European Journal of Operational Research 309 (2023) 993–1009

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Responsive strategic oscillation for solving the disjunctively constrained knapsack problem

Zequn Wei[a], Jin-Kao Hao[b,*], Jintong Ren[c], Fred Glover[d]

[a] School of Economics and Management, Beijing University of Posts and Telecommunications, Beijing 100876, China
[b] LERIA, Université d'Angers, 2 Boulevard Lavoisier, Angers 49045, France
[c] Business School, Hohai University, Nanjing 210098, China
[d] Entanglement, Inc., Boulder, Colorado 80305, USA

ARTICLE INFO

Article history:
Received 4 October 2022
Accepted 7 February 2023
Available online 10 February 2023

Keywords:
Combinatorial optimization
Disjunctively constrained knapsack problem
Heuristics
Strategic oscillation

ABSTRACT

This paper presents a responsive strategic oscillation algorithm for the NP-hard disjunctively constrained knapsack problem, which has a variety of applications. The algorithm uses an effective feasible local search to find high-quality local optimal solutions and employs a strategic oscillation search with a responsive filtering strategy to seek still better solutions by searching along the boundary of feasible and infeasible regions. The algorithm additionally relies on a frequency-based perturbation to escape deep local optimal traps. Extensive evaluations on two sets of 6340 benchmark instances show that the algorithm is able to discover 39 new lower bounds and match all the remaining best-known results. Additional experiments are performed on 21 real-world instances of a daily photograph scheduling problem. The critical components of the algorithm are experimentally assessed.

1. Introduction

The disjunctively constrained knapsack problem (DCKP) belongs to the class of well-known knapsack problems and has received substantial attention over the past two decades. As a general knapsack model, the DCKP can be described as follows. Given a knapsack with a fixed capacity C and a set of items $V = \{1, \dots, n\}$ such that each item i has a profit p_i and a weight w_i, let $G = (V, E)$ be a conflict or incompatibility graph, where V is the set of n items and an edge $\{i, j\} \in E$ exists between i and j if items i and j cannot be placed into the knapsack simultaneously. Then the DCKP is to select a subset of non-conflicting items $S \subseteq V$ (i.e., $\{i, j\} \notin E$, $\forall i, j \in S$) such that the total profit $f(S)$ of the selected items is maximized, while the total weight $W(S)$ does not exceed the given knapsack capacity C. Formally, the DCKP can be written as follows.

$$(DCKP) \qquad \text{Maximize} \quad f(S) = \sum_{i=1}^{n} p_i x_i \tag{1}$$

$$\text{subject to} \quad W(S) = \sum_{i=1}^{n} w_i x_i \le C \tag{2}$$

$$x_i + x_j \le 1, \forall \{i, j\} \in E, \tag{3}$$

$$x_i \in \{0, 1\}, i = 1, \dots, n. \tag{4}$$

where x_i is a binary variable indicating whether item i is packed into the knapsack. Constraint (2) ensures that the capacity is not exceeded. Constraints (3) guarantee that only compatible items are selected and they are also called disjunctive constraints.

The DCKP is able to formulate a number of real-world applications related to public transportation (Hifi, Negre, Saadi, Saleh, & Wu, 2014), scheduling problems (Coniglio, Furini, & San Segundo, 2021; Kleinberg & Tardos, 2006; Van Bevern, Mnich, Niedermeier, & Weller, 2015), network communications (Basagni, 2001) and daily photograph scheduling for earth observation satellite (Vasquez & Hao, 2001; Wei & Hao, 2021). In addition to its practical significance, the DCKP plays an important role in combinatorial optimization, since it is closely related to several other popular NP-hard problems. For example, the DCKP reduces to the maximum weighted independent set problem (Johnson & Garey, 1979) when we ignore the knapsack capacity. The DCKP can also be regarded as a subproblem of the bin packing problem with conflicts (Jansen, 1999), the quadratic knapsack problem (Chen & Hao, 2017a) and the multiple-choice knapsack problem (Chen & Hao, 2014; Kellerer, Pfersch, & Pisinger, 2004).

Since its introduction by Yamada, Kataoka, & Watanabe (2002), various solution algorithms have been proposed in the literature. For instance, in addition to the first implicit enumeration algorithm introduced in Yamada et al. (2002), several exact algorithms have

* Corresponding author.
E-mail address: jin-kao.hao@univ-angers.fr (J.-K. Hao).

https://doi.org/10.1016/j.ejor.2023.02.009

been designed, which are based on branch-and-bound (Bettinelli, Cacchiani, & Malaguti, 2017; Coniglio et al., 2021; Hifi & Michrafy, 2007), branch-and-cut (Salem, Taktak, Mahjoub, & Ben-Abdallah, 2018) and dynamic programming (Gurski & Rehs, 2019; Pferschy & Schauer, 2009; 2017). These methods have been shown to be competitive on several DCKP instances. For example, the dynamic programming algorithm proposed in Pferschy & Schauer (2009) is able to efficiently solve problem instances whose conflict graph is tree, chordal graph or a graph with bounded treewidth. The branch-and-bound algorithm introduced in Coniglio et al. (2021) has achieved remarkable results on the majority of 6240 DCKP benchmark instances with a wide range of conflict densities (see Section 3). This algorithm is the best-performing exact algorithm for the DCKP. Furthermore, it has been shown in Coniglio et al. (2021) that the CPLEX solver with integer linear programming techniques has significant advantages in solving very sparse instances. However, given the NP-hard nature of the DCKP (Yamada et al., 2002), finding the optimal solution is computationally challenging in the general case.

Consequently, a variety of heuristic algorithms have been devised to solve the DCKP approximately. The existing heuristic approaches include the neighborhood search algorithm of Yamada et al. (2002), parallel neighbor algorithms of Hifi et al. (2014); Quan & Wu (2017a,b), scatter search algorithms of Hifi & Otmani (2011), Akeb, Hifi, & Mounir (2011), the iterative rounding algorithm of Hifi (2014), the probabilistic tabu search algorithm of Salem, Hanafi, Taktak, & Abdallah (2017) and the threshold search based memetic algorithm of Wei & Hao (2021). According to their computational results on the popular set of 100 benchmark instances (see Section 3), four algorithms represent the state-of-the-art heuristic approaches for the DCKP, i.e., the parallel neighborhood search algorithm (PNS) (Quan & Wu, 2017b), the cooperative parallel adaptive neighborhood search algorithm (CPANS) (Quan & Wu, 2017a), the probabilistic tabu search algorithm (PTS) (Salem et al., 2017) and the threshold search based memetic algorithm (TSBMA) (Wei & Hao, 2021). In particular, the TSBMA algorithm also reported remarkable results on many of the 6240 DCKP instances tested in Coniglio et al. (2021). However, it failed to achieve most of the best-known results on the 240 random sparse instances (denoted by *SR*).

As the above literature review shows, although considerable research has been devoted to developing solution methods for solving the DCKP, no existing single algorithm dominates the other approaches on all the DCKP benchmark instances in the literature (the no free lunch theorem Wolpert & Macready, 1997). In this work, we propose the first responsive strategic oscillation search algorithm (RSOA) for solving the DCKP. The algorithm adopts the strategic oscillation search framework with an original responsive mechanism to guide the search to oscillate around the boundary of feasible and infeasible regions. Previous investigations have disclosed the general idea of strategic oscillation (SOS, Glover, 1977) to be quite effective for a number of constrained optimization problems, such as the quadratic multiple knapsack problem (García-Martínez, Glover, Rodriguez, Lozano, & Martí, 2014), the capacitated hub location problem (Corberán, Peiró, Campos, Glover, & Martí, 2016), the maximally diverse grouping problem (Gallego, Laguna, Martí, & Duarte, 2013), the quadratic minimum spanning tree problem (Lozano, Glover, García-Martínez, Rodríguez, & Martí, 2014), the α-neighbor p-center problem (Sánchez-Oro, López-Sánchez, Hernández-Díaz, & Duarte, 2022), and the bipartite boolean quadratic programming problem (Wang, Wu, Punnen, & Glover, 2018; Wu, Wang, & Glover, 2020). In this work, we show the benefits of strategic oscillation for solving the DCKP. The main contributions of this work are summarized as follows.

First, unlike the existing DCKP algorithms that confine their search to feasible space, the proposed RSOA algorithm combines both feasible search and infeasible search to effectively examine candidate solutions. In particular, it relies on a responsive filtering strategy to guide the search to strategically oscillate between feasible and infeasible regions. This is the first time such a mixed search approach is investigated for solving the DCKP. The algorithm additionally employs an informed perturbation that is guided by frequency information, able to help the algorithm to get out of local optimum traps.

Second, we provide new lower bounds for 39 instances out of the 6340 benchmark instances in the literature. These bounds can be used for the future studies on the DCKP.

Third, we are making the code of our RSOA algorithm publicly available, to help researchers and practitioners solve problems that can be formulated by the DCKP model.

The rest of the paper is organized as follows. Section 2 presents the proposed RSOA algorithm. Section 3 reports experimental results and comparisons with the state-of-the-art DCKP algorithms. In Section 4, we conduct additional analysis to provide useful insights about the impacts of the key components of RSOA on its performance. Concluding remarks are given in the last section.

2. Responsive strategic oscillation search for the DCKP

The design of the responsive strategic oscillation search algorithm proposed in this work is motivated by three considerations. First, the strategic oscillation (SO) framework has demonstrated its power for solving a wide range of constrained optimization problems as noted in the introduction. Second, the threshold search method has proved to be quite successful for solving the DCKP (Wei & Hao, 2021). Third, existing algorithms for the DCKP only examine feasible solutions. In this work, we integrate threshold search (for examining feasible solutions) and the strategic oscillation framework (for examining both feasible and infeasible solutions) with an original responsive filtering strategy to create an effective search algorithm for the DCKP.

2.1. Main framework

In overview, the proposed responsive strategic oscillation search algorithm RSOA uses a feasible local search procedure to discover high-quality local optima within the feasible space and employs a responsive strategic oscillation search procedure to seek, from a discovered feasible local optimal solution, still better solutions by enlarging the search into infeasible space. When the search is judged to be trapped in a deep local optimum, a frequency based perturbation procedure is applied to lead the search to a distant unseen search region from which a new round of feasible search and strategic oscillation search starts. The main framework of the RSOA algorithm is shown in Algorithm 1.

The algorithm starts from a feasible initial solution generated by the random initialization procedure (line 3). After updating the overall best solution S^* (line 4), the search enters the main loop of the algorithm (lines 5–13). Each iteration of the loop first initializes the frequency counter (an n-vector) $\mathcal{F}$ (line 6), where $\mathcal{F}_i$ is used to record the frequency that each item i appears in visited infeasible solutions. Then the feasible local search procedure (line 7) is run to discover a high-quality local optimal solution. Following this, the search sequentially examines three neighborhoods N, replacing the current solution S by a non-prohibited neighboring solution S' as long as the latter meets a given quality threshold. Upon the termination of the feasible local search procedure (when the maximum number of iterations I_{max1} is reached), the responsive strategic oscillation search procedure (line 8) is started to explore both feasible and infeasible solutions. A responsive filtering strategy is employed to ensure the search visits only promising areas. After conditionally updating the best solution S^* found so far (lines 9–11),

Algorithm 1 Responsive strategic oscillation search for the DCKP.

```
1: Input: Instance I, cut-off time t_max, maximum number of iterations of feasible local search I_max1, maximum number of iterations of strategic oscillation search I_max2, a set of neighborhoods N, perturbation strength ρ.
2: Output: The overall best solution S* found.
3: S ← Random_Initialization(I)      /* S is the current solution */
4: S* ← S                            /* S* is the overall best solution */
5: while Time ≤ t_max do
6:    Initialize the frequency vector F
7:    S_b1 ← Feasible_Local_Search(S, N, I_max1)
8:    (S_b2, F) ← Strategic_Oscillation_Search(S_b1, N, I_max2, F)
9:    if f(S_b2) > f(S*) then
10:      S* ← S_b2
11:   end if
12:   S ← Frequency_Based_Perturbation(S_b2, ρ, F)
13: end while
14: return S*
```

RSOA applies the frequency based perturbation procedure (line 12) based on the frequency counter $\mathcal{F}$ to obtain a new starting solution which is used to seed the next round of feasible search and responsive strategic oscillation search. The algorithm stops when a given cut-off time $t_{\max}$ is reached.

2.2. Solution representation and notations

Given a DCKP instance with knapsack capacity C and conflict graph $G = (V, E)$, a candidate solution S for the instance is represented by $S = (x_1, \dots, x_n)$, where the binary variable x_i takes the value of 1 if item i is selected, and 0 otherwise. The solution S can equivalently be expressed as $S = <A, \bar{A}>$, where $A = \{q : x_q = 1 \text{ in } S\}$ is the set of selected items and $\bar{A} = \{p : x_p = 0 \text{ in } S\}$ is the set of non-selected items.

