RAIRO-Oper. Res. 51 (2017) 627–637
DOI: 10.1051/ro/2016049

RAIRO Operations Research
www.rairo-ro.org

PROBABILISTIC TABU SEARCH WITH MULTIPLE NEIGHBORHOODS FOR THE DISJUNCTIVELY CONSTRAINED KNAPSACK PROBLEM

Mariem Ben Salem[1], Saïd Hanafi[2], Raouia Taktak[3] and Hanêne Ben Abdallah[4]

Abstract. Given a set of items, each with a profit and a weight and a conflict graph describing incompatibilities between items, the Disjunctively Constrained Knapsack Problem is to select the maximum profit set of compatible items while satisfying the knapsack capacity constraint. We develop a probabilistic tabu search heuristic with multiple neighborhood structures. The proposed algorithm is evaluated on a total of 50 benchmark instances from the literature up to 1000 items. Computational results disclose that the proposed tabu search method outperforms recent state-of-the-art approaches. In particular, our approach is able to reach 46 best known solutions and discover 8 new best known solutions out of 50 benchmark instances.

Mathematics Subject Classification. 90C27, 90C59.

Received May 25, 2016. Accepted June 20, 2016.

1. Introduction

This paper deals with the Knapsack Problem with conflicts, also known as the Disjunctively Constrained Knapsack Problem introduced by Yamada *et al.* [39]. This is a variant of the classical $0-1$ Knapsack Problem (KP), where some items are in conflict with others. We are given a knapsack of capacity c, a set $N = \{1, 2, \dots, n\}$ of items, and a set E of pairs of items in conflict, *i.e.*, $E \subset \{(i, j) \in N \times N : i < j\}$. Each pair $(i, j) \in E$ means that items i and j are incompatible. Moreover, with each item $i \in N$ is associated a profit p_i and a weight w_i. The *Disjunctively Constrained Knapsack Problem* (DCKP) consists in determining a maximum-profit set of compatible items to be packed in the knapsack. A natural and compact Integer Linear Programming formulation for the DCKP makes use of a set of binary variables x_i associated with item $i \in N$, taking value 1 if i is packed

Keywords. Probabilistic Tabu Search, knapsack problem, weighted independent set, conflict graph.

[1] FSEGS/MIRACL, Université de Sfax, Tunisia. bensalem.mariem@gmail.com
[2] Université de Valenciennes, LAMIH – UMR CNRS 8201, France
[3] ISIMS/CRNS, Université de Sfax, Tunisia
[4] King Abdulaziz University, Jeddah, Saudi Arabia

in the knapsack, and 0 otherwise. The *standard* formulation of DCKP can be stated as follows:

$$\max \quad \sum_{i\in N} p_i x_i \tag{1.1}$$

$$\text{subject to} \quad \sum_{i\in N} w_i x_i \leq c, \tag{1.2}$$

$$x_i + x_j \leq 1 \qquad \forall (i,j) \in E, \tag{1.3}$$

$$x_i \in \{0,1\} \qquad \forall i \in N, \tag{1.4}$$

where (1.1) denotes the objective, (1.2) represents the knapsack capacity constraint, (1.3) are the disjunctive constraints, and (1.4) the integrality constraints of the variables. Note that other formulations are proposed for DCKP. Hifi and Michrafy [23], propose an *aggregated* formulation and Bettinelli *et al.* [3] propose an equivalent MIP based on the determination of a family of cliques in the conflict graph $G = (N, E)$.

Without loss of generality, we can assume that all input data c, p_i and w_i, for all $i \in N$ are non-negative integers; and $\max\{w_i : i \in N\} \leq c < \sum_{i\in N} w_i$ (otherwise, the variables can be fixed trivially).

The DCKP is an NP-hard combinatorial optimization problem. In fact, when no conflicts are considered, *i.e.*, $E = \emptyset$, the problem reduces to a $0-1$ Knapsack Problem which is proved to be NP-hard [32]. When the knapsack constraint is omitted, *i.e.*, $c \geq \sum_{i\in N} w_i$, the problem becomes the maximum weight Independent Set Problem known to be NP-hard as well [12]. It is well-known that there is no algorithm with a polynomial number of steps in the size of the instances known for solving any NP-Hard problem and that if one is found it would be a polynomial algorithm for all. Therefore, there is a need for heuristics able to quickly produce an approximate solution of high quality, or sometimes an optimal solution but without proof of its optimality. Note that there could be a very large number of local optima for some optimization problems. Usually, it is not hard to get a local optimum but it is not easy to find a global one. Indeed, such search methods may get trapped in a local optimum and miss the global one. Resolving local optima trap problems is one important issue and consists in finding a way to escape from a local optimum within some method.

In order to solve the DCKP, we propose to use a heuristic-based algorithm. We devise a probabilistic tabu search heuristic that makes use of multiple neighborhood structures. The proposed method is evaluated on a total of 50 benchmark instances from the literature. We show that our probabilistic tabu search method outperforms recent state-of-the-art methods. In particular, we have been able to reach 46 best known solutions and discover 8 new best known solutions out of the 50 benchmark instances.

