Soft Comput (2018) 22:2025–2043
https://doi.org/10.1007/s00500-016-2465-7

METHODOLOGIES AND APPLICATION

Optimization algorithms for the disjunctively constrained knapsack problem

Mariem Ben Salem[1] · **Raouia Taktak**[2] · **A. Ridha Mahjoub**[3] · **Hanêne Ben-Abdallah**[4]

Published online: 23 December 2016

Abstract This paper deals with the Knapsack Problem with conflicts, also known as the Disjunctively Constrained Knapsack Problem. The conflicts are represented by a graph whose vertices are the items such that adjacent items cannot be packed in the knapsack simultaneously. We consider a classical formulation for the problem, study the polytope associated with this formulation and investigate the facial aspect of its basic constraints. We then present new families of valid inequalities and describe necessary and sufficient conditions for these inequalities to be facet defining. We also devise separation routines for these inequalities. Using these results, we develop a Branch-and-Cut algorithm for the problem. An extensive computational study is also presented.

Keywords Knapsack problem · Disjunctive constraints · Polytope · Facet · Separation · Branch-and-cut

Communicated by V. Loia.

✉ A. Ridha Mahjoub
mahjoub@lamsade.dauphine.fr

Mariem Ben Salem
bensalem.mariem@gmail.com

Raouia Taktak
raouia.taktak@gmail.com

Hanêne Ben-Abdallah
hbenabdallah@kau.edu.sa

[1] FSEGS/MIRACL, Université de Sfax, Sfax, Tunisia

[2] ISIMS/CRNS, Université de Sfax, Sfax, Tunisia

[3] Université Paris-Dauphine, PSL Research University, CNRS, LAMSADE, 75016 Paris, France

[4] King Abdulaziz University, Jeddah, Saudi Arabia

1 Introduction

In this paper, we consider a variant of the 0–1 Knapsack Problem, where some items are incompatible with others, and cannot then be packed simultaneously in the knapsack. We are given a knapsack of capacity c, a set $V = \{1, 2, \ldots, n\}$ of items, and a graph of conflicts between some items $G = (V, E)$. For each edge $(i, j) \in E$, items i and j are incompatible. With each item $i \in V$ is associated a nonnegative profit p_i and a nonnegative 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.

The DCKP is an NP-hard combinatorial optimization problem since it is a combination of two NP-hard combinatorial optimization problems. Indeed, when no conflicts between items exist (*i.e.*, $E = \emptyset$), the problem reduces to a 0–1 Knapsack Problem (0–1 KP) which is known to be NP-hard Martello and Toth (1990). When the knapsack constraint is omitted, the problem is the maximum independent set which is known to be NP-hard as well Garey and Johnson (1979). For some special classes of conflict graphs, the problem becomes easier. For example, in Pferschy and Schauer (2009), Pferschy and Schauer present pseudo-polynomial algorithms to solve the DCKP for two special classes of graphs, namely graphs of bounded tree-width and chordal graphs.

The DCKP is relatively a recent variant of the Knapsack Problem that has been firstly introduced by Yamada et al. (2002). Yamada et al. (2002) present a heuristic method as well as an implicit enumeration algorithm together with an interval reduction method in order to solve the DCKP to optimality.

Later, more focus has been given to the problem. Heuristic and exact resolution approaches have been proposed.

Senisuka et al. (2005) present an exact algorithm to solve the DCKP. The algorithm combines Lagrangian relaxation techniques with the pegging test for ordinary knapsacks which helps reducing significantly the problem size. The approach proves to be efficient in solving huge instances of the DCKP in a reasonable time of computation. In parallel, Hifi and Michrafy (2007) propose several versions of an exact algorithm for the DCKP, based on a reduction procedure combined with a Branch-and-Bound algorithm. A further exact approach has been proposed by Bettinelli et al. (2014) who present an efficient Branch-and-Bound algorithm using a combination of several upper bounding procedures as well as a variety of branching strategies.

Along with the exact methods and due to the difficulty of the problem, many heuristic methods have been proposed in the literature. These have, in general, provided good results in a reasonable amount of time. Different meta-heuristics have been, in this context, used to solve efficiently the DCKP. These include local branching algorithms Hifi et al. (2009); Akeb et al. (2011), a version of Scatter Search Hifi and Otmani (2011), a parallel large neighborhood search-based heuristic Hifi et al. (2014), and recently a guided neighborhood search Hifi et al. (2015).

In some cases, the DCKP appears as a subproblem of a more general one. Sadykov and Vanderbeck (2013) 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 propose a dynamic programming algorithm for the DCKP when the conflict graph is an interval graph. They develop a Depth-First-Search Branch-and-Bound approach when the conflict graph has no special structure.

To the best of our knowledge, our work constitutes the first contribution considering the polyhedral aspect of the DCKP. The only works presenting close polyhedral studies are those dealing with the 0–1 knapsack polyhedron (see Balas 1975; Balas and Zemel 1978; Weismantel 1997), and polyhedra related to some variants of the Knapsack Problem (see Atamtürk and Narayanan 2009; Boyd 1993; Farias Jr and Nemhauser 2003; Hanafi and Glover 2007; Zeng and Richard 2011), along with the ones dealing with the independent set polyhedron Nemhauser and Trotter (1974).

This paper is organized as follows. In the next section, we state the classical Integer Linear Programming (ILP) formulation used in the literature to model the DCKP and we define the associated polytope and study the facial aspect of the basic constraints. In Sect. 3, we describe some valid inequalities and give necessary conditions and sufficient conditions for these inequalities to be facet defining. In Sect. 4, we devise separation algorithms for these inequalities and describe our Branch-and-Cut algorithm. In Sect. 5, we present an extensive computational study. Some concluding remarks and indications for future work are given in Sect. 6.

2 Formulation and polyhedral analysis

A natural and compact Integer Linear Programming formulation for the DCKP makes use of a set of binary variables x_i associated with items $i \in V$, taking value 1 if item i is packed in the knapsack, and 0 otherwise. The DCKP is equivalent to the following program:

$$\max \sum_{i \in V} p_i x_i \tag{1}$$

$$\sum_{i \in V} w_i x_i \leq c \tag{2}$$

$$x_i + x_j \leq 1 \quad \text{for all } e = (i, j) \in E \tag{3}$$

$$0 \leq x_i \leq 1 \quad \text{for all } i \in V, \tag{4}$$

$$x_i \in \{0, 1\} \quad \text{for all } i \in V, \tag{5}$$

where (1) denotes the objective function, (2) represents the knapsack capacity constraint inequality, (3) are the disjunctive constraint inequalities, and (4) and (5) are the trivial and integrality constraints, respectively.

In the sequel, and w.l.o.g., we will assume that the set of edges E is composed of two subsets E_d and E_c, *i.e.*, $E = E_d \cup E_c$, where E_d is the set of edges representing conflicts due to the disjunction constraints (3), and E_c is the set of edges representing conflicts due to the capacity constraint (2). In other words, for all $e = (i, j) \in E_c$, $w_i + w_j > c$, that is i and j form a *cover*.
We will also assume that $w_i \leq c$ for all $i \in V$, that is any single item induces a solution for the problem.

Denote by DCKP(G) the polytope associated with the DCKP, that is the convex hull of the incidence vectors of all its solutions, *i.e.*,

$$\text{DCKP}(G) = \text{conv}\Big\{x \in \{0, 1\}^n \ : \ x \text{ satisfies } (2)-(5)\Big\}.$$

A solution $S \subseteq V$ of DCKP will be represented by the set of items retained to be packed in the knapsack and that are not in conflict. That is to say, the incidence vector of S, x^S is such that $x_i^S = 1$ if $i \in S$ and $x_i^S = 0$ otherwise, satisfies the DCKP constraints.

We first state the following preliminary result.

Theorem 1 *DCKP(G) is full dimensional.*

Proof Consider the solutions of DCKP $S_0, S_1, \ldots, S_n$ defined as follows, $S_0 = \emptyset$, $S_i = \{i\}$, for all $i \in V$. Clearly $x^{S_0}, x^{S_1}, \ldots, x^{S_n}$ are $n + 1$ affinely independent points of DCKP(G). Consequently, $\dim(\text{DCKP}(G)) = n$. □

Having established the dimension of DCKP(G), next, we will characterize when the trivial and disjunctive inequalities define facets.

Theorem 2 $x_k \geq 0$ *defines a facet of DCKP(G).*

Proof Let $S_0 = \emptyset$, and for all $i \in V \setminus \{k\}$, $S_i = \{i\}$. These constitute $n = |V|$ solutions of DCKP whose incidence vectors satisfy $x_k = 0$ and are affinely independent. □

Theorem 3 *$x_k \leq 1$ defines a facet of DCKP(G) if and only if for all $i \in V \setminus \{k\}$, $(i,k) \notin E$.*

Proof Assume that there is an item $i_0 \in V$ such that $(i_0, k) \in E$, *i.e.*, $x_k + x_{i_0} \leq 1$. Since this inequality dominates $x_k \leq 1$, the latter cannot be facet defining.

Now, suppose that $(i,k) \notin E$ for all $i \in V \setminus \{k\}$. Consider the solutions $S_1, \ldots, S_k, \ldots, S_n$ of DCKP defined by $S_k = \{k\}$, and for all $i \in V \setminus \{k\}$, $S_i = \{i,k\}$. These form n solutions of DCKP whose incidence vectors satisfy $x_k = 1$ and are affinely independent. □

Theorem 4 *For $(k,m) \in E$, $x_k + x_m \leq 1$ defines a facet of DCKP(G) if and only if for all $i \in V \setminus \{k,m\}$, i is not in conflict with at least one of the two items k and m.*

Proof Assume that there exists an item $i^* \in V$ in conflict with both k and m. In this case, we can pack in the knapsack at most one item among k, m and i^*. This implies that the inequality $x_k + x_m + x_{i^*} \leq 1$ is satisfied by every solution of the problem. Since this inequality dominates $x_k + x_m \leq 1$, the latter cannot define a facet.

Assume now that for all $i \in V \setminus \{k,m\}$, i is not in conflict with at least one of the items k and m. Let $U_{k,m} = \{i \in V \setminus \{k,m\} : (i,k) \notin E$, and, $(i,m) \notin E\}$ be the set of items that are not in conflict neither with k nor with m. Let $U_k = \{i \in V \setminus \{k,m\} : (i,k) \notin E$, and, $(i,m) \in E\}$ be the set of items that are not in conflict with k but in conflict with m, and $U_m = \{i \in V \setminus \{k,m\} : (i,m) \notin E$, and, $(i,k) \in E\}$ the set of items that are not in conflict with m but in conflict with k.

Consider the solutions $S_k = \{k\}$, $S_m = \{m\}$, $S_i = \{k,i\}$ for all $i \in U_k \cup U_{k,m}$, $S_j = \{m,j\}$ for all $j \in U_m$. Clearly, these sets constitute a family of n solutions of the problem, whose incidence vectors satisfy the equation $x_k + x_m = 1$. Moreover, as illustrated in Fig. 1, these vectors are affinely independent. □

Along with the basic constraints of the formulation, we have identified new families of valid inequalities for DCKP(G). These will be presented in the next section.

3 Valid inequalities

As mentioned previously, the DCKP can be seen as a combination of two classical problems, namely the knapsack and the independent set problems. Therefore, valid inequalities for these problems have helped us to determine families of valid inequalities for the DCKP.

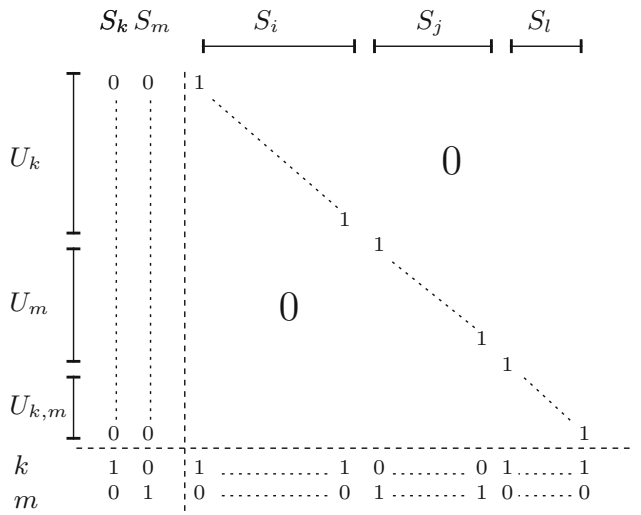

Fig. 1 Solutions incidence matrix

3.1 Clique inequalities

Given a graph $G = (V, E)$, a *clique* of G is a subset of vertices $K \subset V$ such that every two distinct vertices are adjacent.
A clique is said to be *maximal* if it is not strictly contained in another clique.