The unconstrained search space Ω of the given instance includes all non-empty subsets of V, i.e.,

$$\Omega = \{x : x \in \{0, 1\}^n\} \tag{5}$$

The feasible search space Ω^F includes all feasible candidate solutions satisfying the knapsack capacity and the conflict constraints, i.e.,

$$\Omega^F = \left\{ x \in \{0,1\}^n : \sum_{i=1}^{n} w_i x_i \le C; x_i + x_j \le 1, \forall \{i, j\} \in E, 1 \le i, j \le n, i \neq j \right\} \tag{6}$$

In this work, we use the knapsack constraint to delimit the boundary of the feasible and infeasible regions. We define thus the infeasible search space Ω^{IN} to include all candidate solutions satisfying the disjunctive constraints only, i.e.,

$$\Omega^{IN} = \left\{ x \in \{0,1\}^n : \sum_{i=1}^{n} w_i x_i > C; x_i + x_j \le 1, \forall \{i, j\} \in E, 1 \le i, j \le n, i \neq j \right\} \tag{7}$$

To evaluate the infeasibility degree of a solution S of Ω^{IN}, we define its critical value $CV(S)$ as follows.

$$CV(S) = max\left\{ 0, \sum_{i=1}^{n} w_i x_i - C \right\} \tag{8}$$

where C is the knapsack capacity. So for a feasible solution S (i.e., $\sum_{i=1}^{n} w_i x_i \le C$), $CV(S) = 0$, while for an infeasible solution $CV(S) > 0$.

Finally, the fitness of a candidate solution S in the search space Ω is given by the objective value $f(S)$ according to Eq. (1). Given two candidate solutions, if at least one of them is infeasible, they will be assessed by their critical value $CV(S)$ given by Eq. (8) (the smaller the better). Otherwise, the two feasible solutions of Ω^F will be assessed by the objective value $f(S)$ given by Eq. (1) (the larger the better).

2.3. Random initialization

The RSOA algorithm employs a simple random initialization procedure to obtain a feasible starting solution. Specifically, we randomly choose one item i among the non-selected items and add it to the solution if the knapsack constraint and the disjunctive constraints are satisfied. We continue to add new items until the knapsack capacity is reached or no candidate item is available. This initialization procedure has the advantage of being simple and able to provide diversified starting solutions. Our preliminary experiment shows that other more sophisticated procedures don't significantly do better than this simple initialization procedure.

2.4. Feasible local search procedure

The RSOA algorithm adopts a feasible local search procedure (FLS) to achieve an effective local optimization within the feasible search space Ω^F. FLS relies on the threshold search procedure of the TSBMA algorithm (Wei & Hao, 2021) and adopts for the first time the reverse elimination method (REM) (Glover, 1990) for its operation-prohibiting mechanism. Basically, FLS iteratively explores the feasible search space by replacing the current feasible solution by a feasible neighboring solution as long as its quality satisfies a given threshold. We first introduce the main components of the FLS procedure and then present its pseudo-code.

2.4.1. Move operator and neighborhood structure

The FLS procedure relies on three common move operators (*add*, *swap* and *drop*) for the DCKP, which are also applied to other knapsack problems (e.g., Chen & Hao, 2017b; Lai, Hao, Glover, & Lü, 2018; Lai, Hao, & Yue, 2019). Given a solution $S = <A, \bar{A}>$ (see Section 2.2), let mv be one of these move operators and $S' = S \oplus mv$ denote a *neighboring solution* obtained by applying mv to S. Let $Capa(S)$ be a predicate such that $Capa(S) = true$ if solution S satisfies the knapsack constraint, else $Capa(S) = false$. Let $Disj(S)$ be a predicate such that $Disj(S) = true$ if S satisfies the disjunctive constraints, else $Disj(S) = false$. The neighborhoods induced by the *add*, *swap* and *drop* operators can be expressed as follows.

$$N_1^F(S) = \left\{ S' : S' = S \oplus add(p), p \in \bar{A}, Capa(S'), Disj(S') \right\} \tag{9}$$

$$N_2^F(S) = \left\{ S' : S' = S \oplus swap(q, p), q \in A, p \in \bar{A}, Capa(S'), Disj(S') \right\} \tag{10}$$

$$N_3^F(S) = \left\{ S' : S' = S \oplus drop(q), q \in A, Capa(S'), Disj(S') \right\} \tag{11}$$

To quickly evaluate a neighboring solution, a streamlining technique is used to quickly identify the neighboring solution S' violating the disjunctive constraint. Specifically, we employ a conflict vector $\mathcal{V}$ of length n to record the number of conflicts of each candidate non-selected item with respect to the current solution S. Obviously, the value $\mathcal{V}_i$ of each selected item is 0. Once a move operation ($add(p)$, $swap(q, p)$ or $drop(q)$) is performed, the value $\mathcal{V}_j$ of each non-selected item j is updated as follows.

$$\mathcal{V}_j = \begin{cases} \mathcal{V}_j + 1, & \text{if } j \text{ is conflicting with item } p, \\ \mathcal{V}_j - 1, & \text{if } j \text{ is conflicting with item } q, \\ \mathcal{V}_j, & \text{otherwise.} \end{cases} \tag{12}$$

Thus, the time complexity of checking the disjunctive constraints is bounded to $O(n)$, where n is the number of items. (Without the streamlining technique, the time complexity would be $O(n^2)$.)

2.4.2. Reverse elimination method

The FLS procedure employs the reverse elimination method to implement the operation-prohibiting (OP) mechanism introduced in Wei & Hao (2021). REM is a general and efficient method to prevent the search from revisiting previously visited solutions. The procedure operates by employing a *running list* to record all the moves performed during the search process and a residual cancellation sequence (RCS) to trace back through previous solutions according to the *running list*. In our case, the running list size is set to the maximum number of iterations of the corresponding search procedure, i.e., I_{max1} for Algorithm 2 and I_{max2} for Algorithm 3.

Algorithm 2 Feasible local search procedure.

1: **Input**: Input solution S, neighborhoods N_1^F, N_2^F, N_3^F, maximum number of iterations I_{max1}.
2: **Output**: Best solution S_{b1} found during feasible local search.
3: $S_{b1} \leftarrow S$ /* Record the best solution found so far */
4: $T \leftarrow Calculate\_Threshold(S_{b1})$
5: $iter \leftarrow 0$
6: **while** $iter \leq I_{max1}$ **do**
7:   Examine sequentially neighborhoods $N_1^F(S)$, $N_2^F(S)$ and $N_3^F(S)$
8:   **for** N_1^F, N_2^F, N_3^F, let S' be a non-prohibited neighboring solution **do**
9:     **if** $f(S') \geq T$ **then**
10:       $S \leftarrow S'$ /* S' replaces S when the threshold T is satisfied */
11:       **break**;
12:     **end if**
13:   **end for**
14:   **if** $f(S) > f(S_{b1})$ **then**
15:     $S_{b1} \leftarrow S$ /* Update the best solution found during feasible local search */
16:     $T \leftarrow Calculate\_Threshold(S_{b1})$
17:     $iter \leftarrow 0$
18:   **else**
19:     $iter \leftarrow iter + 1$
20:   **end if**
21:   Update the residual cancellation sequence (RCS)
22: **end while**
23: **return** S_{b1}

Considering the fact that an encountered solution can be revisited only if it is a neighboring solution of the current solution, a move is prohibited if RCS can be reversed by this move. Interested readers are referred to Glover (1990) for a detailed presentation of this technique. Unlike the hash vector based OP mechanism of Wei & Hao (2021) where collisions may occur, REM can record exactly each previously visited solution with less computing resources. Our experiment indicates that REM has a better performance than the hash vector based OP mechanism of Wei & Hao (2021).

2.4.3. Main scheme of FLS procedure

As shown in Algorithm 2, the FLS procedure first performs some initialization tasks (lines 3–5). Then the search enters the 'while' loop (lines 6–22) to improve the input solution S iteratively by sequentially exploring three neighborhoods N_1^F to N_3^F (see Wei & Hao (2021) for more details). Each iteration of the 'while' loop performs three operations. First, a threshold search (lines 7–13) is applied to sequentially explore the three neighborhoods N_1^F, N_2^F, N_3^F (see Section 2.4.1). For this, a non-prohibited neighboring solution S' is accepted to replace the current solution S as long as S' reaches a dynamic quality threshold T. The threshold T is a value slightly below the current best objective value and is defined as $T = f(S_{b1}) - \delta$ where S_{b1} is the best solution found so far during the FLS procedure and δ is a positive valued parameter (see Section 3.2 for its settings). The threshold search exits if a neighboring solution is accepted or the three neighborhoods are exhausted. Second, the outcome of the threshold search is used to update the best solution S_{b1} found so far by the FLS procedure (lines 14–15) and recalculate the quality threshold T (line 16). Third, the residual cancellation sequence (RCS) needed by the operation-prohibiting mechanism is updated (see Section 2.4.2). The FLS procedure stops when the best solution S_{b1} cannot be improved during I_{max1} consecutive 'while' loops. Finally, S_{b1} is returned to seed the responsive strategic oscillation procedure to further improve the best solution found.

Algorithm 3 Strategic oscillation search procedure.

Input: Input solution S_{b1}, neighborhood N^+, maximum number of iterations I_{max2}, frequency counter $\mathcal{F}$.
Output: Best solution S_{b2} found.
Initialize the responsive critical limits CL_1, CL_2
$r_{IN} \leftarrow 0$ /* Initialize the infeasibility rate of neighboring solutions */
$S \leftarrow S_{b1}$ /* S is the current solution */
$S_{b2} \leftarrow S$ /* Record the best solution found so far */
$iter \leftarrow 0$
while $iter \leq I_{max2}$ **do**
  for Each non-prohibited neighboring solution S' in N^+ **do**
    Calculate the critical value $CV(S')$ by Eq. (8)
    Filter out unpromising neighboring solutions with r_{IN}, CL_1, CL_2, $CV(S')$
    Identify the non-prohibited neighboring solution S' with the minimum $CV(S')_{\min}$ value or the best S' with the same $CV(S')_{\min}$ value
  end for
  $S \leftarrow S'$
  if $(f(S) > f(S_{b2})) \wedge (W(S) \leq C)$ **then**
    $S_{b2} \leftarrow S$ /* A better feasible solution is encountered */
    $iter \leftarrow 0$
  else
    $iter \leftarrow iter + 1$
  end if
  Update the residual cancellation sequence (RCS)
  Update the infeasibility rate r_{IN}
  Update the responsive critical limits CL_1, CL_2
  if $(W(S) > C)$ **then**
    Update the frequency counter $\mathcal{F}$ /* An infeasible solution is encountered */
  end if
end while
return S_{b2}

2.5. Strategic oscillation search procedure

Strategic oscillation (Glover, 1977) was initially proposed with the purpose of crossing back and forth between feasible and infeasible space. By searching around the feasible and infeasible boundary, strategic oscillation is designed to find high-quality solutions that cannot be reached if the search is confined to feasible space.

In our case, we define the oscillation boundary by the knapsack capacity constraint and relax this constraint to allow the search to visit candidate solutions exceeding the knapsack capacity. Our strategic oscillation procedure SOS combines a responsive neighborhood filtering strategy (RNF) within strategic oscillation to traverse the boundary of the feasible space and infeasible space. The general SOS procedure follows two general principles. First, it is preferable to allow the search to transit between the spaces Ω^F and Ω^{IN}, rather than staying in a single space all the time. Second, as discussed in Glover & Hao (2011), a critical level is required to prevent the search from going too far into the feasible or infeasible region.

2.5.1. Main scheme of SOS procedure

As shown in Algorithm 3, after some initialization tasks (lines 3–7), the SOS procedure performs the 'while' loop to examine candidate solutions (lines 8–27). Each iteration of the loop calculates, according to Eq. (8), the critical value $CV(S')$ of each non-prohibited neighboring solution S' within the neighborhood N^+ (line 10), where N^+ is the union of three relaxed neighborhoods (see Section 2.5.2 for these relaxed neighborhoods.) Then the responsive neighborhood filtering strategy RNF (see below) is applied to filter out unpromising neighboring solutions (line 11). The non-prohibited neighboring solution S' with the minimum $CV(S')_{\min}$ value or the best S' (according to the objective value $f(S')$) with the same $CV(S')_{\min}$ value is then identified (line 12) and used to replace the current solution (line 14). Such a neighboring solution corresponds to a promising solution that is closest to the boundary of the feasible and infeasible regions. The best solution S_{b2} found during SOS is updated if S is better than S_{b2} (lines 15–20). Since we adopt the same REM method to record previously encountered solutions, RCS is updated after each solution transition (line 21). The infeasibility rate (r_{IN}) and the critical limits (CL_1 and CL_2) used for the RNF strategy (see below) are updated according to the current status of the search traversing the search spaces Ω^F and Ω^{IN} (lines 22–23). If the current solution S is infeasible, the frequency counter $\mathcal{F}$ (used in the perturbation procedure) is updated (lines 24–26). Finally, the SOS procedure terminates after I_{max2} consecutive iterations without improving the best solution S_{b2}.

2.5.2. Enlarged neighborhood structures

The SOS procedure explores the neighborhood N^+, which is the union of the neighborhoods N_1^+, N_2^+, N_3^+ induced by the *add*, *swap* and *drop* moves, such that the disjunctive constraints are always satisfied while the capacity constraint is not necessarily satisfied.

$$N_1^+(S) = \left\{S' : S' = S \oplus add(p),\ p \in \bar{A},\ Disj(S)\right\} \tag{13}$$

$$N_2^+(S) = \left\{S' : S' = S \oplus swap(q,p),\ q \in A,\ p \in \bar{A},\ Disj(S)\right\} \tag{14}$$

$$N_3^+(S) = \left\{S' : S' = S \oplus drop(q),\ q \in A,\ Disj(S)\right\} \tag{15}$$

Unlike the feasible neighborhoods N_1^F to N_3^F, a solution in $N^+ = N_1^+ \cup N_2^+ \cup N_3^+$ may or may not satisfy the capacity constraint. As a result, the SOS procedure explores both feasible and infeasible solutions.