Some definitions and notations will be presented in the following. Remark first that the DCKP can be presented by an undirected graph $G = (N, E)$, where N is the set of items, and E are edges representing conflicts between items. Let n and m be the cardinalities of N and E, respectively. A *clique* of a graph G is a complete subgraph of G. A subset of N is called an *independent set* if no two adjacent vertices belong to it. The *density* η of a graph $G = (N, E)$ is defined as the ratio between $|E|$ and the cardinality of the edge set of the complete graph having the same number of vertices. Let X and f respectively denote the set of feasible solutions and a real-valued objective function. Each solution x has an associated neighborhood $\mathcal{N}(x) \subseteq \{0,1\}^n$. Generally, a neighborhood $\mathcal{N}(x)$ is defined with respect to a given metric (or quasi-metric) function. Then, on one hand, a solution $x^* \in \mathcal{N}(x)$ is, with respect to neighborhood $\mathcal{N}(x^*)$, a local minimum for DCKP if $f(x^*) \geq f(x)$, $\forall x \in \mathcal{N}(x^*)$. On the other hand, a solution $x^* \in X$ is an optimal solution (global optimum) for DCKP if $f(x^*) \geq f(x)$, for all $x \in X$.

This paper is organized as follows. In the next section, we review some related work from the literature that deal with the DCKP or with some close variants of the problem. In the third section, we describe the heuristic used to solve the problem. The last section will be devoted to the computational experimentations and discussion of the obtained results.

2. Literature review

In this section, we give an overview on previous works that have dealt with DCKP, and related problems. As previously mentioned, the DCKP is an NP-hard optimization problem. In [33], Pferschy and Schauer show that the DCKP is even strongly NP-hard for general conflict graphs. The authors present pseudo-polynomial algorithms to solve the DCKP for two special classes of conflict graphs, namely graphs of bounded treewidth (including trees and series-parallel graphs) and chordal graphs (including interval graphs). Based on these algorithms, the authors derived Fully Polynomial Time Approximation Schemes (FPTAS) for the DCKP for these classes of graphs. The authors show, however, that the DCKP is strongly NP-hard for perfect conflict graphs and hence does not permit an FPTAS.

Yamada *et al.* [39] present a heuristic method as well as an implicit enumeration algorithm and an interval reduction method in order to solve to optimality the DCKP. A combination of all these methods allows the authors to solve instances with up to 1000 items with a density of incompatible items taking one of the following values $\{0.001, 0.002, 0.005, 0.01, 0.02\}$. In a further work, Senisuka *et al.* [36] propose a method that solves the DCKP using the Lagrangian relaxation combined with the pegging test for ordinary KP. An upper bound is derived using the Lagrangian relaxation, and then a lower bound is obtained by applying a 2-opt neighborhood search method. A pegging approach is then used in order to reduce significantly the size of the problem. Experiments are held on uncorrelated and weakly-correlated instances having items between 1000 and 16 000 and a density ranging in $\{0.1, 0.2, 0.4\}$. In [22], Hifi and Michrafi propose a reactive local search based algorithm in order to solve the DCKP. An initial solution is computed using two complementary greedy procedures. A degrading procedure is then applied in order to escape to local optima and to introduce a diversification in the search space. The authors also opt for a memory list used in order to forbid the repetition of configurations. Computational results prove the performance of the two versions of the algorithm compared to the results obtained by Cplex for instances of 500 items (densities 0.1 and 0.3), and 1000 items (densities 0.05, 0.07 and 0.09). Later, Hifi and Michrafy [23] propose several versions of an exact algorithm for the DCKP. In the first version, the authors apply a three-phase approach starting with a lower bound, then use a reduction procedure combined with an exact Branch-and-Bound algorithm. The second version is based on a combination of the reduction procedure and a dichotomous search in order to speed up the search process. And finally, in the third one, the authors enhance the previous algorithm using an equivalent ILP model for the problem, and dominating constraints as well as cover cuts. These algorithms have been tested on instances with $n = 1000$, a capacity c varying in $[2000, 4000]$, and a conflict density varying in $[0.007, 0.016]$.

In further works, Hifi *et al.* [1, 24–27] devise heuristic methods for the DCKP.

In [1, 24], the DCKP is solved using local branching based algorithm. In [24], a two-phase-based algorithm combining a rounding solution stage with a restricted exact solution procedure is proposed. In the first phase, the rounding procedure is used to fix a subset of the items of the LP. In the second phase, a local-branching restricted exact method is used to solve the reduced problem. In [1], three versions of local branching based algorithms are proposed. The first is a direct adaptation of the local branching method. The second combines local branching with a rounding solution procedure. And finally, the combined second algorithm is improved by the use of a diversification strategy. The three algorithms are proved to be efficient to solve a set of problem instances of the literature. In [26], the authors propose a version of the Scatter Search (SS) in order to solve the DCKP. The approach is based on the first level of SS using both starting phase and evolutionary phase. The heuristic is applied on an equivalent DCKP model enhanced with two families of valid inequalities. Other heuristic procedures have been later developed. In [25], Hifi *et al.* propose a parallel large neighborhood search-based heuristic to solve the DCKP. The approach introduces a large neighborhood search heuristic in a parallel model. This parallel programming aspect is designed using MPI (Message Passing Interface). The authors prove that their approach provides high quality solutions compared with the ones given by Cplex, and the ones previously obtained in the literature. Recently, Hifi *et al.* [27] propose a guided neighborhood search to solve the DCKP. The authors investigate the structure of the problem which is a combination of two combinatorial optimization problems, namely the maximum independent set and the classical $0-1$ KP. The proposed approach is a hybrid method that combines two local search procedures, a deterministic and a random one. The random local search is based on a modified ant colony optimization system. An exhaustive experimental study shows the efficiency of the used method on a benchmark of instances from the literature.

