A Cutting Plane Based Algorithm for the Multiple Knapsack Problem

C.E. Ferreira * A. Martin R. Weismantel

Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany

Abstract

In this paper we describe a cutting plane based algorithm for the multiple knapsack problem. We use our algorithm to solve some practical problem instances arising in the layout of electronic circuits and in the design of main frame computers, and we report on our computational experience. This includes a discussion and evaluation of separation algorithms, an LP-based primal heuristic and some implementation details. The paper is based on the polyhedral theory for the multiple knapsack polytope developed in our companion paper [**?**] and meant to turn this theory into an algorithmic tool for the solution of practical problems.

* On leave from University of São Paulo, Brazil.

1 Introduction

Given a set N of *items* and a set M of *knapsacks*. With every item $i \in N$ there is associated a *weight* $f_i > 0$, and every knapsack $k \in M$ has a *capacity* $F_k > 0$. Moreover, an objective function $c_{ik} \in \mathbb{R}$, $i \in N, k \in M$ is given, which reflects the cost if item i is assigned to knapsack k. The task is to assign a subset of the set of items to the set of knapsacks that yields minimum cost. This problem is called the *weighted multiple knapsack problem*.

Closely related to the (weighted) multiple knapsack problem are the single 0/1 knapsack problem and the generalized assignment problem. The single 0/1 knapsack problem is the special case of the multiple knapsack problem where $|M| = 1$.

This problem is $\mathcal{NP}$-hard (cf. [**?**]) and has been extensively studied in terms of approximation algorithms (see, for instance, [**?**]), in terms of branch and bound methods (see, for example, [**?**]) and from a polyhedral point of view (see, for instance, [**?**], [**?**], [**?**], [**?**]). The generalized assignment problem is a generalization of the multiple knapsack problem where every item i may have a particular weight f_{ik} for each knapsack k. The corresponding polyhedron was investigated in [**?**].

Our motivation for studying the multiple knapsack problem came from two applications, namely, the design of processors for main frame computers and the layout of electronic circuits. We will briefly describe both applications now.

The first problem arises in the (global) design of a main frame computer. We are given a set of electronic components. The most important property of the electronic circuits – for our purposes – is the area that these components cover. The electronic components have to be integrated on printed circuit boards, multi chip modules or other devices. Each of these devices is defined by several technical properties that we do not intend to describe here. Two properties of devices are important for us. Every device k has a *capacity* F_k, representing its "area" or the weight it can hold and a *cut capacity* S_k, describing the number of wires that can be connected to this device. The electronic components have certain contact points, called pins, from which wires can extend to pins of other components. In the logical design phase it is determined which pins of which components have to be connected by a wire to ensure certain functional properties. It is customary to call a collection of pins that have to be connected a net.

The task is to assign the electronic components to the devices in such a way that a certain objective function is minimized and a number of technical side constraints is satisfied. Among them are two essential requirements.

- For each device k the sum of the areas of the electronic components that are assigned to this device must not exceed the capacity F_k.
- The number of nets that must leave some device k must not exceed its cut capacity S_k.

The mathematical problem that arises by thoroughly modelling all (or at least the most important) aspects of this question is a rather complicated integer program. The full model appears to be hopelessly difficult – at least for the present state of integer programming technology. Thus, we investigated a hierarchy of combinatorial relaxations of the complete model. A first relaxation of the general

model is the multiple knapsack problem, where we neglect the nets completely and concentrate on the packing aspect of the problem. In knapsack terminology, the items correspond to the electronic components and the devices to the knapsacks. An appropriate objective function for the multiple knapsack problem is determined heuristically. For more details we refer the reader to [**?**].