2.5.3. Responsive filtering strategy

The current solution S may be infeasible since the knapsack capacity constraint is relaxed during the SOS procedure. As indicated in Glover & Hao (2011), the strategic oscillation is to guide the search to cross back and forth between feasible and infeasible regions. The responsive filtering strategy is designed to prevent the search from staying in the space Ω^F or Ω^{IN} for a long time. To keep the search from becoming trapped deeply in infeasible regions of Ω^{IN}, only neighboring solutions S' with a critical value satisfying $CV(S') \le CL_1$ are considered, where CL_1 is the so-called critical limit. The initial value of CL_1 is determined automatically by the maximum item weight and the minimum item weight of the given DCKP instance, setting $CL_1 = (argmax\{w_i : i = 1,\dots,n\} + argmin\{w_i : i = 1,\dots,n\})/2$. On the other hand, the critical limit CL_2 is employed to keep the search from becoming mired in the feasible region of Ω^F, i.e., only neighboring solutions S' verifying $CV(S') \ge CL_2$ are considered. The initial value of CL_2 is equal to 0 and CL_1 and CL_2 dynamically evolve as follows.

We inspect the feasibility of the 100 neighboring solutions most recently visited, to define the infeasibility ratio by $r_{IN} = N_{infea}/100$ where N_{infea} is the number of infeasible solutions among these 100 neighboring solutions. If $r_{IN} = 0$ (i.e., no infeasible solution is encountered), we increase the value of CL_2 by 1 to push the search towards infeasible regions. If, on the contrary, $r_{IN} = 1$ (all 100 solutions most recently encountered are infeasible), the value of CL_1 is reduced by 1 to push the search into the feasible region. CL_2 works only when $r_{IN} = 0$, while CL_1 is always active. As a result, the algorithm searches along the boundary of the spaces Ω^F and Ω^{IN} within a controlled range. The impact of the SOS procedure and the RNF strategy on the performance of the proposed algorithm is analyzed in Sections 4.1 and 4.2.

2.6. Frequency-based perturbation

The fact that the SOS procedure may accept infeasible solutions is beneficial for diversifying the search. However, the degree of this diversification may not be sufficiently great to allow the algorithm to escape deep local optima. Thus, we design a frequency-based perturbation to drive the search to enter new unseen regions, based on a frequency counter $\mathcal{F}$ that records the number of times each item appears in an infeasible solution encountered in the SOS procedure (Algorithm 3, lines 24–26). Then the perturbation procedure is carried out according to $\mathcal{F}$.

As shown in Algorithm 4, we first sort all the selected items in a descending order according to $\mathcal{F}$ (line 3). Then we remove the top $\rho \times |S_{b2}|$ most frequently selected items (line 4), where ρ is a parameter called perturb strength and $|S_{b2}|$ indicates the number of selected items in solution S_{b2}. Afterwards, new items are randomly added into the solution S until the knapsack capacity is reached (lines 5–12). The perturbed solution S will be used as the input solution for the next round of the RSOA algorithm.

Algorithm 4 Frequency-based perturbation procedure.

1: **Input**: Input solution S_{b2}, perturb strength ρ, frequency counter $\mathcal{F}$.
2: **Output**: The perturbed solution S.
3: Sort items according to $\mathcal{F}$
4: /* Remove the top $\rho \times |S_{b2}|$ most frequently selected items */
   $S \leftarrow Drop\_items(S_{b2}, \rho, \mathcal{F})$
5: **while** $W(S) \le C$ **do**
6:    Randomly pick one non-selected item j
7:    **if** $w(j) + W(S) \le C$ **then**
8:       $S \leftarrow Add\_one\_item(j, S)$
9:    **else**
10:       **break**;
11:    **end if**
12: **end while**
13: **return** S

2.7. Discussions

In this section, we discuss the differences between the proposed RSOA algorithm and the existing solution methods.

Although RSOA is derived from the strategic oscillation framework, it has the following distinguishing features. First of all, RSOA employs the original responsive filtering strategy to guide the

Table 1
Experimental environments of the compared algorithms.

Algorithm	Programming language	Processor	CPU (GHz)	RAM (GB)	Operating system
BCM (Bettinelli et al., 2017)	C	Intel Xeon E3-1220	3.1	16	Linux
CFS & ILP$_{1,2}$ (Coniglio et al., 2021)	C++/CPLEX 12.8	Intel Xeon E5-2690	3.0	128	Linux
PTS (Salem et al., 2017)	Java	Intel Pentium I5-6500	3.2	4	-
PNS (Quan & Wu, 2017b)	C++	-	3.06	-	-
CPANS (Quan & Wu, 2017a)	C++	Intel Xeon E5-4640	2.6	-	-
TSBMA (Wei & Hao, 2021)	C++	Intel Xeon E5-2670	2.5	2	Linux
RSOA (this work)	C++	Intel Xeon E5-2670	2.5	2	Linux

search to oscillate around the boundary of feasible and infeasible regions and reinforce the SOS framework. Second, the known SOS solution methods often adopt the tabu heuristic for local optimization (García-Martínez et al., 2014; Wang et al., 2018). In our case, RSOA uses the threshold search technique to enhance the performance of the SOS framework.

Compared to the TSBMA heuristic algorithm introduced in Wei & Hao (2021), which also uses the threshold search, RSOA distinguishes itself by the fact that RSOA explores both feasible and infeasible regions, while TSBMA only explores feasible regions. Moreover, RSOA adopts the REM method to record the encountered neighboring solutions, which consumes less computing resources than the solution-based tabu technique applied in TSBMA. Finally, for its local search, RSOA uses the SOS framework instead of the population-based memetic framework in TSBMA.

The way of exploring the neighborhoods in the FLS procedure (see Section 2.4) of our algorithm is similar to the variable neighborhood search (VNS) (Hansen, Mladenovic, & Moreno-Pérez, 2010), we analyze their relationships here. First, the FLS procedure adopts the threshold search technique to accept both improving and deteriorating neighboring solutions that meet the given threshold, while only improving neighboring solutions are accepted in the usual VNS framework. Second, VNS typically uses the best improvement strategy to explore the given neighborhoods, while our FLS procedure stops exploring the current neighborhood once it finds a neighboring solution satisfying the quality threshold. Finally, the frequency-based perturbation strategy of RSOA can be viewed as an informative shaking procedure for VNS.

3. Computational results and comparisons

To evaluate the performance of our RSOA algorithm, we perform extensive experiments on a large number of DCKP benchmark instances commonly tested in the literature. The computational results of the RSOA algorithm as well as the state-of-the-art reference algorithms are provided in this section.

3.1. Benchmark instances

Two sets (Set I and Set II) of 6340 popular DCKP benchmark instances are adopted for our experimental studies. Each instance can be characterized by three parameters: n represents the number of items, C is the knapsack capacity and η is the edge density of the corresponding conflict graph (conflict density), i.e., $\eta = 2m/n(n-1)$ where m is the number of disjunctive constraints.

Set I (Hifi & Michrafy, 2006; Quan & Wu, 2017a) includes 100 popular instances with n ranging from 500 to 2000, C from 1800 to 4000 and η from 0.04 to 0.40. These instances are denoted by xIy where $x = \{1, \ldots, 20\}$ and $y = \{1, \ldots, 5\}$. These instances were largely tested in the literature such as (Akeb et al., 2011; Hifi, 2014; Hifi et al., 2014; Hifi & Otmani, 2012; Quan & Wu, 2017a; 2017b; Salem et al., 2017; Wei & Hao, 2021).

Set II (Bettinelli et al., 2017; Coniglio et al., 2021) includes 6240 instances with n ranging from 60 to 1000, C from 150 to 15,000 and η from 0.001 to 0.90. These instances are further divided into ten classes, with 720 instances in each of the first eight classes (C1, C3, C10, C15, R1, R3, R10, R15) and 240 instances in each of the other two very sparse classes (SC, SR). In this paper, the four correlated instance classes C1 to C15 are denoted by *CC* and the other four random classes R1 to R15 are denoted by *CR*. These instances are relatively new and were tested in Bettinelli et al. (2017), Coniglio et al. (2021), Wei & Hao (2021).

More details of these two sets of benchmark instances are summarized in Wei & Hao (2021). All instances are available online[1].

Finally, like in Wei & Hao (2021), we also test our algorithm on the 21 benchmark instances from the real-life daily photograph scheduling problem (DPSP) of the earth observation satellite SPOT5 (Bensana, Lemaitre, & Verfaillie, 1999; Vasquez & Hao, 2001).

3.2. Experimental settings and reference algorithms

To assess the performance of our RSOA algorithm, we employ four state-of-the-art algorithms for the instances of Set I: the parallel neighborhood search algorithm (PNS) (Quan & Wu, 2017b), the cooperative parallel adaptive neighborhood search algorithm (CPANS) (Quan & Wu, 2017a), the probabilistic tabu search algorithm (PTS) (Salem et al., 2017) and the threshold search based memetic algorithm (TSBMA) (Wei & Hao, 2021). For the instances of Set II, we compare RSOA against three best-performing exact approaches: the two branch-and-bound algorithms BCM (Bettinelli et al., 2017) and CFS (Coniglio et al., 2021), the CPLEX solver (denoted by ILP) (Coniglio et al., 2021) as well as the best heuristic algorithm TSBMA (Wei & Hao, 2021). Both RSOA and TSBMA are heuristic algorithms, and consequently we use TSBMA as our main reference algorithm, which is the latest DCKP algorithm and has achieved excellent results on both benchmark sets.

The proposed RSOA algorithm was coded in C++[2] and compiled by the g++ compiler with the -O3 option. The main experimental environments for the compared algorithms are shown in Table 1. Following Wei & Hao (2021), the cut-off time was set to be 1000 seconds for Set I and 600 seconds for Set II. Corresponding to Wei & Hao (2021), we ran our RSOA algorithm 20 times independently for the instances of Set I and 10 times for the instances of Set II. According to the above experimental settings, the computing resources consumed by our experiment are the same or less than those in the existing literature.

The four parameters required by RSOA are shown in Table 2. The value of I_{max1} and I_{max2} are automatically determined according to the number of items n in each instance. The parameter δ to calculate the quality threshold T of the feasible local search adopts the same settings as in Wei & Hao (2021), where $MinP$ is the minimum profit of each instance and $rand(20)$ is a random integer in $[1 \ldots 20]$. For the perturbation strength ρ, we use the automatic parameter tuning tool Irace (López-Ibáñez, Dubois-Lacoste, Cáceres, Birattari, & Stützle, 2016) to determine its value. The candidate val-

[1] https://doi.org/10.17632/gb5hhjkygd.1.

[2] The code of our algorithm will be made available upon the publication of the paper at https://github.com/Zequn-Wei/DCKP_RSOA.

Table 2
Parameter settings of RSOA.

Parameters	Section	Description	Value
I_{max1}	2.4	*maximum number of iterations of FLS*	$10 \times n$
I_{max2}	2.5	*maximum number of iterations of SOS*	$5 \times n$
δ	2.4	*threshold parameter*	$n/10$ (for Set I) $MinP + rand(20)$ (for Set II)
ρ	2.6	*perturbation strength*	0.6

Table 3
Summarized comparisons of the RSOA algorithm against each reference algorithm on the 100 DCKP instances of Set I.

Algorithm pair	Instance (total)	Indicator	#Wins	#Ties	#Losses	$p - value$
RSOA vs. PTS (Salem et al., 2017)	$1Iy - 10Iy$ (50)	f_{best}	8	42	0	1.40e-2
		f_{avg}	45	5	0	5.34e-9
RSOA vs. PNS (Quan & Wu, 2017b)	$1Iy - 10Iy$ (50)	f_{best}	9	41	0	8.91e-3
	$11Iy - 20Iy$ (50)	f_{best}	27	23	0	5.56e-6
RSOA vs. CPANS (Quan & Wu, 2017a)	$1Iy - 10Iy$ (50)	f_{best}	0	50	0	NA
	$11Iy - 20Iy$ (50)	f_{best}	31	19	0	1.18e-6
RSOA vs. TSBMA (Wei & Hao, 2021)	$1Iy - 10Iy$ (50)	f_{best}	0	50	0	NA
		f_{avg}	12	34	4	7.44e-2
	$11Iy - 20Iy$ (50)	f_{best}	2	48	0	3.46e-1
		f_{avg}	15	16	19	3.47e-1

ues of ρ are from 0.1 to 0.9 with a step size of 0.1. This experiment is carried out on 8 instances from Set I with a cut-off time of 200 seconds. From the outcome of the experiment, the final value of ρ was set to 0.6. Unless stated otherwise, these values of the four parameters are consistently adopted for all our experiments reported in this paper and can be considered as the default settings of the RSOA algorithm.

