX solver under a time limit of 8 hours, and a state-of-the-art QKP heuristic IHEA (Chen and Hao, 2017) on the KPF instance. In Table 10, the results reported by CPLEX and MA are directly compiled from (D'Ambrosio et al., 2023). Column 'Ins. group' shows the instance scenario and type, and column 'n' gives the number of items for each type of instance. For each scenario, instance type and value of n, 10 different instances were used. Column '$Avg\ sol$' indicates the average solution value reported by CPLEX across 10 instances within the time limit of 3 hours. Column '$time(s)$' shows the running time elapsed by CPLEX, and the symbol '-' given in this column demonstrates that there is 0 instance belonging to a group is solved to optimality. Column '$Avg\ gap(\%)$' and '$AvgT(s)$' denote the average percentage gap of the best objective value to the solution reported by CPLEX and the average running time in seconds to find the final objective value for the compared algorithms. The percentage gaps of the best objective values are calculated as $(f_{sol} - f)/f_{sol}$, where f_{sol} is the solution reported by CPLEX and f is the best objective value obtained by the compared algorithms. Row '*Average*' shows the average values of each indicator across each set of instances. For Tables 11-13, column 'sol' and 'ub' show respectively the solution value and upper bound reported by CPLEX by solving the models (1)-(5) with a cutoff time of 8h and maximal allowed

Table 10
Comparative results between HESM and the reference algorithms on the KPFS benchmarks. The best results are indicated in bold.

		CPLEX		MA		HESM	
Ins. group	n	*Avg sol*	$time(s)$	*Avg gap*$(\%)$	$AvgT(s)$	*Avg gap*$(\%)$	$AvgT(s)$
scenario1_nc	300	684.00	3325.66	0.38	3.86	**-0.31**	1.31
scenario1_nc	500	561.70	-	0.14	11.95	**-2.85**	4.26
scenario1_c	300	**769.50**	-	0.10	2.59	0.56	2.86
scenario1_c	500	834.10	-	-0.99	18.96	**-5.13**	4.33
scenario1_fc	300	751.30	-	**-0.09**	3.03	0.36	1.92
scenario1_fc	500	802.60	-	-2.08	17.38	**-3.02**	3.98
scenario2_nc	300	**299.60**	357.50	0.17	1.37	**0.00**	0.72
scenario2_nc	500	**227.60**	2642.25	0.41	2.45	**0.00**	1.45
scenario2_nc	700	**186.90**	5133.72	0.31	3.24	0.05	3.13
scenario2_nc	800	**174.30**	5837.46	**0.00**	3.80	1.78	4.10
scenario2_nc	1000	**145.90**	5869.85	1.29	4.31	1.58	5.23
scenario2_c	300	443.90	3158.58	0.00	1.89	**-0.52**	0.54
scenario2_c	500	343.20	-	-0.96	3.16	**-1.43**	1.98
scenario2_fc	300	464.40	5003.56	-0.22	2.21	**-1.96**	0.64
scenario2_fc	500	403.40	-	-0.79	4.61	**-11.80**	1.13
scenario3_nc	300	**1033.30**	3.04	0.55	1.23	0.02	2.68
scenario3_nc	500	**1404.50**	2086.83	1.25	8.03	0.16	10.02
scenario3_c	300	**968.30**	369.69	0.84	1.23	4.13	6.97
scenario3_c	500	**1453.80**	-	0.99	5.16	1.16	27.00
scenario3_fc	300	**955.00**	675.73	0.32	1.54	2.83	5.20
scenario3_fc	500	**1435.60**	-	0.81	6.12	1.01	22.45
scenario4_nc	300	**908.70**	4.85	0.63	1.32	0.10	1.84
scenario4_nc	500	1178.70	119.36	0.89	13.01	**-0.06**	8.00
scenario4_nc	700	**1428.30**	4134.69	1.86	36.40	0.15	31.27
scenario4_nc	800	**1475.40**	3538.67	1.74	55.01	0.16	31.43
scenario4_nc	1000	**1544.40**	7089.20	2.02	89.64	0.19	59.01
scenario4_c	300	**906.20**	165.61	1.08	1.16	1.30	5.04
scenario4_c	500	**1321.50**	8882.10	1.01	5.91	1.14	17.07
scenario4_fc	300	**895.80**	224.82	0.68	1.23	1.08	6.01
scenario4_fc	500	**1323.40**	-	0.74	4.80	0.67	15.50
Average		844.18	-	0.44	10.55	-0.29	9.57

memory of 16 GB. For each instance, if the time limit (or allocated memory) is reached and the optimal solution is not obtained, we mark the solution with a symbol 'TL' (or 'OM'). Optimal solutions are marked by the symbol '*'. Column '$gap(\%)$' presents the percentage gap of the solution reported by CPLEX or the best objective value obtained by IHEA (Chen and Hao, 2017) and HESM to the upper bound value. Columns 'f_{best}' and f_{avg} give the best objective value and average objective value for the compared methods across 10 independent runs.

