An effective hybrid search method for the quadratic knapsack problem with conflict graphs

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, China

Accepted to *Journal of the Operational Research Society*, May 2023

ARTICLE HISTORY
Compiled May 11, 2023

ABSTRACT
The quadratic knapsack problem (QKP) is a variant of the well-known knapsack problem and arises in a variety of real life applications. The quadratic knapsack problem with conflict graphs (QKPCG) further extends QKP by considering the conflicts of items. In this work, we propose an effective hybrid search method based on the framework of memetic algorithm to tackle QKPCG. The method integrates a randomized uniform-based crossover operator to generate promising offspring solutions, a multi-neighborhood tabu search to perform local optimization, and a streamline technique to speed up the evaluation of candidate solutions. The method shows a competitive performance compared to the state-of-the-art approaches in the literature. It finds 3 improved best-known solutions and matches the best-known solutions for all the remaining cases out of the 45 benchmark instances. We investigate the effects of the key ingredients of the algorithm.

KEYWORDS
Quadratic knapsack problem; conflict graphs; heuristics; memetic framework; tabu search

1. Introduction

The knapsack problem (KP) is a canonical combinatorial optimization problem which finds various applications such as resources allocation (Aisopos, Tserpes, & Varvarigou, 2013), logistics (Perboli, Gobbato, & Perfetti, 2014) and telecommunications (Van der Merwe, & Hattingh, 2006). Given a knapsack with its capacity and a set of items, each with its associated weight and profit, KP seeks to select a subset of items to maximize the total profit of the selected items without exceeding the knapsack capacity. For a comprehensive review of KP, the readers are referred to (Kellerer, Pferschy, & Pisinger, 2004).

The quadratic knapsack problem with conflict graphs (QKPCG) extends KP by considering quadratic profits between items and conflicts of items. In QKPCG, we

CONTACT Zhong-Zhong Jiang. Email: zzjiang@mail.neu.edu.cn

are given a knapsack with a predefined capacity c, a set of n items $V = \{1, 2, ..., n\}$ and a conflict graph $G = (V, E)$ where $E \subset \{(i, j) \in V \times V, i < j\}$ specifies the incompatibility of items ($(i, j) \in E$ means that items i and j cannot be packed in the knapsack simultaneously). Let $w = \{w_1, w_2, ..., w_n\}$ be the set of item weights, and $p = \{p_1, p_2, ..., p_n\}$ be the set of item profits. Let $q = \{q_{ij} : (i, j) \notin E\}$ represent the set of quadratic profits of compatible items. QKPCG is to select a subset of compatible items to maximize the total profit of the selected items, while satisfying the capacity constraint. QKPCG is NP-hard (Shi, Wu, & Meng, 2017) and thus computationally challenging.

Formally, let x_i ($i \in V$) be the decision variable such that $x_i = 1$ if item i is placed in the knapsack, and $x_i = 0$ otherwise. QKPCG can then be expressed as the following quadratic integer program (Dahmani, & Hifi, 2021; Shi, Wu, & Meng, 2017).

$$\textbf{Maximize} \quad f = \sum_{i\in V} p_i x_i + \sum_{i\in V} \sum_{j\in V, j>i} q_{ij} x_i x_j \tag{1}$$

$$\textbf{subject to} \quad \sum_{i\in V} w_i x_i \leq c \tag{2}$$

$$x_i + x_j \leq 1, \quad \forall (i, j) \in E \tag{3}$$

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

The objective function (1) maximizes the overall collected profits. The capacity constraint (2) guarantees that the total weights of the items in the knapsack do not exceed its capacity. The disjunctive constraints (3) ensure that the items placed in the knapsack must be compatible, and the constraints (4) impose that each variable takes a binary value.

QKPCG has a number of practical applications in domains like portfolio management (Dahmani, & Hifi, 2021). A real world case study in establishing an economic development strategy in the environmental protection zone for QKPCG is presented in (Shi et al., 2017). Given the relevance of QKPCG, several solution methods have been proposed in the literature. In Section 2, we review these existing works as well as some related problems. We notice that compared to KP and QKP, much less efforts have been dedicated to QKPCG since its introduction in 2017 (Shi, Wu, & Meng, 2017). In fact only three heuristic algorithms have been developed for practically solving QKPCG problem. These algorithms have contributed to find satisfying solutions to a set of benchmark instances of the problem, their performances vary according to the tested instances and lack stability across the whole range of the benchmark. This work aims thus to advance the state-of-the-art of solving more efficiently and effectively QKPCG. For this, we propose a competitive heuristic algorithm, which is able to find, in a short computing time, high quality solutions on the commonly used benchmark instances. The main contributions of this work are summarized as follows.

- We present the first hybrid search method (HSM) based on the memetic framework for tackling QKPCG, which possesses three original characteristics. First, to generate promising offspring solutions, HSM applies a dedicated randomized uniform-based crossover operator (RUX), which commits to preserving common items shared by parent solutions. The motivation behind the design of RUX is that high quality solutions are observed to share a large number of common items (see Section 5.2). Second, to explore efficiently the search space around each newly generated solution by the crossover operator, HSM adopts a tabu search optimization procedure which relies on the union of three complementary neighborhoods. Finally, a fast incremental evaluation technique is devised to quickly evaluate each candidate move, improving the computational efficiency of the proposed approach.

- Computational assessments on the set of 45 benchmark instances commonly used in the literature indicate that the proposed algorithm obtains highly competitive results compared to the existing best-performing QKPCG methods. It finds 3 novel lower bounds and matches the best-known solutions for all the remaining cases in a short computing time.

The rest of the paper is organized as follows. Section 2 reviews existing related works. Section 3 describes the methodology of the proposed algorithm. Section 4 presents the computational assessment and comparisons. Section 5 shows an analysis of two key ingredients of the algorithm. Conclusions and perspectives are given in the last section.

2. Related works

To our knowledge, three practical algorithms have been proposed for QKPCG. Shi et al. (Shi, Wu, & Meng, 2017) developed the first neighborhood search-based metaheuristic (NSBM) for the problem. NSBM adopts a two-phase construction procedure to generate a starting solution and a sophisticated large neighborhood search to explore the solution space. Dahmani and Hifi (Dahmani, & Hifi, 2021) proposed a modified descent method-based (MDM) heuristic for QKPCG, which combines a descent-based intensification search and a diversification procedure with degrading and re-optimization strategies to enhance the search efficiency. The reported results indicate that the MDM method has a superior performance compared to the reference algorithms. Dahmani et al. (Dahmani, Hifi, Saadi, & Yousef, 2020) introduced a population-based search algorithm (PBSA), which integrates binary particle swarm optimization (BPSO) with a fast and efficient local search procedure. In PBSA, BPSO maintains a population of solutions and uses the local search to either improve the quality of each binary solution or repair its infeasibility with the check and repair operators following the profit/weight ratio (Chih, 2015, 2018). Their study showed that PBSA was very competitive compared with the state-of-the-art algorithms including NSBM (Shi, Wu, & Meng, 2017) and MDM (Dahmani, & Hifi, 2021).