The second application we have in mind arises in the layout of electronic circuits. Here, one major subproblem (the so-called placement problem) is to assign a set of logical units (cells) to locations on a given rectangle (silicon) subject to certain technical side constraints. In general, cells are of rectangular shape and have a particular weight representing its area. Due to the inherent complexity of the placement problem and its large scale, it is further decomposed in practice. In a first step, the given rectangle is subdivided and it is determined which cells are assigned to which of the subareas such that the total weight of the cells that are assigned to the same subarea does not exceed the corresponding area capacity. This process of iteratively subdividing areas and assigning cells to subareas is continued until every subarea contains at most one cell. If we interpret cells as items and subareas as knapsacks, we can associate with every problem arising in this decomposition scheme a multiple knapsack problem. In fact, the multiple knapsack problem does not reflect the whole situation, since – besides the area requirements – there are many additional side constraints which are to be taken into account and a complicated objective function must be minimized which strongly depends on the underlying fabrication technology. Nevertheless, solving the corresponding multiple knapsack problems seems to be a reasonable starting point to attack the much more complicated placement problems.

We consider the multiple knapsack problem from a polyhedral point of view. Let an *instance* (N, M, f, F) *of the multiple knapsack problem* be given, i. e., a set N of items, a set M of knapsacks, a weight vector $f = (f_i)_{i \in N}$ and a capacity vector $F = (F_k)_{k \in M}$. We introduce variables $x_{ik} \in \mathbb{R}^{N \times M}$ with the interpretation $x_{ik} = 1$, if item i is assigned to knapsack k, and $x_{ik} = 0$, otherwise. The *multiple knapsack polytope* $MK(N \times M, f, F)$ is defined as the convex hull of all feasible solutions to the multiple knapsack problem. It is easy to see that the following relation holds.

$$
\begin{aligned}
MK(N \times M, f, F) = \operatorname{conv}\{x \in \mathbb{R}^{N \times M} \mid & \\
& \textstyle\sum_{i \in N} f_i x_{ik} \le F_k, && \text{for all } k \in M; \quad (1)\\
& \textstyle\sum_{k \in M} x_{ik} \le 1, && \text{for all } i \in N; \quad (2)\\
& x_{ik} \in \{0, 1\}, && \text{for all } i \in N, k \in M\}.
\end{aligned}
$$

The constraints (1) are called *knapsack constraints* and the constraints (2) *SOS (Special Ordered Set) constraints*. The weighted multiple knapsack problem can

be solved – in principle – via the following linear program:

$$(??.1) \qquad \begin{array}{ll} \min & \sum_{i \in N} \sum_{k \in M} c_{ik} x_{ik} \\ & x \in MK(N \times M, f, F). \end{array}$$

In order to apply linear programming techniques we need a description of the polytope $MK(N \times M, f, F)$ by means of inequalities. This issue was addressed in [**?**]. Since the number of facet-defining inequalities is exponential in the size of the input, we use a cutting plane based algorithm to solve (**??**.1). Here, the idea is the following.

Start with a small set of valid inequalities, for example, the SOS and knapsack inequalities. These inequalities define a polyhedron P' that contains $MK(N \times M, f, F)$. Optimize the linear objective function over P' and let y be an optimal solution. Obviously, y yields a lower bound for the optimum value of the weighted multiple knapsack problem. If y is also feasible, y is an optimal solution for the weighted multiple knapsack problem. Otherwise, there exists a valid inequality for $MK(N \times M, f, F)$ that is violated by y. Thus, we must solve the *separation problem* which is to find a valid inequality that is violated by y. If such an inequality is found, we add it to the linear program and solve it again. The procedure of iteratively solving linear programs and adding violated constraints is commonly called a *cutting plane algorithm*.

A cutting plane algorithm ends with an optimal solution or (at least) with a lower bound for the weighted multiple knapsack problem. The latter case is not avoidable in general, since we do not know a complete description of the multiple knapsack polytope, and exact separation routines are not available for all known classes of facet-defining inequalities. If we intend to find an optimal solution of the problem we must embed the procedure into an enumeration scheme.