If K is a clique of G, then all the vertices of K are pairwise in conflict. This implies that one can pack at most one element from K in the knapsack; therefore, the following inequality is valid for DCKP(G).

$$\sum_{i \in K} x_i \leq 1. \tag{6}$$

Theorem 5 *Inequality (6) defines a facet of DCKP(G) if and only if K is maximal.*

Proof If K is not maximal, then there is an element $j \in V \setminus K$ such that $K' = K \cup \{j\}$ is a clique. Therefore, $\sum_{i \in K'} x_i = \sum_{i \in K} x_i + x_j \leq 1$ is valid for DCKP(G). Since this inequality dominates (6), the latter cannot define a facet.

Now suppose K is maximal. Then for every item $j \in V \setminus K$, there is an item $j' \in K$ such that $(j, j') \notin E$. Consider the sets $S_i = \{i\}$ for all $i \in K$, and $S_j = \{j, j'\}$ for all $j \in V \setminus K$, where $j' \in K$ is an item of K not adjacent to j. It is clear that these sets are solutions of the DCKP. Moreover, their incidence vectors satisfy (6) with equality and are affinely independent. □

3.2 Cover inequalities

A *cover* for the DCKP is a set C of items such that $\sum_{i \in C} w_i > c$. Clearly, a cover cannot be packed in the knapsack. Hence,

the following inequalities are valid for the DCKP(G)

$$\sum_{i \in C} x_i \leq |C| - 1 \quad \text{for all } C \in \mathcal{C} \tag{7}$$

where $\mathcal{C}$ is the set of the covers of the DCKP instance. Inequalities (7) will be called *cover inequalities*.
A cover C is called *minimal* if

$$\sum_{i \in C \setminus \{j\}} w_i \leq c \quad \text{for all } j \in C.$$

The following theorem characterizes the covers that may induce facets for DCKP(G). For this, note that if a cover is not minimal, then it contains a proper subset which is a cover. Moreover, this set may consist of two elements i, j such that $(i, j) \in E$, that is to say i and j are either in conflict or form a cover.

Theorem 6 *A cover inequality (7), induced by a cover C, defines a facet for DCKP(G) if and only if*

(1) C is minimal,
(2) for all $j \in V \setminus C$, $w_j < w_{j^*} = max\{w_j : j \in C\}$,
(3) every item $j \in V \setminus C$ is not in conflict with more than one item of C, and if j is in conflict with item j' of C, then $(C \setminus \{j'\}) \cup \{j\}$ is not a cover.

Proof Necessity

(1) If C is not minimal, then a proper subset $\widetilde{C}$ of C is a cover. Hence,

$$\sum_{i \in \widetilde{C}} x_i \leq |\widetilde{C}| - 1 \tag{8}$$

is valid for DCKP(G). By summing inequality (8) together with $x_i \leq 1$ for all $i \in C \setminus \widetilde{C}$, we obtain (7). Since $C \setminus \widetilde{C} \neq \emptyset$, inequality (7) is a linear combination of valid inequalities and thus it cannot define a facet.

(2) If there exists an item $j \in V \setminus C$ such that $w_j \geq w_{j^*}$, then the inequality

$$\sum_{i \in C} x_i + x_j \leq |C| - 1 \tag{9}$$

is also valid for DCKP(G). In fact, item j cannot be packed with $|C| - 1$ items of C. However, inequality (7) is redundant with respect to (9) and $x_j \geq 0$. It cannot, therefore, define a facet.

(3) If an item j of $V \setminus C$ is in conflict with two items i_1, i_2 of C, then any solution of DCKP containing j cannot contain more than $|C| - 2$ elements from C. Hence, its incidence vector cannot satisfy inequality (7) with equality. This implies that inequality (7) is equivalent to $x_j \geq 0$. Since (7) is not a positive multiple of $x_j \geq 0$, it cannot be facet defining. Also, if an item j of $V \setminus C$ is in conflict with an item j' and $(C \setminus \{j'\}) \cup \{j\}$ is a cover, then it follows, along the same way, that inequality (7) is equivalent to $x_j \geq 0$ and cannot hence define a facet.

Sufficiency

Suppose that Conditions (1) $-$ $-$(3) are all satisfied. Let $ax \leq \alpha$ denote inequality (7), and let us suppose there is a facet defining inequality $bx \leq \beta$ of DCKP(G) such that $\{x \in \text{DCKP}(G) \mid ax = \alpha\} \subseteq \{x \in \text{DCKP}(G) \mid bx = \beta\}$. We will show that $b = \rho a$.

By (1), it follows that any set $Q \subset C$ such that $|Q| = |C| - 1$ is a solution of DCKP. Let i, $j \in C$, and consider the solutions

$$S_1 = C \setminus \{i\}, \; S_2 = C \setminus \{j\}.$$

As $ax^{S_1} = ax^{S_2} = \alpha$, we have that $bx^{S_1} = bx^{S_2}$. This yields $b_i = b_j$. As i and j are arbitrary in C, it follows that all the b_i's are the same in C, and thus

$$b_i = \rho \text{ for all } i \in C \quad \text{for some } \rho \in \mathbb{R}. \tag{10}$$

Now let $j \in V \setminus C$. By (2) we have that $w_j < w_{j^*}$, and by 3) j is in conflict with at most one element of C. Suppose, for instance, that j is in conflict with an element, say j', of C. Consider the set $S = (C \setminus \{j'\}) \cup \{j\}$. By 3), S is a solution of DCKP. Moreover, we have that $ax^S = \alpha$, and hence $bx^S = \beta$. As $S \setminus \{j\} = C \setminus \{j'\}$ is also a solution of the problem and $ax^{S \setminus \{j\}} = \alpha$, we have $bx^{S \setminus \{j\}} = \beta$. But this implies that $b_j = bx^S - bx^{S \setminus \{j\}} = 0$. If j is not in conflict with any item of C, as by (2), $w_j < w_{j^*}$, $(C \setminus \{j^*\}) \cup \{j\}$ is a solution of DCKP. And, similarly, it follows that $b_j = 0$. Thus, we obtain that $b_j = 0$ for all $j \in V \setminus C$. This together with (10) yields $b = \rho a$, and the proof is complete. □

Balas (1975), Hammer et al. (1975) and Wolsey (1975) observed that when C is *minimal* the cover inequalities (7) are the strongest. Balas (1975) proposes a way to strengthen the cover inequalities. Let $w^* = \max_{j \in C} w_j$, and consider the *extension* $E(C) = C \cup \{j \in V \setminus C : w_j \geq w^*\}$. Then, a valid inequality for the DCKP(G) called the *Extended Cover Inequality* (ECI) is given by

$$\sum_{j \in E(C)} x_j \leq |C| - 1. \tag{11}$$

Balas (1975) and Wolsey (1975) also showed that, given a minimal cover C, there exists at least one facet defining

Lifted Cover Inequality having the following form:

$$\sum_{j\in C} x_j + \sum_{j\in V\setminus C} \alpha_j x_j \leq |C| - 1, \tag{12}$$

where $\alpha_j \geq 0$ for all $j \in V \setminus C$.

Inequalities (12) can be obtained by a sequential lifting from inequalities (11). This means that the lifting coefficients α_j, $j \in V\setminus C$, are computed one by one in a given order. Suppose that $V\setminus C = \{j_1, \ldots, j_t\}$, and that $\alpha_{j_1}, \ldots, \alpha_{j_{t-1}}$ are computed, that is to say the inequality $\sum_{j\in C} x_j + \sum_{i=1}^{t-1} \alpha_{j_i} x_{j_i} \leq |C| - 1$ is valid for KP. In order to determine coefficient α_{j_t}, one can compute $\alpha_0 = \max\{\sum_{j\in C} x_j + \sum_{i=1}^{t-1} \alpha_{j_i} x_{j_i} : x \text{ solution of KP}, x_{j_t} = 1\}$, and set $\alpha_{j_t} = |C| - 1 - \alpha_0$. The coefficients α_{j_i}, $i = 1, \ldots, t$ depend on the order in which they are computed.

As it appears, the computation of each coefficient α_j requires the resolution of a KP. Zemel (1989) showed that given a fixed cover C and a fixed sequence of lifting, the lifting coefficients can be computed in $O(n|C|)$.

Gu et al. (1998) referred to inequalities (12) as *Simple Lifted Cover Inequality*. This has been later generalized by Van Roy and Wolsey (1987) who derived the *General Lifted Cover Inequality* of the form

$$\sum_{j\in C\setminus D} x_j + \sum_{j\in V\setminus C} \alpha_j x_j + \sum_{j\in D} \beta_j x_j \leq |C \setminus D| + \sum_{j\in D} \beta_j - 1, \tag{13}$$

where C is a cover, $D \subset C$, $\alpha_j \geq 0$ for all $j \in V\setminus C$, and $\beta_j \geq 0$ for all $j \in D$.

As for inequalities (12), inequalities (13) can be obtained from (11) by a sequential lifting. For more details on lifting techniques in combinatorial optimization, one can refer to Wolsey and Nemhauser (1999).

We will follow Gu et al. (1998) in referring to the computation of the lifting coefficients α and β as *up-lifting* and *down-lifting*, respectively.

3.3 Odd-cycle and hypercycle inequalities

A *cycle* in a graph is a sequence $v_1, e_1, v_2, \ldots, v_k, e_k, v_1$ of nodes and edges such that $e_i = (v_i, v_{i+1})$, $i = 1, \ldots, k-1$ and $e_k = (v_k, v_1)$. We will also denote a cycle C by its sequence of nodes and write $C = (v_1, \ldots, v_k)$. A cycle of k nodes is said of *length* k. A cycle is said to be *even* (*odd*) if its length is even (odd). A *chord* of a cycle is an edge joining two non-consecutive nodes of the cycle. A cycle is called *simple* if its nodes $v_1, \ldots, v_k$ are all different.

Odd cycles induce valid inequalities for the independent set problem and hence for the DCKP(G). Consider a cycle C of G where nodes are $1, \ldots, k$ with k odd. Then the inequalities

$$x_i + x_{i+1} \leq 1, \quad \text{for all} \quad i = 1, \ldots, k,$$

where the indices are taken modulo k, are valid for DCKP(G). By summing these inequalities, dividing by 2 and rounding down the right-hand side, we obtain the inequality

$$\sum_{i\in C} x_i \leq \frac{k-1}{2}, \tag{14}$$

which is valid for DCKP(G). Inequalities of type (14) are called *Odd-Cycle Inequalities*. Inequalities (14) may define facets for DCKP(G). A necessary condition for inequality (14) to be facet defining is that the cycle C does not contain a chord. If C contains a chord (i_0, j_0), $i_0 < j_0$, then one of the cycles $C_1 = (1, \ldots i_0, j_0, \ldots, k)$ and $C_2 = (i_0, \ldots, j_0)$, say C_1, is odd. It is not hard to see that (14) can be obtained as a linear combination of the odd-cycle inequality induced by C_1 and the disjunctive inequalities $x_{i_0+l} + x_{i_0+l+1} \leq 1$, $l = 1, \ldots, j_0 - 2$. Inequalities (14) can be strengthened by the so-called lifted odd-cycle inequalities to be facet defining for the independent set polytope Nemhauser and Trotter (1974) and the DCKP(G) as well.

In what follows, we are going to introduce a more general class of valid inequalities for the DCKP(G). For this, let us first give an example.