Table 11
Comparative results between HESM and the reference algorithms including CPLEX and a QKP heuristic (i.e., IHEA) on the O set benchmarks. The best results are marked in bold.

	CPLEX				IHEA				HESM			
Ins.	sol	ub	$gap(\%)$	$time(s)$	f_{best}	f_{avg}	$gap(\%)$	$AvgT(s)$	f_{best}	f_{avg}	$gap(\%)$	$AvgT(s)$
O500_01	2626(TL)	2627.21	0.05	18108.69	**2629**	**2627.60**	-0.07	586.28	**2629**	2627.20	-0.07	127.32
O500_02	**2660**(TL)	2695.80	1.33	28800.09	**2660**	2651.90	1.33	821.58	**2660**	**2658.40**	1.33	314.76
O500_03	2515(OM)	2561.30	1.81	16027.66	**2521**	2513.30	1.57	451.31	2516	**2516.00**	1.77	125.14
O500_04	**2556**(*)	2557.51	0.06	13716.09	2544	2544.00	0.53	243.83	**2556**	**2556.00**	0.06	141.10
O500_05	2624(TL)	2626.37	0.09	28004.03	**2633**	2628.60	-0.25	795.35	**2633**	**2633.00**	-0.25	526.39
O500_06	**2615**(*)	2616.99	0.08	5822.70	2612	2597.50	0.19	586.85	**2615**	**2614.20**	0.08	342.14
O500_07	2627(OM)	2680.47	1.99	10834.50	**2632**	2628.30	1.81	295.74	**2632**	**2632.00**	1.81	233.44
O500_08	**2556**(*)	2556.70	0.03	2765.88	2555	2555.00	0.07	396.66	**2556**	**2556.00**	0.03	134.27
O500_09	**2613**(*)	2615.40	0.09	6419.00	**2613**	2612.20	0.09	291.63	**2613**	**2613.00**	0.09	273.92
O500_10	2557(OM)	2596.12	1.51	21480.91	**2558**	**2558.00**	1.47	407.18	**2558**	**2558.00**	1.47	131.50
O700_01	3587(OM)	3657.42	1.93	15806.03	3586	3568.00	1.95	554.62	**3588**	**3588.00**	1.90	639.18
O700_02	**3424**(TL)	3511.05	2.48	28800.36	**3424**	3402.10	2.48	956.43	**3424**	**3423.60**	2.48	1149.59
O700_03	**3679**(TL)	3715.81	0.99	28800.19	3677	3665.20	1.04	816.91	3671	**3669.20**	1.21	674.05
O700_04	**3664**(TL)	3706.00	1.13	28800.09	3659	3648.40	1.27	794.48	**3664**	**3663.10**	1.13	1021.74
O700_05	**3647**(OM)	3704.41	1.55	20027.00	3645	3632.30	1.60	494.48	**3647**	**3644.90**	1.55	962.37
O700_06	3595(OM)	3621.52	0.73	20664.75	**3598**	3590.80	0.65	841.84	**3598**	**3598.00**	0.65	1294.95
O700_07	**3542**(TL)	3594.16	1.45	28800.14	**3542**	3534.30	1.45	372.17	3541	**3541.00**	1.48	471.32
O700_08	3609(OM)	3688.97	2.17	14985.88	**3614**	3599.50	2.03	1143.13	3613	**3607.50**	2.06	1112.01
O700_09	**3553**(TL)	3666.85	3.10	28800.34	3552	3543.90	3.13	608.28	**3553**	**3546.30**	3.10	1045.86
O700_10	3645(OM)	3729.01	2.25	11809.42	3649	3627.10	2.15	615.99	**3650**	**3648.50**	2.12	812.94