In this paper we discuss how to adapt the general cutting plane method to the weighted multiple knapsack problem. In detail, the paper is organized as follows. In Section 2 a short review is given on several classes of valid and facet-defining inequalities for the multiple knapsack polytope. Section 3 is devoted to the separation problem for these classes of inequalities. We investigate the computational complexity of the separation problem for one class and present several heuristic procedures to find violated inequalities. This issue includes lifting and complementing of inequalities. In Section 4 we deal with implementational details. In particular, a primal heuristic to find a good feasible solution for the multiple knapsack problem is presented. We have tested our cutting plane based algorithm on instances arising in the applications mentioned above. The computational results we have obtained are shown in Section 5.

2 The Multiple Knapsack Polytope

In this section we outline several classes of inequalities that play an important role in our cutting plane algorithm. For many of these classes we give necessary and sufficient conditions such that the corresponding inequalities are valid or facet-defining. For the proofs we refer the interested reader to [**?**]. We suppose throughout this chapter that an instance (N, M, f, F) of the multiple knapsack problem is given. In case $|M| = 1$, the quadruple (N, M, f, F) defines an instance of the single knapsack problem. We will frequently abbreviate this instance by the triple (N, f, F_k) where $M = \{k\}$. First of all, let us fix some notation.

It will turn out that we often refer to subinstances of the multiple knapsack problem where certain items are not feasible for certain knapsacks. Thus, we will define the polyhedron in a more general frame. Suppose $A_i \subseteq N$ and $B_i \subseteq M$ for $i = 1, \ldots, t$ are given, and let $T := \bigcup_{i=1}^{t} A_i \times B_i$. Define the polytope

$$MK(T, f, F) := \text{conv}\{x \in \mathbb{R}^T \mid \begin{array}{ll} \sum_{i:(i,k)\in T} f_i x_{ik} \leq F_k, & k \in \bigcup_{l=1}^{t} B_l, \\ \sum_{k:(i,k)\in T} x_{ik} \leq 1, & i \in \bigcup_{j=1}^{t} A_j, \\ x_{ik} \in \{0,1\}, & (i,k) \in T\}. \end{array}$$

In this notation the polytope corresponding to the multiple knapsack problem defined at the beginning coincides with $MK(N \times M, f, F)$, which we will often abbreviate by MK. It is easy to see that MK is full dimensional if and only if $f_i \leq F_k$ for all $i \in N$ and $k \in M$. Similarly, the dimension of the polytope $MK(T, f, F)$ equals $|T| = \sum_{i=1}^{t} |A_i||B_i|$ if and only if $f_i \leq F_k$ for all $(i, k) \in T$. For the remainder of this paper we assume that $f_i \leq F_k$ for all $i \in N$, $k \in M$.

A set $S \subseteq N$ is a *cover* with respect to some knapsack $k \in M$ if $\sum_{i\in S} f_i > F_k$. The cover is *minimal* with respect to k if $\sum_{i\in S\setminus\{s\}} f_i \leq F_k$ for all $s \in S$. Let $d \geq 1$ be some integer. We say that a subset of items $S \subseteq N$ is a *d-cover* with respect to some knapsack $k \in M$, if S is a cover and every subset $D \subseteq S$ with $|D| = |S| - d$ satisfies $f(D) \leq F_k$ and $f(D \cup \{s\}) > F_k$ for all $s \in S \setminus D$. Using this notation a minimal cover is a 1-cover. A set $N' \cup \{z\}$ with $N' \subseteq N$ and $z \in N \setminus N'$ is called a *(1,d)-configuration* with respect to some knapsack $k \in M$ if

1. $\sum_{j\in N'} f_j \leq F_k$;
2. $K \cup \{z\}$ is a minimal cover with respect to knapsack k, for all $K \subset N'$ with $|K| = d$.

For $I \subseteq N$ and a vector $x \in \mathbb{R}^N$ we set $x(I) := \sum_{i\in I} x_i$.

For the rest of this section we briefly review some of the valid inequalities for the multiple knapsack polyhedron.

First of all, note that the *trivial inequalities* $x_{ik} \geq 0$ define facets of MK for all $i \in N, k \in M$. In case $m \geq 2$, the SOS constraints define facets of MK.