The disjunctively constrained knapsack problem (DCKP) also known as the knapsack problem with conflict graph is closely related to QKPCG. The difference between these two problems is their objective function: a quadratic objective function for QKPCG and a linear objective function for DCKP. The quadratic function suggests a higher difficulty for solution methods, especially to obtain exact solutions or high quality approximate solutions for large-scale instances. Compared to QKPCG, DCKP has attracted much more attention in the last two decades, and various solution approaches have been proposed for solving it, including exact algorithms (Coniglio, Furini, & San Segundo, 2021; Hifi, & Michrafy, 2007; Yamada, Kataoka, & Watanabe, 2002), approximation algorithms (Pferschy, & Schauer, 2017) and heuristic algorithms (Hifi, & Michrafy, 2006; Salem, Hanafi, Taktak, & Abdallah, 2017; Wei, & Hao, 2021).

According to the computational results shown in the literature on QKPCG, NSBM (Shi et al., 2017), MDM (Dahmani, & Hifi, 2021) and PBSA (Dahmani et al., 2020) are the current best performing algorithms for tackling QKPCG, and we use them as the reference methods in the numerical experimental comparisons. Meanwhile, it is known that the memetic search framework is among the most powerful general approaches for grouping problems such as graph coloring (Lü, & Hao, 2010) and bin packing (Spencer et al., 2019). Until now, this method is still unexplored for QKPCG. Therefore, this work aims at filling this gap by studying for the first time the potential of the memetic search framework to tackle QKPCG. As demonstrated by the computational results in Section 4, this method is indeed very effective and robust.

3. Hybrid search method for QKPCG

The proposed HSM method relies on the memetic algorithm (Moscato, 1999), which combines population-based evolutionary principle and neighborhood-based local search. The basic rationale behind memetic algorithms is to take advantage of these two complementary search strategies (Hao, 2012), by combining the exploration capability of population-based search and exploitation power of local search. The memetic framework has been successfully applied to a variety of difficult combinatorial optimization problems, such as job-shop scheduling problems (Constantino, & Segura, 2022; Zhou, He, Ma, Lim, & Pratap, 2022; Zobolas, Tarantilis, & Ioannou, 2009), critical node detection problems (Alozie, Arulselvan, Akartunalı, & Pasiliao, 2022) and knapsack problems (Li, Tang, & Liu, 2020; Wei, & Hao, 2021).

3.1. Main scheme

Algorithm 1: Hybrid search method for QKPCG

```
   Input: $I$: QKPCG graph, population size $p_s$, maximum allowed time $t_{max}$
   Output: Best found solution $S^*$
1  $P = \{S^1, ..., S^p\} \leftarrow pop\_initial(p_s)$ /* Section 3.3, population initialization */
2  $S^* \leftarrow best(P)$ /* $S^*$ records the best solution encountered so far */
3  while $t_{max}$ is not reached do
4      Select randomly two parents $S^a$, $S^b$ from $P$
5      $S^o \leftarrow cross\_over(S^a, S^b)$ /* Section 3.4, generate an offspring solution */
6      $S^o \leftarrow tabu\_search(S^o)$ /* Section 3.5, improve the offspring */
7      if $f(S^o) > f(S^*)$ then
8          $S^* \leftarrow S^o$ /* Update the recorded best solution */
9      $P \leftarrow pop\_update(P, S^o)$ /* Update the population */
```

The general scheme of HSM is presented in Algorithm 1. HSM starts with an initial population (see line 1, Section 3.3) consisting of p_s individuals where p_s is the population size. Then HSM executes a series of generations (see lines 3-9) until it reaches a stopping condition (typically a given cutoff time t_{max}). At each generation, the randomized uniform-based crossover operator is applied to produce an offspring solution S^o from two parent solutions randomly chosen in the population (see lines 4-5, Section 3.4). Afterward, S^o is improved by the multi-neighborhood tabu search procedure (see line 6, Section 3.5). Finally, S^o is utilized to update the best recorded solution S^* (lines 7-8) as well as the population (line 9). To update the population, S^o just replaces the worst solution in the population if S^o has a better objective value and S^o does not exist already in the population. The main components of HSM are depicted in the following subsections.

3.2. Search space and evaluation function

For a given QKPCG instance $I = (V, E, c, w)$, the search space Ω explored by the proposed HSM algorithm contains all the possible subsets of V satisfying the capacity constraint and disjunctive constraint, i.e., $\Omega = \{S : S \subset V$ such that $\sum_{i \in S} w_i \leq c$ and $\forall i, j \in S, (i, j) \notin E$ $\}$. For any solution $S \in \Omega$, its quality is evaluated by the objective value (Eq. (1)). A solution S is better than another solutionS' only if $f(S) > f(S')$.

3.3. Population initialization

Algorithm 2: Randomized construction method

```
   Input: QKPCG graph $I$
   Output: A feasible random solution $S$
1  $S \leftarrow \emptyset$
2  Identify the item $m$ with the minimum weight $w_m$
3  $cw \leftarrow c$ /* $cw$ is the residual weight of the knapsack */
4  $R \leftarrow \{1, 2, ..., n\}$ /* $R$ is the set of unvisited items */
5  Perform a random permutation operation on $R$ /* $R$ is a random permutation of
     the unvisited item set $\{1, 2, ..., n\}$ */
6  $t \leftarrow 1$
7  while $w_m \leq cw$ do
8      Select the $t$-th element $R_t$ of $R$
9      if $w_{R_t} \leq cw$ then
10         if $R_t$ is compatible with each element in $S$ then
11             $S \leftarrow S \cup \{R_t\}$
12             $cw \leftarrow cw - w_{R_t}$
13     $t \leftarrow t + 1$
14 return $S$
```

The initial population P is composed of p_s feasible individuals (solutions) and each solution is constructed by a randomized construction method (RCM, Algorithm 2). Starting from an empty set S, RCM first initializes the set of unvisited items $R = \{1, 2, ..., n\}$. And then a random permutation operation is carried out on R by executing n random swaps (lines 4-5, Algorithm 2), leading to a randomized set $R = \{R_1, R_2, ..., R_n\}$, where $R_i \in \{1, 2, ..., n\}$ and $R_i \neq R_j$ for any different i and j $(i, j \in \{1, 2, ..., n\}, i \neq j)$. Then, RCM repeats a number of iterations to visit the items and returns a feasible random solution S at the end (lines 7-14, Algorithm 2). At iteration t, the t-th element R_t of R is visited (line 8, Algorithm 2) and inserted into the solution S, so long as the weight of R_t does not exceed the residual knapsack capacity $cw = c - \sum_{i \in S} w_i$ (line 9, Algorithm 2) and R_t is compatible with the selected items in S (line 10, Algorithm 2). RCM stops once the knapsack capacity is reached, that is, the residual knapsack capacity cw is less than the minimum weight w_m of the set of item weights.