3.3. Computational results and comparisons

The performance of the RSOA algorithm is assessed by comparing it with the reference algorithms introduced in Section 3.2, reporting computational results for the instances of sets I and II. A further comparison is made between RSOA and the main reference algorithm TSBMA by applying these algorithms to an additional set of real-world instances from the earth observation satellite SPOT5. All solution certificates reported in this section are available online.[3]

3.3.1. Computational results on the 100 instances of Set I

Table 3 summarizes the comparisons between the RSOA algorithm and each reference algorithm on the 100 instances of Set I, with the reference results obtained from Wei & Hao (2021). The first three columns of the table indicate the pairs of compared algorithms, the instance names and the performance indicators. Note that some of the performance indicators are not available in the literature. Columns 4 to 6 give the number of instances where RSOA attains a better, equal or worse result compared to each reference algorithm. The last column shows the $p - values$ obtained from the Wilcoxon signed-rank test (Wilcoxon, 1992) with a significance level of 0.05. NA in the last column indicates that the results of the compared algorithms are the same.

Table 3 shows that the RSOA algorithm dominates the reference algorithms PTS, PNS and CPANS in terms of the best objective value f_{best} and average objective value f_{avg}. Compared with the main reference algorithm TSBMA, our algorithm is able to obtain improved f_{best} values for two instances and obtains all the best-known results for the remaining 98 instances. However, instances in which the $p - values$ are greater than 0.05 mean that the difference between RSOA and TSBMA is not statistically significant in these cases. Detailed results of our algorithm on the instances of Set I are reported in the Appendix.

3.3.2. Computational results on the 6240 instances of Set II

Table 4 summarizes the comparisons of the RSOA algorithm against each reference algorithm on the 6240 instances of Set II. Column 1 shows the name of each instance class and column 2 indicates the number of instances corresponding to each class. The numbers of instances for which an algorithm reaches the known optimal solution are given in columns 3 to 7. Column 8 indicates the number of improved best results (new lower bounds) obtained by our RSOA algorithm. The last four columns give a detailed comparison between RSOA and the heuristic algorithm TSBMA, as well as the $p - values$ from the Wilcoxon signed-rank test. The summarized results for each column are presented in the last three rows.

From Table 4, we observe that for the 5760 instances *CC* and *CR* with conflict densities from 0.1 to 0.9, our RSOA algorithm obtains all the 5389 known optimal solutions. For the 480 very sparse instances *SC* and *SR* with conflict densities from 0.001 to 0.05, the RSOA algorithm achieves 419 out of 429 proved optimal solutions. Our algorithm successfully solves the remaining 10 instances (4 for *SC* and 6 for *SR*) with a longer cut-off time (2000 seconds) or with more repeated runs (100 times), as shown within parentheses in the table. In addition, RSOA discovers 37 new lower bounds, which have not previously been reported in the literature. Compared to the main reference heuristic algorithm TSBMA, our RSOA algorithm obtains 270 better, 5970 equal and no worse results for the 6240 instances of Set II. Notably, RSOA performs very well on the very sparse instance class *SR*, which proved challenging for the reference algorithm TSBMA. Finally, the small $p - value$ (< 0.05) discloses that the performance difference between RSOA and TSBMA is statistically significant for this second benchmark set.

3.3.3. Computational results on 21 real-life instances

To complete the comparison, we also tested the 21 benchmark instances from the real-life daily photograph scheduling problem (DPSP) of the earth observation satellite SPOT5 (Bensana et al., 1999; Vasquez & Hao, 2001). The number of items (photographs) in these DPSP instances ranges from 8 to 1057. As indicated in Wei & Hao (2021), this problem is equivalent to the DCKP model when ignoring the ternary constraints. As a result, the DPSP can be solved via any DCKP algorithm by adding a simple repair procedure. We adopted the same repair procedure and experimental

[3] https://github.com/Zequn-Wei/DCKP_RSOA.

Table 4
Summarized comparisons of the RSOA algorithm against each reference algorithm on the 6240 DCKP instances of Set II.

Class	Total	$ILP_{1,2}$ (Coniglio et al., 2021)	BCM (Bettinelli et al., 2017)	CFS (Coniglio et al., 2021)	TSBMA (Wei & Hao, 2021)	RSOA (this work)		RSOA vs. TSBMA (Wei & Hao, 2021)			
		Solved	Solved	Solved	Solved	Solved	New LB	#Wins	#Ties	#Losses	$p-value$
*C*1	720	**720**	**720**	**720**	**720**	**720**	0	0	720	0	NA
*C*3	720	584	**720**	**720**	716	**720**	0	**4**	716	0	6.79e-2
*C*10	720	446	552	**617**	**617**	**617**	**9**	**9**	711	0	7.47e-3
*C*15	720	428	550	**600**	**600**	**600**	**3**	**3**	717	0	1.09e-1
*R*1	720	**720**	**720**	**720**	717	**720**	0	**3**	717	0	1.09e-1
*R*3	720	680	**720**	**720**	**720**	**720**	0	0	720	0	NA
*R*10	720	508	630	**670**	669	**670**	**1**	**3**	717	0	1.09e-1
*R*15	720	483	590	**622**	**622**	**622**	**14**	**15**	705	0	6.53e-4
SC	240	**200**	109	156	194	196(**200**)	0	**2**	238	0	5.64e-1
SR	240	**229**	154	176	9	223(**229**)	**10**	**231**	9	0	2.20e-16
CC and *CR*	5760	4569	5201	**5389**	5381	**5389**	**27**	**37**	5723	0	3.20e-06
SC and *SR*	480	**429**	263	332	204	419(**429**)	**10**	**233**	247	0	2.20e-16
Grand total	6240	4998	5424	5721	5585	**5808**(**5818**)	**37**	**270**	5970	0	2.20e-16

settings as our main reference algorithm TSBMA for these real-life instances, i.e., 10 repeated runs with a cut-off time of one hour per run.

The computational results on the 21 DPSP instances are shown in Table 5. The first five columns give the name, number of photographs and number of conflict constraints of each instance. The symbol * in the first column indicates that the optimal value of any instance is known. Column 6 presents the known optimal values (indicated by *) or best-known values (*Opt*/*BKV*), which have been obtained by specially designed DPSP algorithms. Columns 7 to 12 show the detailed results of the proposed RSOA algorithm as well as the reference algorithm TSBMA. The $gap(\%)$ is defined by $(f_{best} - BKV)/BKV$ and the better results between the two compared algorithms are highlight in bold. The row #Avg shows the average value of each column. According to the f_{best} and f_{avg} values of RSOA and TSBMA, we give the $p - values$ from the Wilcoxon signed-rank test in the last row.

Table 5 discloses that RSOA performs very well compared to TSBMA, by achieving 11 better and 8 equal f_{best} values while obtaining a worse result for only two cases. Although TSBMA has better #Avg values in terms of f_{avg}, the $p - value$ of 5.71e-01 (> 0.05) indicates that there is no significant difference between TSBMA and RSOA for f_{avg}. From the row #Avg, we observe that RSOA achieves a smaller $gap(\%)$ than TSBMA (-0.742 against -1.025). With reference to the best-known values *BKV*, our RSOA algorithm matches ten optimal results against eight for TSBMA. In terms of f_{best}, the small $p - value$ (< 0.05) confirms RSOA performs significantly better than TSBMA. These computational results on the DPSP instances again show the utility of our RSOA algorithm for solving this real-world problem.

4. Additional experiments and discussions

To shed light on the RSOA algorithm's performance, this section presents a series of experiments to assess two main components: the strategic oscillation procedure and the responsive filtering strategy. We also show the first analysis on the distribution of feasible and infeasible solutions and the landscape of the DCKP.

4.1. Effectiveness of the strategic oscillation search procedure

As indicated in Section 2.5, the RSOA algorithm adopts a strategic oscillation procedure SOS to allow the search to explore infeasible regions within Ω^{IN}. In this section, we investigate the influence of SOS on the algorithm by creating a variant $RSOA_1^-$ of RSOA that disables the SOS procedure. As a result, $RSOA_1^-$ will only explore the feasible search space. For the experiment, we tested 240 very sparse instances *SR*, for which the RSOA algorithm demonstrated its effectiveness compared to the main reference algorithm TSBMA. Each instance was solved 10 times independently with a cut-off time of 600 seconds per run.

The experimental results are summarized in Table 6. The first two columns give the name of each group of 10 instances. The remaining columns show the number of instances where the RSOA algorithm obtains better (#Wins), equal (#Ties) or worse (#Losses) results compared to the $RSOA_1^-$ variant in terms of f_{best} and f_{avg}. We also present the $p - values$ from the Wilcoxon signed-rank test according to these two performance indicators. The last row shows a summary of each column.

The results of Table 6 clearly demonstrate the dominance of the RSOA algorithm over the $RSOA_1^-$ variant. Specifically, RSOA achieves better f_{best} values for 221 out of the 240 instances and no worse values compared to $RSOA_1^-$. The effectiveness of RSOA is even more evident when considering the f_{avg} indicator, according to which RSOA dominates $RSOA_1^-$ for all instances tested. The

Table 5
Computational results of the RSOA algorithm and comparison with the best-known values (BKV) on the 21 real-life instances of the SPOT5 daily photograph scheduling problem.

Instance	Photographs	Number of constraints			Opt/BKV	TSBMA (Wei & Hao, 2021)			RSOA (this work)		
		C2	C3_1	C3_2		f_{best}	f_{avg}	$gap(\%)$	f_{best}	f_{avg}	$gap(\%)$
8*	8	17	0	4	10*	10	10	0	10	10	0.000
54*	67	389	23	29	70*	70	69.60	0.000	70	70	0.000
29*	82	610	0	19	12,032*	12,032	12031.40	0.000	12,032	12030.40	0.000
42*	190	1762	64	57	10,8067*	108,067	108,067	0.000	108,067	108,067	0.000
28*	230	6302	590	58	56,053*	56,053	56,053	0.000	56,053	56,053	0.000
5*	309	13,982	367	250	115*	111	107.40	-3.478	**115**	108.40	0.000
404*	100	919	18	29	49*	49	47.80	0.000	49	49	0.000
408*	200	2560	389	64	3082*	3075	3074.20	-0.227	**3078**	3077.80	-0.130
412*	300	6585	389	122	16,102*	16,094	16092.40	-0.050	**16,096**	15496.20	-0.037
11*	364	9456	4719	164	22,120*	22,111	22109.10	-0.041	**22,117**	20,814	-0.014
503*	143	705	86	58	9096*	9096	8994.60	0.000	9096	7798.60	0.000
505*	240	2666	526	104	13,100*	13,096	12995.40	-0.031	**13,099**	11803.70	-0.008
507*	311	5545	2293	131	15,137*	15,132	15127.30	-0.033	**15,137**	13334.70	0.000
509*	348	7968	3927	152	19,215*	19,113	19110.60	-0.531	**19,115**	17720.90	-0.520
1401	488	11,893	2913	213	176,056	170,056	167960.50	-3.408	**171,062**	169660.80	-2.837
1403	665	14,997	3874	326	176,140	170,146	167848.80	-3.403	**172,141**	169245.70	-2.270
1405	855	24,366	4700	480	176,179	168,185	167882.40	-4.537	**170,175**	168484.10	-3.408
1021	1057	30,058	5875	649	176,246	**170,247**	168049.70	-3.404	169,240	168145.10	-3.975
1502*	209	296	29	102	61,158*	61,158	61,158	0.000	61,158	61157.80	0.000
1504	605	5106	882	324	124,243	124,238	124135.50	-0.004	**124,240**	124236.90	-0.002
1506	940	19,033	4775	560	168,247	**164,241**	161639.30	-2.381	164,239	159341.80	-2.382
#Avg	-	-	-	-	63453.19	62018.10	**61550.67**	-1.025	**62209.00**	61271.71	-0.742
$p-value$	-	-	-	-	-	2.48e-2	5.71e-01	-	-	-	-

Table 6
Effectiveness of the strategic oscillation search procedure on the performance of the RSOA algorithm (RSOA vs. $RSOA_1^-$).

Instance	Total	Indicator: f_{best}				Indicator: f_{avg}			
		#Wins	#Ties	#Losses	$p-value$	#Wins	#Ties	#Losses	$p-value$
SR_500_1000_r0.01	10	**9**	1	0	7.58e-3	**10**	0	0	5.06e-3
SR_500_1000_r0.02	10	**7**	3	0	1.78e-2	**10**	0	0	5.06e-3
SR_500_1000_r0.05	10	**4**	6	0	6.79e-2	**10**	0	0	5.06e-3
SR_500_2000_r0.01	10	**10**	0	0	5.03e-3	**10**	0	0	5.06e-3
SR_500_2000_r0.02	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
SR_500_2000_r0.05	10	**6**	4	0	2.77e-2	**10**	0	0	5.06e-3
SR_500_1000_r0.001	10	**10**	0	0	7.69e-3	**10**	0	0	5.06e-3
SR_500_1000_r0.002	10	**10**	0	0	7.63e-3	**10**	0	0	5.06e-3
SR_500_1000_r0.005	10	**8**	2	0	1.16e-2	**10**	0	0	5.06e-3
SR_500_2000_r0.001	10	**10**	0	0	5.03e-3	**10**	0	0	5.06e-3
SR_500_2000_r0.002	10	**9**	1	0	7.69e-3	**10**	0	0	5.06e-3
SR_500_2000_r0.005	10	**10**	0	0	5.03e-3	**10**	0	0	5.06e-3
SR_1000_1000_r0.01	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
SR_1000_1000_r0.02	10	**9**	1	0	7.63e-3	**10**	0	0	5.06e-3
SR_1000_1000_r0.05	10	**9**	1	0	7.69e-3	**10**	0	0	5.06e-3
SR_1000_2000_r0.01	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
SR_1000_2000_r0.02	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
SR_1000_2000_r0.05	10	**10**	0	0	5.03e-3	**10**	0	0	5.06e-3
SR_1000_1000_r0.001	10	**10**	0	0	5.01e-3	**10**	0	0	5.06e-3
SR_1000_1000_r0.002	10	**10**	0	0	5.03e-3	**10**	0	0	5.06e-3
SR_1000_1000_r0.005	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
SR_1000_2000_r0.001	10	**10**	0	0	5.03e-3	**10**	0	0	5.06e-3
SR_1000_2000_r0.002	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
SR_1000_2000_r0.005	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
Summary	240	**221**	19	0	2.20e-16	**240**	0	0	2.20e-16

small $p-values$ indicate that the performance differences are statistically significant. This experiment confirms the usefulness of the strategic oscillation search procedure adopted by the proposed algorithm.