Let (N, M, f, F) be an instance of the multiple knapsack problem. With each of the $|M|$ knapsack constraints we can associate the single knapsack polyhedron

$$SK(N, f, F_k) := \text{conv}\{x \in \mathbb{R}^N \mid \sum_{i \in N} f_i x_i \leq F_k, x_i \in \{0, 1\}, i \in N\}.$$

Obviously, for every $k \in M$ the relation $MK(N \times M, f, F) \subseteq SK(N, f, F_k)$ holds, and hence, a first question arising in this context is, whether knowledge about these single knapsack polyhedra can be used to describe the corresponding multiple knapsack polytope. Indeed, the answer to this question is yes, since the subsequent lemma states that all nontrivial facet-defining inequalities associated with the single knapsack polytopes are inherited by MK (cf. [**?**]).

Lemma 2.1 *Let $V \subseteq N$ and $k \in M$ be given. Suppose $a^T x \leq \alpha$ is a nontrivial facet-defining inequality of $SK(V, f, F_k)$. Then, $\overline{a}^T x \leq \alpha$ defines a facet of $MK(V \times M, f, F)$, where $\overline{a} \in \mathbb{R}^{V \times M}$ and*

$$\overline{a}_{il} := \begin{cases} a_i, & \text{if } l = k; \\ 0, & \text{otherwise.} \end{cases}$$

Inequalities that are valid for some polytope $MK(N \times M, f, F)$ and obtained by applying Lemma 2.1 will be called *individual*. More precisely, let $a^T x \leq \alpha$ be a nontrivial facet-defining inequality of $SK(N, f, F_k)$ and set $\overline{a}_{il} = a_i$, if $l = k$ and $\overline{a}_{il} = 0$, otherwise ($i \in V, l \in M$). Then, the inequality $\overline{a}^T x \leq \alpha$ which is valid for $MK(N \times M, f, F)$ is called individual. Valid inequalities for the polytope $MK(N \times M, f, F)$ that cannot be obtained by applying Lemma 2.1 will be called *joint*. Examples of individual inequalities are the minimal cover inequality and the (1,d)-configuration inequality that we want to present now.

Suppose that $S \subseteq N$ is a minimal cover with respect to some knapsack $k \in M$. The inequality

$$\sum_{i \in S} x_{ik} \leq |S| - 1$$

is called *minimal cover inequality* corresponding to S and k. In [**?**], [**?**] and [**?**] it was shown that the minimal cover inequality corresponding to S and k defines a facet of $SK(S, f, F_k)$ (and, thus, of $MK(S \times M, f, F)$).

Another well known class of individual inequalities consists of the (1,d)-configuration inequalities. Suppose that $N' \cup \{z\} \subseteq N$ is a (1,d)-configuration with respect to some knapsack $k \in M$. The inequality

$$\sum_{i \in N'} x_{ik} + (|N'| - d + 1)x_{zk} \leq |N'|$$

is called *(1,d)-configuration inequality* corresponding to $N' \cup \{z\}$ and k. In [**?**] it was shown that the (1,d)-configuration inequality corresponding to $N' \cup \{z\}$ and k defines a facet of $SK(N' \cup \{z\}, f, F_k)$ (and, therefore, of $MK((N' \cup \{z\}) \times M, f, F)$).

Besides minimal cover and $(1, d)$-configuration inequalities there are classes of joint inequalities that are valid for the multiple knapsack polytope (cf. [**?**], [**?**]). Among them are the *heterogeneous two-cover inequalities*, the *extended cover inequalities* and the *multiple cover inequalities*. In the remainder of this section we will briefly review some of the results from our companion paper.

In [**?**] a theorem is presented which allows to extend certain classes of facet-defining inequalities of the multiple knapsack polytope. Here, we will restrict the discussions just to the special case where a minimal cover inequality is to be extended. This yields the so-called *extended cover inequalities*.