The generated solution S is then improved by the multi-neighborhood tabu search procedure (see Section 3.5). The resulting improved solution is inserted into the population if it is not identical to any existing solution in the population. The process stops as soon as the number of solutions in the population reaches p_s.

The time complexity of RCM algorithm can be estimated as follows. At the beginning of RCM, a random permutation R is obtained in $O(n)$ time (lines 4-5, Algorithm 2). At each RCM iteration, an item R_t is inspected for insertion into the current solution S in $O(|S|)$ (lines 8-10, Algorithm 2), where $|S|$ is the number of selected items in S. Therefore, the total time complexity of RCM is bounded by $O(|S| \times n)$.

3.4. Randomized uniform-based crossover operator

Algorithm 3: Randomized uniform-based crossover operator

```
Input: Two randomly selected parents $S^a$ and $S^b$
Output: A feasible offspring solution $S^o$
1  $S^o \leftarrow \emptyset$
2  Identify the item $m$ with the minimum weight $w_m$
3  $cw \leftarrow c$ /* $cw$ is the residual weight of the knapsack */
4  $R \leftarrow \{1, 2, ..., n\}$ /* $R$ is the set of unvisited items */
5  Make a random permutation operation on $R$ /* $R$ is a random permutation of the
     unvisited item set $\{1, 2, ..., n\}$ */
6  $t \leftarrow 1$
7  while $cw \geq w_m$ do
8      Select the $t$-th element $R_t$ of $R$
9      $coin \leftarrow rand(0, 1)$ /* $coin$ takes the value 0 or 1 with equal probability */
10     if $coin \neq 0$ then
11         if $R_t$ is selected in the knapsack of $S^a$ and can fit into the knapsack of $S^o$
             then
12             $S^o \leftarrow S^o \cup \{R_t\}$
13             $cw \leftarrow cw - w_{R_t}$
14     else
15         if $R_t$ is selected in the knapsack of $S^b$ and can fit into the knapsack of $S^o$
             then
16             $S^o \leftarrow S^o \cup \{R_t\}$
17             $cw \leftarrow cw - w_{R_t}$
18     $t \leftarrow t + 1$
19 return $S^o$
```

The randomized uniform-based crossover operator (denoted as RUX, see Algorithm 3) is a crucial component of HSM algorithm. RUX is applied to generate a diversified offspring solution from two randomly selected parent solutions at each generation of HSM. Usually, an effective crossover is expected to inherit good properties ("building blocks") of parent solutions and sustain diversity in relation to the parents (Hao, 2012; Neri, & Cotta, 2012). For QKPCG, a preliminary experiment indicated that high quality solutions share a number of items (refer to Section 5.2), which might be part of an optimal solution.

RUX follows the general principle described above and generates a feasible offspring solution S^o which inherits solution features from a randomly selected parent. Specifically, RUX firstly initializes $S^o = \emptyset$, the set of unvisited items $R = \{1, 2, ..., n\}$, and then performs a random permutation operation on R as illustrated in Section 3.3 to obtain a random set $R = \{R_1, R_2, ..., R_n\}$ (lines 4-5, Algorithm 3). Next, RUX performs a series of iterations to build the offspring solution S^o (lines 7-18, Algorithm 3) until the knapsack capacity is reached. At each iteration t, the t-th element R_t of R is selected and a parent solution S^r is taken randomly between the two parents S^a and S^b (lines 8-9, Algorithm 3). Then R_t is added to S^o, as long as it appears in S^r and adding R_t to S^o does not violate the capacity constraint nor the disjunctive constraint (lines 10-13 or lines 14-17, Algorithm 3). The time complexity of RUX is bounded by $O(|S^o| \times n)$.

3.5. Multi neighborhood tabu search

For local optimization, HSM algorithm employs a multi neighborhood tabu search (MNTS), which has been successfully applied to tackle a number of combinatorial optimization problems (Glover, & Laguna, 1998; Wu, Hao, & Glover, 2012). In the following subsections, we describe the neighborhood structures and fast incremental evaluation technique, as well as the neighborhood exploration strategy of MNTS procedure.

3.5.1. Neighborhood structures and fast incremental evaluation technique

The MNTS method jointly exploits three neighborhoods, including the add neighborhood N_a, the drop neighborhood N_d and the swap neighborhood N_s, which were also used in previous studies like (Dahmani, & Hifi, 2021). However, unlike (Dahmani, & Hifi, 2021) where these neighborhoods are applied according to a specific order, MNTS exploits the union of these neighborhoods. Moreover, to ensure a high computational efficiency, MNTS adopts for the time an incremental streamlining evaluation technique to assess the quality of each neighboring solution. Note that this fast evaluation technique can benefit other local search algorithms based on these neighborhoods.

Given an incumbent solution $S \subset V$, the *Add* operator denoted as $Add(i)$ inserts an item $i \in V \backslash S$ into S such that all the constraints (Constraints (2-4)) are satisfied. To quickly compute the move gain of a candidate move, MNTS introduces the first fast incremental evaluation technique for QKPCG which was widely used for other quadratic optimization problems, such as the capacitated clustering problem (Lai, & Hao, 2016) and the quadratic assignment problem (Zhou, Hao, & Duval, 2020). The main idea is to hold a n-dimensional vector γ, where its element $\gamma[i]$ records the sum of profits between item i and all other items selected in current solution S, i.e., $\gamma[i] = \sum_{j \in S, j \neq i} q_{ij}$. The move value of an $Add(i)$ operation can then be efficiently calculated as:

$$\Delta_f(Add(i)) = p_i + \gamma[i] \tag{5}$$

Clearly, an *Add* move always leads to an improved solution since its associated move gain is always positive. After executing an $Add(i)$ move, the vector γ is updated in $O(n)$ time as follows: $\gamma[j] = \gamma[j] + q_{ij}$, $\forall j \in V, j \neq i$.

The size of N_a neighborhood consisting of all the *Add* candidate moves is bounded by $O(|V \backslash S|)$.