4.2. Influence of the responsive filtering strategy

The responsive filtering strategy is another key component of the RSOA algorithm. To investigate its influence on the RSOA algorithm, we performed an experiment by disabling this strategy to produce the procedure denoted by $RSOA_2^-$. We adopted the same experimental settings and benchmark instances as in Section 4.1. The experimental results are reported in Table 7 with the same information as in Table 6.

Table 7 discloses that RSOA outperforms $RSOA_2^-$ by obtaining better f_{best} results for 187 out of the 240 instances and equal results for the 53 remaining instances. RSOA achieves better or equal f_{avg} values for most of the instances. The $p-values$ smaller than 0.05 indicate significant differences between the results of the RSOA algorithm and the $RSOA_2^-$ variant. We conclude that the responsive filtering strategy plays an important role for the performance of the RSOA algorithm.

Table 7
Influence of the responsive filtering strategy on the performance of RSOA algorithm. (RSOA vs. $RSOA_2^-$).

Instance	Total	Indicator: f_{best}				Indicator: f_{avg}			
		#Wins	#Ties	#Losses	$p-value$	#Wins	#Ties	#Losses	$p-value$
SR_500_1000_r0.01	10	**8**	2	0	1.17e-2	**10**	0	0	5.06e-3
SR_500_1000_r0.02	10	**7**	3	0	1.80e-2	**10**	0	0	5.06e-3
SR_500_1000_r0.05	10	0	10	0	NA	**3**	7	0	1.03e-1
SR_500_2000_r0.01	10	**9**	1	0	7.63e-3	**10**	0	0	5.06e-3
SR_500_2000_r0.02	10	**4**	6	0	6.79e-2	**10**	0	0	5.06e-3
SR_500_2000_r0.05	10	0	10	0	NA	0	10	0	NA
SR_500_1000_r0.001	10	**9**	1	0	7.69e-3	**10**	0	0	5.06e-3
SR_500_1000_r0.002	10	**9**	1	0	7.63e-3	**10**	0	0	5.06e-3
SR_500_1000_r0.005	10	**7**	3	0	1.80e-2	**10**	0	0	5.06e-3
SR_500_2000_r0.001	10	**9**	1	0	7.63e-3	**10**	0	0	5.06e-3
SR_500_2000_r0.002	10	**9**	1	0	7.47e-3	**10**	0	0	5.06e-3
SR_500_2000_r0.005	10	**9**	1	0	7.58e-3	**10**	0	0	5.06e-3
SR_1000_1000_r0.01	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
SR_1000_1000_r0.02	10	**10**	0	0	7.63e-3	**10**	0	0	5.06e-3
SR_1000_1000_r0.05	10	**7**	3	0	1.80e-2	**8**	1	1	1.09e-2
SR_1000_2000_r0.01	10	**10**	0	0	5.03e-3	**10**	0	0	5.06e-3
SR_1000_2000_r0.02	10	**10**	0	0	5.03e-3	**10**	0	0	5.06e-3
SR_1000_2000_r0.05	10	0	10	0	NA	**4**	3	3	7.35e-1
SR_1000_1000_r0.001	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
SR_1000_1000_r0.002	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
SR_1000_1000_r0.005	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
SR_1000_2000_r0.001	10	**10**	0	0	5.00e-3	**10**	0	0	5.06e-3
SR_1000_2000_r0.002	10	**10**	0	0	5.06e-3	**10**	0	0	5.06e-3
SR_1000_2000_r0.005	10	**10**	0	0	5.03e-3	**10**	0	0	5.06e-3
Summary	240	**187**	53	0	2.20e-16	**215**	21	4	2.20e-16

4.3. Analysis of the feasible and infeasible solutions

As noted in Section 1, this is the first work that employs strategic oscillation for solving the DCKP, allowing the search to transition between feasible and infeasible regions. To shed light on why such an approach is meaningful for solving the DCKP, we adopted the perspective of the study (Porumbel, Hao, & Kuntz, 2010) on the graph coloring problem and investigated the following fundamental questions.

(1) Are high-quality feasible solutions close to each other or separated by large distances?
(2) Are high-quality infeasible solutions close to each other or separated by large distances?
(3) Are high-quality feasible solutions close to or far away from high-quality infeasible solutions?
(4) Are high-quality solutions clustered in spheres of small diameter or scattered everywhere in the search space?

To answer these questions we executed our RSOA algorithm to solve four representative instances:18I2 (Set I, 2000 items), C10_8_0_3_r0.1 (Set II, class C10, 501 items), 1000_2000_r0.01-0 (Set II, class SR, 1000 items), 1000_2000_r0.005-0 (Set II, class SR, 1000 items). Our preliminary experiment indicates that these instances are hard for RSOA due to their large size or low conflict density. For each instance, we ran RSOA 10 times with the same cut-off time as indicated in Section 3.2. For each run, we recorded the top 50 best feasible solutions in terms of the largest objective values and the top 50 high-quality infeasible solutions in terms of the least critical values CV (defined in Section 2.2), thus obtaining 500 high-quality feasible solutions and 500 good infeasible solutions for each instance. Then we used the Hamming distance to measure the distance between each pair of solutions (feasible or infeasible).

The frequencies of the distances between each pair of solutions are shown in the histograms of Figs. 1, 2, 3. The X-axis in each sub-figure gives the distances between the solutions and the Y-axis indicates the number of paired solutions having a specific distance. Each sub-figure includes the distances of the 124,750 paired solutions for each instance, where the distance of 0 between two identical solutions is ignored. From these figures, we observe that the distances between high-quality solutions are quite small. Most of the paired solutions are distributed in a cluster with a diameter less than $0.1n$, where n is the number of items of each instance.

To visualize the spatial distribution of the sampled high-quality solutions in the solution space, we map the locations of these solutions from n-dimensional space into the Euclidean space R^3 by employing the multidimensional scaling (MDS) procedure (Kruskal, 1964) using the classical *cmdscale* algorithm to generate the solution distributions in R^3. The spatial distributions of these high-quality feasible and infeasible solutions are shown in Fig. 4. Each point on the sub-figure represents a high-quality solution, where a blue point represents a feasible solution and a red point represents an infeasible solution.

From Fig. 4, we observe that high-quality feasible solutions are grouped in several clusters, rather than scattered throughout the search space. Since there are boundaries between feasible and infeasible solutions, it makes sense to allow the search to cross back and forth over these boundaries to locate high-quality solutions. Moreover, we observe that if an infeasible solution is far from clusters of feasible solutions, there are usually no other high-quality feasible solutions around it. Therefore, to ensure the effectiveness of the search, we should prevent the search from going too deeply into infeasible regions. These findings provide foundations for the design of our RSOA algorithm regarding the use of strategic oscillation.

4.4. Instance space analysis

To further understand the behavior of our RSOA algorithm as well as the reference algorithms, we provide an instance space analysis (ISA, Smith-Miles, Baatar, Wreford, & Lewis, 2014). Basically, ISA helps to get insights into the algorithm performance on instances of different features. We employ the recently emerging toolkit MATILDA that provides visualized ISA results. Interested readers are referred to Muñoz & Smith-Miles (2020), Smith-Miles, Muñoz, & Neelofar (2020) for more details about MATILDA.

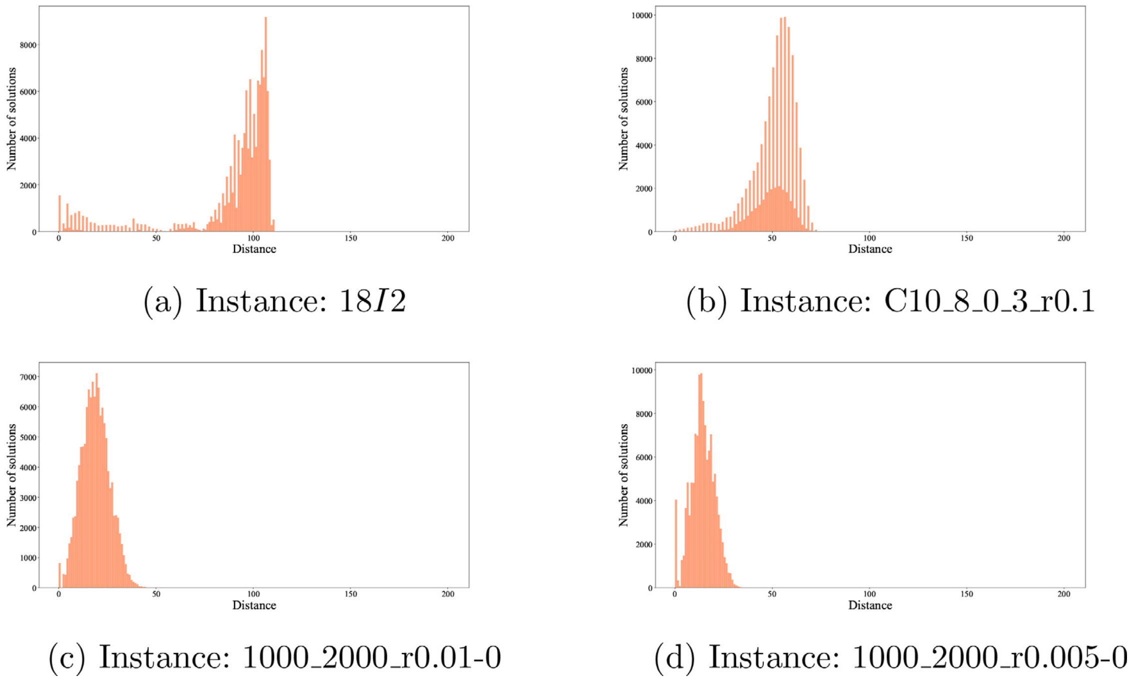


Fig. 1. Frequencies of distances between each pair of high-quality feasible solutions.

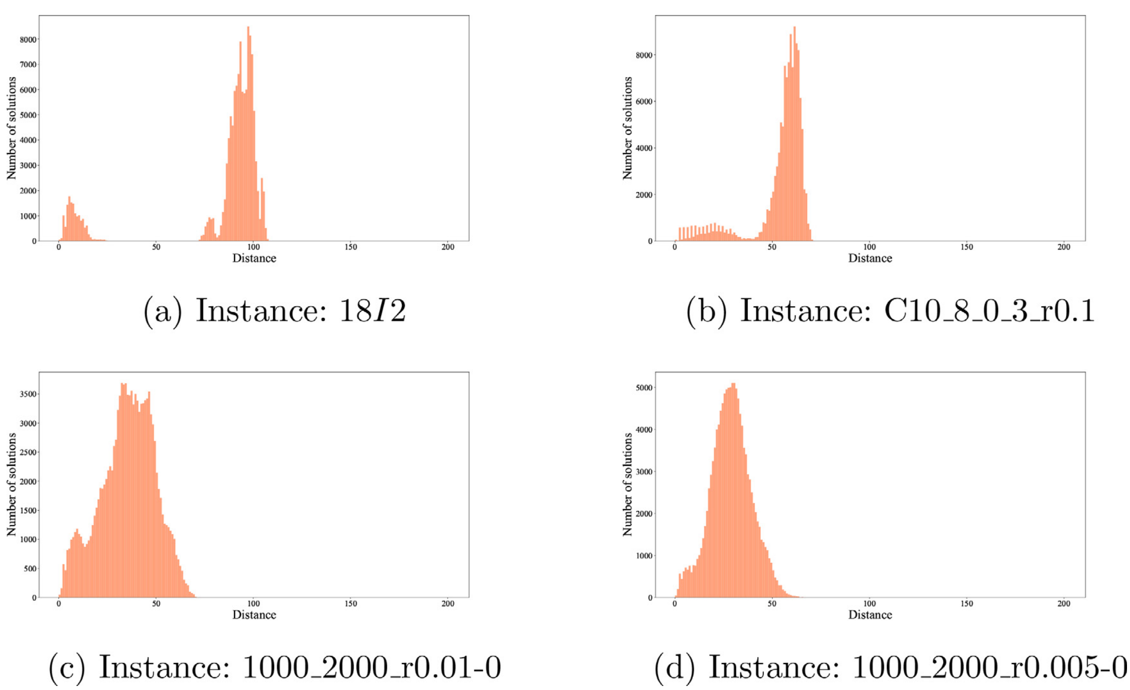
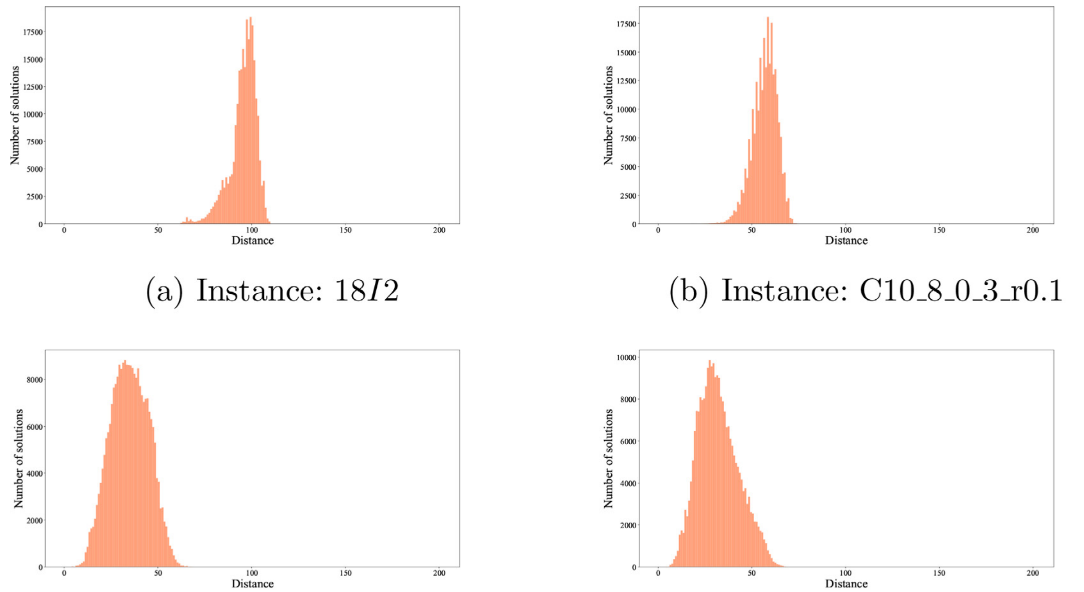