Let $S \subseteq N$ be a minimal cover with respect to some knapsack $k \in M$ and let $M' \subseteq M$ be a subset of knapsacks with $k \in M'$. Let us choose a positive integer $r \leq \min\{|N \setminus S|, |M \setminus M'|\}$, sets $T_1, \ldots, T_r$ that are mutually disjoint subsets of $N \setminus S$ and a subset $\{k_1, \ldots, k_r\}$ of $M \setminus M'$. We call the inequality

$$\sum_{i \in S} x_{ik} + \sum_{v=1}^{r} \sum_{i \in S \cup T_v} x_{ik_v} \leq |S| - 1 + \sum_{v=1}^{r} |T_v|$$

the *extended cover inequality* corresponding to S, $T_1, \ldots, T_r$, $k, k_1, \ldots, k_r$. It is valid for MK if and only if $T_v \cup \{i\}$ is a cover with respect to k_v for all $i \in S$ and $v = 1, \ldots, r$. Necessary and sufficient conditions when these inequalities define facets were given in [**?**].

Finally, let us introduce the class of *multiple cover inequalities* which can be viewed as a generalization of minimal cover inequalities to several knapsacks. These inequalities were introduced in [**?**], and in [**?**] some conditions were derived when they define facets.

Let an instance of the multiple knapsack problem (N, M, f, F) be given. Suppose, $S \subseteq N$ and $J \subseteq M$ such that $\sum_{i \in S} f_i > \sum_{k \in J} F_k$. Under these assumptions, the inequality

$$\sum_{i \in S} \sum_{j \in J} x_{ij} \leq |S| - 1$$

is called multiple cover inequality. It is valid for the polytope $MK(N \times M, f, F)$.

For the rest of this paper we will deal with the task how to integrate these inequalities in a cutting plane algorithm and we will discuss their use in solving practical problems.

3 The Separation Problem

In this section we deal with the separation problems for the classes of inequalities presented in [**?**]. Formally, the separation problem for a given class of inequalities can be stated as follows:

> Given an instance (N, M, f, F) of the multiple knapsack problem, a vector $y \in [0,1]^{N \times M}$ and a class of valid inequalities for $MK(N \times M, f, F)$. Decide, whether y satisfies all inequalities of the given class, if not, find an inequality of that class which y violates.

All the classes of inequalities shown in the previous section include the concept of a cover. So, it seems natural to look at the separation problem for the minimal cover inequalities first. Obviously, the problem of finding a minimal (with respect to some weighting of the items) cover is $\mathcal{NP}$-hard, because it reduces to the single knapsack problem. This fact indicates that the separation problem for the minimal cover inequalities is also $\mathcal{NP}$-hard. We have not found an explicit proof of this result in the literature. Therefore, we give a short proof here.

Though this result does not prove that all the other separation problems are $\mathcal{NP}$-hard as well, we conjecture that these problems are not solvable in polynomial time. Thus, we put our emphasis on developing separation heuristics, which we will discuss in the following. We conclude this section by describing the main ideas of lifting and complementing inequalities.

3.1 Separation of Minimal Cover Inequalities

Theorem 3.1 *The separation problem for the minimal cover inequalities is $\mathcal{NP}$-hard.*

Proof.

Obviously, it suffices to show that the following decision problem (SMC) of the separation problem for the minimal cover inequalities is $\mathcal{NP}$-complete.

(SMC) Given an instance (V, a, b) of the single knapsack problem and a vector $x \in [0,1]^V$. Does there exist a subset $S \subseteq V$ such that $\sum_{i \in S} a_i \geq b + 1$, $\sum_{i \in S \setminus \{j\}} a_i \leq b$ for all $j \in S$ and $\sum_{i \in S} x_i > |S| - 1$?

We show that the decision problem associated with the knapsack problem (KP), which is known to be $\mathcal{NP}$-complete [**?**], can be reduced to (SMC).

(KP) Given an instance (R, w, k) of the single knapsack problem, an objective function vector $c \in \mathbb{N}^R$ and a positive integer number $f \in \mathbb{N}$. Does there exist a subset $S' \subseteq R$ such that $\sum_{i \in S'} c_i \geq f$ and $\sum_{i \in S'} w_i \leq k$?