O800_01	4164(OM)	4326.14	3.75	14991.78	4181	4159.90	3.35	1021.62	**4187**	**4183.70**	3.22	1192.04
O800_02	4060(OM)	4212.00	3.61	7561.22	4064	4048.30	3.51	1177.49	**4067**	**4066.40**	3.44	827.99
O800_03	4064(OM)	4296.84	5.42	4026.53	4102	4088.10	4.53	1005.96	**4109**	**4106.70**	4.37	1317.70
O800_04	4052(OM)	4203.88	3.61	8781.02	4039	4013.60	3.92	638.29	**4057**	**4051.40**	3.49	1204.98
O800_05	4080(OM)	4218.75	3.29	14503.56	**4090**	4085.30	3.05	798.90	**4090**	**4088.60**	3.05	1253.60
O800_06	4240(OM)	4380.00	3.20	9891.25	4248	4237.00	3.01	989.81	**4250**	**4246.40**	2.97	1357.24
O800_07	4112(OM)	4292.59	4.21	16752.16	4114	4102.40	4.16	903.39	**4128**	**4121.80**	3.83	1320.69
O800_08	4057(OM)	4163.73	2.56	9299.03	**4061**	4050.40	2.47	806.80	**4061**	**4057.60**	2.47	1376.91
O800_09	4063(OM)	4194.23	3.13	8125.17	4067	4051.00	3.03	868.61	**4082**	**4079.00**	2.68	1089.25
O800_10	4129(OM)	4273.00	3.37	11194.13	4125	4117.80	3.46	811.18	**4130**	**4128.90**	3.35	976.61
01000_01	**4940**(TL)	5141.05	3.91	28800.09	4912	4880.70	4.46	1164.00	4935	**4930.60**	4.01	1408.30
01000_02	4966(OM)	5227.64	5.00	17468.33	4940	4918.60	5.50	765.55	**4982**	**4976.40**	4.70	1211.55
01000_03	5146(OM)	5407.49	4.84	10415.67	5145	5131.90	4.85	1457.54	**5177**	**5170.60**	4.26	1575.59
01000_04	5134(OM)	5327.18	3.63	14710.67	5137	5120.70	3.57	982.27	**5138**	**5131.40**	3.55	1453.31
01000_05	5127(OM)	5291.55	3.11	8431.30	5106	5097.30	3.51	1117.99	**5138**	**5136.60**	2.90	1218.50
01000_06	5052(OM)	5271.00	4.15	8367.95	5063	5027.20	3.95	801.72	**5079**	**5075.60**	3.64	1559.52
01000_07	5071(OM)	5329.54	4.85	14590.88	5096	5069.30	4.38	1141.09	**5119**	**5116.10**	3.95	1382.29
01000_08	5161(OM)	5398.52	4.40	14583.97	5171	5134.80	4.21	1060.10	**5193**	**5188.80**	3.81	1471.44
01000_09	5101(OM)	5305.07	3.85	8348.90	5103	5075.80	3.81	1096.49	**5104**	**5099.20**	3.79	1531.58
01000_10	5176(OM)	5367.10	3.56	11927.61	5171	5149.80	3.65	1148.70	**5184**	**5176.60**	3.41	1409.39
#Best	12				13	2			**35**	**39**		
Average	3844.73	3958.91	2.56	15551.12	3845.95	3832.2	2.52	795.81	3853.18	3850.66	2.37	941.81

Table 12
Comparative results between HESM and the reference algorithms including CPLEX and a QKP heuristic (i.e., IHEA) on the LK set benchmarks. The best results are marked in bold.