Consider the DCKP given by the system

$$(P_1) \begin{cases} 5x_1 + 4x_2 + 3x_3 + 5x_4 + 2x_5 \leq 11, \\ x_1 + x_4 \leq 1, \\ x_4 + x_5 \leq 1. \end{cases}$$

Observe that $C_1 = \{1, 2, 3\}$ and $C_2 = \{2, 3, 4\}$ are covers. Thus, the following inequalities are valid for DCKP(G),

$$x_1 + x_2 + x_3 \leq 2,$$
$$x_2 + x_3 + x_4 \leq 2.$$

By summing these inequalities together with $x_1 + x_4 \leq 1$, we obtain the inequality $2(x_1 + x_2 + x_3 + x_4) \leq 5$. By dividing by 2 and rounding down the right-hand side, we obtain that

$$x_1 + x_2 + x_3 + x_4 \leq 2 \tag{15}$$

is valid for DCKP(G). Moreover, (15) defines a facet. In fact, it is easy to see that the sets $S_1 = \{1, 2\}$, $S_2 = \{1, 3\}$, $S_3 = \{2, 3\}$, $S_4 = \{3, 4\}$, $S_5 = \{3, 4, 5\}$ are solutions of the problem. Moreover, their incidence vectors satisfy (15) with equality and are affinely independent.

In what follows, we are going to show this as a special case of a more general class of facets. This will be presented within the framework of hypergraphs. Given a finite

set $V = \{v_1, \ldots, v_n\}$, a *hypergraph* H on V is a family $\mathcal{E} = \{E_1, E_2, \ldots, E_m\}$ of subsets of V such that

$$E_i \neq \emptyset \quad \text{for } i = 1, \ldots, m,$$
$$\bigcup_{i=1}^{m} E_i = V.$$

The elements $v_1, \ldots, v_n$ of V are called *nodes*, and the sets $E_1, \ldots, E_m$ are called *hyperedges*. A hypergraph $H = (V, \mathcal{E})$ is said to be *simple* if no hyperedge is strictly contained in another hyperedge, that is to say $E_i \not\subset E_j$ for all $E_i, E_j \in \mathcal{E}$. A graph without loops is a hypergraph where each hyperedge consists of exactly two nodes. Given a hypergraph $H = (V, \mathcal{E})$, a *hypercycle* is a sequence $(v_1, E_1, v_2, \ldots, v_k, E_k, v_1)$ with $\{v_i, v_{i+1}\} \subseteq E_i$, for $i = 1, \ldots, k$, where the indices are modulo k. Note that the E_i's may not be all different. Also note that the E_i's may contain nodes different from $v_1, \ldots, v_k$. A cycle in a graph corresponds to the case where $|E_i| = 2$ for $i = 1, \ldots, k$. Now consider the DCKP along with the corresponding conflict graph $G = (V, E)$, and let us associate with it the hypergraph $H = (V, \mathcal{E})$ where $\mathcal{E}$ is the set of minimal covers of the problem together with the sets $\{i, j\}$ such that $(i, j) \in E$, *i.e.*, i and j are in conflict. Also as a cover, which contains two items in conflict may be considered as a non-minimal cover, we will suppose w.l.o.g., that no hyperedge strictly contains an edge of E. In other words, the hyperedges will correspond to the covers C such that $C\backslash\{i\}$ is a solution of DCKP(G) for all $i \in C$. The hypergraph $H = (V, \mathcal{E})$, associated with problem (P_1) above, is depicted in Fig. 2. Here $V = \{1, 2, 3, 4, 5\}$ and $\mathcal{E}$ contains the hyperedges $E_1 = \{1, 2, 3\}$, $E_2 = \{2, 3, 4\}$, $E_3 = \{1, 4\}$, and $E_4 = \{4, 5\}$.

Now consider a hypercycle of $H = (V, \mathcal{E})$ whose hyperedges are $E_1, \ldots, E_k$. Let $W = \bigcup_{i=1}^{k} E_i$. Let $W' \subseteq W$ be a subset of W such that every node of W' appears in exactly q hyperedges among $E_1, \ldots, E_k$. For every $j \in W\backslash W'$ let ρ_j be the number of hyperedges, among $E_1, \ldots, E_k$, to which j belongs.

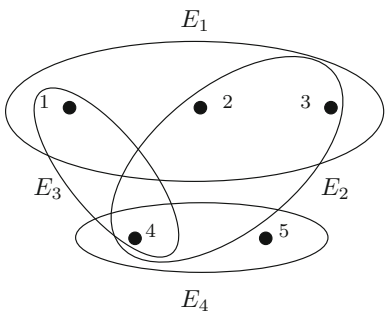

Fig. 2 The hypergraph associated with (P_1)

Suppose that $\rho_j < q$ for all $j \in W\backslash W'$ and $\sum_{i=1}^{k}(|E_i| - 1) + \sum_{j \in W\backslash W'}(q - \rho_j)$ is not a multiple of q. Now consider the following valid cover inequalities

$$\sum_{j \in E_i} x_j \leq |E_i| - 1 \quad \text{for all } i = 1, \ldots, k,$$
$$(q - \rho_j)x_j \leq q - \rho_j \quad \text{for all } j \in W\backslash W'.$$

By summing these inequalities, we obtain the inequality

$$q\left(\sum_{i \in W} x_i\right) \leq \sum_{i=1}^{k}(|E_i| - 1) + \sum_{j \in W\backslash W'}(q - \rho_j).$$

As the right-hand side of the above inequality is not a multiple of q, by dividing by q and rounding down the right-hand side of the resulting inequality, we obtain the following valid inequality

$$\sum_{i \in W} x_i \leq \left\lfloor \frac{\sum_{i=1}^{k} |E_i| + \sum_{j \in W\backslash W'}(q - \rho_j) - k}{q} \right\rfloor. \tag{16}$$

Since each node of W' is in q E_i's and each node j of $W\backslash W'$ is in ρ_j E_i's, we have that $\sum_{i=1}^{k} |E_i| + \sum_{j \in W\backslash W'}(q - \rho_j) = q|W|$.
Hence, inequality (16) can be written as

$$\sum_{i \in W} x_i \leq |W| - \left\lceil \frac{k}{q} \right\rceil. \tag{17}$$

Inequalities of type (17) will be called *Hypercycle Inequalities*.

Remark 1 Any solution T of DCKP whose incidence vector satisfies (17) with equality is such that $|W\backslash T| = \left\lceil \frac{k}{q} \right\rceil$ and $W\backslash T$ covers $\mathcal{E} = \{E_1, \ldots, E_k\}$. Otherwise, one of the covers would be included in $W \cap T$, which is not possible.

In order to illustrate the hypercycle inequalities, consider the problem (P_1) given above. Consider the hypercycle $(1, E_1, 3, E_2, 4, E_3, 1)$ whose hyperedges are $E_1 = \{1, 2, 3\}$, $E_2 = \{2, 3, 4\}$, $E_3 = \{1, 4\}$. Observe that every node i of $E_1 \cup E_2 \cup E_3$ belongs to exactly two sets among E_1, E_2, E_3. So here $W = \{1, 2, 3, 4\}$, $k = 3$, $q = 2$ and $W' = W$. The corresponding hypercycle inequality (16) is nothing but inequality (15).
Now suppose that in (P_1), items 2 and 4 are also in conflict. Thus, $E_2 = \{2, 3, 4\}$ is no more hyperedge of the associated

hypergraph and must then be replaced by $E_2' = \{2, 4\}$. Consider the hypercycle whose hyperedges are E_1, E_2', E_3. Here we have $W = \{1, 2, 3, 4\}$, $k = 3$, $q = 2$, $W' = \{1, 2, 4\}$, $\rho = 1$. We still obtain inequality (15).

Observe that inequality (15) is an extended cover inequality obtained from cover $\{1, 2, 3\}$. However, the hypercycle inequalities may be different from the extended and lifted cover inequalities as shown in the following example. Consider the problem

$$(P_2) \begin{cases} x_1 + 4x_2 + 3x_3 + x_4 + x_5 + x_6 + x_7 \leq 7, \\ x_4 + x_5 \leq 1, \\ x_4 + x_6 \leq 1, \\ x_1 + x_5 \leq 1, \\ x_1 + x_6 \leq 1. \end{cases}$$

Clearly, the sets $E_1 = \{1, 2, 3\}$, $E_2 = \{2, 3, 4\}$, $E_3 = \{2, 3, 5\}$ are minimal covers for (P_2). Consider the hypercycle in the related hypergraph, induced by the hyperedges E_1, E_2, E_3 together with $E_4 = \{1, 6\}$, $E_5 = \{1, 5\}$, $E_6 = \{4, 5\}$, $E_7 = \{4, 6\}$. Here we have $W = \{1, 2, 3, 4, 5, 6\}$, $k = 7$, $q = 3$, $W' = \{1, 2, 3, 6\}$, $\rho_4 = 1$, $\rho_6 = 2$. The corresponding hypercycle inequality is given by

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 3.$$

This is different from an extended and a lifted cover inequality. Moreover, this inequality is facet defining for the associated DCKP(G).

As a further example, consider the DCKP whose constraints are

$$\begin{aligned} x_1 + 2x_2 + 3x_3 + x_4 &\leq 5, \\ x_1 + x_4 &\leq 1. \end{aligned}$$

By considering the hypercycle induced by the hyperedges $\{1, 2, 3\}$, $\{2, 3, 4\}$ and $\{1, 4\}$ of the related hypergraph, we obtain the inequality

$$x_1 + x_2 + x_3 + x_4 \leq 2.$$

which is valid and facet defining for the associated polytope. Let us remark that odd-cycle inequalities (14), obtained from the conflict graph G, are nothing but the hypercycle inequalities when the hyperedges are all edges of G.

We have the following result which gives necessary and sufficient conditions for a hypercycle inequality to be facet defining.

Theorem 7 *A hypercycle inequality (17) defines a facet of DCKP(G) if and only if the following hold.*

(1) For every node $j \in V \backslash W$, there is a set $S \subset W$ of $\left\lceil \frac{k}{q} \right\rceil$ items which covers the hyperedges E_1, ..., E_k and such that $(W \backslash S) \cup \{j\}$ is not a cover.

(2) There are $|W|$ sets S_1, ..., $S_{|W|} \subset W$ which cover E_1, ..., E_k such that $|S_i| = \left\lceil \frac{k}{q} \right\rceil$, and $T_i = W \backslash S_i$ is not a cover for $i = 1, \ldots, |W|$ and x^{S_1}, ..., $x^{S_{|W|}}$ are affinely independent. (Note that two items i, j such that $(i, j) \in E$ are considered as a cover and any set containing a cover is a cover).

(3) k is not a multiple of q.

Proof Necessity

(1) Suppose that for some $j \in V \backslash W$, any set $S \subset W$ of $\left\lceil \frac{k}{q} \right\rceil$ items that covers $E_1, \ldots, E_k$ is such that $(W \backslash S) \cup \{j\}$ is a cover. By Remark 1, it then follows that j cannot belong to any solution of the problem whose incidence vector satisfies (17) with equality. This implies that (17) is equivalent to the inequality $x_j \geq 0$. Since (17) is not a positive multiple of $x_j \geq 0$ and DCKP(G) is full dimensional, this implies that (17) does not define a facet.

(2) First of all, observe that the incidence vectors of subsets S_1, ..., S_l of W with $|S_i| = \left\lceil \frac{k}{q} \right\rceil$ for $i = 1, \ldots, l$ are affinely independent if and only if the incidence vectors of the sets $T_1, \ldots, T_l$ where $T_i = W \backslash S_i$, for $i = 1, \ldots, l$, are affinely independent. Suppose that the statement does not hold. By Remark 1 and the observation above, it follows that there do not exist sufficiently many ($|V|$) solutions of the problem whose incidence vectors satisfy (17) with equality and are affinely independent. Hence, (17) cannot define a facet.

(3) If k is a multiple of q, then clearly, (17) can be obtained as a linear combination of valid inequalities of DCKP(G) and cannot, thus, be facet defining.

Sufficiency:

Suppose that 1)-3) hold. Let us denote (17) by $ax \leq \alpha$ and let $bx \leq \beta$ be a facet defining inequality of DCKP(G) such that $\{x \in \text{DCKP}(G) \mid ax = \alpha\} \subseteq \{x \in \text{DCKP}(G) \mid bx = \beta\}$. We will show that $b = \rho a$ for some $\rho \in \mathbb{R}$.