First of all, note that (SMC) belongs to $\mathcal{NP}$, since it can be verified in linear time whether a given set $S \subseteq V$ is a solution to (SMC). Let us now show that (KP) can be reduced to (SMC) in polynomial time.

Let an instance $\mathcal{I}_{KP}$ of (KP) be given, i. e., an instance (R, w, k) of the single knapsack problem, a weight vector $c \in \mathbb{N}^R$ and a positive integer number $f \in \mathbb{N}$. We define an instance $\mathcal{I}_{SMC}$ of (SMC) by setting

$$\begin{aligned} V &:= R; \\ x_i &:= 1 - \tfrac{w_i}{k} + \epsilon, \quad \text{for all } i \in V; \\ a &:= c; \\ b &:= f - 1; \end{aligned}$$

where $0 < \epsilon < \frac{1}{k|V|}$.

Now we show that $\mathcal{I}_{KP}$ has a solution if and only if $\mathcal{I}_{SMC}$ has one.

Let S be a solution of $\mathcal{I}_{SMC}$. Then, $\sum_{i \in S} a_i \geq b + 1$ and $\sum_{i \in S} x_i > |S| - 1$. By substitution, we obtain $\sum_{i \in S} c_i \geq f$ and $\sum_{i \in S}(1 - \frac{w_i}{k} + \epsilon) > |S| - 1$. Thus, $|S|(1+\epsilon) - \frac{1}{k}\sum_{i \in S} w_i > |S| - 1$, or equivalently $\frac{1}{k}\sum_{i \in S} w_i < 1 + \epsilon|S|$. Since $\epsilon < \frac{1}{k|V|}$ and $w_i \in \mathbb{N}$, we have that $\sum_{i \in S} w_i \leq k$. So, S is a solution of $\mathcal{I}_{KP}$.

Conversely, let S' be a solution of $\mathcal{I}_{KP}$. Without loss of generality we can assume that S' is minimal, i. e., every proper subset of S' is no solution of $\mathcal{I}_{KP}$. Then, $\sum_{i \in S'} c_i \geq f$ and $\sum_{i \in S'} w_i \leq k$. Substituting c and f, we obtain $\sum_{i \in S'} a_i \geq b + 1$. Since S' is minimal, we have that $\sum_{i \in S' \setminus \{j\}} a_i \leq b$ for all $j \in S$. Dividing $\sum_{i \in S'} w_i \leq k$ by $-k$ and adding $|S'|$ to both sides, we obtain $|S'| - \frac{1}{k}\sum_{i \in S'} w_i \geq$

$|S'| - 1$, which is equivalent to $\sum_{i \in S'}(1 - \frac{w_i}{k}) \geq |S'| - 1$. Since $\epsilon > 0$, we get $\sum_{i \in S'}(1 - \frac{w_i}{k} + \epsilon) > |S'| - 1$, and, hence, $\sum_{i \in S'} x_i > |S'| - 1$. Thus, S' is a solution of $\mathcal{I}_{SMC}$. ■

In [**?**] a pseudopolynomial algorithm was presented that solves the separation problem for minimal cover inequalities. The algorithm is based on a transformation of the problem into a network flow problem with a pseudopolynomial number of vertices and edges. By applying network flow techniques the problem can be solved with a pseudopolynomial time and space complexity.

In our code we have integrated a heuristic to separate minimal cover inequalities that was introduced in [**?**]. This procedure can be described as follows.

Separation heuristic for minimal cover inequalities
Input: An instance of the single knapsack problem (N, f, C) and a vector $x' \in [0,1]^N$.
Output: A violated minimal cover inequality or the procedure fails.