The *Drop* operator denoted by $Drop(i)$ removes an item i from the current solution S. The move gain of dropping an item i can be quickly calculated by:

$$\Delta_f(Drop(i)) = -p_i - \gamma[i] \tag{6}$$

Obviously, a *Drop* move always deteriorates the quality of the current solution. When a $Drop(i)$ move is performed, the vector γ will be updated in $O(n)$ time: $\gamma[j] = \gamma[j] - q_{ij}$, $\forall j \in V, j \neq i$.

The size of N_d induced by the *Drop* operator is bounded by $O(|S|)$.

The *Swap* operator exchanges two items i and j (denoted as $Swap(i, j)$) where $i \in S$ and $j \notin S$, while satisfying all the constraints. For a given $Swap(i, j)$ move, its move gain can be expressed by:

$$\Delta_f(Swap(i, j)) = p_j - p_i + \gamma[j] - \gamma[i] - q_{ij} \tag{7}$$

One can notice that the move gain of a *Swap* move can be negative or non-negative. In other words, a *Swap* move can improve or worsen a solution with respect to the objective function or keep the objective value unchanged. Since a *Swap* move can be regarded as the compound operation of an *Add* move followed by a *Drop* move (or a *Drop* move followed by an *Add* move), γ is then consecutively updated two times according to the *Add* move and *Drop* move in $O(n)$ time.

The size of N_s is bounded by $O(|S| \times |V\backslash S|)$ and is usually much larger than that of N_a and N_d.

3.5.2. Exploration with tabu search

Algorithm 4: Multi neighborhood tabu search

```
   Input: A feasible solution $S$, the search depth of tabu search $sd$
   Output: Best feasible solution found $S^*$
1  $t \leftarrow 0$ /* Non-improve iteration counter */
2  $S^* \leftarrow S$ /* $S^*$ records the best feasible solution encountered so far */
3  $tl[i] \leftarrow 0$, for each $i \in V$ /* Initialize the tabu list $tl$ */
4  while $t < sd$ do
5      Select a best admissible neighboring solution $S'$ from the union of $N_a$, $N_d$ and
        $N_s$
6      $S \leftarrow S'$
7      Update the tabu list $tl$
8      if $f(S) > f(S^*)$ then
9          $S^* \leftarrow S$
10         $t \leftarrow 0$
11     else
12         $t \leftarrow t + 1$
13 return $S^*$
```

When there are several neighborhoods, one important issue is how to apply the neighborhoods to efficiently explore the search space (Lü, Hao, & Glover, 2011). In the case of QKPCG, a *Swap* move might result in a neighboring solution that has a better quality than any solution obtained by an *Add* move. Furthermore, when no *Add* move can be applied, a *Drop* move may lead to a neighboring solution better than any solution produced by a *Swap* move. Therefore, there is no absolute advantage of a move operator (and the induced neighborhood) over another one. Based on these considerations, MNTS procedure examines the N_a, N_d and N_s neighborhoods as the union set $N_a \cup N_d \cup N_s$.

The general scheme of MNTS procedure is given in Algorithm 4. Starting from an input feasible solution, MNTS replaces the current solution S with the admissible highest-gain neighboring solution S' (ties are broken randomly) from the union of N_a, N_d and N_s neighborhoods (lines 5-6, Algorithm 4). Thanks to the fast evaluation technique above, each iteration can be achieved in $O(|S| + |V\backslash S| + |S| \times |V\backslash S|)$ time. To avoid short term cycling, each time an item i is removed from the solution, it is

Table 1. Scaling factors of the processors used in each reference algorithm, with respect to the processor used in this work.

Algorithm	Reference	Processor type	CPU frequency (GHz)	Scaling factor
HSM	-	Intel Xeon E5-2630 v3	2.40	1.00
NSBM	(Shi et al., 2017)	Intel Core i5	3.10	1.29
MDM	(Dahmani, & Hifi, 2021)	Intel core 2 duo	2.53	1.05
PBSA	(Dahmani et al., 2020)	Intel core 2 duo	2.53	1.05

marked as tabu and forbidden to go back to the solution for the next tt iterations where tt is a parameter called the tabu tenure (line 7, Algorithm 4). The tabu status of a move is overridden if the move results in a solution with improved quality (aspiration criterion). A move is regarded as admissible if it is not marked as tabu or it meets the aspiration criterion. The best feasible solution found S^* is updated whenever an improved solution S is obtained, i.e., $f(S) > f(S^*)$ (lines 8-9, Algorithm 4). MNTS is repeated until the best recorded solution S^* cannot be updated for sd consecutive iterations, where sd is a parameter called the search depth of MNTS, and returns the best feasible solution during the search process.

4. Computational experiments

To evaluate the performance of HSM algorithm, we present computational experiments on a set of benchmark instances from the literature.

4.1. Benchmark instances

Our computational assessments are based on a set of 45 benchmark instances[1], which are grouped into 9 classes (noted from 1qkpcg to 9qkpcg) where each class consists of 5 instances. These instances are introduced by Shi et al. (Shi et al., 2017) and later used in other studies (Dahmani, & Hifi, 2021; Dahmani et al., 2020). The instances are characterized by: the number of items $n \in \{100, 150, 200\}$, and the density of the graph d (calculated by $d = \frac{|E|}{n*(n-1)}$ where $|E|$ is the number of pairs of incompatible items) varies from 2% to 8%. All instances were generated according to the standard model used by Billionnet and Soutif (Billionnet, & Soutif, 2004) for the quadratic functions, and the specificity of the disjunctive constraints as studied by Yamada et al. (Yamada et al., 2002).

4.2. Experimental settings

The HSM algorithm was programmed in C++ language[2] and compiled using the g++ 7.3.0 compiler with the "-O3" option. HSM was executed on a computing platform with an Intel Xeon E5-2630 v3 processor (2.4 GHz) and 1 GB RAM under the Linux operating system. Following the reference algorithms (Dahmani, & Hifi, 2021; Dahmani

[1]`https://github.com/neteasefans/QKPCG.git`

[2]The source codes of the HSM algorithm will be publicly available at the GitHub page

Table 2. Values of parameters tuned with the 'irace' package.

Parameter	Section	Description	Candidate values	Final value
p_s	3.3	size of population	{5, 10, 20, 30, 50}	10
tt	3.5.2	tabu tenure	{10, 20, 30, 50, 100}	20
sd	3.5.2	search depth of tabu search	{1000, 2000, 5000, 10000, 20000}	10000

et al., 2020), HSM was performed independently ten times on each instance with different random seeds. The cutoff time t_{max} for each run was set to 200 seconds as in (Dahmani, & Hifi, 2021; Dahmani et al., 2020; Shi et al., 2017).