By (2), it follows that the sets $T_i = W \backslash S_i, i = 1, \ldots, |W|$, are solutions of the DCKP. Moreover, as x^{S_1}, ..., $x^{S_{|W|}}$ are affinely independent, it follows that x^{T_1}, ..., $x^{T_{|W|}}$ so are. Let A be the square matrix whose columns are x^{T_1}, ..., $x^{T_{|W|}}$. As these vectors are affinely independent and are so linearly independent since they do not contain the zero vector, we have that A is non-singular. Moreover, as $|T_i| = |W| - \lceil \frac{K}{q} \rceil$

for all $i = 1, \ldots, |W|$ it follows that system

$$\lambda A = (\beta, \ldots, \beta)$$

has a unique solution given by $\lambda_i = \frac{\beta}{|W| - \left\lceil \frac{k}{q} \right\rceil}$, for all $i = 1, \ldots, |W|$.

As $T_1, \ldots, T_{|W|}$ are such that $ax^{T_i} = \alpha$ for $i = 1, \ldots, |W|$, and thus $bx^{T_i} = \beta$ for $i = 1, \ldots, |W|$, it follows that $b_i = \frac{\beta}{|W| - \left\lceil \frac{k}{q} \right\rceil} = \rho$ for $i = 1, \ldots, |W|$. Now let j be an element of $V \backslash W$. By 1) there is $S \subset W$ with $S = \left\lceil \frac{k}{q} \right\rceil$ which covers $E_1, \ldots, E_k$ and such that $T = (W \backslash S) \cup \{j\}$ is not a cover. Thus, $T \backslash \{j\}$ and T are both solutions of DCKP. As $ax^{T \backslash \{j\}} = ax^T = \alpha$, and hence $bx^{T \backslash \{j\}} = bx^T = \beta$, it follows that $b_j = 0$. Altogether we have that

$$b_i = \rho \quad \text{for all } i \in W,$$
$$b_i = 0 \quad \text{for all } i \in V \backslash W.$$

Thus, $b = \rho a$, and the proof is complete. □

In what follows, we characterize a special case in which inequality (17) may define a facet.

Theorem 8 *Consider a hypercycle $(v_1, E_1, v_2, \ldots, v_k, E_k, v_1)$ in H and suppose that $E_1, \ldots, E_k$ are distinct sets, and $v_1, \ldots, v_k$ are distinct nodes. Then inequality (17) defines a facet of DCKP(G) if the following hold.*

(1) Each node belongs to at most two sets among $E_1, \ldots, E_k$ (that is to say $q = 2$).
(2) k is odd.
(3) For any set $S = \{v, v_{i+2}, \ldots, v_{i+2l}\}$ where $v \in E_i$ and l is such that $k = 2l + 1$, the set $W \backslash S$ does not contain a cover. Here the indices are taken modulo k.
(4) For every $j \in V \backslash W$, there is a set S among those introduced in (3) such that $(W \backslash S) \cup \{j\}$ is not a cover.

Proof First remark that from 1), it follows that any set $S \subset W$ of the form $\{v, v_{i+2}, \ldots, v_{i+2l}\}$, where $v \in E_i$, covers all the hyperedges $E_1, \ldots, E_k$. Moreover by 3), the complementary set $W \backslash S$ in W induces a solution of DCKP.

Now as before, let us denote (17) by $ax \leq \alpha$ and let $bx \leq \beta$ be facet defining inequality such that $\{x \in \text{DCKP}(G) \mid ax = \alpha\} \subseteq \{x \in \text{DCKP}(G) \mid bx = \beta\}$. We will show that $b = \rho a$ for some $\rho \in \mathbb{R}$.

Consider the sets

$$W_1 = W \backslash S_1 \text{ with } S_1 = \{v_1, v_3, \ldots, v_k\},$$
$$W_2 = W_1 \backslash S_2 \text{ with } S_2 = \{v_2, v_3, \ldots, v_k\}.$$

It is not hard to see that W_1 and W_2 are solutions of DCKP such that $ax^{W_1} = ax^{W_2} = \alpha$. Thus, $bx^{W_1} = bx^{W_2} = \beta$, implying that $b_{v_1} = b_{v_2}$. As nodes $v_1, \ldots, v_k$ play similar roles, by symmetry, it follows that

$$b_{v_i} = b_{v_j} = \rho \text{ for all } \quad i, j \in \{1, \ldots, k\}. \tag{18}$$

Now let v be a node of E_1 different from v_1. We have that $W_3 = (W_1 \backslash \{v\}) \cup \{v_1\}$ is also a solution of DCKP with $ax^{W_3} = \alpha$. Hence, $bx^{W_3} = \beta$. As $bx^{W_1} = \beta$, this yields $b_v = b_{v_1}$. Since v is arbitrary in E_1, it follows that $b_v = \rho$ for all $v \in E_1$. And by (18), we obtain that $b_v = \rho$ for all $v \in W$.

To complete the proof, using 4), we can show in a similar way as in Theorem 6. that $b_j = 0$ for all $j \in V \backslash W$. Hence, $b = \rho a$. □

Note that the solutions to the DCKP induce an independence system whose circuits are precisely the minimal covers (see Schrijver 2002 for basic notions on the independence systems). Euler et al. (1987) introduce a generalization of the odd-cycle inequalities to independence systems. These more general inequalities have a structure different from that of (17), and, in some cases, can be seen as a special case of (17).

In what follows, we present new families of valid inequalities that combine the knapsack and independent set problems structures.

3.4 Clique-cover inequalities

Proposition 1 *Let C be a cover of V and $K \subset V$ a clique. Let $C^* \subset C$ such that $|C^*| = |K|$, and suppose that for all $i \in C^*$, there exists a unique item $j \in K$ that is in conflict with i. Then the following inequality*

$$\sum_{i \in K} x_i + \sum_{j \in C^*} x_j \leq \left\lfloor \frac{|C| + |K|}{2} \right\rfloor, \tag{19}$$

is valid for the DCKP(G).

Proof The inequality is obtained by a Chvátal-Gomory procedure. We have

$$\begin{aligned} x(K) &\leq 1, \\ x_i + x_j &\leq 1, && \text{for all } i \in C^*, j \in K \\ x(C) &\leq |C| - 1, \\ -x_l &\leq 0, && \text{for all } l \in C \setminus C^* \end{aligned}$$

Inequalities (19) are obtained by adding the previous inequalities, dividing by 2, and then rounding down the right-hand side. □

3.5 Clique-cover partition inequalities

Proposition 2 *Consider a clique $K \subset V$ and let $K_1, K_2, \ldots, K_r$ (r is even) be a partition of K, i.e., $\cup_i K_i = K$, and $K_i \cap K_j = \emptyset$ for all $i \neq j$. Consider a subset of items $T \in V$ such that for all $i = 1, 2, \ldots, r$, $T \cup K_i$ is a cover. Then the inequality*

$$x(K) + \frac{r}{2}x(T) \leq \left\lfloor \frac{r|T| + |K| - r + 1}{2} \right\rfloor, \tag{20}$$

is valid for the DCKP(G).

Proof The inequality is obtained by a Chvátal-Gomory procedure. We have

$$\begin{aligned} x(K) &\leq 1, \\ x(T) + x(K_i) &\leq |T| + |K_i| - 1, \text{ for all } i = 1, 2, \ldots, r. \end{aligned}$$

Inequalities (20) are obtained by adding the previous inequalities, dividing by 2, and then rounding down the right-hand side. □

In the following section, we devise a Branch-and-Cut for the DCKP based on the polyhedral results given before.

4 Branch-and-cut algorithm

The Branch-and-Cut algorithm alternates a cutting plane phase and a branching phase. During the cutting plane phase, we add, if there exist, violated inequalities. This is known as the separation process. In our Branch-and-Cut algorithm, we devise separation routines for the valid inequalities (6), (11), (12), and (14). Depending on the class of valid inequalities, we devise heuristic or exact procedures of separation.

Moreover, taking advantage of the close relationship with the classical KP, we develop a greedy heuristic allowing us to have an initial feasible solution for the DCKP.

4.1 Initial solution and preprocessing

In order to have an initial lower bound, we generate a feasible solution for the DCKP using the greedy heuristic given in Algorithm 1. The idea of this heuristic is the following. We first start by sorting items $j \in \{1, \ldots, n\}$ in a decreasing order of $\frac{p_j}{w_j + |\mathcal{V}_j|}$, where $\mathcal{V}_j$ represents the set of items that are in conflict with item j. Note that the fraction used for sorting the items refers to the classical $\frac{p_j}{w_j}$ of the KP, but also includes the item's degree in the conflict graph. That is to say, it is more interesting to begin with items having the less conflicts in order to pack more items in the knapsack during the subsequent iterations. Once the items are sorted, we put the first in the knapsack and automatically delete all its neighbors in the conflict graph. This process continues while the knapsack capacity is not violated.

Algorithm 1: Initial solution using greedy heuristic

```
Data: (G, p, w)
Result: An initial solution for the DCKP
Let x be a vector;
for j ∈ V do
    x_j = −1;
Set Δ = c, F = V;
/* Δ denotes the current knapsack capacity */
/* F is the set of nodes that can be added to the current solution */
while F ≠ ∅ do
    let i = argmax{ p_j / (w_j + |V_j|) : j ∈ F };
    if w_i ≤ Δ then
        x_i = 1, Δ = Δ − w_i, F = F \ V_i;
        for j ∈ V_i do
            x_j = 0;
    else
        x_i = 0, F = F \ {i};
return solution x ;
```

4.2 Separation routines

Let $\bar{x} \in \mathbb{R}^n$ be the current fractional solution to be cut, that is the optimal solution of the linear relaxation of the ILP given by (1)–(5). Separating a valid inequality consists in finding one or more inequalities that are violated by $\bar{x}$, or show that such inequality does not exist.

In what follows, we describe separation routines for the valid inequalities (6), (11), (12), and (14).

4.2.1 Clique inequalities separation

Clique inequalities (6) represent an interesting family of valid inequalities that are easily generated. For this reason, we choose to generate a set of valid clique inequalities within the first linear relaxation of the Branch-and-Cut tree's root node. We hence solve the linear relaxation of the ILP given by (21).

$$\begin{cases} \max & \sum_{i \in V} p_i x_i \\ & \sum_{i \in V} w_i x_i \leq c \\ & \sum_{j \in K} x_j \leq 1 \quad \text{for all } K \in \mathcal{K}, \\ & x_i + x_j \leq 1 \quad \text{for all } (i, j) \in E : \{i, j\} \nsubseteq K, \text{ for all } K \in \mathcal{K}, \\ & x_i \in \{0, 1\} \quad \text{for all } i \in V, \end{cases} \tag{21}$$

where $\mathcal{K}$ is a family of cliques.

Since identifying the whole set of cliques is NP-hard, we choose to generate a set of cliques $\mathcal{K}$ using a heuristic method. To this end, we used the greedy Algorithm 2.

Let $\mathcal{K} = \emptyset$ be the set of cliques that we are looking for, and consider $K = \emptyset$ that initially denotes an empty clique. For each item, say i, in V, we iterate the following. We first put i in K. Then, we add to K an item j in conflict with i such that $|\mathcal{V}_j|$ is maximum. After that, among all the other nodes, we add the one that is universal to items in K (*i.e.*, that is to say adjacent to all the items of K), and with a maximum degree in G. The process is repeated until no more universal node can be added. At the end, if $|K| > 2$, we obtain a clique K that we add to the family of cliques $\mathcal{K}$.

Algorithm 2: Clique generation heuristic

```
Data: (G, p, w)
Result: A set of cliques
Let 𝒦 = ∅ denote the set of cliques;
for i = 1 to n do
    Let K = {i};
    Set F = V \ i;
    while F ≠ ∅ do
        let j = argmax{|𝒱_l| : l ∈ F};
        if (j, l) ∈ E for all l ∈ K then
            K = K ⋃ {j};
        F = F \ {j};
    if |K| > 2 and K ∉ 𝒦 then
        𝒦 = 𝒦 ⋃ {K};
return the set of cliques 𝒦 ;
```

Concerning now the separation phase, we decided to apply, with slight modifications, a simple greedy heuristic presented by Nemhauser and Sigismondi (1992) for the independent set problem. The idea of the heuristic is detailed in Algorithm 3 and can be described as follows. We choose a vertex, say j, of maximum weight regarding $\bar{x}$, and set $K = \{j\}$. Then we iterate the following process. Determine, if it exists, a maximum-weight vertex according to $\bar{x}$, say t, from all the vertices that are universal to K (*i.e.*, t is adjacent to every vertex in K). Add t to K and repeat the procedure until no more universal vertex can be found. If $|K| > 2$ and $\bar{x}(K) > 1$, then the K-Clique inequality is violated. The whole process is then iterated for another initial vertex j until either we find a violated inequality or a maximum number of iterations is reached. In our algorithm, we generate, if possible, only one violated clique inequality per iteration.