Along with heuristic based approaches, exact methods have been also used to solve the DCKP. In [3], Bettinelli *et al.* devise efficient Branch-and-Bound approaches to solve the DCKP. The authors develops a clique-based formulation for the problem with a tight relaxation used during the branching and bounding phases. They discuss several upper bounding procedures as well as efficient branching strategies. Both are combined in four Branch-and-Bound algorithms tested on instances inspired from instances of the Bin Packing Problem with conflicts.

Further works study close variants to the DCKP. In [2,7], the authors consider the two-dimensional DCKP. They propose a GRASP based heuristic. An initial solution is computed using a greedy randomized procedure. This solution is then improved using perturbation and diversification in order to best explore the research space.

The DCKP can also be seen as a subproblem of a more general problem, which is the *Disjunctively Constrained Bin Packing Problem* (DCBPP). In [35] Sadykov and Vanderbeck propose a Branch-and-Price algorithm to solve the bin packing problem with conflicts, and prove that the associated pricing subproblem is nothing but a DCKP. The authors solve efficiently the DCKP in special cases. They propose a dynamic programming algorithm to solve the DCKP when the conflict graph is an interval graph. Also, they develop a depth-first-search branch-and-bound approach when the conflict graph has no special structure. A similar approach has been previously proposed by Pisinger and Sigurd [34] who use a Dantzig−Wolfe decomposition for the two-dimensional bin packing problem, and prove that the pricing reduces to a two-dimensional knapsack problem.

The DCBPP, also known as the bin packing problem with conflicts, has also been studied in [8,9,13,28–31].

Among other interesting related problems, we can cite the multidimensional 0-1 Knapsack Problem. Several methods of resolution have been developed to solve the MKP. In particular, in [20] Hanafi and Fréville propose an efficient Tabu Search (TS) for the MKP. This TS is based on strategic oscillation and surrogate constraint information providing a balance between the intensification and diversification phases. More details about the TS approach are given in the next section. For a deeper idea about the MKP, the reader is referred to the following papers [4,10,11].

3. Probabilistic tabu search

In this section, we propose a Probabilistic Tabu Search for DCKP. The Tabu Search (TS) metaheuristic was proposed by Fred Glover (1986) [14], its principle is based on procedures designed to cross boundaries of feasibility or local optimality. TS guides a local search procedure to explore the solution space beyond local optimality by using adaptive memory to create a flexible search. TS starts from an initial solution, feasible or infeasible, and moves iteratively from one solution to its neighbor until a chosen termination criterion is satisfied. Tabu search allows moves that deteriorate the objective function value of the current solution x to be chosen. TS may be viewed as a dynamic neighborhood method since the moves are selected from a modified neighborhood $\mathcal{N}^*(x)$ of the current solution x. In fact, short term structures restrict the neighborhood $\mathcal{N}(x)$, *i.e.*, $\mathcal{N}^*(x) = \mathcal{N}(x) - TL$ where TL is the tabu list, while longer term structures expand the neighborhood $\mathcal{N}(x)$, *i.e.*, $\mathcal{N}^*(x) = \mathcal{N}(x) \cup ES$, where ES is the set of elite solutions. The tabu list TL keeps track of solutions attributes that have changed during the recent past where attributes correspond to information about solution properties (attributes) that change in moving from one solution to another. The adaptive memory of TS is used to create a balance between search intensification and diversification. Intensification strategies intensify the search around solutions historically found good while diversification strategies drive the search into regions dissimilar to those already examined. An extensive description of the TS metaheuristic can be found in [17,19].

The Probabilistic Tabu Search (PTS) is a variant of TS where the move is chosen probabilistically from the pool of those evaluated (or from a subset of the best members of this pool), weighting the moves so that those with higher evaluations are especially favored [15,16]. Several implementations of the probabilistic tabu search have been developed see for example Soriano and Gendreau [37], Crainic *et al.* [5], Glover and Lokketangen [18], or Xu *et al.* [38].

3.1. Initial solution

Several procedures can be designed to generate an initial solution for the DCKP by exploiting the fact that it is a combination of the maximum weighted independent set problem (WISP) and the classical $0{-}1$ KP. For example, Yamada *et al.* [39] adapt the greedy procedure applied to the KP (*cf.*, [6,32]) by adding items selected in decreasing order of $\frac{p_i}{w_i}$ while checking the disjunctive constraints. Similarly, constructive heuristics based on WISP can be designed by adding procedure to check the knapsack constraint. In this work, we propose a greedy heuristic where the selection of items take into account the profit, the weight, the degree of the items (number of conflicts they have), and also the density of the conflict graph. The pseudo code of the proposed heuristic is given in Algorithm 1. The greedy heuristic assumes that the items are sorted in the decreasing order of the ratio $\frac{p_i}{w_i+\alpha\eta d_i}$. $\delta(i) = \{j \in N : (i,j) \in E\}$, that is the set of items incompatible with i, $d_i = |\delta(i)|$ is the degree of i, η is the density of the conflict graph $G = (N, E)$, and α is a parameter. In our experimentation the parameter α is set to 0.1. The algorithm starts with an empty knapsack, *i.e.*, $x_i = 0$ for all $i \in N$. Then the unselected items are scanned in the given order and at time the current item can be inserted in the knapsack if the capacity constraint is not violated. Once an item i is selected, all items $j \in \delta(i)$ in conflict with i are forbidden to be selected in the next iterations.