	CPLEX				IHEA				HESM			
Ins.	sol	ub	$gap(\%)$	$time(s)$	f_{best}	f_{avg}	$gap(\%)$	$AvgT(s)$	f_{best}	f_{avg}	$gap(\%)$	$AvgT(s)$
LK500_01	2706(TL)	2905.56	6.87	28003.04	2726	2701.60	6.18	581.70	**2727**	**2727.00**	6.15	41.48
LK500_02	2711(OM)	2961.29	8.45	14266.91	2720	2690.60	8.15	533.80	**2740**	**2740.00**	7.47	144.43
LK500_03	2586(OM)	2915.57	11.30	14242.50	2548	2548.00	12.61	48.27	**2639**	**2639.00**	9.49	95.05
LK500_04	2660(TL)	2883.71	7.76	28012.58	**2665**	2661.70	7.58	99.70	**2665**	**2665.00**	7.58	165.36
LK500_05	2689(TL)	2941.19	8.57	28800.16	2680	2673.70	8.88	95.75	**2695**	**2694.90**	8.37	268.79
LK500_06	2738(OM)	2974.65	7.96	11449.70	**2755**	2751.50	7.38	83.76	**2755**	**2754.60**	7.38	157.43
LK500_07	2678(OM)	2972.89	9.92	10069.34	2686	2685.00	9.65	288.01	**2708**	**2706.40**	8.91	184.50
LK500_08	2675(OM)	2894.23	7.57	9006.24	2677	2677.00	7.51	233.84	**2681**	**2680.00**	7.37	229.34
LK500_09	2646(OM)	2831.02	6.54	16671.14	2639	2638.00	6.78	640.97	**2654**	**2654.00**	6.25	82.65
LK500_10	2640(OM)	2925.48	9.76	13265.83	2668	2665.80	8.80	541.87	**2675**	**2675.00**	8.56	174.84
LK700_01	3726(OM)	4170.41	10.66	23984.58	3719	3697.50	10.82	693.75	**3761**	**3758.10**	9.82	412.18
LK700_02	3611(TL)	4042.56	10.68	28800.09	3560	3531.80	11.94	523.12	**3613**	**3610.00**	10.63	590.98
LK700_03	3820(OM)	4236.89	9.84	18488.13	3811	3794.30	10.05	265.84	**3835**	**3829.20**	9.49	948.28
LK700_04	3818(OM)	4225.83	9.65	16939.03	3786	3763.50	10.41	483.46	**3844**	**3840.80**	9.04	842.19
LK700_05	3826(TL)	4251.98	10.02	28313.95	3822	3805.90	10.11	320.02	**3851**	**3844.40**	9.43	423.23
LK700_06	3684(OM)	4060.66	9.28	16895.26	3696	3677.60	8.98	221.90	**3719**	**3714.70**	8.41	713.70
LK700_07	3668(OM)	4055.68	9.56	20169.14	3652	3614.40	9.95	444.88	**3685**	**3682.00**	9.14	488.41
LK700_08	3773(OM)	4240.44	11.02	13315.41	3788	3746.40	10.67	726.13	**3800**	**3797.40**	10.39	810.64
LK700_09	3643(OM)	4129.75	11.79	13636.89	3636	3611.10	11.96	900.01	**3648**	**3646.70**	11.67	419.62
LK700_10	3821(OM)	4203.07	9.09	13679.88	3789	3751.60	9.85	551.45	**3825**	**3819.30**	9.00	618.18
LK800_01	4274(TL)	4818.80	11.31	28413.25	4280	4260.80	11.18	354.57	**4314**	**4313.00**	10.48	831.33
LK800_02	4194(OM)	4762.66	11.94	23497.75	4171	4121.50	12.42	760.70	**4225**	**4219.90**	11.29	660.07
LK800_03	4236(TL)	4780.71	11.39	28800.06	4212	4182.20	11.90	789.70	**4266**	**4257.30**	10.77	822.43
LK800_04	4192(TL)	4728.82	11.35	28800.09	4149	4129.90	12.26	868.43	**4223**	**4221.80**	10.70	852.91
LK800_05	4228(TL)	4732.95	10.67	28800.08	4235	4194.40	10.52	704.95	**4239**	**4236.50**	10.44	651.49
LK800_06	4387(OM)	4901.94	10.50	22540.38	4331	4275.60	11.65	571.01	**4428**	**4425.70**	9.67	612.14
LK800_07	4272(OM)	4869.81	12.28	26535.27	4220	4171.00	13.34	409.37	**4302**	**4292.10**	11.66	1014.68
LK800_08	4125(OM)	4660.64	11.49	22528.73	4134	4083.90	11.30	346.10	**4162**	**4154.90**	10.70	1002.61