4.2.2 Odd-cycle inequalities separation

The separation problem of the odd-cycle inequalities (14) can be solved exactly in polynomial time as it has been shown by Grötschel et al. (2012). The separation procedure is described

Algorithm 3: Separation of the Clique inequalities

```
Data: Fractional Solution x̄
Result: A violated clique inequality
Sort items in V such that x̄_j ≥ x̄_{j+1} for all j ∈ V;
Let i = 1 ; stop = false ;
while i < n and stop=false do
    Let K = {i};
    Set F = V \ {i};
    while F ≠ ∅ do
        let j = argmax{x̄_l : l ∈ F};
        if (j, l) ∈ E for all l ∈ K then
            K = K ⋃ {j};
        F = F \ {j};
    if |K| > 2 and Σ_{j∈K} x̄_j > 1 then
        stop = true;
return the violated Clique inequality K ;
```

in Mahjoub (2010). Consider the current fractional solution $\bar{x}$. Consider $e = (i, j) \in E$ and let $z_e = 1 - \bar{x}_i - \bar{x}_j$. It is clear that, since $\bar{x}$ satisfies (3) and (4), $z_e \geq 0$ for all $e \in E$. Using this, inequalities (14) can be written as

$$\sum_{e \in C} z_e \geq 1 \quad \text{for all} \quad C \text{ odd cycle of } G. \tag{22}$$

As a consequence, separating inequalities (14) with respect to $\bar{x}$ reduces to separating inequalities (22) with respect to z, and this can be ensured as follows. Consider graph G and let z_e be the weight of edge e for all $e \in E$. In order to separate inequalities (22), one has to look for a minimum-weight cycle in a graph with nonnegative weights. This problem can be solved in polynomial time. To this end, consider the bipartite graph $\widetilde{G} = (V' \cup V'', \widetilde{E})$ obtained from G in the following way : for each vertex $v \in V$, consider two vertices $v' \in V'$ and $v'' \in V''$, and for each edge (u, v) consider two edges $(u', v'') \in \widetilde{E}$ and $(u'', v') \in \widetilde{E}$ with the same weight $\tilde{z}_{u'v''} = \tilde{z}_{u''v'} = z_{uv}$. Obviously, looking for a minimum-weight odd cycle in G with respect to z going through a vertex v is nothing but determining a minimum-weight path in $\widetilde{G}$ with respect to $\tilde{z}$ between $v'$and v''. Since $\tilde{z} \geq 0$, this can be done in polynomial time, using for instance Dijkstra's algorithm. At the end of this step, we have (if it exists) a minimum-weight cycle in G, say C. If $\sum_{e \in C} z_e < 1$, then a violated odd-cycle inequality is detected.

4.2.3 Hypercycle inequalities separation

In what follows, we will describe a heuristic for separating the hypercycle inequalities. First, note that a hypercycle inequality (17) can also be written as

$$\sum_{i \in W} (1 - x_i) \geq \left\lceil \frac{k}{q} \right\rceil.$$

The heuristic works as follows. First, we generate a set of minimal covers $\{E_1, \ldots, E_r\}$ where $|E_i| \geq 3$, for $i = 1, \ldots, r$. Let $\mathcal{E}' = \{E_1, \ldots, E_r, E_{r+1}, \ldots, E_s\}$, where $E_{r+1}, \ldots, E_s$ are all the edges of the conflict graph G. Let $U = \bigcup_{i=1}^{s} E_i$.

Then we construct a bipartite graph $\Gamma = (U \cup \mathcal{E}', F)$ whose nodes on the left are the nodes in U, and the nodes on the right correspond to the elements of $\mathcal{E}'$. The set F of edges in Γ is defined as follows. We consider an edge between a node u of U and a set E_i of $\mathcal{E}'$ if $u \in E_i$. We associate with each E_i the weight $\sum_{j \in E_i}(1 - \bar{x}_j)$. We associate to the nodes of U the weight zero. Each path in Γ between a node u and a set E_k, such that $u \in E_k$, is a hypercycle in the hypergraph $H = (V, \mathcal{E})$. Remark that since Γ is bipartite each path alternates between the nodes of U and the sets of $\mathcal{E}'$. The heuristic will compute a minimum weight path between u and E_h for all $E_h, h = 1, \ldots, s$ and $u \in E_h$. Each computed path induces a hypercycle inequality for which we determine the corresponding q. If the weight of this path is $< \left\lceil \frac{k}{q} \right\rceil$, then the corresponding hypercycle inequality is violated by $\bar{x}$.

In our Branch-and-Cut algorithm, such inequalities were not quite efficient in the resolution. In fact, separating the hypercycle inequalities generally takes huge time, and in the majority of the cases we did not succeed to find a violated inequality. For this reason, we decided to not include them in the separation process.

4.2.4 Cover inequalities separation

We now turn our attention to the cover inequalities. The separation algorithm of the cover inequalities (7) is an NP-hard problem Klabjan et al. (1998), and it seems to be the same for the extended cover inequalities (11). Crowder et al. (1983) show that the separation problem associated with cover inequalities is equivalent to the following 0–1 knapsack-like problem:

$$\begin{cases} \min \sum_{j \in V}(1 - \bar{x}_j) y_j \\ \quad \sum_{j \in V} w_j y_j > c \\ \quad\quad y_j \in \{0, 1\} \qquad \forall j \in V, \end{cases} \tag{23}$$

where y_j is a binary variable taking 1 if j is to be inserted into the cover C, and 0 otherwise. A cover inequality is hence violated when the optimal solution of (23), say y^*, has an objective value less than 1. In this case, the cover $C = \{j \in V : y_j^* = 1\}$ yields a violated cover inequality.

In order to speed up the separation, Crowder et al. (1983) proposed a heuristic method that runs in $\mathcal{O}(n\,log(n))$. This consists in inserting items into C in a non-decreasing order regarding $\frac{1-\bar{x}_j}{w_j}$ until a cover is obtained.

Now, concerning the extended cover inequalities (ECI) (11), Gabrel and Minoux (2002) proposed an exact separation which reduces to the resolution of a sequence of 0–1 knapsack-like problems. In our algorithm, we choose to separate these inequalities using a heuristic algorithm inspired from the ones proposed by Kaparis and Letchford (2010b). This heuristic is described in Algorithm 4. We first begin by sorting items in a non-decreasing order of $\frac{1-\bar{x}_j}{w_j}$ and store them in a list L. We also initialize the cover C^* to the empty set and w^* to the capacity of the knapsack c. We then remove an item from the head of the sorted list L. If its weight is larger than w^*, then we ignore it; otherwise, we insert it in C^*. If C^* is a minimal cover, then we try to extend it to obtain a violated extended cover inequality. If the obtained extended cover inequality is not violated, we delete the heaviest items from C^*, form a new extended cover inequality induced by C^*, and check again whether this inequality is violated or not. The whole process is then iterated until either a violated ECI is found or the list L is totally explored.

Algorithm 4: Separation of the Extended Cover

Data: Fractional Solution $\bar{x}$
Result: Violated Extended Cover inequality
1 Sort items $j \in V$ such that $\frac{1-\bar{x}_j}{w_j} \leq \frac{1-\bar{x}_{j+1}}{w_{j+1}}$;
2 Let $L = V$ with order;
3 Let $C^* = \emptyset$ and $w^* = c$;
4 Set $stop = false$;
5 **while** $L \neq \emptyset$ *and stop=false* **do**
6 Let $i = L[0]$;
7 $L = L \setminus \{i\}$;
8 **if** $w_i < w^*$ **then**
9 $C^* = C^* \bigcup \{i\}$;
10 **if** C^* *is cover* **then**
11 **if** *the ECI corresponding to* C^* *is violated* **then**
12 stop = true;
13 **else**
14 Set $w^* = \max_{j \in C^*} w_j$;
15 $C^* = C^* \setminus \{j \in C^* : w_j = ?w^*\}$;
16 **return** the violated Extended Cover inequality ECI ;

Along with the extended cover inequalities, we also separate the Simple Lifted Cover inequalities (12). To this end, we devise the routine described in Algorithm 5.

We first begin by forming a cover C, then making the cover C minimal. We consider then two subsets of C, *i.e.*, F and R, corresponding to the subsets of elements of C having $\bar{x}_j > 0$ and $\bar{x}_j = 0$, respectively. Steps 9 and 11 of the Algorithm 6 refer to the up-lifting phase for the two subsets F and R. This corresponds to calculating the α coefficients for inequalities (12) given in Algorithm 6. Note that in Algorithm 6, one has to solve a knapsack problem. In our case, we use a greedy heuristic to this end. We first begin by sorting items in a non-increasing order of $\frac{z_i}{w_i}$, where $z_i = 1$ if item $i \in C$, and $z_i = \alpha_i$ otherwise. We then pack the sorted items

in the knapsack while the corresponding capacity constraint is not violated.

For a deep description of the covers inequalities separations, the reader is referred to Kaparis and Letchford (2008, 2010a, b).

Algorithm 5: Separation of the Lifted Cover inequalities

Data: A fractional solution $\bar{x}$
Result: A violated Lifted Cover inequality
Sort items $j \in V$ such that $\frac{1-\bar{x}_j}{w_j} \leq \frac{1-\bar{x}_{j+1}}{w_{j+1}}$;
Let $C = \emptyset$, $j = 1$ and $\Delta = c$;
while $j < n$ *and* $\Delta > 0$ **do**
  $C = C \bigcup \{j\}$;
  $\Delta = \Delta - w_j$;
  $j = j + 1$;
Make the cover minimal: delete elements from C;
Let $F = \{j \in C : \bar{x}_j > 0\}$;
Let $R = \{j \in C : \bar{x}_j = 0\}$;
Up-lifting in F;
If the resulting inequality is not violated, stop;
Up-lifting in R;
return The violated Lifted Cover inequality;

Algorithm 6: The Up-lifting procedure

Data: A cover inequality
Result: The lifting coefficients $\alpha_j, \ j \in V \setminus C$
Let $j_1, \ldots, j_r$ be an ordering of $V \setminus C$;
Set $t = 1$;
Consider the valid inequality $\sum_{i=1}^{t-1} \alpha_{j_i} x_{j_i} + \sum_{j \in C} x_j \leq |C| - 1$;
`/* Solve the following knapsack problem */`

$$\zeta_t = \begin{cases} \max \sum_{i=1}^{t-1} \alpha_{j_i} x_{j_i} + \sum_{j \in C} x_j, \\ s.t. \sum_{i=1}^{t-1} w_{j_i} x_{j_i} + \sum_{j \in C} w_j x_j \leq c - w_{j_t}, \\ x \in \{0,1\}^{|C|+t-1}. \end{cases}$$

Set $\alpha_{j_t} = |C| - 1 - \zeta_t$;
Stop if $t = r$;
return The lifting coefficients $\alpha_j, \ j \in V \setminus C$;

Using the previous polyhedral analysis and the devised separation algorithms, we propose a Branch-and-Cut algorithm to solve the DCKP. An experimental study is held on a set of problem instances. This will be presented in the next section.

5 Computational results

The Branch-and-Cut algorithm is implemented in C++ and tested on Bi-Xeon quad-core E5507 2.27GHz with 8Go of RAM, running under Linux. We use CPLEX 12.5 as a linear solver.

Problem instances are generated using David Pisinger's instance generator (used in Pisinger (1999) and Martello et al. (1997), see http://www.diku.dk/~pisinger/codes.html) and following the generated instances by Hifi and Michrafy (2007) and Yamada et al. (2002). We tested two kind of instances: uncorrelated and strongly correlated. For the uncorrelated instances, items' weights and profits are randomly generated from 1 to 100. Concerning the strongly correlated ones, items' weights are randomly generated from 1 to 100, and each profit $p_i = w_i + 10$ for $i = 1, \ldots, n$. We tested three groups of instances containing 100–1000 items.