To evaluate the performance of HSM, we conduct experimental comparisons with three state-of-the-art QKPCG heuristics in the literature, including the NSBM (Shi et al., 2017), MDM (Dahmani, & Hifi, 2021) and PBSA (Dahmani et al., 2020). The numerical results reported by these methods are directly compiled from the corresponding literature. To make a relatively fair comparison in running time, we used the CPU frequency as the main indicator to compare the speeds of the processors used by HSM and the reference methods. Table 1 shows the processer type and its frequency used in each algorithm, as well as resulting scaling factor in relation to the processor used in this work (Intel Xeon E5-2630 v3, 2.4 GHz) which servers as a basis. The processor used in this paper is thus a little slower than that utilized by the reference approaches. Note that the running time reported by all algorithms is provided for indicative purposes only, since the runtime of each algorithm is influenced by some additional factors including the programming language, data structures, operating system and compiler options.

4.3. Parameter tuning

The HSM algorithm has only 3 parameters: the population size p_s, the tabu tenure tt and the search depth of tabu search sd. These parameters are tuned with an automatic tuning package named 'irace' (López-Ibáñez, Dubois-Lacoste, Cáceres, Birattari, & Stützle, 2016) dedicated to off-line parameter configuration for parameterized algorithms. In this tuning experiment, we selected randomly 5 instances from the set of 45 benchmarks as the training instances, and set the tuning budget of HSM to 1000. The candidate values of each parameter serving as the input of the 'irace' software are chosen according to our preliminary experiments. Table 2 presents the candidate values as well as the final value determined by 'irace' for each parameter.

4.4. Computational results and comparisons with state-of-the-art algorithms

Table 3 shows the detailed results of HSM as well as the results of three reference heuristics (NSBM (Shi et al., 2017), MDM (Dahmani, & Hifi, 2021) and PBSA (Dahmani et al., 2020)) on the set of 45 benchmark instances. Column 'Ins.' indicates the instance name and 'f_{bk}' reports the best-known result compiled from all published results in the literature. Columns 'f_{best}', 'f_{avg}' and '$t_{avg}(s)$' show respectively the best objective value, the average objective value, and the average running time in seconds to reach the final objective value over 10 independent executions. Note that the running time of MDM is unavailable. Columns '$\#hits$' and 'σ' represent respectively the number of runs reaching f_{best}, and the standard deviation of the objective values obtained across 10 independent runs for HSM. Row '#Best' records the number of

Table 3. Comparative results between HSM and three reference algorithms on the set of 45 benchmarks. The best results are marked in bold.