Solve the following linear program

$$\begin{array}{rl} \min & \sum_{i \in N}(1 - x'_i)\tilde{s}_i \\ s.t. & \sum_{i \in N} f_i \tilde{s}_i \geq C + 1, \\ & 0 \leq \tilde{s}_i \leq 1 \text{ for all } i \in N. \end{array}$$

Set $S := \{i \in N \mid \tilde{s}_i > 0\}$.
Reduce S to a minimal cover.
Lift the corresponding inequality to a facet of $SK(N, f, C)$.
If the resulting inequality is violated add it to the list of violated inequalities.

We store all minimal covers we obtain by applying this procedure in a certain structure, called "pool". We use these minimal covers as substructures for some other separation routines (see next subsection). Note that the minimal cover S found by the heuristic defines a facet for the polytope $SK(S, f, C)$, but not necessarily for $SK(N, f, C)$. In order to obtain a facet-defining inequality for $SK(N, f, C)$ (and thus for $MK(N \times M, f, F)$) the inequality must be lifted. This can be done in polynomial time, as we will see in the last subsection.

The linear program in the heuristic can be solved efficiently by using Dantzig's procedure. For ease of exposition suppose for the moment that $N = \{1, \ldots, n\}$ and that $\frac{(1-x_1)}{f_1} \leq \ldots \leq \frac{(1-x_n)}{f_n}$. Set $c := \min\{i \in \{1, \ldots, n\} \mid \sum_{j=1}^{i} f_j > C\}$.

Then, the solution of the linear program is given by $s^* = (s_i^*)_{i \in N}$, where

$$s_i^* = \begin{cases} 1, & 1 \leq i \leq c-1; \\ 0, & c+1 \leq i \leq n; \\ \frac{C+1-\sum_{j=1}^{c-1} f_j}{f_c}, & i = c. \end{cases}$$

3.2 Further Separation Heuristics

In this subsection we present several procedures that we implemented to find violated inequalities for the multiple knapsack polytope. For the exposition of these separation heuristics we always assume that an instance (N, M, f, F) of the multiple knapsack problem is given and that $x' \in [0,1]^{N \times M}$ is the fractional point to be cut off.

Separation of $(1, d)$-Configuration Inequalities

In order to find violated $(1, d)$-configuration inequalities we have been implementing and evaluating two heuristics. One of these was introduced in [**?**] and can be described as follows.

Heuristic 1
Choose a minimal cover S (with respect to some knapsack k) from the pool.
Let $z \in S$ be an item with maximum weight.
Set $N' := S \setminus \{z\}$ and $d := |N'|$.
For every item $i \in N \setminus S$ such that $\sum_{j \in N' \cup \{i\}} f_j \leq F_k$,
    if $\sum_{j \in N' \cup \{i\}} f_j \leq C$ and if for all $K \subseteq N' \cup \{i\}$ with $|K| = d$,
    the set $K \cup \{z\}$ is a cover with respect to k,
        set $N' := N' \cup \{i\}$,
        lift the corresponding (1,d)-configuration inequality and
        check whether it is violated.

The second algorithm for finding $(1, d)$-configuration inequalities uses a threshold parameter t and partitions the set of items N into two sets L and S. For the items in S the corresponding weight is smaller or equal than t and the items in L have weight greater than t. The special element of the $(1, d)$-configuration is now chosen among the items in L and the other elements in the $(1, d)$-configuration are selected in S by solving a linear program. More precisely, the algorithm can be described as follows.

Heuristic 2

Set $L := \{i \in N \mid f_i > t\}$ and $S := \{i \in N \mid f_i \leq t\}$.
Choose a knapsack $k \in M$.
Let $z \in L$ be an item such that $x'_z = \max\{x'_i \mid i \in L\}$.
Solve the following linear program

$$\begin{array}{rl} \max & \sum_{i \in S} x'_i t_i \\ s.t. & \sum_{i \in S} f_i t_i \leq F_k, \\ & 0 \leq t_i \leq 1, \text{ for all } i \in S. \end{array}$$

Set $N' := \{i \in S \mid t_i = 1\}$.
If $N' \cup \{z\}$ is a $(1, d)$-configuration,
lift the corresponding inequality and check whether it is violated.