The knapsack capacity for each group of instances is calculated using the formula proposed by David Pisinger:

$$c = \frac{l \times \sum_{j \in V} w_j}{S + 1},$$

where l and S are fixed parameters. We set S to 1000, and l to 5 and 10, respectively. Concerning the disjunctive aspect, conflicts between items are randomly generated and the density η of the conflict graph G takes, as in Hifi and Michrafy (2007) and Yamada et al. (2002), the following values $\eta = 0.005,\ 0.007,\ 0.009,\ 0.010$, and 0.020. Moreover, we set a time limit to 10,800 s.

In order to evaluate the performance of our Branch-and-Cut algorithm, we compare our results to the results given by running Cplex on the original formulation for the DCKP, *i.e.*, ILP (1)–(5), without taking into account the valid inequalities. Note that we also disable the valid inequalities generated by default by Cplex, in order to be able to compare it with the Branch-and-Cut. Recall that in our Branch-and-Cut algorithm we separated the valid inequalities (6), (14), (11) and (12). After testing different order of separation, we chose to separate the valid inequalities in the following order, which proved to be the most effective. We first separate the cliques (6), then the simple lifted cover inequalities (12), then the extended cover inequalities (11), and finally the odd cycles (14).

The results are reported in Tables 1, 2, 3 and 4. The three first columns of each table represent the instance main characteristics. Then the 8 following columns report results corresponding to the Branch-and-Cut algorithm. Finally, the remaining 4 columns give the results obtained by running Cplex.

Entries of the tables are the following		
n	:	Number of items
m	:	Number of edges in G (disjunction constraints)
η	:	Conflict graph density
Nodes	:	Number of nodes in the Branch-and-Cut tree
Gap-final	:	Relative error between the best lower bound and the best upper bound
Gap-root	:	Relative error between the best lower bound and the upper bound at the root
ECI	:	Number of generated Extended Cover Inequalities
LCI	:	Number of generated Lifted Cover Inequalities
Cliques	:	Number of generated cliques inequalities
OddCycles	:	Number of generated odd-cycle inequalities
CPU	:	Total time of execution (in seconds)

The value of Gap-final is used to analyze the efficiency of the algorithm. When it is equal to 0, this means that the instance is solved to optimality. The value of Gap-root points the quality of the linear relaxation at the root node of the Branch-and-Cut tree before branching compared to the best lower bound obtained at the end. Note also that the number of disjunctive constraints, *i.e.* m, is linked to the conflict graph's order and density through the following formula $m = \eta \frac{n(n-1)}{2}$.

In total, we tested the Branch-and-Cut algorithm and Cplex over 170 instances. These can be classified to easy, medium, and hard instances. The difficulty of an instance highly depends on the number of items n, the number of disjunctive constraints m, and parameter l which is directly impacting the value of the knapsack capacity (c.f., the capacity formula given above). The higher the values of n and m are, the harder the instance is. We also note that instances with capacities calculated with $l = 10$ are a bit harder than those with $l = 5$.

For the group of easy instances, our algorithm was able to solve to optimality the instances within a small amount of time. In fact, for 61 instances we reached optimality with less than 300 s. Medium instances took much more time to be solved to optimality (8 instances), and hard ones reached the time limit without having an optimal solution. Within the group of hard instances, we find some "soft" instances for which we obtained a final gap less than or equal to 10% (23 instances). Strongly hard ones reached, however, important and sometimes huge values of final gaps, showing hence the hardness of the tested instances.

Based on the results of the four tables, we can conclude that our Branch-and-Cut algorithm outperforms Cplex results for almost all the instances. In fact, applying the Branch-and-Cut algorithm, we have been able to reduce the gap values for all the instances. Our gaps at the root are always better than those of Cplex, which proves the efficiency of the generated valid inequalities to have a tightened linear relaxation. These valid inequalities were in fact extremely crucial in improving the resolution of hard instances. In fact, for instances with more than 600 items, our Branch-and-Cut algorithm provides acceptable values of gaps for which Cplex reached very huge values. Our algorithm was also able to solve to optimality instances that Cplex could not solve within the time limit. Consider Table 1 and take for instance the instance with $n = 500$, and $\eta = 0.1$ that Cplex could not solve and got a final gap equal to 6.84%. The same instance has been solved to optimality by the Branch-and-Cut algorithm in around 4080 s. The same case happened for the instance with $n = 300$ and $\eta = 0.2$ and the instance with $n = 400$ and $\eta = 0.07$ in Table 2. We also notice that compared to Cplex results, the use of the Branch-and-Cut algorithm implies a reduction of the CPU time as well as the tree size (given by the number of nodes).

Through the experimental results, we also note that our Branch-and-Cut algorithm was performant, being able to guarantee optimality for 69 instances over 170 in general hard instances. Nine instances have been solved to optimality at the root node of the Branch-and-Cut tree (5 for Cplex), and for 68 instances we have a very interesting gap at the root not exceeding 10% (47 instances solved by Cplex have a root gap less than 10%). This proves that we succeed in tightening the DCKP linear relaxation using our valid inequalities.

As it has been mentioned above, we separate 4 families of valid inequalities, *i.e.*, the cliques, the odd cycles, the extended and the lifted covers. Based on experimentations, we can see that the number of generated inequalities of each family depends on the group of instance. For almost all the instances, we notice that the number of generated odd-cycle inequalities is the most important, reaching more than 10,000 for some hard instances. For the other families of inequalities, we generate a reasonable number. Moreover, as said before, the number of generated inequalities of each family of valid inequality directly depends on the instance. For example, when the instances are of small size (*i.e.*, regarding n and m), we generate very small number of cliques and odd cycles. This is obvious because for such instances the graph of conflict is sparse. As much as the instance becomes bigger, we generate more and more cliques and odd cycles since the corresponding conflict graph becomes denser. Obviously, there is a very close relationship between the different families of valid inequalities, namely the cliques and odd cycles, and the properties related to the conflict graph (perfect, tree, claw-free,...). In fact, the conflict graph structure plays a prominent role in determining which families are more likely to appear and be effective in strengthening the linear relaxation of the problem. Recall that, in our case, conflicts between items are randomly generated, which means the structure of the conflict graph is not particular, and that the number of generated cliques and odd cycles depends only on the density of the conflict graph. For the two families of cover inequalities, *i.e.*, the extended and the lifted ones, we remark that the gen-

Table 1 Uncorrelated instances with $l = 5$

Instance			Branch-and-Cut								Cplex			
n	m	η	Nodes	Gap-final	Gap-root	ECI	LCI	Cliques	OddCycles	CPU	Nodes	Gap-final	Gap-root	CPU
100	247	0.05	1	0.00	0.00	0	0	0	0	0.01	1	0.00	0.44	0.00
100	346	0.07	1	0.00	0.00	0	0	0	0	0.01	1	0.00	0.15	0.00
100	445	0.09	3	0.00	0.38	2	0	0	0	0.02	15	0.00	0.38	0.01
100	495	0.1	1	0.00	0.00	0	0	0	0	0.01	4	0.00	0.01	0.01
100	990	0.2	7	0.00	3.18	6	2	1	1	0.06	16	0.00	17.72	0.04
200	995	0.05	10	0.00	0.28	8	6	0	0	0.09	49	0.00	0.12	0.03
200	1393	0.07	17	0.00	0.38	3	1	0	0	0.14	58	0.00	0.43	0.11
200	1791	0.09	14	0.00	0.63	20	9	0	0	0.17	141	0.00	0.53	0.12
200	1990	0.1	36	0.00	2.13	39	24	0	0	0.45	283	0.00	2.59	0.94
200	3980	0.2	110	0.00	18.09	19	8	20	21	3.81	422	0.00	33.80	7.65
300	2242	0.05	8	0.00	0.43	4	1	0	0	0.32	115	0.00	0.30	0.40
300	3139	0.07	8	0.00	1.58	2	1	0	0	0.59	62	0.00	2.35	0.89
300	4036	0.09	26	0.00	3.02	11	1	2	2	1.23	198	0.00	8.77	3.16
300	4485	0.1	112	0.00	7.12	37	13	4	11	4.99	518	0.00	14.18	12.09
300	8970	0.2	1306	0.00	26.10	0	0	54	129	140.91	13,241	0.00	48.19	558.21
400	3990	0.05	14	0.00	0.44	1	1	0	0	0.94	313	0.00	1.40	2.84
400	5586	0.07	100	0.00	4.52	33	16	4	14	6.65	919	0.00	11.20	28.65
400	7182	0.09	252	0.00	8.77	9	3	4	53	24.89	1516	0.00	19.73	67.11
400	7980	0.1	2261	0.00	17.03	23	12	36	432	255.46	21,615	0.00	30.49	1050.50
400	15,960	0.2	50,751	5.75	41.33	0	0	358	3799	10,800	170,337	17.34	87.68	10,800
500	6237	0.05	290	0.00	6.17	73	31	3	17	16.32	1020	0.00	9.48	21.58
500	8732	0.07	28,346	0.00	17.62	100	43	44	2307	5799.25	168,568	3.86	30.27	10,800
500	11,227	0.09	3100	0.00	15.48	40	15	32	596	613.15	76203	0.00	31.35	4590.42
500	12,475	0.1	17,556	0.00	19.57	18	6	86	2358	4079.55	184,684	6.84	39.46	10,800
500	24,950	0.2	36,773	29.85	58.99	0	0	240	708	10,800	109,854	64.38	134.75	10,800
600	8985	0.05	831	0.00	8.15	90	30	5	72	78.0089	4437	0.00	13.47	287.31
600	12,579	0.07	28,346	0.00	17.62	100	43	44	2307	6541.56	168,568	3.86	30.24	10,800
600	16,173	0.09	43,234	10.95	27.15	18	4	107	4475	10,800	148,353	18.64	47.61	10,800
600	17,970	0.1	38,137	10.43	26.74	58	15	131	3803	10,800	136,759	19.68	52.53	10,800
700	12,232	0.05	15,951	0.00	13.20	103	50	16	2312	4566.04	162,692	4.20	21.44	10,800
700	17,125	0.07	36,071	10.06	21.87	50	22	64	2823	10,800	142,310	23.77	45.31	10,800
700	22,018	0.09	29,930	25.87	39.93	0	0	111	1954	10,800	107,643	48.35	76.12	10,800
700	24,465	0.1	24,776	27.34	42.61	0	0	131	1884	10,800	104,920	57.51	88.33	10,800
800	15,980	0.05	37,207	3.21	14.08	22	7	20	2230	10,800	128,724	10.42	26.25	10,800
800	22,372	0.07	26,240	10.54	22.20	1	1	60	1455	10,800	119,280	33.74	54.37	10,800
800	28,764	0.09	25,021	36.23	48.65	0	0	116	1911	10,800	93,559	68.94	97.86	10,800
800	31,960	0.1	20,355	35.76	49.45	1	1	125	1221	10,800	88,836	75.37	104.47	10,800
900	20,227	0.05	23,988	9.81	19.18	24	10	26	7120	10,800	112,276	24.14	39.16	10,800
900	28,318	0.07	21,564	26.19	36.37	2	2	97	3921	10,800	86,910	47.92	68.59	10,800
900	36,409	0.09	16,757	45.39	56.89	0	0	198	3285	10,800	79,646	79.33	102.33	10,800
900	40,455	0.1	15,846	61.45	74.10	0	0	205	2387	10,800	64,479	89.26	117.31	10,800
1000	24,975	0.05	25,249	16.39	23.85	14	5	48	1730	10,800	92,720	34.58	48.10	10,800
1000	34,965	0.07	19,158	36.82	46.04	11	8	88	1020	10,800	76,146	67.85	85.60	10,800
1000	44,955	0.09	15,140	63.56	73.01	0	0	134	831	10,800	66,315	98.10	121.46	10,800
1000	49,950	0.1	1,, 919	68.67	77.48	0	0	102	505	10,800	65,025	107.04	133.64	10,800