Algorithm 1: Greedy heuristic.

Data: An instance of DCKP
Result: A feasible solution x.
Sort items such that $\frac{p_i}{w_i+\alpha\eta d_i} \geq \frac{p_{i+1}}{w_{i+1}+\alpha\eta d_{i+1}}$, $i = 1, 2, \ldots, n-1$;
for $i = 1$ to n do $x_i = 0$;
Set $\varDelta = c$, $F = N$;
while $F \neq \emptyset$ **do**
    select the first item i in F;
    if $w_i \leq \varDelta$ **then**
        $x_i = 1$, $\varDelta = \varDelta - w_i$, $F = F - \delta(i)$;
        for $j \in \delta(i)$ **do**
            $x_j = 0$;
    else
        $x_i = 0, F = F - \{i\}$;
return x ;

3.2. Neighborhood structure and its exploration

The definition of neighborhood structure is one of the most critical features of local search. Each neighbor x' is reached from the current solution x by applying a single or multiple elementary moves, *i.e.*, a series of local modifications of x. Neighborhood structures for the knapsack problem and the independent set problem usually involve the so-called "Add" and "Drop" elementary moves that set a variable x_i to one or zero (*i.e.*, complementing a variable x_i to $1 - x_i$).

In our PTS, we use neighborhoods in dynamic ways including multiple Add and Drop moves. Let k be a non-negative integer, the neighborhood of a solution x is a subset of the search space defined by:

$$\mathcal{N}^k(x) = \{x' \text{ obtained from } x \text{ by dropping } k \text{ items and adding other ones while feasiblity is maintained}\}.$$

In our experimentation, we choose during the process to vary the value of parameter k in $\{1, 3, 5, 7\}$. Since the size of the neighborhood $\mathcal{N}^k(x)$ becomes larger when the parameter k increases, candidate list strategies are used to restrict the number of solutions considered on a given iteration. In order to make a balance between intensification and diversification phases, we develop four types of candidate list strategies using semi-randomly sample from members of $\mathcal{N}^k(x)$. Each candidate list $CL^h(x)$, $h = 1, \ldots, 4$ is subset of $\mathcal{N}^k(x) \setminus TL$, where TL is the tabu list. In our experiments, we set the cardinality of each candidate list $|CL^h(x)| = 30$. Each element of $CL^h(x)$ for $h = 1, \ldots, 4$ is obtained by dropping k items from the current solution x. The main difference between these candidate lists arises in the phase of adding new items to the current solution. More precisely, after removing the k items, each solution of the four candidate lists is constructed as follows:

- $CL^1(x)$: The variables i fixed to 0 in the current solution x are sorted in decreasing order according to the ratio $\frac{p_i}{w_i + \alpha\eta d_i}$, and we add items while the feasibility is satisfied.
- $CL^2(x)$: The variables i fixed to 0 in the current solution x are sorted in decreasing order according to $p_i - \eta d_i$ and we add items while the feasibility is satisfied.
- $CL^3(x)$: The variables i fixed to 0 in the current solution x are sorted in decreasing order according to $p_i - \eta d_i$ and at each iteration, we add an item randomly from the l first ones while the feasibility is satisfied, where l is a parameter set to value 3 in our numerical experiments.
- $CL^4(x)$: We add items randomly while the feasibility is satisfied.

Note that the candidate lists $CL^1(x)$ and $CL^2(x)$ are used to reinforce the aggressive aspect of TS (*i.e.*, intensification phase), while $CL^3(x)$ and $CL^4(x)$ incorporate randomness to diversify the search (diversification phase). Algorithm 2 describes the strategy used to explore the neighborhood $\mathcal{N}^k(x)$ of a given current solution x. First a candidate list CL^h is selected with some probabilities. In our algorithm, candidate lists $CL^1(x)$, $CL^2(x)$, $CL^3(x)$ and $CL^4(x)$ are chosen with probabilities $\frac{1}{2}$, $\frac{1}{4}$, $\frac{3}{20}$ and $\frac{1}{10}$, respectively. Moreover, in order to support the aggressive character of the TS, we chose the best not tabu neighbor solution in each candidate list $CL^h(x)$. This means once a candidate list $CL^h(x)$ is chosen, we chose the neighborhood solution $x' \in CL^h(x)$ such that:

$$x' = \text{argmax}\{py : y \in CL^h(x) - TL\}.$$

Algorithm 2: Exploration of neighborhood $\mathcal{N}^k(x)$.

Data: A solution x, $\mathcal{N}^k$ and Tabu list TL.
Result: The neighborhood solution $x' \in \mathcal{N}^k(x)$.

Function `Exploration`($x, \mathcal{N}^k, TL$) ;
Generate a real value $r \in [0, 1]$ randomly;
if $r \in [0, 0.5]$ **then**
  set $h = 1$;
if $r \in [0.5, 0.75]$ **then**
  set $h = 2$;
if $r \in [0.75, 0.9]$ **then**
  set $h = 3$;
if $r \in [0.9, 1]$ **then**
  set $h = 4$;
Choose the best neighborhood $x' = \text{argmax}\{py : y \in CL^h(x) - TL\}$;
return x';

In our implementation, the tabu list consists of the pair (px, wx), representing the cost of the visited solution x and its consumption of the resource, respectively. It is managed in a static way where its size is fixed in our experiments to $|TL| = 10$.