		NSBM			MDM		PBSA			HSM				
Ins.	f_{bk}	f_{best}	f_{avg}	$t_{avg}(s)$	f_{best}	f_{avg}	f_{best}	f_{avg}	$t_{avg}(s)$	f_{best}	f_{avg}	$t_{avg}(s)$	$\#hits$	σ
1qkpcg1	**17071**	16642	16280.20	117.78	17061	16859.50	**17071**	16859.50	35.00	**17071**	**17071.00**	0.24	10	0.00
1qkpcg2	**13500**	12272	12006.20	125.34	**13500**	13332.20	**13500**	13369.80	15.00	**13500**	**13466.90**	38.37	6	41.46
1qkpcg3	**16156**	15843	15612.60	148.14	**16156**	16111.50	**16156**	16111.50	21.44	**16156**	**16156.00**	0.68	10	0.00
1qkpcg4	**19921**	19659	19594.20	102.30	**19921**	19714.90	**19921**	19806.50	15.33	**19921**	**19921.00**	1.81	10	0.00
1qkpcg5	**16870**	15499	15378.80	37.18	**16870**	16833.70	**16870**	16843.70	24.11	**16870**	**16870.00**	0.14	10	0.00
2qkpcg1	**11217**	11205	10895.00	82.78	11172	11172.00	**11217**	11181.00	5.89	**11217**	**11217.00**	3.95	10	0.00
2qkpcg2	**10933**	10565	10438.20	88.83	10911	10812.00	**10933**	10842.60	15.89	**10933**	**10933.00**	5.21	10	0.00
2qkpcg3	11248	11163	11084.80	47.02	11232	11212.70	11248	11218.10	18.11	**11312**	**11312.00**	4.51	10	0.00
2qkpcg4	**15599**	**15599**	**15599.00**	156.07	**15599**	**15599.00**	**15599**	**15599.00**	6.11	**15599**	**15599.00**	0.47	10	0.00
2qkpcg5	**12580**	**12580**	12541.60	19.28	**12580**	12442.60	**12580**	12520.60	14.00	**12580**	**12580.00**	20.20	10	0.00
3qkpcg1	**7455**	**7455**	7337.00	17.00	**7455**	**7455.00**	**7455**	**7455.00**	14.33	**7455**	**7455.00**	0.25	10	0.00
3qkpcg2	**7343**	**7343**	**7343.00**	57.44	**7343**	7266.80	**7343**	**7343.00**	8.78	**7343**	**7343.00**	0.19	10	0.00
3qkpcg3	**7285**	7258	7200.60	88.58	**7285**	**7285.00**	**7285**	**7285.00**	10.89	**7285**	**7285.00**	0.30	10	0.00
3qkpcg4	**8006**	7991	7988.60	59.85	**8006**	7899.80	**8006**	**8006.00**	18.11	**8006**	**8006.00**	0.14	10	0.00
3qkpcg5	**7350**	**7350**	7156.40	4.38	**7350**	7192.60	**7350**	7307.60	23.11	**7350**	**7350.00**	0.18	10	0.00
4qkpcg1	**20726**	19457	18931.40	29.84	20401	20116.40	**20726**	20116.40	24.56	**20726**	**20635.10**	94.49	7	138.35
4qkpcg2	**21677**	19240	18749.00	97.11	**21677**	21406.80	**21677**	21406.80	20.00	**21677**	**21677.00**	20.85	10	0.00
4qkpcg3	**21953**	20864	20538.20	73.48	21897	21077.20	**21953**	21332.20	35.56	**21953**	**21953.00**	12.05	10	0.00
4qkpcg4	**31123**	31082	30737.80	3.01	**31123**	**31123.00**	**31123**	**31123.00**	18.00	**31123**	**31123.00**	0.57	10	0.00
4qkpcg5	**20792**	18727	18083.40	124.47	20781	20660.60	**20792**	20660.60	43.33	**20792**	**20792.00**	7.72	10	0.00
5qkpcg1	**15517**	14871	14745.00	110.44	15353	15157.70	**15517**	15189.90	14.89	**15517**	**15517.00**	8.39	10	0.00
5qkpcg2	15529	15068	14956.20	1.62	15529	15511.10	15529	15511.10	18.11	**15594**	**15561.50**	33.51	5	32.50
5qkpcg3	**15370**	15132	14946.20	161.43	15178	15086.10	**15370**	15086.10	26.44	**15370**	**15370.00**	37.87	10	0.00
5qkpcg4	**18954**	18380	18272.20	161.64	**18954**	18757.20	**18954**	18938.80	28.44	**18954**	**18954.00**	3.81	10	0.00
5qkpcg5	**15715**	15607	15313.20	44.84	**15715**	15686.10	**15715**	15702.20	28.56	**15715**	**15715.00**	4.43	10	0.00
6qkpcg1	**8969**	**8969**	8840.00	136.26	8698	8688.00	**8969**	8688.00	24.33	**8969**	**8969.00**	14.25	10	0.00
6qkpcg2	**9658**	9193	9098.80	18.43	**9658**	9403.50	**9658**	9403.50	21.22	**9658**	**9658.00**	1.02	10	0.00
6qkpcg3	**8578**	8508	8504.40	79.86	**8578**	8375.80	**8578**	8555.10	29.44	**8578**	**8578.00**	7.79	10	0.00
6qkpcg4	**9657**	9630	9156.40	147.43	**9657**	8761.00	**9657**	**9657.00**	10.56	**9657**	**9657.00**	3.54	10	0.00
6qkpcg5	**9338**	9335	9335.00	141.68	**9338**	**9338.00**	**9338**	**9338.00**	24.89	**9338**	**9338.00**	1.77	10	0.00
7qkpcg1	**27010**	24602	24187.80	127.89	**27010**	26628.30	**27010**	26659.40	61.22	**27010**	**26842.00**	28.96	2	84.00
7qkpcg2	**30343**	28686	28354.60	152.25	30290	30039.90	**30343**	30115.50	48.67	**30343**	**30319.60**	36.25	7	49.41
7qkpcg3	**26685**	23420	22444.60	43.48	26539	26437.10	**26685**	26497.40	47.11	**26685**	**26685.00**	7.14	10	0.00
7qkpcg4	**34110**	32641	32350.20	22.36	**34110**	33687.20	**34110**	33687.20	44.00	**34110**	**34110.00**	46.48	10	0.00
7qkpcg5	**25947**	22735	21762.40	90.17	**25947**	25815.40	**25947**	25815.40	71.44	**25947**	**25846.70**	60.41	4	93.07
8qkpcg1	**18668**	18583	18583.00	10.45	18378	18152.30	**18668**	18331.90	25.22	**18668**	**18629.20**	83.91	6	59.73
8qkpcg2	**20021**	19620	19406.20	126.94	19841	19656.30	**20021**	19813.50	41.56	**20021**	**20021.00**	32.34	10	0.00
8qkpcg3	17911	17469	16998.60	59.92	17911	17743.50	17911	17806.90	58.11	**17974**	**17917.30**	48.02	1	18.90
8qkpcg4	**21168**	20715	20524.00	100.65	**21168**	20904.40	**21168**	**21168.00**	34.67	**21168**	**21168.00**	10.54	10	0.00
8qkpcg5	**18757**	18480	18000.40	38.88	**18757**	18701.50	**18757**	18701.50	48.89	**18757**	**18757.00**	12.18	10	0.00
9qkpcg1	**11180**	**11180**	**11180.00**	110.27	**11180**	10914.00	**11180**	10914.00	50.78	**11180**	**11180.00**	19.91	10	0.00
9qkpcg2	**10598**	10577	10534.20	63.77	**10598**	10286.30	**10598**	10338.60	54.56	**10598**	**10598.00**	43.05	10	0.00
9qkpcg3	**11022**	10869	10852.60	12.30	**11022**	10849.30	**11022**	10849.30	49.89	**11022**	**11022.00**	2.58	10	0.00
9qkpcg4	**11751**	**11751**	11308.00	150.00	**11751**	11433.80	**11751**	11707.00	35.67	**11751**	**11729.00**	56.79	9	66.00
9qkpcg5	**10761**	10713	10602.40	47.27	10743	10517.10	**10761**	10517.10	38.00	**10761**	**10761.00**	33.33	10	0.00
#Best	42	8	3		28	5	42	9		**45**	**45**			
Avg.	16044.93	15433.96	15194.50	80.89	16004.96	15824.54	16044.93	15897.34	29.43	**16049.20**	**16036.63**	**18.90**		

Table 4. Results of the Wilcoxon signed-rank test between HSM and the reference algorithms on the set of 45 benchmark instances, with a significance level of 0.05.

Comparison	R^{+}_{best}	R^{-}_{best}	p-value	R^{+}_{avg}	R^{-}_{avg}	p-value
HSM vs. NSBM	37	0	1.14e-7	42	0	1.65e-8
HSM vs. MDM	17	0	2.93e-4	40	0	3.57e-8
HSM vs. PBSA	3	0	0.11	36	0	1.68e-7

instances for which a corresponding algorithm produces the best result among the compared algorithms, with respect to the best/average objective value. Row 'Avg.' denotes the average values of the corresponding indicators.

We can observe from Table 3 that HSM is able to find 3 improved best-known solutions (i.e., 2qkpcg3, 5qkpcg2 and 8qkpcg3), while matching the best-known solution for all the remaining instances. In terms of the average objective value, HSM yields the best results in 45 instances, while NSBM, MDM and PBSA report respectively the best result in 3, 5 and 9 cases. Moreover, HSM hits consistently the best objective value in each run for many instances, indicating the high robustness of the proposed approach, which is also confirmed by the small standard deviation (σ) on most test instances. Additionally, HSM is more computationally efficient than NSBM and PBSA in most cases. To detect whether there are statistical differences between the compared methods, Table 4 provides the results of Wilcoxon signed-rank test with a significance level of 0.05 as recommended in (Carrasco, García, Rueda, Das, & Herrera, 2020). Column R^{+}_{best} (R^{+}_{avg}) reports the sum of ranks for the cases where HSM outperforms the compared algorithm with respect to the best (average) objective value, while R^{-}_{best} (R^{-}_{avg}) represents the sum of ranks for the opposite cases. Table 4 indicates that HSM performs significantly better than the compared approaches NSDM, MDM and PBSA, except the case between HSM and PBSA in terms of the best objective value with a p-value > 0.05. These observations demonstrate the superiority of HSM, in terms of the solution quality and computing efficiency, compared to the state-of-the-art algorithms.