Table 2 Uncorrelated instances with $l = 10$

Instance			Branch-and-Cut								Cplex			
n	m	η	Nodes	Gap-final	Gap-root	ECI	LCI	Cliques	OddCycles	CPU	Nodes	Gap-final	Gap-root	CPU
100	247	0.05	2	0.00	0.25	0	0	0	0	0.01	162	0.00	0.38	0.04
100	346	0.07	6	0.00	0.23	2	0	0	0	0.03	84	0.00	0.35	0.06
100	445	0.09	1	0.00	0.21	0	0	0	0	0.01	1	0.00	0.21	0.00
100	495	0.1	28	0.00	2.67	15	9	0	1	0.09	132	0.00	2.52	0.06
100	990	0.2	21	0.00	7.54	1	0	6	5	0.26	89	0.00	15.55	0.63
200	995	0.05	26	0.00	0.21	23	12	0	0	0.19	194	0.00	0.28	0.17
200	1393	0.07	41	0.00	2.47	29	17	0	3	0.50	269	0.00	3.63	1.03
200	1791	0.09	39	0.00	3.92	16	6	4	9	0.79	283	0.00	9.95	1.98
200	1990	0.1	64	0.00	6.00	49	22	1	14	1.38	384	0.00	12.93	4.90
200	3980	0.2	146	0.00	16.24	0	0	21	16	8.66	3017	0.00	47.04	67.53
300	2242	0.05	67	0.00	3.27	50	17	0	6	2.48	505	0.00	6.40	4.39
300	3139	0.07	390	0.00	9.32	49	26	2	105	17.68	2586	0.00	18.65	69.10
300	4036	0.09	310	0.00	8.68	3	2	11	40	21.53	3774	0.00	20.82	138.75
300	4485	0.1	2062	0.00	16.38	7	4	21	383	156.28	32,615	0.00	32.77	906.14
300	8970	0.2	39,341	0.00	44.85	0	0	299	1044	6154.42	311,652	40.67	116.01	10,800
400	3990	0.05	3797	0.00	11.25	52	29	1	426	253.40	22,268	0.00	18.07	510.46
400	5586	0.07	23,940	0.00	16.46	4	3	31	4749	3765.15	284,901	2.87	29.19	10,800
400	7182	0.09	47,050	6.52	24.05	2	0	88	7614	10,800	325,979	15.92	50.07	10,800
400	7980	0.1	47,956	13.32	30.72	0	0	146	8014	10,800	316,780	32.17	68.91	10,800
400	15, 960	0.2	40,904	57.82	92.60	0	0	168	271	10,800	180,179	102.62	187.73	10,800
500	6237	0.05	37,444	1.81	14.24	27	12	14	7752	10,800	277,193	5.73	23.10	10,800
500	8732	0.07	35,359	20.37	31.85	4	1	87	6615	10,800	175,741	42.21	64.95	10,800
500	11,227	0.09	38,454	26.27	41.75	0	0	168	6988	10,800	212,334	54.22	84.63	10,800
500	12,475	0.1	40,261	22.43	36.39	0	0	176	5058	10,800	197,321	49.97	80.33	10,800
500	24,950	0.2	22,229	90.82	117.51	0	0	34	46	10,800	104,919	160.28	233.28	10,800
600	8985	0.05	37,375	11.41	21.55	18	6	21	8370	10,800	208,657	20.60	37.24	10,800
600	12,579	0.07	33,158	12.41	22.55	4	1	83	6257	10,800	158,181	84.05	112.11	10,800
600	16,173	0.09	30,552	48.96	61.92	0	0	195	3965	10,800	158,181	84.05	113.38	10,800
600	17,970	0.1	28,568	46.42	59.91	0	0	173	2863	10,800	149,536	88.22	121.42	10,800
700	12,232	0.05	30,753	24.64	33.38	0	0	46	7367	10,800	164,053	42.56	59.41	10,800
700	17,125	0.07	26,705	51.52	62.70	0	0	134	4059	10,800	145,868	94.21	119.66	10,800
700	22,018	0.09	23,008	65.76	78.09	0	0	136	1604	10,800	115,391	124.06	155.46	10,800
700	24,465	0.1	19,870	67.19	80.44	0	0	143	931	10,800	117,491	125.22	160.55	10,800
800	15,980	0.05	26,084	34.27	42.32	0	0	43	3961	10,800	87,782	61.82	77.17	10,800
800	22,372	0.07	21,315	58.37	66.90	0	0	121	3379	10,800	115,179	96.29	118.57	10,800
800	28,764	0.09	16,461	89.70	100.96	0	0	117	1107	10,800	99,850	128.93	158.44	10,800
800	31,960	0.1	14,684	99.92	110.54	0	0	93	422	10,800	46,248	156.40	184.32	10,800
900	20,227	0.05	22,181	45.59	52.07	0	0	65	6990	10,800	109,889	72.45	87.06	10,800
900	28,318	0.07	16,465	68.35	77.12	0	0	115	3653	10,800	106,975	126.86	151.24	10,800
900	36,409	0.09	12,159	94.10	102.07	0	0	93	709	10,800	93,020	165.11	196.53	10,800
900	40,455	0.1	10,045	124.00	134.93	0	0	57	177	10,800	83,209	183.76	219.44	10,800

erated number depends not only on the size of the instance, but also on its type and capacity (and hence parameter l). In general, the number of generated cover inequalities is more important for the strongly correlated instances (Tables 3, 4) compared to the uncorrelated ones (Tables 1, 2). It is clearly bigger for instances with $l = 5$ than for those with $l = 10$, since capacities for the second group is bigger. We also note that for huge instances, (more than 500 items, $l = 10$), we are no more able to generate extended and lifted cover inequalities. This is obvious for these families of inequalities. Recall that in order to separate these inequalities, we use a heuristic method solving a knapsack or a knapsack-like problem with

Table 3 Strongly correlated instances with $l = 5$

Instance			Branch-and-Cut								Cplex			
n	m	η	Nodes	Gap-final	Gap-root	ECI	LCI	Cliques	OddCycles	CPU	Nodes	Gap-final	Gap-root	CPU
100	247	0.05	16	0.00	0.00	25	0	0	0	0.05	4	0.00	0.00	0.01
100	346	0.07	92	0.00	0.00	120	7	0	0	0.25	5	0.00	0.00	0.01
100	445	0.09	197	0.00	0.00	254	4	0	0	1.16	5	0.00	0.00	0.01
100	495	0.1	540	0.00	0.00	377	4	0	1	2.14	2334	0.00	2.18	0.46
200	995	0.05	1	0.00	0.23	0	0	0	1	0.08	193	0.00	0.23	0.36
200	1393	0.07	1	0.00	0.95	1	1	0	0	0.07	1198	0.00	1.12	2.22
200	1791	0.09	293	0.00	1.82	474	47	2	2	6.79	1678	0.00	2.61	4.30
200	1990	0.1	1501	0.00	2.60	2412	238	5	12	88.38	3172	0.00	5.50	8.39
300	2242	0.05	39	0.00	0.84	17	0	1	4	1.98	179	0.00	2.19	2.19
300	3139	0.07	1449	0.00	2.13	1955	150	1	28	91.91	4145	0.00	3.64	23.67
300	4036	0.09	1483	0.00	4.41	850	112	16	262	112.55	4018	0.00	7.07	70.41
300	4485	0.1	2092	0.00	5.35	203	32	19	529	181.97	6752	0.00	8.83	149.99
400	3990	0.05	445	0.00	2.49	123	16	0	48	37.30	3081	0.00	4.17	70.31
400	5586	0.07	33137	0.19	5.32	10,767	1839	17	4630	10,800	153,887	0.00	8.06	3883.62
400	7182	0.09	17,804	0.00	6.00	680	171	45	3103	3321.01	136,735	0.00	9.62	5594.97
400	7980	0.1	56,196	1.63	7.02	136	24	116	5885	10,800	293,983	3.05	13.14	10,800
500	6237	0.05	42,994	1.29	5.41	946	152	6	5418	10,800	330,593	1.16	7.47	10,800
500	8732	0.07	47,034	2.65	6.79	269	59	51	6021	10,800	271,224	4.28	11.69	10,800
500	11,227	0.09	41,337	5.76	9.82	97	33	121	7676	10,800	223,641	9.54	17.92	10,800
500	12,475	0.1	45,661	6.51	11.35	70	11	204	6804	10,800	182,486	10.00	18.64	10,800
600	8985	0.05	41,001	2.11	5.55	59	2	14	6469	10,800.1	218,199	3.66	9.10	10,800
600	12,579	0.07	37,767	6.22	9.40	49	19	65	7524	10,800	177060	9.33	15.53	10,800
600	16,173	0.09	33,690	8.91	11.81	135	50	171	4753	10,800	149,776	14.59	21.36	10,800
600	17,970	0.1	33,574	11.90	15.17	99	53	212	4414	10,800	132,624	16.52	23.84	10,800
700	12,232	0.05	29,680	5.58	8.48	138	48	29	10522	10,800	187,259	8.13	12.91	10,800
700	17,125	0.07	28,335	8.53	11.27	15	11	113	9328	10,800	152,268	13.77	19.76	10,800
700	22,018	0.09	27,815	13.39	16.18	53	20	286	6416	10,800	115,993	20.17	26.11	10,800
700	24,465	0.1	24,943	13.81	16.65	105	38	245	4295	10,800	108,295	21.04	27.81	10,800
800	15,980	0.05	26,222	6.88	9.15	139	36	41	8577	10,800	138,085	10.75	15.47	10,800
800	22,372	0.07	27,344	12.06	14.43	59	18	144	5868	10,800	113,199	18.23	23.28	10,800
800	28,764	0.09	23,309	14.48	16.77	21	6	203	3394	10,800	97,043	22.44	28.59	10,800
800	31,960	0.1	19,067	15.67	17.84	26	11	235	2127	10,800	83,675	24.30	30.98	10,800
900	20,227	0.05	28,333	9.60	11.57	128	31	42	3270	10,800	119,317	15.79	19.97	10,800
900	28,318	0.07	20,497	13.56	15.54	14	6	116	1670	10,800	72,385	22.12	26.75	10,800
900	36,409	0.09	15,899	16.70	18.60	36	5	156	1036	10,800	81,174	26.76	32.29	10,800
900	40,455	0.1	14,142	18.34	20.33	2	2	132	593	10,800	79,987	30.07	36.30	10,800
1000	24,975	0.05	23,104	10.40	12.12	149	56	70	3462	10,800	103,098	17.87	21.63	10,800
1000	34,965	0.07	15,789	15.65	17.33	19	5	117	1369	10,800	85,208	26.02	30.59	10,800
1000	44,955	0.09	10,999	19.13	20.81	0	0	89	289	10,800	68,456	28.87	34.12	10,800
1000	49,950	0.1	10,606	20.49	22.24	0	0	78	169	10,800	70,896	35.42	41.44	10,800

a greedy algorithm. The bigger the instance is, the looser the quality of solution of the knapsack subproblem is. Consequently, it would be interesting if we could improve the separation of these families for huge-sized instances.