3.3. Probabilistic tabu search algorithm

The probabilistic tabu search (PTS) algorithm starts with an initial solution x^0 obtained by the greedy constructive heuristic (see Algorithm 1). The initial solution becomes the current solution $x = x^0$ and it is inserted in the tabu list $TL = \{x^0\}$. The PTS algorithm cyclically explores the neighborhood structures $\mathcal{N}^k$ for $k \in \{1, 3, 5, 7\}$ one after another according to the established order. As soon as the neighborhood structure $\mathcal{N}^7$ is reached, the PTS resumes the search in the first neighborhood structure $\mathcal{N}^1$. More precisely, the neighborhood $\mathcal{N}^k$ is changed by setting $k = (k+2)$ *modulo* 8 after each q iterations (in our implementation, we set $q = 200$ iterations). At each iteration, the neighbor solution x' of the current solution is chosen from the current neighborhood structure $\mathcal{N}^k$ by calling the $exploration(x, \mathcal{N}^k, TL)$ function. To restrict the number of solutions examined in $\mathcal{N}^k(x)$, the exploration function selects a candidate list strategy with some probability. The process is repeated until a stopping condition is verified and the best solution x^* found so far is updated if an improvement occurs. The stopping criterion of the PTS algorithm used in our implementation is a time limit in seconds (see the next section on Computational Results).

Algorithm 3: Probabilistic Tabu Search Algorithm.

Data: An instance of DCKP
Result: The best feasible solution found so far x^*.
Construct an initial solution x^0 using a greedy heuristic;
Set $x^* = x^0$; $x = x^0$;
$TL = \{x^0\}$;
$iter = 0$;
Select an initial value for $k = 1$;
while *The stopped criterion is not met* **do**
  $x' = \text{Exploration}(x, \mathcal{N}^k, TL)$;
  $TL = TL + \{x'\}$;
  if $px' > px^*$ **then**
    $x^* = x'$;
  Set $x = x'$;
  $iter = iter + 1$;
  if $iter$ *modulo* $q = 0$ **then**
    $k = (k+2)$ *modulo* 8;
return x^*;

4. Computational results

We evaluated the Probabilistic Tabu Search (PTS) method described above on the set of instances generated by Hifi et Michrafi in [22]. The main characteristics of the instances are grouped in Table 1. The first column of Table 1 reports the names of the groups of tested instances. These groups are labeled jIy, $j \in \{1, 2, \ldots, 10\}$ (where $y \in \{1, \ldots, 5\}$). For each group of instances, the remaining columns of Table 1 give respectively the number n of items, the number $m = |E|$ of edges in the conflict graph, the density η of the graph, and the capacity c of the knapsack. Instances labeled from $1Iy$ to $4Iy$, where $y \in \{1, \ldots, 5\}$, represent the medium-size instances with $n = 500$, a capacity $c = 1800$, and a density η ranging from 0.1 to 0.4. Note that the number m of disjunctive constraints depends directly of the graph density. Second group of instances labeled from $5Iy$ to $10Iy$, where $y \in \{1, \ldots, 5\}$, contains larger instances with $n = 1000$, $c = 1800$, or 2000 and a density η varying from 0.05 to 0.10.

The PTS algorithm was coded in Java and tested on an Intel Pentium Core i5-6500, 3.2 GHz, 4Gb RAM.

TABLE 1. Characteristics of the tested instances [22,27].

Inst.	n	m	η	c
$1Iy$	500	12 475	0.10	1800
$2Iy$	500	24 950	0.20	1800
$3Iy$	500	37 425	0.30	1800
$4Iy$	500	49 900	0.40	1800
$5Iy$	1000	24 975	0.05	1800
$6Iy$	1000	29 970	0.06	2000
$7Iy$	1000	44 955	0.07	2000
$8Iy$	1000	39 960	0.08	2000
$9Iy$	1000	44 955	0.09	2000
$10Iy$	1000	49 950	0.10	2000

To calibrate the parameters of our PTS, several experiments carried by varying each parameter. The final choice of the parameter values that are used in our experiments are:

(1) The size of each candidate list is fixed to $|CL^h(x)| = 30$ for $h = 1, 2, 3, 4$.
(2) The l first Add moves in the definition of candidate list $LC^3(x)$, is fixed to $l = 3$.
(3) The probabilities β_h to choose the current candidate list $CL^h(x)$, are fixed as follows: $\beta_1 = \frac{1}{2}$, $\beta_2 = \frac{1}{4}$, $\beta_3 = \frac{3}{20}$ and $\beta_4 = \frac{1}{10}$.
(4) The size of the tabu list is fixed to $|TL| = 10$.
(5) The number of moves performed before changing the current neighborhood, is fixed to $q = 200$.
(6) The time limit T computed in seconds to run the PTS algorithm, is varying in the set $\{250, 500, 1000\}$.

To evaluate the performance of the proposed PTS algorithm, we conduct extensive experiments on the 50 benchmark instances. The assessment is performed by comparing our results to those of the state-of-the-art methods and the current best known results reported in the literature [21,27].
In order to avoid the random aspect of PTS algorithm and have an average overview, for each group of instance jIy, where $j \in \{1, \ldots, 10\}$ and $y \in \{1, \ldots, 5\}$, tests are computed 10 times.