LK800_09	4297(OM)	4800.54	10.49	25322.06	4208	4177.70	12.34	1142.76	**4310**	**4307.50**	10.22	1001.06
LK800_10	4274(OM)	4790.50	10.78	18265.47	4231	4211.50	11.68	750.18	**4288**	**4287.00**	10.49	790.81
LK1000_01	5134(TL)	5936.30	13.52	28800.05	5067	4998.40	14.64	1039.34	**5183**	**5168.00**	12.69	1171.42
LK1000_02	5104(TL)	6002.85	14.97	28800.11	5057	5023.70	15.76	1229.89	**5161**	**5157.20**	14.02	1444.96
LK1000_03	5347(TL)	6180.86	13.49	28800.08	5259	5168.50	14.91	913.07	**5377**	**5360.50**	13.01	1275.02
LK1000_04	5410(TL)	6150.60	12.04	28800.06	5409	5312.10	12.06	1238.89	**5435**	**5429.00**	11.63	1063.41
LK1000_05	5266(TL)	6006.52	12.33	28800.06	5214	5137.50	13.19	1086.38	**5297**	**5284.30**	11.81	1452.03
LK1000_06	5279(OM)	6126.10	13.83	25159.99	5294	5169.10	13.58	938.77	**5352**	**5345.90**	12.64	1179.19
LK1000_07	5234(TL)	6057.87	13.60	28800.05	5190	5147.20	14.33	1507.46	**5276**	**5260.70**	12.91	1380.15
LK1000_08	5309(TL)	6138.16	13.51	28800.05	5244	5215.30	14.57	363.93	**5393**	**5388.30**	12.14	1106.53
LK1000_09	5304(TL)	6075.48	12.70	28800.03	5300	5212.20	12.76	853.50	**5351**	**5333.70**	11.92	1421.95
LK1000_10	5368(TL)	6137.00	12.53	28800.05	5341	5260.20	12.97	844.42	**5404**	**5394.50**	11.94	1142.97
#Best	0				2	0			**40**	**40**		
Average	3983.83	4487.05	10.77	22646.84	3964.23	3928.49	11.14	609.04	4012.65	4007.91	10.14	717.21

Table 13
Comparative results between HESM and the reference algorithms including CPLEX and a QKP heuristic (i.e., IHEA) on the MF set benchmarks. The best results are marked in bold.

Ins.	CPLEX				IHEA				HESM			
	sol	ub	$gap(\%)$	$time(s)$	f_{best}	f_{avg}	$gap(\%)$	$AvgT(s)$	f_{best}	f_{avg}	$gap(\%)$	$AvgT(s)$
MF500_01	2360(OM)	2565.85	8.02	17191.55	2367	2363.60	7.75	614.51	**2368**	**2368.00**	7.71	502.39
MF500_02	2280(OM)	2546.01	10.45	15159.58	2313	2308.60	9.15	909.22	**2319**	**2318.80**	8.92	1209.61
MF500_03	2277(OM)	2468.79	7.77	11537.94	2283	2276.80	7.53	472.82	**2284**	**2283.90**	7.49	559.90
MF500_04	2249(OM)	2456.52	8.45	15999.17	2266	2264.70	7.76	535.82	**2273**	**2273.00**	7.47	763.32
MF500_05	2297(OM)	2528.23	9.15	20811.09	2325	2322.20	8.04	671.84	**2327**	**2327.00**	7.96	38.13
MF500_06	2312(OM)	2525.42	8.45	19568.14	**2327**	2324.10	7.86	884.05	**2327**	**2327.00**	7.86	1189.01
MF500_07	2261(OM)	2556.47	11.56	19703.97	2291	2286.40	10.38	206.65	**2294**	**2294.00**	10.27	54.29
MF500_08	2194(OM)	2409.82	8.96	18456.23	**2215**	2199.60	8.08	758.85	**2215**	**2215.00**	8.08	16.19
MF500_09	2214(OM)	2480.25	10.73	7659.08	**2272**	2266.10	8.40	598.02	**2272**	**2272.00**	8.40	33.11
MF500_10	2291(OM)	2524.80	9.26	20353.00	**2320**	**2317.60**	8.11	173.19	2319	2317.20	8.15	153.08
MF700_01	3081(OM)	3562.08	13.51	17510.16	3127	3113.90	12.21	420.42	**3130**	**3127.80**	12.13	1182.09
MF700_02	3008(OM)	3457.53	13.00	19714.66	3058	3045.80	11.56	1108.57	**3059**	**3057.80**	11.53	818.12