Fig. 2. Frequencies of distances between each pair of high-quality infeasible solutions.

In our case, the experiment is based on the 480 very sparse instances SC and SR, on which the performance of existing DCKP algorithms varies greatly. In this experiment, the proposed RSOA algorithm and the four leading reference algorithms introduced in Section 3.2 are considered, i.e., BCM (Bettinelli et al., 2017), CFS (Coniglio et al., 2021), ILP (Coniglio et al., 2021) and TSBMA (Wei & Hao, 2021). For the parameter settings of MATILDA, we set the value of the parameter 'opts.perf.epsilon' (threshold of good performance) to 0.01, while keeping the other parameters as their default values. For the description of instance characteristics, we adopted the conflict density feature (denoted by Density) defined in Section 3.1 and the 44 features for the 0-1 knapsack problem introduced in MATILDA.

Fig. 5 shows the distributions of the 480 benchmark instances and Density feature in the 2D instance space. The values of the Density feature in the right figure were scaled ranging from 0 (dark bule) to 1 (light yellow). From Fig. 5 we can observe that the two sets of instances SC and SR are clustered in two regions of the instance space and the Density values in each region are scattered.



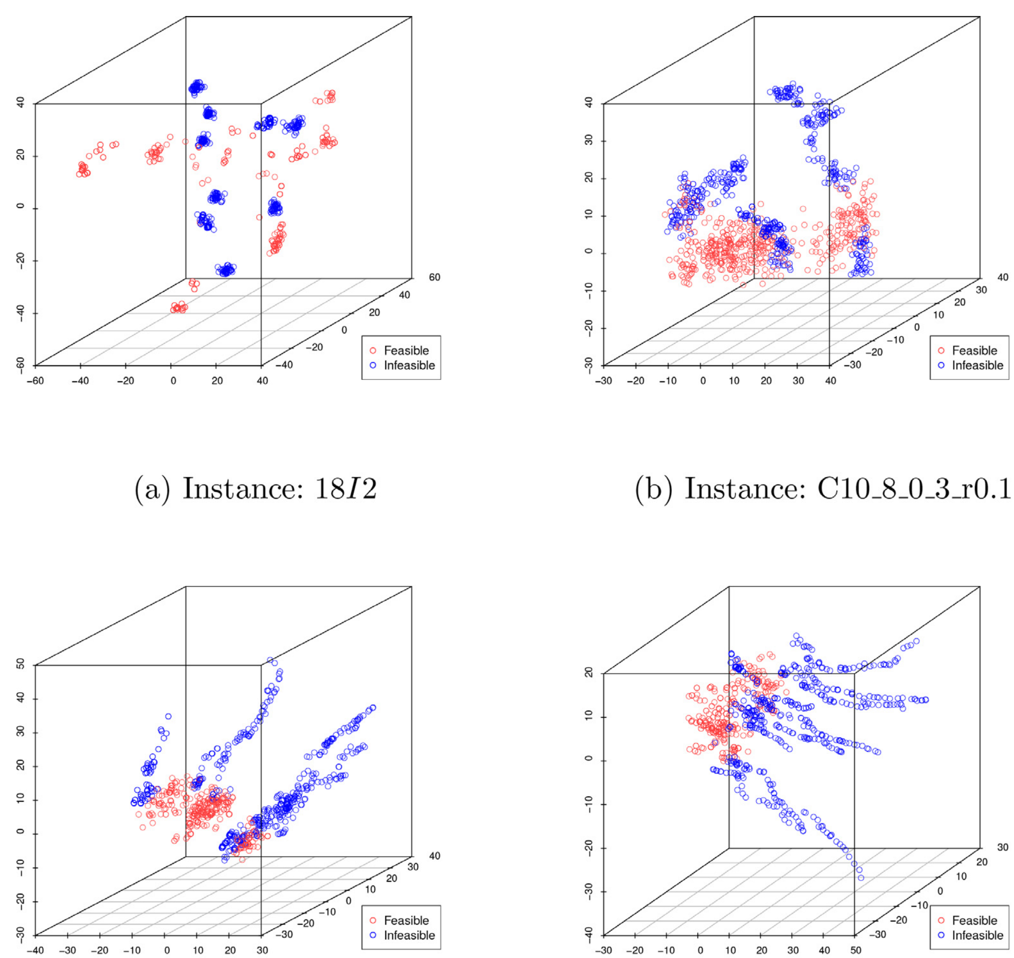
(a) Instance: $18I2$

(b) Instance: C10_8_0_3_r0.1

(c) Instance: 1000_2000_r0.01-0

(d) Instance: 1000_2000_r0.005-0

Fig. 3. Frequencies of distances between each pair of feasible and infeasible high-quality solutions.



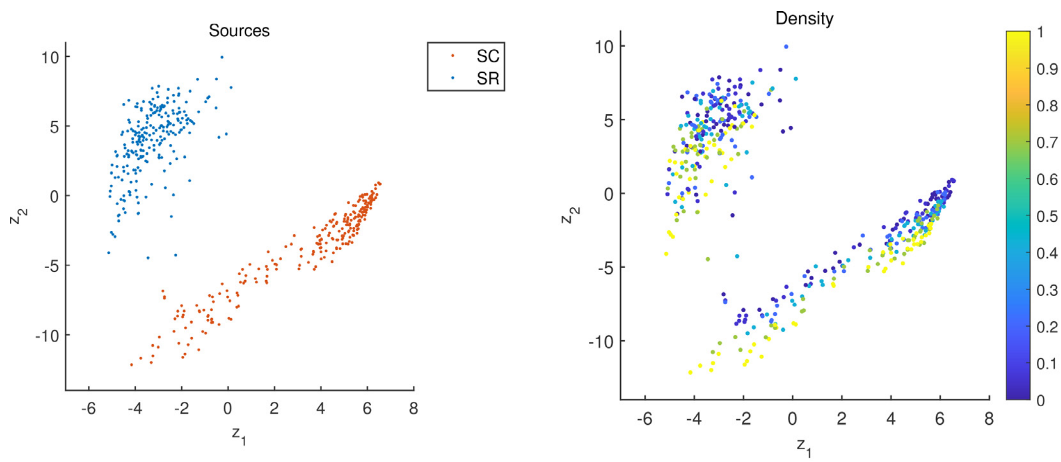
(a) Instance: $18I2$

(b) Instance: C10_8_0_3_r0.1

(c) Instance: 1000_2000_r0.01-0

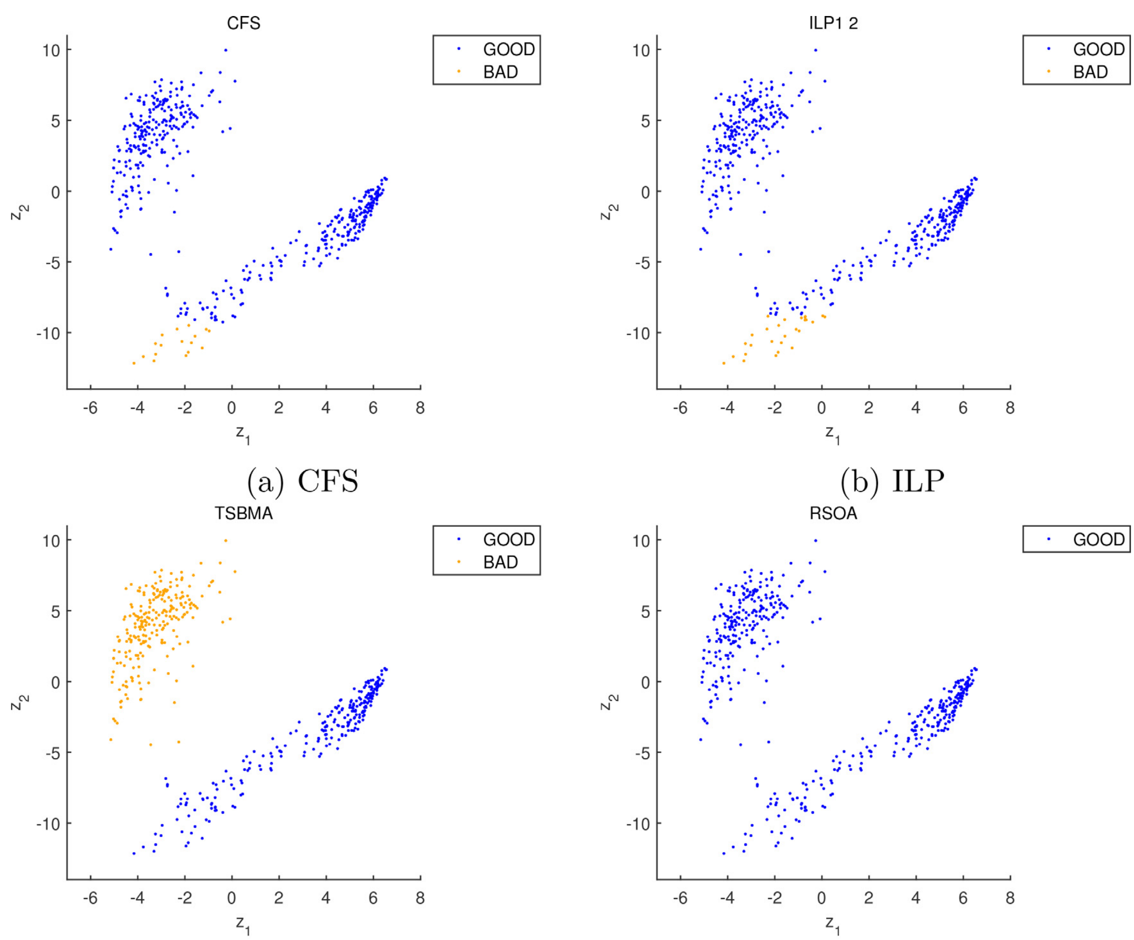
(d) Instance: 1000_2000_r0.005-0

Fig. 4. Spatial distributions of feasible and infeasible high-quality solutions.



(a) Distribution of instances (b) Distributions of Density feature

Fig. 5. Distributions of benchmark instances and Density feature in the 2D instance space.



(a) CFS (b) ILP (c) TSBMA (d) RSOA

Fig. 6. Prediction results for SVM model on the tested algorithms.

Figure 6 presents the prediction results for the Support Vector Machine (SVM) classification model of MATILDA on the tested algorithms. This figure helps us to better understand the strengths and weaknesses of each algorithm on the given instances. We can observe from Fig. 6 (d) that our RSOA algorithm is able to achieve a good performance on all the instance points. SVM predicts that such a result can not be achieved by the other reference algorithms. Specifically, from Fig. 6(a) and (b), we can find that the exact methods CFS and ILP will perform well on a majority of the instances, while show weaknesses on the instances of set SC with large Density values. Figure 6(c) discloses that TSBMA will show a good performance on the instances of set SC and have trouble on the instances of set SR. These outcomes of SVM predictions are consistent with the results reported in Table 4.