In Table 2, the first column (*inst.*) corresponds to the name of the instance and the second column (*best*) represents the best known values obtained in [21] or [27]. The best known values are obtained by [27] except for two instances $10I2$ and $10I4$ which are provided by [21] (values in bold). For each time limit $T = 250$, $T = 500$, or $T = 1000$ seconds, we give the maximum, minimum, average and #*best* value obtained from the 10 previously reported values. The column #*best* corresponds to the number of times PTS found the best known value or discovers a new one. Values that are in bold correspond to the best known, for which our method (*i.e.*, the PTS) is at least as good as the best known results of methods proposed in [21,27]. Values that are marked with an asterix denote the instances where our method outperforms the previously mentioned ones.
From Table 2, we can deduce that our algorithm performs very well compared to the methods developed in [21,27]. For the majority of the instances, we have reached the best known values for the different groups of instances (values in bold). In fact, our PTS approach is able to reach 46 best known values over the 50 benchmarks, that is more than 90% of the tested instances. Moreover, for 8 over 50 of the instances groups, we have been able to get *new best known* solutions (values in bold with asterix). For these groups of instances, our PTS algorithm outperforms the Iterative Rounding Search (IRS) method developed in [21] and the Hybrid Guided Neighborhood Search (HGNS) algorithm of [27]. For only 4 over 50 instances, our algorithm gives a result not better than those of [21,27]. However, we have been very close to the best known values, *i.e.*, at most 6% from the best known values. Moreover, for 35 over the 50 groups of instances, we have been able to reach best known values with less than 250 seconds. For 8 over 50, we got best known values with a time limit set

TABLE 2.

		$T = 250$ s				$T = 500$ s				$T = 1000$ s			
inst.	Best	max	min	avg	$\#best$	max	min	avg	$\#best$	max	min	avg	$\#best$
1I1	2567	**2567**	2561	2566	8	**2567**	2563	2566.6	9	**2567**	2567	2567	10
1I2	2594	**2594**	2594	2594	10	**2594**	2594	2594	10	**2594**	2594	2594	10
1I3	2320	**2320**	2320	2320	10	**2320**	2320	2320	10	**2320**	2320	2320	10
1I4	2310	**2310**	2309	2309.9	9	**2310**	2310	2310	10	**2310**	2310	2310	10
1I5	2330	**2330**	2320	2323	3	**2330**	2320	2325	5	**2330**	2320	2321	1
2I1	2118	2115	2105	2113.5	0	**2118**	2110	2115.2	1	**2118**	2110	2115.2	1
2I2	2110	**2110**	2107	2109.5	8	**2110**	2110	2110	10	**2110**	2110	2110	10
2I3	2132	2113	2100	2107.4	0	2119	2109	2112.4	0	2119	2109	2112.4	0
2I4	2109	**2109**	2097	2103.1	1	**2109**	2100	2105.6	1	**2109**	2100	2105.6	1
2I5	2114	2110	2099	2105	0	**2114**	2110	2110.4	1	**2114**	2110	2110.4	1
3I1	1845	**1845**	1688	1734.2	1	**1845**	1700	1760.3	2	**1845**	1700	1760.3	2
3I2	1795	**1795**	1629	1734.2	3	**1795**	1672	1767.5	4	**1795**	1672	1767.5	4
3I3	1774	**1774**	1712	1742.3	3	**1774**	1739	1757	3	**1774**	1739	1757	3
3I4	1792	**1792**	1511	1680.7	1	**1792**	1663	1767.4	6	**1792**	1663	1767.4	6
3I5	1794	1775	1706	1742.5	0	1772	1719	1750.9	0	**1794**	1719	1755.5	1
4I1	1330	**1330**	1321	1327.3	7	**1330**	1321	1329.1	9	**1330**	1321	1329.1	9
4I2	1378	**1378**	1317	1371.9	9	**1378**	1303	1370.5	9	**1378**	1303	1370.5	9
4I3	1374	**1374**	1327	1352.6	5	**1374**	1334	1370	9	**1374**	1334	1370	9
4I4	1353	**1353**	1298	1321.8	1	**1353**	1298	1337.6	2	**1353**	1298	1337.6	2
4I5	1354	1330	1298	1321.3	0	**1354**	1330	1333.2	1	**1354**	1330	1333.2	1
5I1	2690	**2700***	2680	2694	9	**2700***	2690	2697.9	10	**2700***	2690	2697.9	10
5I2	2700	**2700**	2690	2696.9	6	**2700**	2690	2699	9	**2700**	2690	2699	9
5I3	2690	**2690**	2690	2690	10	**2690**	2680	2689	9	**2690**	2680	2689	9
5I4	2700	**2700**	2680	2693	4	**2700**	2690	2699	9	**2700**	2690	2699	9
5I5	2680	**2680**	2670	2675.9	5	**2689***	2680	2682.7	10	**2689***	2680	2682.7	10
6I1	2850	**2850**	2840	2845	5	**2850**	2830	2843	5	**2850**	2830	2843	5
6I2	2829	**2830***	2820	2823.8	4	**2830***	2820	2829	9	**2830***	2820	2829	9
6I3	2830	**2830**	2820	2824	4	**2830**	2830	2830	10	**2830**	2830	2830	10
6I4	2829	2820	2810	2817	0	**2830***	2820	2824.7	4	**2830***	2820	2824.7	4
6I5	2830	**2840***	2820	2826.8	5	**2840***	2820	2825	4	**2840***	2820	2825	4
7I1	2780	**2780**	2760	2768	1	**2780**	2760	2770	1	**2780**	2760	2771	2
7I2	2770	**2770**	2760	2766	6	**2770**	2760	2763.5	3	**2780***	2750	2769.8	8
7I3	2760	**2770***	2750	2759	7	**2770***	2750	2762	9	**2770***	2760	2762	10
7I4	2800	2790	2770	2785	0	2790	2770	2783.6	0	**2800**	2789	2791.9	2
7I5	2760	**2770***	2750	2757	6	**2770***	2759	2762.8	9	**2770***	2750	2763.6	9
8I1	2720	**2720**	2707	2711	1	**2720**	2710	2718.9	8	**2720**	2710	2718.9	8
8I2	2720	2710	2690	2700.6	0	**2720**	2700	2713.6	3	**2720**	2700	2713.6	3
8I3	2740	2730	2660	2713	0	**2740**	2720	2731.5	4	**2740**	2720	2731.5	4
8I4	2720	2719	2690	2704.7	0	**2720**	2710	2712	2	**2720**	2710	2712	2
8I5	2710	**2710**	2688	2698.4	1	**2710**	2700	2705	5	**2710**	2700	2705	5
9I1	2670	**2670**	2640	2659.3	1	**2670**	2660	2666.9	6	**2670**	2660	2666.9	6
9I2	2666	2660	2648	2655.3	0	**2670***	2659	2661.7	2	**2670***	2659	2661.7	2
9I3	2670	**2670**	2644	2656.1	1	**2670**	2660	2666.5	5	**2670**	2660	2666.5	5
9I4	2668	2657	2630	2647.4	0	2663	2650	2657.3	0	2663	2650	2657.3	0
9I5	2670	**2670**	2646	2655.2	1	**2670**	2650	2662	3	**2670**	2650	2662	3
10I1	2620	**2620**	2609	2611.7	1	**2620**	2609	2613.7	2	**2620**	2609	2613.7	2
10I2	**2642**	2629	2606	2615.3	0	2630	2610	2620.8	0	2630	2610	2620.8	0
10I3	2620	**2620**	2600	2609	1	**2620**	2600	2614.5	5	**2620**	2600	2614.5	5
10I4	**2621**	2620	2590	2606	0	2620	2600	2609.7	0	2620	2600	2609.7	0
10I5	2630	2620	2600	2612.3	0	2627	2610	2617.6	0	2627	2610	2617.6	0
avg	2401.6	2399.4	2367.5	2385.1	3.2	2401.3	2378.4	2392.9	5.0	2402.2	2378.7	2393.3	5.1