Finally, the experimental results demonstrate that the proposed HSM algorithm is a highly competitive approach for finding high-quality solutions for the existing QKPCG benchmark. This has some practical implications. First, as indicated in the introduction, QKPCG is a general model for a number of real problems. Given that the source codes of our HSM algorithm will be publicly available, researchers and practitioners can adopt our algorithm to solve problems that can be formulated by QKPCG. Indeed, thanks to the high efficiency and effectiveness of the algorithm, one can expect that our algorithm helps to find better solutions or solve some problems that cannot be solved with existing methods. Second, the proposed approach is stochastic nature, implying that it is easy to run the algorithm multiple times to obtain many high-quality solutions. This feature is of interest for decision-making because the decision-maker can easily obtain a set of alternative solutions from which the best decision can be made based on additional criteria. Finally, our approach could be adapted to other related problems constrained by a conflict graph such as the disjunctively constrained knapsack problem and flow shop scheduling problem with conflict graphs.

Table 5. The results of Wilcoxon signed-rank test of HSM with a multi-start version of MNTS on the set of 45 benchmarks, with a significance level of 0.05.

Comparison	R^+_{best}	R^-_{best}	p-value	R^+_{avg}	R^-_{avg}	p-value
HSM vs. MS-MNTS	1	0	0.32	10	1	7.65e-3

5. Discussions

In this section, additional experiments are reported to analyze the impacts of the population-based framework to the proposed algorithm. In addition, we present experimental results to support the design of the proposed crossover operator.

5.1. Advantage of the population-based framework

We made an experiment to assess the effects of the population-based framework. For this purpose, we showed a comparison of the proposed HSM algorithm with a multi-start version of MNTS procedure (denoted as MS-MNTS) where the crossover operator and the population mechanism are removed. For a fair comparison, HSM and MS-MNTS were run independently 10 times per instance, under the experimental settings described in Section 4.2.

The results of Wilcoxon signed-rank test shown in Table 5 indicate the superiority of HSM compared to MS-MNTS, by finding better results in 1 (10) instances with respect to the best (average) objective value. This experiment demonstrates the effectiveness of the population-based framework to the overall performance of HSM.

5.2. The motivation behind the uniform-based crossover operator

To illustrate the rationale behind the proposed crossover operator, we provide an experimental analysis of structural similarities between high-quality solutions. Given two solutions S^a and S^b, their similarity is defined as the proportion of the commonly shared items: $sim(S^a, S^b) = \frac{|S^a \cap S^b|}{|S^a \cup S^b|}$.

We run HSM method 100 independent times on 10 randomly selected instances, and record the best solution found for each run with a cutoff time of $\frac{n}{20}$ seconds. We then calculate the maximum similarity (denoted as sim_{max}), the average similarity (denoted sim_{avg}) and the minimum similarity (denoted sim_{min}) between any two solutions for each instance. The results of the solution similarities are displayed in Fig. 1. From Fig. 1, one observes that the average similarity between high-quality solutions is very high, over 0.85 for each instance, which suggests that a large number of shared items might form building blocks of a globally optimal solution, and provides a solid basis for the design of the uniform-based crossover operator in this work.

6. Conclusions

This work investigates the quadratic knapsack problem with conflict graphs, which is a variant of the classic knapsack problem. An effective and robust hybrid search method is developed to tackle the problem, which features a randomized uniform-based crossover operator to create promising offspring solutions, a multi-neighborhood tabu search to efficiently explore the search space, and a fast incremental evaluation

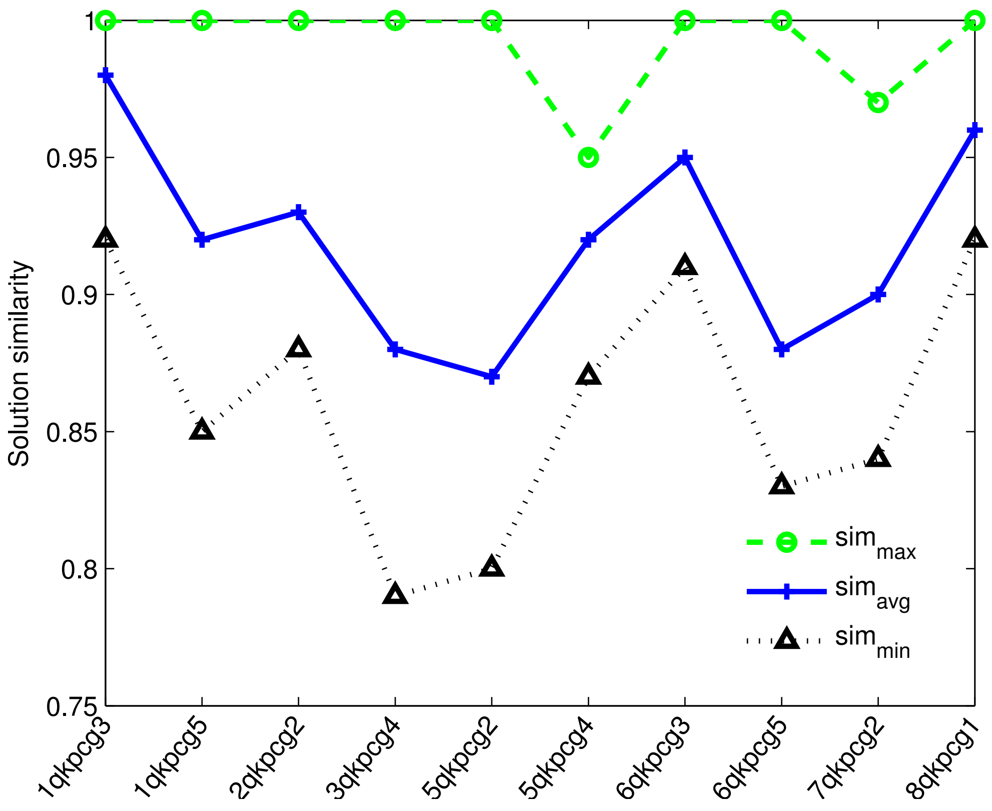

Figure 1. Similarity between high-quality solutions on 10 randomly selected instances from the set of 45 benchmarks.

technique to speed up the examination of neighboring solutions.

We demonstrate the performance of the proposed algorithm HSM on the set of 45 commonly used benchmarks. The comparative results reveal that HSM competes favorable with the current best algorithms in the literature. In particular, it updates the best-known solutions (new lower bounds) for 3 instances and matches the best-known results for all the remaining cases within a short running time. Additional experiments assess the usefulness of the population-based framework as well as the multi-neighborhood tabu search procedure and justify the design of the randomized uniform-based crossover.

The proposed HSM method is very effective and robust for QKPCG by obtaining high quality solutions in a reasonable time frame. It would be possible to adapt its basic ideas to design effective algorithms for other similar problems such as the disjunctively constrained knapsack problem. Also, the algorithm can be used to solve practical problems that can be formulated as QKPCG and the availability of its source codes facilitates such applications.