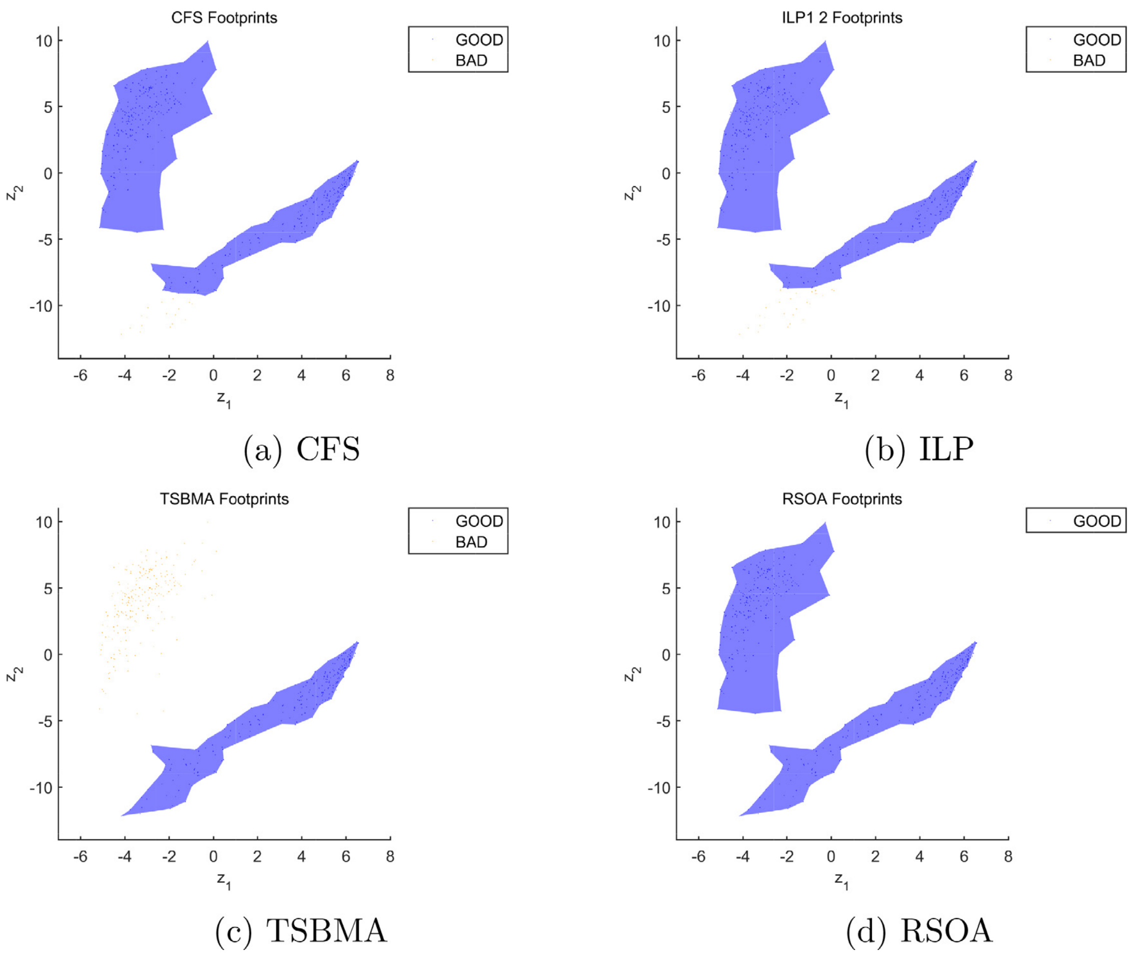


Fig. 7. Footprint analysis of the tested algorithms.

Finally, we present the predicted footprints of each algorithm to see the strengths and weaknesses of each algorithm in the 2D instance space. As shown in Fig. 7, footprint refers to an area (blue shading) where an algorithm is expected to achieve a good performance. From the footprints, we can gain insights about the algorithm performance on the instances of different features. The largest footprint of RSOA in Fig. 7 indicates that our RSOA algorithm is able to perform well in the instance space of SC and SR, while the other reference algorithms show some weaknesses.

5. Conclusions and perspectives

The disjunctively constrained knapsack problem (DCKP) plays an important role in combinatorial optimization due to its theoretical and practical significance. Previous solution methods have only focused on searching feasible regions. In this work, we have designed a responsive strategic oscillation search algorithm (RSOA) which explores for the first time both feasible and infeasible regions.

We have assessed the algorithm on two sets of 6340 DCKP benchmark instances as well as 21 real-world instances of the daily photograph scheduling problem of the earth observation satellite SPOT5. Our algorithm obtains the best-known results for all the 6340 DCKP benchmark instances (an outcome unmatched by all previous algorithms). Moreover, it finds improved best-known results for 2 instances of Set I and 37 instances of Set II. The proposed algorithm also shows its relevance for the real-world daily photograph scheduling problem, outperforming the best DCKP algorithm applied to this scheduling problem. In addition to these computational results, we have experimentally studied the key algorithmic components to shed light on their roles in the performance of the algorithm. We have also carried out the first investigation of the distribution of high-quality solutions, which discloses the merit of the rationale underlying our algorithm.

From the perspective of future work, two directions could be pursued. First, in the present work, we used the responsive filtering strategy to limit the search around the boundary between feasible and infeasible regions. It is worth investigating other possibilities to achieve this goal. Second, the proposed algorithm combines the strategic oscillation (SO) framework with threshold search to examine both feasible and infeasible solutions. Such an idea is general enough to be tested on other constrained optimization problems.

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 Fundamental Research Funds for the Central Universities (Grant No. 2022RC64).

Appendix A. Detailed results on the 100 DCKP instances of Set I

This appendix shows the computational results of our RSOA algorithm and the reference algorithms on the 100 DCKP instances of Set I (the results of the reference algorithms are from Wei & Hao, 2021). In Tables A.1 and A.2, the first column presents the name

Table A.1
Comparison between the RSOA algorithm and the four state-of-the-art algorithms on the 50 DCKP instances of Set I (1*Iy* to 10*Iy*).

Instance	*BKV*	PNS (Quan & Wu, 2017b)	CPANS (Quan & Wu, 2017a)		PTS (Salem et al., 2017)		TSBMA (Wei & Hao, 2021)				RSOA (this work)			
		f_{best}	f_{best}	$t_{avg}(s)$	f_{best}	f_{avg}	f_{best}	f_{avg}	std	$t_{avg}(s)$	f_{best}	f_{avg}	std	$t_{avg}(s)$
1*I*1	2567	2567	2567	17.133	2567	2567	2567	2567	0	163.577	2567	2567	0	76.647
1*I*2	2594	2594	2594	12.623	2594	2594	2594	2594	0	19.322	2594	2594	0	3.280
1*I*3	2320	2320	2320	14.897	2320	2320	2320	2320	0	6.060	2320	2320	0	2.450
1*I*4	2310	2310	2310	13.063	2310	2310	2310	2310	0	10.969	2310	2310	0	9.855
1*I*5	2330	2330	2330	20.757	2330	2321	2330	2330	0	63.663	2330	2330	0	8.358
2*I*1	2118	2118	2118	21.710	2118	2115.20	2118	2117.70	0.46	330.797	2118	2118	0	131.154
2*I*2	2118	2112	2118	129.390	2110	2110	2118	2111.60	3.20	705.755	2118	2116.80	2.86	426.448
2*I*3	2132	2132	2132	23.820	2119	2112.40	2132	2132	0	210.108	2132	2131.35	2.83	152.244
2*I*4	2109	2109	2109	31.377	2109	2105.60	2109	2109	0	14.182	2109	2109	0	13.391
2*I*5	2114	2114	2114	20.040	2114	2110.40	2114	2114	0	99.133	2114	2114	0	96.935
3*I*1	1845	1845	1845	34.683	1845	1760.30	1845	1845	0	3.780	1845	1845	0	1.513
3*I*2	1795	1795	1795	107.993	1795	1767.50	1795	1795	0	3.029	1795	1795	0	0.682
3*I*3	1774	1774	1774	22.490	1774	1757	1774	1774	0	3.585	1774	1774	0	7.950
3*I*4	1792	1792	1792	27.953	1792	1767.40	1792	1792	0	3.275	1792	1792	0	1.460
3*I*5	1794	1794	1794	34.820	1794	1755.50	1794	1794	0	9.159	1794	1794	0	2.663
4*I*1	1330	1330	1330	37.307	1330	1329.10	1330	1330	0	1.967	1330	1330	0	0.330
4*I*2	1378	1378	1378	40.827	1378	1370.50	1378	1378	0	3.926	1378	1378	0	0.947
4*I*3	1374	1374	1374	100.183	1374	1370	1374	1374	0	2.431	1374	1374	0	0.478
4*I*4	1353	1353	1353	26.930	1353	1337.60	1353	1353	0	4.167	1353	1353	0	2.803
4*I*5	1354	1354	1354	81.113	1354	1333.20	1354	1354	0	6.196	1354	1354	0	7.036
5*I*1	2700	2694	2700	122.637	2700	2697.90	2700	2700	0	78.215	2700	2700	0	56.455
5*I*2	2700	2700	2700	111.160	2700	2699	2700	2700	0	57.300	2700	2700	0	16.205
5*I*3	2690	2690	2690	73.640	2690	2689	2690	2690	0	18.566	2690	2690	0	14.793
5*I*4	2700	2700	2700	130.913	2700	2699	2700	2700	0	52.807	2700	2700	0	29.078
5*I*5	2689	2689	2689	279.377	2689	2682.70	2689	2687.65	3.21	289.966	2689	2685.40	4.41	529.143
6*I*1	2850	2850	2850	104.623	2850	2843	2850	2850	0	57.997	2850	2850	0	32.735
6*I*2	2830	2830	2830	93.887	2830	2829	2830	2830	0	76.883	2830	2830	0	61.418
6*I*3	2830	2830	2830	203.677	2830	2830	2830	2830	0	157.597	2830	2830	0	83.878
6*I*4	2830	2824	2830	160.587	2830	2824.70	2830	2830	0	328.817	2830	2830	0	165.146
6*I*5	2840	2831	2840	112.947	2840	2825	2840	2833.10	4.22	378.393	2840	2836.90	3.60	514.102
7*I*1	2780	2780	2780	186.970	2780	2771	2780	2779.40	1.43	483.465	2780	2780	0	207.449
7*I*2	2780	2780	2780	161.117	2780	2769.80	2780	2775.50	4.97	372.935	2780	2779	3.00	406.568
7*I*3	2770	2770	2770	136.310	2770	2762	2770	2768.50	3.57	393.018	2770	2768.50	3.57	386.366
7*I*4	2800	2800	2800	123.957	2800	2791.90	2800	2795.50	4.97	162.060	2800	2797.50	4.30	314.032
7*I*5	2770	2770	2770	149.933	2770	2763.60	2770	2770	0	290.591	2770	2770	0	158.003
8*I*1	2730	2720	2730	472.153	2720	2718.90	2730	2724	4.90	484.264	2730	2725.50	4.97	579.706
8*I*2	2720	2720	2720	109.373	2720	2713.60	2720	2720	0	214.760	2720	2720	0	80.060
8*I*3	2740	2740	2740	112.847	2740	2731.50	2740	2739.55	1.96	207.311	2740	2740	0	159.391
8*I*4	2720	2720	2720	253.230	2720	2712	2720	2715.35	4.85	518.579	2720	2718.55	2.84	475.087
8*I*5	2710	2710	2710	115.777	2710	2705	2710	2710	0	67.003	2710	2710	0	49.843
9*I*1	2680	2678	2680	134.023	2670	2666.90	2680	2679.70	0.71	316.210	2680	2680	0	217.079
9*I*2	2670	2670	2670	158.397	2670	2661.70	2670	2669.90	0.44	238.149	2670	2669.95	0.22	302.939
9*I*3	2670	2670	2670	123.280	2670	2666.50	2670	2670	0	161.176	2670	2670	0	140.003
9*I*4	2670	2670	2670	137.690	2663	2657.30	2670	2668.90	2.49	522.294	2670	2669.95	0.22	336.999
9*I*5	2670	2670	2670	131.247	2670	2662	2670	2670	0	98.124	2670	2670	0	94.351
10*I*1	2624	2620	2624	244.020	2620	2613.70	2624	2621.45	1.72	348.617	2624	2621.45	1.88	513.555
10*I*2	2642	2630	2630	144.867	2630	2620.80	2630	2630	0	182.474	2630	2630	0	175.915
10*I*3	2627	2620	2627	198.050	2620	2614.50	2627	2621.40	2.80	326.099	2627	2620.35	1.53	259.592
10*I*4	2621	2620	2620	148.997	2620	2609.70	2620	2620	0	105.609	2620	2620	0	120.01
10*I*5	2630	2627	2630	170.620	2627	2617.60	2630	2629.50	2.18	307.851	2630	2629	3.00	383.384
#Avg	2403.68	2402.36	2403.42	112.508	2402.18	2393.26	2403.42	2402.47	0.96	179.244	2403.42	2402.82	0.79	156.198

Table A.2
Comparison between the RSOA algorithm and the three state-of-the-art algorithms on the 50 DCKP instances of Set I (11*Iy* to 20*Iy*).