to 500 seconds. And for only 2 instances, we needed to reach 1000 seconds as time limit in order to find a best value. This shows that our PTS algorithm gives performs very well within a reasonable amount of time.

In the last row of Table 2, we report the average over all the groups of the tested instances. According to this row, we note that our PTS algorithm has the best total average value. This shows that in average our method performs better than all the previous ones developed in the literature.

5. Concluding remarks

In this paper, we presented a Probabilistic Tabu Search (PTS) heuristic with multiple neighborhood structures for The Disjunctively Constrained Knapsack Problem (DCKP). The DCKP is is a combination of two well-known combinatorial optimization problems, namely the maximum independent set and the standard Knapsack Problem, *i.e.*, an extension of the knapsack problem in which a conflict graph describing incompatibilities between items is given. Our PTS algorithm starts with an initial solution constructed by a new greedy heuristic. The proposed greedy heuristic selects items in non-increasing order of their efficiency taking in account the profit, weight, degree of the items and also the density of the conflict graph. The PTS algorithm uses multiple neighborhood structures based on drop and add moves which are explored cyclically. Four candidate lists are proposed for a given neighborhood structure where two are used to reinforce the aggressive aspect of TS (*i.e.* intensification phase) while the other incorporates randomness to diversifies the search (diversification phase). At each iteration a candidate list is selected with some probabilities. The proposed algorithm is evaluated on a total of 50 benchmark instances from the literature up to 1000 items. Computational results disclose that the proposed tabu search method outperforms a state-of-the-art approach. In particular, our approach is able to reach 46 best known solutions and discover 8 new best known solutions out of 50 benchmark instances.

Acknowledgements. The authors would like to thank Oualid Guemri for substantial discussion on the problem and Mhand Hifi for providing the benchmark of test instances. We are grateful for the comments of the referees which helped a lot to improve the exposition of the paper.

References

[1] H. Akeb, M. Hifi and M. Elhafedh O.Ah. Mounir, Local branching-based algorithms for the disjunctively constrained knapsack problem. *Comput. Industrial Eng.* **60** (2011) 811–820.

[2] Th. Alves de Queiroz and F.K. Miyazawa, Approaches for the 2d 0-1 knapsack problem with conflict graphs. In *Computing Conference (CLEI), 2013 XXXIX Latin American.* IEEE (2013) 1–8.

[3] A. Bettinelli, V. Cacchiani and E. Malaguti, Bounds and algorithms for the knapsack problem with conflict graph. Technical Report OR-14-16, DEIS–University of Bologna, Bologna, Italy (2014).

[4] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi and Ph. Michelon, A multi-level search strategy for the 0–1 multidimensional knapsack problem. *Discrete Appl. Math.* **158** (2010) 97–109.

[5] T.G. Crainic, M. Gendreau, P. Soriano and M. Toulouse, A tabu search procedure for multicommodity location/allocation with balancing requirements. *Ann. Oper. Res.* **41** (1993) 359–383.

[6] G.B. Dantzig, Discrete-variable extremum problems. *Oper. Res.* **5** (1957) 266–288.