MF700_03	3204(OM)	3572.71	10.32	12845.86	**3225**	3204.70	9.73	373.01	3224	**3223.70**	9.76	616.33
MF700_04	3087(OM)	3627.88	14.91	11117.17	**3247**	3225.50	10.50	948.86	**3247**	**3244.90**	10.50	762.56
MF700_05	3132(OM)	3592.27	12.81	16699.41	3218	3202.80	10.42	836.89	**3246**	**3246.00**	9.64	167.56
MF700_06	3056(OM)	3497.00	12.61	18332.98	**3133**	3127.20	10.41	490.28	**3133**	**3132.70**	10.41	599.53
MF700_07	2964(OM)	3461.27	14.37	13189.48	3048	3018.80	11.94	932.06	**3052**	**3050.30**	11.82	913.92
MF700_08	3122(OM)	3526.40	11.47	12477.92	3176	3159.70	9.94	843.41	**3177**	**3175.80**	9.91	965.92
MF700_09	3162(OM)	3553.54	11.02	19799.23	3207	3195.90	9.75	773.80	**3219**	**3218.60**	9.41	1037.33
MF700_10	3172(OM)	3560.15	10.90	16795.25	**3216**	3209.10	9.67	464.04	**3216**	**3213.70**	9.67	724.19
MF800_01	3638(OM)	4118.86	11.67	23143.39	3697	3668.20	10.24	821.71	**3702**	**3701.30**	10.12	901.79
MF800_02	3707(OM)	4053.81	8.56	28800.17	3724	3711.70	8.14	642.86	**3735**	**3734.20**	7.86	962.19
MF800_03	3643(OM)	4147.67	12.17	9372.48	3665	3637.20	11.64	758.61	**3684**	**3682.60**	11.18	841.44
MF800_04	3466(OM)	4011.33	13.59	12453.17	3519	3500.60	12.27	1034.33	**3529**	**3526.30**	12.02	1410.84
MF800_05	3738(OM)	4094.26	8.70	17401.45	3753	3747.60	8.34	480.44	**3754**	**3754.00**	8.31	396.85
MF800_06	3733(OM)	4176.76	10.62	12528.47	3765	3753.00	9.86	764.78	**3782**	**3782.00**	9.45	415.73
MF800_07	3510(OM)	4181.85	16.07	9216.42	3676	3665.90	12.10	686.18	**3688**	**3686.70**	11.81	947.20
MF800_08	3216(OM)	4039.09	20.38	10309.25	3611	3595.70	10.60	727.81	**3612**	**3606.20**	10.57	1007.44
MF800_09	3517(OM)	3980.71	11.65	19233.69	3600	3560.30	9.56	766.85	**3608**	**3604.90**	9.36	791.96
MF800_10	3620(OM)	4061.97	10.88	13568.69	3635	3608.60	10.51	513.45	**3643**	**3641.00**	10.31	741.42
MF1000_01	4383(OM)	4994.40	12.24	14714.33	4457	4437.50	10.76	1060.82	**4470**	**4466.20**	10.50	816.85
MF1000_02	4384(OM)	5027.37	12.80	10598.92	4451	4411.60	11.46	1023.52	**4471**	**4464.90**	11.07	1514.09
MF1000_03	4612(OM)	5214.57	11.56	10593.81	4636	4602.00	11.10	904.53	**4645**	**4639.50**	10.92	1332.18
MF1000_04	4522(OM)	5119.94	11.68	23007.08	4562	4536.00	10.90	791.74	**4592**	**4589.20**	10.31	1324.51
MF1000_05	4449(OM)	5056.91	12.02	18866.23	4518	4489.10	10.66	376.20	**4531**	**4528.40**	10.40	1329.76
MF1000_06	4578(OM)	5057.41	9.48	24270.31	4597	4578.80	9.10	1038.57	**4604**	**4603.20**	8.97	1326.59
MF1000_07	4561(OM)	5164.49	11.69	14552.89	4605	4569.50	10.83	631.54	**4612**	**4609.50**	10.70	1344.36
MF1000_08	4510(TL)	5129.77	12.08	28800.13	4531	4488.40	11.67	907.97	**4572**	**4559.60**	10.87	1518.82
MF1000_09	4617(OM)	5094.03	9.36	13171.78	4642	4630.40	8.87	701.34	**4643**	**4636.80**	8.85	1005.37
MF1000_10	4586(OM)	5181.95	11.50	9649.64	4660	4644.30	10.07	655.44	**4678**	**4674.70**	9.73	1388.05
#Best	0				8	1			**38**	**39**		
Average	3367.83	3809.50	11.41	16229.35	3423.45	3406.74	9.95	707.63	3431.40	3429.46	9.76	840.60