Instance	*BKV*	PNS (Quan & Wu, 2017b)	CPANS (Quan & Wu, 2017a)		TSBMA (Wei & Hao, 2021)				RSOA (this work)			
		f_{best}	f_{best}	$t_{avg}(s)$	f_{best}	f_{avg}	std	$t_{avg}(s)$	f_{best}	f_{avg}	std	$t_{avg}(s)$
11*I*1	4960	4950	4950	333.435	4960	4960	0	4.594	4960	4960	0	7.384
11*I*2	4940	4940	4928	579.460	4940	4940	0	14.305	4940	4940	0	18.914
11*I*3	4950	4920	4925	178.400	4950	4950	0	69.236	4950	4950	0	75.143
11*I*4	4930	4890	4910	320.067	4930	4930	0	139.197	4930	4930	0	137.875
11*I*5	4920	4890	4900	222.053	4920	4920	0	100.178	4920	4920	0	88.264
12*I*1	4690	4690	4690	230.563	4690	4687.65	2.22	416.088	4690	4688.45	1.56	588.182
12*I*2	4680	4680	4680	502.600	4680	4680	0	224.000	4680	4680	0	213.861
12*I*3	4690	4690	4690	229.116	4690	4690	0	215.103	4690	4690	0	112.227
12*I*4	4680	4680	4676	367.330	4680	4679.50	2.18	256.300	4680	4679.50	2.18	218.957
12*I*5	4670	4670	4670	487.563	4670	4670	0	79.190	4670	4670	0	150.989
13*I*1	4539	4533	4533	395.985	4539	4534.80	3.60	415.880	4539	4535.55	3.58	434.457
13*I*2	4530	4530	4530	573.718	4530	4528	4.00	361.229	4530	4528.55	3.46	328.305
13*I*3	4540	4530	4540	901.620	4540	4531	3.00	498.622	4540	4531.50	3.57	612.543
13*I*4	4530	4530	4530	315.076	4530	4529.15	2.29	366.951	4530	4529.95	0.22	288.374
13*I*5	4537	4537	4537	343.240	4537	4534.20	3.43	425.064	4537	4534.55	3.34	344.715
14*I*1	4440	4440	4440	483.156	4440	4440	0	205.733	4440	4439.95	0.22	241.302
14*I*2	4440	4440	4440	735.505	4440	4439.40	0.49	438.190	4440	4438.45	3.06	489.488
14*I*3	4439	4439	4439	614.733	4439	4439	0	146.119	**4440**	4439.40	0.49	490.091
14*I*4	4435	4435	4434	533.908	4435	4431.50	2.06	106.389	4435	4431.30	2.00	623.823
14*I*5	4440	4440	4440	473.448	4440	4440	0	160.900	4440	4439.50	2.18	201.209
15*I*1	4370	4370	4370	797.125	4370	4369.95	0.22	321.296	4370	4369.75	0.54	299.153
15*I*2	4370	4370	4370	676.703	4370	4370	0	181.021	4370	4369.90	0.44	235.340
15*I*3	4370	4370	4370	612.792	4370	4369.25	1.84	315.575	4370	4369.50	1.07	499.235
15*I*4	4370	4370	4370	649.398	4370	4369.85	0.36	424.873	4370	4369.50	0.92	405.624
15*I*5	4379	4379	4379	678.354	4379	4373.15	4.29	359.003	4379	4371.80	3.60	366.956
16*I*1	5020	4980	4980	286.130	5020	5020	0	205.964	5020	5020	0	135.759
16*I*2	5010	4990	4980	232.825	5010	5010	0	342.824	5010	5010	0	252.130
16*I*3	5020	5000	5009	199.880	5020	5020	0	155.070	5020	5020	0	122.188
16*I*4	5020	4997	5000	831.750	5020	5020	0	86.324	5020	5020	0	87.126
16*I*5	5060	5020	5040	982.970	5060	5060	0	32.837	5060	5060	0	214.644
17*I*1	4730	4730	4721	422.640	4730	4729.70	0.64	388.541	4730	4729.95	0.22	350.570
17*I*2	4720	4710	4710	248.770	4720	4719.50	2.18	300.275	4720	4719.50	2.18	195.371
17*I*3	4729	4720	4720	454.317	4729	4723.60	4.41	343.016	4729	4727.20	3.60	323.057
17*I*4	4730	4720	4720	432.900	4730	4730	0	288.961	4730	4730	0	242.384
17*I*5	4730	4720	4720	102.468	4730	4726.85	4.50	366.752	4730	4729.80	0.40	304.977
18*I*1	4568	4566	4566	225.010	4568	4565.80	3.40	269.545	4568	4567.25	1.30	488.992
18*I*2	4560	4550	4550	288.862	4560	4551.40	3.01	13.884	4560	4551.65	3.40	776.628
18*I*3	4570	4570	4570	328.555	4570	4569.40	2.20	466.748	4570	4568.40	3.56	407.121
18*I*4	4568	4560	4560	511.527	4568	4565.20	3.12	264.931	4568	4562.30	3.18	214.938
18*I*5	4570	4570	4570	651.887	4570	4567.95	3.46	572.589	4570	4564.35	4.46	532.574
19*I*1	4460	4460	4460	506.945	4460	4456.65	3.48	459.570	4460	4454.90	4.23	552.430
19*I*2	4460	4459	4459	666.900	4460	4453.25	4.17	307.224	4460	4452.85	4.09	287.043
19*I*3	4469	4460	4460	608.913	4469	4462.05	4.04	485.550	4469	4459.50	2.67	159.261
19*I*4	4460	4450	4450	476.755	4460	4453.20	3.89	430.824	4460	4452.10	3.32	455.170
19*I*5	4466	4460	4460	508.730	4466	4460.75	1.61	40.752	4466	4460.85	1.96	422.412
20*I*1	4390	4389	4388	957.410	4390	4383.20	3.36	929.372	4390	4383.30	3.52	598.771
20*I*2	4390	4390	4387	756.908	4390	4381.80	3.78	299.673	4390	4381.20	2.91	344.526
20*I*3	4389	4383	4389	966.010	4389	4387.90	2.77	568.988	**4390**	4385.75	4.07	165.729
20*I*4	4389	4388	4380	993.630	4389	4380.40	1.98	657.694	4389	4380.20	2.18	45.058
20*I*5	4390	4389	4389	772.495	4390	4386.40	4.05	646.570	4390	4383.95	4.44	448.259
#Avg	4614.14	4606.88	4607.58	513.011	4614.14	4611.83	1.80	303.390	4614.18	4611.64	1.76	311.991

of each instance and the second column gives the best-known values (BKV) reported in the literature. The remaining columns show the detailed results of the compared algorithms according to four performance indicators: the best objective value f_{best}, the average objective value f_{avg}, the standard deviation std and the average run time $t_{avg}(s)$ to reach the f_{best} value. The average values of each column are summarized in the last row. The two improved best results (new lower bounds for 14I3 and 20I3) discovered by RSOA are indicated in bold.

References

Akeb, H., Hifi, M., & Mounir, M. E. O. A. (2011). Local branching-based algorithms for the disjunctively constrained knapsack problem. *Computers & Industrial Engineering, 60*(4), 811–820.

Basagni, S. (2001). Finding a maximal weighted independent set in wireless networks. *Telecommunication Systems, 18*(1), 155–168.

Bensana, E., Lemaitre, M., & Verfaillie, G. (1999). Earth observation satellite management. *Constraints, 4*(3), 293–299.

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.

Chen, Y., & Hao, J.-K. (2014). A “reduce and solve” approach for the multiple-choice multidimensional knapsack problem. *European Journal of Operational Research, 239*(2), 313–322.

Chen, Y., & Hao, J.-K. (2017a). An iterated “hyperplane exploration” approach for the quadratic knapsack problem. *Computers & Operations Research, 77*, 226–239.

Chen, Y., & Hao, J.-K. (2017b). 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.

Corberán, Á., Peiró, J., Campos, V., Glover, F., & Martí, R. (2016). Strategic oscillation for the capacitated hub location problem with modular links. *Journal of Heuristics, 22*(2), 221–244.

Gallego, M., Laguna, M., Martí, R., & Duarte, A. (2013). Tabu search with strategic oscillation for the maximally diverse grouping problem. *Journal of the Operational Research Society, 64*(5), 724–734.

García-Martínez, C., Glover, F., Rodriguez, F. J., Lozano, M., & Martí, R. (2014). Strategic oscillation for the quadratic multiple knapsack problem. *Computational Optimization and Applications, 58*(1), 161–185.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences, 8*(1), 156–166.

Glover, F. (1990). Tabu search-part II. *ORSA Journal on Computing, 2*(1), 4–32.

Glover, F., & Hao, J.-K. (2011). The case for strategic oscillation. *Annals of Operations Research, 183*(1), 163–173.

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*(3), 411–432.

Hansen, P., Mladenovic, N., & Moreno-Pérez, J. A. (2010). Variable neighbourhood search: Methods and applications. *Annals of Operations Research, 175*(1), 367–407.

Hifi, M. (2014). An iterative rounding search-based algorithm for the disjunctively constrained knapsack problem. *Engineering Optimization, 46*(8), 1109–1122.

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.

Hifi, M., Negre, S., Saadi, T., Saleh, S., & Wu, L. (2014). A parallel large neighborhood search-based heuristic for the disjunctively constrained knapsack problem. In *2014 IEEE international parallel & distributed processing symposium workshops* (pp. 1547–1551). IEEE.

Hifi, M., & Otmani, N. (2011). A first level scatter search for disjunctively constrained knapsack problems. In *2011 International conference on communications, computing and control applications (CCCA)* (pp. 1–6). IEEE.

Hifi, M., & Otmani, N. (2012). An algorithm for the disjunctively constrained knapsack problem. *International Journal of Operational Research, 13*(1), 22–43.

Jansen, K. (1999). An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization, 3*(4), 363–377.

Johnson, D. S., & Garey, M. R. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman.

Kellerer, H., Pfersch, U., & Pisinger, D. (2004). *Knapsack problems*. Spinger.

Kleinberg, J. M., & Tardos, É. (2006). *Algorithm design*. Addison-Wesley.

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika, 29*(1), 1–27.

Lai, X., Hao, J., Glover, F. W., & Lü, Z. (2018). A two-phase tabu-evolutionary algorithm for the 0-1 multidimensional knapsack problem. *Information Sciences, 436-437*, 282–301.

Lai, X., Hao, J.-K., & Yue, D. (2019). Two-stage solution-based tabu search for the multidemand multidimensional knapsack problem. *European Journal of Operational Research, 274*(1), 35–48.

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.

Lozano, M., Glover, F., García-Martínez, C., Rodríguez, F. J., & Martí, R. (2014). Tabu search with strategic oscillation for the quadratic minimum spanning tree. *IIE Transactions, 46*(4), 414–428.

Muñoz, M. A., & Smith-Miles, K. (2020). Instance space analysis: A toolkit for the assessment of algorithmic power. Source code is available at https://github.com/andremun/InstanceSpace

Pferschy, U., & Schauer, J. (2009). The knapsack problem with conflict graphs. *Journal Graph Algorithms and Applications, 13*(2), 233–249.

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

Porumbel, D. C., Hao, J.-K., & Kuntz, P. (2010). A search space “cartography” for guiding graph coloring heuristics. *Computers & Operations Research, 37*(4), 769–778.

Quan, Z., & Wu, L. (2017a). Cooperative parallel adaptive neighbourhood search for the disjunctively constrained knapsack problem. *Engineering Optimization, 49*(9), 1541–1557.

Quan, Z., & Wu, L. (2017b). Design and evaluation of a parallel neighbor algorithm for the disjunctively constrained knapsack problem. *Concurrency and Computation: Practice and Experience, 29*(20), e3848.

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.

Salem, M. B., Taktak, R., Mahjoub, A. R., & Ben-Abdallah, H. (2018). Optimization algorithms for the disjunctively constrained knapsack problem. *Soft Computing, 22*(6), 2025–2043.

Sánchez-Oro, J., López-Sánchez, A., Hernández-Díaz, A., & Duarte, A. (2022). Grasp with strategic oscillation for the α-neighbor p-center problem. *European Journal of Operational Research*.

Smith-Miles, K., Baatar, D., Wreford, B., & Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research, 45*, 12–24.

Smith-Miles, K., Muñoz, M. A., & Neelofar (2020). Melbourne algorithm test instance library with data analytics (MATILDA). Aavailable at https://matilda.unimelb.edu.au

Van Bevern, R., Mnich, M., Niedermeier, R., & Weller, M. (2015). Interval scheduling and colorful independent sets. *Journal of Scheduling, 18*(5), 449–469.

Vasquez, M., & Hao, J.-K. (2001). A “logic-constrained” knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite. *Computational Optimization and Applications, 20*(2), 137–157.

Wang, Y., Wu, Q., Punnen, A. P., & Glover, F. (2018). Adaptive tabu search with strategic oscillation for the bipartite boolean quadratic programming problem with partitioned variables. *Information Sciences, 450*, 284–300.

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

Wilcoxon, F. (1992). Individual comparisons by ranking methods. In S. Kotz, & N. L. Johnson (Eds.), *Breakthroughs in statistics* (pp. 196–202). Springer.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82.

Wu, Q., Wang, Y., & Glover, F. (2020). Advanced tabu search algorithms for bipartite boolean quadratic programs guided by strategic oscillation and path relinking. *INFORMS Journal on Computing, 32*(1), 74–89.

Yamada, T., Kataoka, S., & Watanabe, K. (2002). Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Journal of Information Processing Society of Japan, 43*(9), 2864–2870.