[7] Th. Alves de Queiroz and F. Keidi Miyazawa, Problema da mochila 0-1 bidimensional com restriçoes de disjunçao (2012).

[8] L. Epstein and A. Levin, On bin packing with conflicts. *SIAM J. Optim.* **19** (2008) 1270–1298.

[9] L. Epstein, A. Levin and R. van Stee, Two-dimensional packing with conflicts. *Acta Inf.* **45** (2008) 155–175.

[10] A. Fréville, The multidimensional 0–1 knapsack problem: An overview. *Europ. J. Oper. Res.* **155** (2004) 1–21.

[11] A. Fréville and S. Hanafi, The multidimensional 0-1 knapsack problembounds and computational aspects. *Ann. Oper. Res.* **139** (2005) 195–227.

[12] M.R. Garey and D.S. Johnson, Computers and intractability: a guide to the theory of np-completeness. San Francisco, LA: Freeman (1979).

[13] M. Gendreau, G. Laporte and F. Semet, Heuristics and lower bounds for the bin packing problem with conflicts. *Comput. Oper. Res.* **31** (2004) 347–358.

[14] F. Glover, Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **13** (1986) 533–549.

[15] F. Glover, Tabu search-part i. *ORSA J. Comput.* **1** (1989) 190–206.

[16] F. Glover, Tabu thresholding: Improved search by nonmonotonic trajectories. *ORSA J. Comput.* **7** (1995) 426–442.

[17] F. Glover and M. Laguna, Tabu search. Springer (1997).

[18] F. Glover and A. Lokketangen, Probabilistic tabu search for zero-one mixed integer programming problems. University of Colorado, Boulder, USA (1994).

[19] S. Hanafi, On the convergence of tabu search. *J. Heuristics* **7** (2001) 47–58.

[20] S. Hanafi and A. Fréville, An efficient tabu search approach for the 0–1 multidimensional knapsack problem. *Europ. J. Oper. Res.* **106** (1998) 659–675.

[21] M. Hifi, An iterative rounding search-based algorithm for the disjunctively constrained knapsack problem. *Engrg. Optim.* **46** (2014) 1109–1122.

[22] M. Hifi and M. Michrafy, A reactive local search-based algorithm for the disjunctively constrained knapsack problem. *J. Oper. Res. Soc.* **57** (2006) 718–726.

[23] M. Hifi and M. Michrafy, Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem. *Comput. Oper. Res.* **34** (2007) 2657–2673.

[24] M. Hifi, S. Negre and M.Q.A. Mounir, Local branching-based algorithm for the disjunctively constrained knapsack problem. In *CIE 2009 – International Conference on Computers & Industrial Engineering.* IEEE (2009) 279–284.

[25] M. Hifi, S. Negre, T. Saadi, S. Saleh and L. Wu, A parallel large neighborhood search-based heuristic for the disjunctively constrained knapsack problem. In *IPDPSW 2014: International Parallel & Distributed Processing Symposium Workshops.* IEEE (2014) 1547–1551.

[26] M. Hifi and N. Otmani, A first level scatter search for disjunctively constrained knapsack problems. In *International Conference on Communications, Computing and Control Applications (CCCA).* IEEE (2011) 1–6.

[27] M. Hifi, S. Saleh, L. Wu and J. Chen, A hybrid guided neighborhood search for the disjunctively constrained knapsack problem. *Cogent Engrg.* **2** (2015) 1068969.

[28] K. Jansen, An approximation scheme for bin packing with conflicts. *J. Combin. Optim.* **3** (1999) 363–377.

[29] A. Khanafer, F. Clautiaux, S. Hanafi and E.-Gh. Talbi, The min-conflict packing problem. *Comput. Oper. Res.* **39** (2012) 2122–2132.

[30] A. Khanafer, F. Clautiaux and E.-Ghazali Talbi, New lower bounds for bin packing problems with conflicts. *Europ. J. Oper. Res.* **206** (2010) 281–288.

[31] M. Maiza and M.S. Radjef, Heuristics for solving the bin-packing problem with conflicts. *Appl. Math. Sci.* **5** (2011) 1739–1752.

[32] S. Martello and P. Toth, Knapsack problems: algorithms and computer implementations. John Wiley & Sons, Inc. (1990).

[33] U. Pferschy and J. Schauer, The knapsack problem with conflict graphs. *J. Graph Algorithms Appl.* **13** (2009) 233–249.

[34] D. Pisinger and M. Sigurd, Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS J. Comput.* **19** (2007) 36–51.

[35] R. Sadykov and F. Vanderbeck, Bin packing with conflicts: a generic branch-and-price algorithm. *INFORMS J. Comput.* **25** (2013) 244–255.

[36] A. Senisuka, B. You and T. Yamada, Reduction and exact algorithms for the disjunctively constrained knapsack problem. In: *International symposium on operations research and its applications; ISORA'05.* Lecture Notes in Operations Research (2005) 241–254.

[37] P. Soriano and M. Gendreau, Diversification strategies in tabu search algorithms for the maximum clique problem. *Ann. Oper. Res.* **63** (1996) 189–207.

[38] J. Xu, S.Y. Chiu and F. Glover, Probabilistic tabu search for telecommunications network design. *Comb. Optim. Theory Practice* **1** (1996) 69–94.

[39] T. Yamada, S. Kataoka and K. Watanabe, Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Inform. Process. Soc. Jpn. J.* **43** (2002).