All the arguments above show that our Branch-and-Cut algorithm is efficient for solving the DCKP. Note, however, that for one instance (see Table 3, instance $n = 400$, $\eta = 0.07$), Cplex was able to reach optimality but not the Branch-and-Cut algorithm. Our algorithm has oddly a better gap at the root than Cplex's one (5.32% for the Branch-and-Cut, 8.06% for Cplex), and it was very near to optimality (final gap equal to 0.19%). This is mainly due to the number of generated extended covers and lifted covers inequalities which reach 10,767 and 1839, respectively. At this stage, one

Table 4 Strongly correlated instances with $l = 10$

Instance			Branch-and-Cut								Cplex			
n	m	η	Nodes	Gap-final	Gap-root	ECI	LCI	Cliques	OddCycles	CPU	Nodes	Gap-final	Gap-root	CPU
100	247	0.05	20	0.00	0.00	22	0	0	0	0.06	4	0.00	0.00	0.01
100	346	0.07	1	0.00	0.00	0	0	0	0	0.01	8	0.00	0.00	0.01
100	445	0.09	7	0.00	0.51	12	0	0	0	0.07	49	0.00	0.52	0.05
100	495	0.1	1	0.00	0.09	0	0	0	0	0.01	44	0.00	0.38	0.04
100	990	0.2	540	0.00	4.83	131	63	3	26	3.458	1084	0.00	9.76	2.67
200	995	0.05	602	0.00	0.00	851	45	0	0	11.7535	1864	0.00	0.94	3.37
200	1393	0.07	406	0.00	1.41	425	67	1	7	10.1888	745	0.00	2.63	5.45
200	1791	0.09	32	0.00	1.56	0	0	4	9	1.1102	462	0.00	2.78	6.74
200	1990	0.1	1212	0.00	3.76	49	19	11	544	56.71	13889	0.00	8.40	163.13
200	3980	0.2	77,721	1.15	8.58	17	14	493	6890	10,800	464,451	0.00	18.97	6103.17
300	2242	0.05	368	0.00	1.75	160	27	0	92	24.5311	2636	0.00	3.07	56.25
300	3139	0.07	12,515	0.00	3.53	89	30	17	2645	1301.46	249,704	0.00	6.60	5536.75
300	4036	0.09	48,336	2.06	5.51	24	8	69	10473	10,800	516,532	2.44	9.66	10,800
300	4485	0.1	52,848	3.73	7.58	40	27	110	10769	10,800	651,610	4.24	12.50	10,800
300	8970	0.2	84,625	7.35	11.84	0	0	423	2263	10,800	462,214	12.46	25.38	10,800
400	3990	0.05	38,856	0.54	3.31	200	74	5	7687	10,800	408,974	0.92	5.72	10,800
400	5586	0.07	53,857	3.05	5.49	19	15	45	7349	10,800	355,690	4.77	10.29	10,800
400	7182	0.09	46,274	5.96	8.57	51	29	175	9010	10,800	293,740	8.13	14.07	10,800
400	7980	0.1	50,873	7.11	9.94	36	26	215	7448	10,800	436,858	11.44	18.90	10,800
400	15960	0.2	49,995	17.92	21.77	0	0	261	497	10,800	257,266	39.20	51.82	10,800
500	6237	0.05	33,807	4.07	6.08	33	15	17	10591	10,800	262,019	6.12	9.82	10,800
500	8732	0.07	35,345	6.89	8.98	2	1	99	11190	10,800	230,617	12.00	16.60	10,800
500	11,227	0.09	40,450	9.73	12.10	17	6	266	8008	10,800	197,699	16.04	21.42	10,800
500	12,475	0.1	41,819	10.52	12.83	0	0	310	3924	10,800	290,677	15.33	21.26	10,800
500	24,950	0.2	26,664	42.59	45.51	0	0	89	108	10,800	177,213	74.63	86.61	10,800
600	8985	0.05	29,370	7.07	8.72	3	2	29	10856	10,800	238,654	10.14	13.70	10,800
600	12,579	0.07	33,348	9.02	10.67	0	0	157	7229	10,800	168,148	16.13	20.17	10,800
600	16,173	0.09	30,007	11.82	13.63	7	7	250	3003	10,800	144,996	20.18	24.92	10,800
600	17,970	0.1	31,490	13.85	15.65	0	0	285	2741	10,800	144,860	26.06	31.36	10,800
700	12,232	0.05	30,179	8.77	10.24	9	5	61	10400	10,800	116,826	15.07	17.82	10,800
700	17,125	0.07	26,342	11.82	13.29	0	0	137	4035	10,800	137,211	20.80	24.59	10,800
700	22,018	0.09	21,134	14.43	15.91	0	0	192	2088	10,800	126,935	38.85	43.94	10,800
700	24,465	0.1	19,815	18.24	19.76	0	0	166	900	10,800	117,362	44.34	50.05	10,800
800	15,980	0.05	25,185	9.69	10.72	0	0	81	8884	10,800	143,579	14.24	16.87	10,800
800	22,372	0.07	21,294	12.58	13.72	0	0	161	2194	10,800	113,037	35.22	38.77	10,800
800	28,764	0.009	16,209	21.37	22.51	0	0	112	680	10,800	97,300	58.42	63.17	10,800
800	31,960	0.1	14,099	31.41	32.73	0	0	76	433	10,800	76,526	63.01	67.62	10,800
900	20,227	0.05	22,524	9.76	10.78	0	0	79	5713	10,800	135,516	20.35	23.01	10,800
900	28,318	0.07	15,758	14.33	15.41	0	0	104	1057	10,800	97,675	56.58	60.41	10,800
900	36,409	0.09	11,056	20.99	22.18	0	0	63	392	10,800	81,766	71.63	76.61	10,800
900	40,455	0.1	8778	47.66	48.94	0	0	43	248	10,800	65,160	97.34	103.02	10,800
1000	24,975	0.05	17,449	13.29	14.24	0	0	60	2260	10,800	450,14	31.13	33.40	10,800
1000	34,965	0.07	10,494	21.41	22.36	0	0	44	283	10,800	86,958	61.63	65.47	10,800
1000	44,955	0.09	7764	42.61	43.85	0	0	41	244	10,800	27818	94.41	98.37	10,800
1000	49,950	0.1	6318	58.12	59.05	0	0	13	110	10,800	67937	123.33	129.66	10,800

can think first in improving the separation procedures that we are using, and second in devising an efficient primal heuristic that can help us pruning some uninteresting branches of the Branch-and-Cut tree.

6 Concluding remarks

In this paper, we studied a variant of the Knapsack Problem, that is when some of the items are in conflict with some others. We presented a 0–1 ILP formulation for the problem and study the associated polytope. New families of valid inequalities are then identified. A facial study of the basic and valid inequalities is held. We then discussed the separation problem of the valid inequalities and used the whole study to develop a Branch-and-Cut algorithm. Experimental results show that the Branch-and-Cut algorithm performs well compared to Cplex, mainly for hard instances. However, as pointed for some cases, it would be interesting to boost our Branch-and-Cut algorithm, first by improving our separation procedures and second by devising efficient primal heuristics. This can indeed help prune uninteresting branches of the Branch-and-Cut tree and accelerate the resolution mainly for hard instances.

It would also be interesting to use the results obtained in this work in order to study some extensions of the DCKP. A possible extension consists in imposing an order for the selection of items. One can also think of other variations extending the problem for more than one dimension, *i.e.*, the Bi-dimensional (or Multi-dimensional) Disjunctively Constrained Knapsack Problem. Another stimulating perspective would be to extend the provided theoretical results to several other problems for which the DCKP appears as a subproblem, such as the Bin Packing Problem with conflicts.

Acknowledgements We would like to thank the referees for their valuable comments which helped to improve the presentation of the paper.

Compliance with ethical standards

Conflict of interest Mariem Ben Salem declares that she has no conflict of interest. Dr. Raouia Taktak declares that she has no conflict of interest. Prof. Dr. A. Ridha Mahjoub declares that he has no conflict of interest. Prof. Dr. Hanêne Ben-Abdallah declares that she has no conflict of interest.

Human and animal rights This article does not contain any studies with human participants or animals performed by any of the authors.

References

Akeb H, Hifi M, Mounir MEOA (2011) Local branching-based algorithms for the disjunctively constrained knapsack problem. Comput Ind Eng 60(4):811–820

Atamtürk A, Narayanan V (2009) The submodular knapsack polytope. Discrete Optim 6(4):333–344

Balas E (1975) Facets of the knapsack polytope. Math Program 8(1):146–164

Balas E, Zemel E (1978) Facets of the knapsack polytope from minimal covers. SIAM J Appl Math 34(1):119–148

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

Boyd EA (1993) Polyhedral results for the precedence-constrained knapsack problem. Discrete Appl Math 41(3):185–201

Crowder H, Johnson EL, Padberg M (1983) Solving large-scale zero-one linear programming problems. Oper Res 31(5):803–834

Euler R, Jünger M, Reinelt G (1987) Generalizations of cliques, odd cycles and anticycles and their relation to independence system polyhedra. Math Oper Res 12(3):451–462

de Farias Jr IR, Nemhauser GL (2003) A polyhedral study of the cardinality constrained knapsack problem. Math Program 96(3):439–467

Gabrel V, Minoux M (2002) A scheme for exact separation of extended cover inequalities and application to multidimensional knapsack problems. Oper Res Lett 30(4):252–264

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of np-completeness. Freeman, San Francisco

Grötschel M, Lovász L, Schrijver A (2012) Geometric algorithms and combinatorial optimization. Springer, Berlin

Gu Z, Nemhauser GL, Savelsbergh MW (1998) Lifted cover inequalities for 0–1 integer programs: computation. INFORMS J Comput 10(4):427–437

Hammer PL, Johnson EL, Peled UN (1975) Facet of regular 0–1 polytopes. Math Program 8(1):179–206

Hanafi S, Glover F (2007) Exploiting nested inequalities and surrogate constraints. Eur J Oper Res 179(1):50–63

Hifi M, Michrafy M (2007) Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem. Comput Oper Res 34(9):2657–2673

Hifi M, Negre S, Mounir MQA (2009) Local branching-based algorithm for the disjunctively constrained knapsack problem. In: IEEE international conference on computers and industrial engineering, 2009. pp 279–284

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: IEEE international processing symposium workshops (IPDPSW) parallel and distributed, pp 1547–1551

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

Hifi M, Saleh S, Wu L, Chen J (2015) A hybrid guided neighborhood search for the disjunctively constrained knapsack problem. Cogent Eng 2(1):1068,969

Kaparis K, Letchford AN (2008) Local and global lifted cover inequalities for the 0–1 multidimensional knapsack problem. Eur J Oper Res 186(1):91–103

Kaparis K, Letchford AN (2010a) Cover inequalities. Wiley Encyclopedia of Operations Research and Management Science

Kaparis K, Letchford AN (2010b) Separation algorithms for 0–1 knapsack polytopes. Math Program 124(1–2):69–91

Klabjan D, Nemhauser GL, Tovey C (1998) The complexity of cover inequality separation. Oper Res Lett 23(1):35–40

Mahjoub AR (2010) Polyhedral approaches. In: Paschos V (ed) Concepts of combinatorial optimization. ISTE-Wiely, pp 261–324

Martello S, Pisinger D, Toth P (1997) Dynamic programming and tight bounds for the 0–1 knapsack problem. Københavns Universitet, Datalogisk Institut

Martello S, Toth P (1990) Knapsack problems: algorithms and computer implementations. Wiley, New York

Nemhauser G, Sigismondi G (1992) A strong cutting plane/branch-and-bound algorithm for node packing. J Oper Res Soc 43(5):443–457

Nemhauser GL, Trotter LE Jr (1974) Properties of vertex packing and independence system polyhedra. Math Program 6(1):48–61

Pferschy U, Schauer J (2009) The knapsack problem with conflict graphs. J Graph Algorithms Appl 13(2):233–249

Pisinger D (1999) Core problems in knapsack algorithms. Oper Res 47(4):570–575

Sadykov R, Vanderbeck F (2013) Bin packing with conflicts: a generic branch-and-price algorithm. INFORMS J Comput 25(2):244–255

Schrijver A (2002) Combinatorial optimization: polyhedra and efficiency. Springer, Berlin

Senisuka A, You B, Yamada T (2005) Reduction and exact algorithms for the disjunctively constrained knapsack problem. In: International symposium, operational research Bremen

Van Roy TJ, Wolsey LA (1987) Solving mixed integer programming problems using automatic reformulation. Oper Res 35(1):45–57

Weismantel R (1997) On the 0/1 knapsack polytope. Math Program 77(3):49–68

Wolsey LA (1975) Faces for a linear inequality in 0–1 variables. Math Program 8(1):165–178

Wolsey LA, Nemhauser GL (1999) Integer and combinatorial optimization. Wiley-Interscience, New York

Yamada T, Kataoka S, Watanabe K (2002) Heuristic and exact algorithms for the disjunctively constrained knapsack problem. Inform Proces Soc Jpn J 43(9)

Zemel E (1989) Easily computable facets of the knapsack polytope. Math Oper Res 14(4):760–764

Zeng B, Richard JPP (2011) A polyhedral study on 0–1 knapsack problems with disjoint cardinality constraints: facet-defining inequalities by sequential lifting. Discrete Optim 8(2):277–301