On the other hand, for several benchmark instances, the success rate of the algorithm to attain its best solutions remains low, implying that further amelioration is still possible. For instance, it would be interesting to improve the population management by considering both solution quality and distance (Hao, 2012) and pseudo random number assignment strategies (Chih, 2023). To enhance the local search component, it would be useful to investigate search strategies examining both feasible and infeasible solutions like in (Wei, & Hao, 2023). Finally, as a heuristic algorithm, HSM cannot ensure the optimality of the obtained solutions nor an approximation ratio relative to the optimal solution. To fill the gap, efforts are needed to develop practical exact and approximation algorithms.

Acknowledgments

We are grateful to the reviewers for their valuable comments and suggestions, which helped us to improve the paper. This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 71671033, 72201053) and China Postdoctoral Science Foundation (Grant No. 2021M700716).

References

Aisopos, F., Tserpes, K., & Varvarigou, T. (2013). Resource management in software as a service using the knapsack problem model. International Journal of Production Economics, 141(2), 465-477.

Alozie, G. U., Arulselvan, A., Akartunalı, K., & Pasiliao Jr, E. L. (2022). A heuristic approach for the distance-based critical node detection problem in complex networks. Journal of the Operational Research Society, 73(6), 1347-1361.

Billionnet, A., & Soutif, É. (2004). An exact method based on Lagrangian decomposition for the quadratic knapsack problem. European Journal of operational research, 157(3), 565-575.

Carrasco, J., García, S., Rueda, M. M., Das, S., & Herrera, F. (2020). Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. Swarm and Evolutionary Computation, 54, 100665.

Chih, M. (2015). Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem. Applied Soft Computing, 26, 378-389.

Chih, M. (2018). Three pseudo-utility ratio-inspired particle swarm optimization with local search for multidimensional knapsack problem. Swarm and Evolutionary Computation, 39, 279-296.

Chih, M. (2023). Stochastic stability analysis of particle swarm optimization with pseudo random number assignment strategy. European Journal of Operational Research, 305(2), 562-593.

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.

Dahmani, I., & Hifi, M. (2021). A modified descent method-based heuristic for binary quadratic knapsack problems with conflict graphs. Annals of Operations Research, 298(1), 125-147.

Dahmani, I., Hifi, M., Saadi, T., & Yousef, L. (2020). A swarm optimization-based search algorithm for the quadratic knapsack problem with conflict graphs. Expert Systems with Applications, 148, 113224.

Glover, F. , & Laguna, M. (1998). Tabu search. In: DU, DZ., & Pardalos, P. M. (Eds.), Handbook of Combinatorial Optimization (pp. 2093-2229). Boston: Springer.

Hao, J. K. (2012). Memetic Algorithms in Discrete Optimization. In: Neri, F., Cotta, C., & Moscato, P. (Eds.), Handbook of Memetic Algorithms, Studies in Computational Intelligence, Vol 379 (pp. 73-94). Berlin: Springer.

Hifi, M., & Michrafy, M. (2006). A reactive local search-based algorithm for the disjunctively constrained knapsack problem. Journal of the Operational Research Society, 57(6), 718-726.

Hifi, M., & Michrafy, M. (2007). Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem. Computers & operations research, 34(9), 2657-2673.

Kellerer, H., Pferschy, U., & Pisinger, D. (2004). Multidimensional Knapsack Problems. In: Knapsack Problems (pp. 235-283). Berlin: Springer.

Lai, X., & Hao, J. K. (2016). Iterated variable neighborhood search for the capacitated clustering problem. Engineering Applications of Artificial Intelligence, 56, 102-120.

Li, Z., Tang, L., & Liu, J. (2020). A memetic algorithm based on probability learning for solving the multidimensional knapsack problem. IEEE Transactions on Cybernetics, 52(4), 2284-2299.

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.

Lü, Z., & Hao, J. K. (2010). A memetic algorithm for graph coloring. European Journal of Operational Research, 203(1), 241-250.

Lü, Z., Hao, J. K., & Glover, F. (2011). Neighborhood analysis: a case study on curriculum-based course timetabling. Journal of Heuristics, 17(2), 97-118.

Moscato, P. (1999). Memetic algorithms: A short introduction. In New ideas in optimization (pp. 219-234). McGraw-Hill Ltd., UK.

Neri, F., & Cotta, C. (2012). Memetic algorithms and memetic computing optimization: A literature review. Swarm and Evolutionary Computation, 2, 1-14.

Perboli, G., Gobbato, L., & Perfetti, F. (2014). Packing problems in transportation and supply chain: new problems and trends. Procedia-Social and Behavioral Sciences, 111, 672-681.

Pferschy, U., & Schauer, J. (2017). Approximation of knapsack problems with conflict and forcing graphs. Journal of Combinatorial Optimization, 33(4), 1300-1323.

Salem, M. B., Hanafi, S., Taktak, R., & Abdallah, H. B. (2017). Probabilistic Tabu search with multiple neighborhoods for the Disjunctively Constrained Knapsack Problem. RAIRO-Operations Research, 51(3), 627-637.

Shi, X., Wu, L., & Meng, X. (2017). A new optimization model for the sustainable development: Quadratic knapsack problem with conflict graphs. Sustainability, 9(2), 1-10.

Spencer, K. Y., Tsvetkov, P. V., & Jarrell, J. J. (2019). A greedy memetic algorithm for a multiobjective dynamic bin packing problem for storing cooling objects. Journal of Heuristics, 25, 1-45.

Van der Merwe, D. J., & Hattingh, J. M. (2006). Tree knapsack approaches for local access network design. European Journal of Operational Research, 174(3), 1968-1978.

Wei, Z., & Hao, J. K. (2021). A threshold search based memetic algorithm for the disjunctively constrained knapsack problem. Computers & Operations Research, 136, 105447.

Wei, Z., Hao, J. K. Ren, J. & Glover F. (2023). Responsive strategic oscillation for solving the disjunctively constrained knapsack problem. European Journal of Operational Research (in press) doi.org/10.1016/j.ejor.2023.02.009

Wu, Q., Hao, J. K., & Glover, F. (2012). Multi-neighborhood tabu search for the maximum weight clique problem. Annals of Operations Research, 196(1), 611-634.

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), 2864-2870.

Zhou, F., He, Y., Ma, P., Lim, M. K., & Pratap, S. (2022). Capacitated disassembly scheduling with random demand and operation time. Journal of the Operational Research Society, 73(6), 1362-1378.

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), 1503-1505.

Zobolas, G. I., Tarantilis, C. D., & Ioannou, G. (2009). A hybrid evolutionary algorithm for the job shop scheduling problem. Journal of the Operational Research Society, 60(2), 221-235.