Problem	Heuristic 1	Heuristic 2
cl2 0% red.	3	106
cl2 1% red.	4	162
cl2 2% red.	1	91
cl2 3% red.	1	124
cl2 4% red.	1	143
dm1 36.75% red.	1	9
dm1 36.8% red.	1	9
dm2 27 % red.	1	4
dm2 28 % red.	0	9
dm2 29 % red.	1	6
dm2 30 % red.	0	16

Table 1: Comparison of heuristics for (1,d)-configuration separation.

Table **??** subsumes the performance of both heuristics on some problem instances (see Section 5 for more details on these instances). In column 2 the total number of violated inequalities found by the first procedure is given. Accordingly, the numbers shown in column 3 correspond to the number of violated $(1, d)$-configuration inequalities that were obtained by applying Heuristic 2. On all these examples the second heuristic performs much better than does Heuristic 1. Our explanation is that Heuristic 1 takes a cover that was stored in the pool. However, minimal covers in the pool correspond to inequalities that were violated in some previous iteration, but are mostly satisfied by the current fractional point. Hence, choosing one item from the cover as the special element of the configuration and adding some other items that do not belong to the cover may not give rise to a violated inequality. In contrast, the performance of Heuristic 2 only

depends on the fractional point x' and on the parameter t. In our implementation we sort the items in increasing order according to their weights, w.l.o.g. $N = \{1, \ldots, n\}$, $f_1 \leq f_2 \leq \ldots \leq f_{|N|}$, and run the heuristic with all possible sets L and S such that $S = \{1, \ldots, i\}$, $L = \{i+1, \ldots, |N|\}$ and $f_i < f_{i+1}$.

Separation of Multiple Cover Inequalities

Given a set $J \subseteq M$ of knapsacks. A multiple cover with respect to J is a set of items $S \subseteq N$ such that $f(S) > F(J)$. The procedure we suggest in order to find violated multiple cover inequalities, is a two-stage process. First, we determine a set $\mathcal{M}$ of candidate sets $M' \subseteq M$. For every of these sets $M' \in \mathcal{M}$ we then try to determine a set of items that defines a multiple cover. The latter problem is solved, in principle, by applying the minimal cover separation routines to the "aggregated" knapsack with capacity $F(M')$. More precisely, suppose, $x' \in \mathbb{R}^{N \times M}$ is the current fractional point and $M' \subseteq M$ is the set of knapsacks for which we want to find a multiple cover. We define the instance (N, f, C) of the single knapsack problem by setting $C := F(M')$. Moreover we define a new fractional point, $y \in \mathbb{R}^N$ say, where $y_i := \sum_{k \in M'} x'_{ik}$ for all $i \in N$. With input (N, f, C) and y, the routines for finding minimal cover inequalities are called. If these calls yield a violated minimal cover inequality, it is transformed back to the original space and, hence, corresponds to a violated multiple cover inequality.

Unfortunately, there are $2^{|M|} - (|M| + 2)$ different sets of knapsacks for which we could perform the algorithm just described. This would, even for small $|M|$, result in an immense running time. Therefore, we generate a set $\mathcal{M}$ of candidates $M' \subseteq M$ according to some heuristic rules which we briefly explain now. For every item $i \in N$ we determine $B_i := \{k \in M \mid x'_{ik} > 0\}$, where x' is the current LP solution. If $2 \leq |B_i| \leq |M| - 1$, we consider B_i as one candidate set for which we try to find a multiple cover as described before. In case $B_i = M$, we simply add all subsets of M of cardinality $|M| - 1$ to the set $\mathcal{M}$ of candidate sets. The idea for this (heuristic) selection of the candidate sets is that B_i is a subset of knapsacks where still (at least) one item is in conflict with its "correct position". Thus, additional inequalities are needed that provide further information how to handle this conflict.

Proceeding in this way, the number of different candidate sets that are generated does not exceed the number $|N| + |M| - 1$.

Separation of Extended Cover Inequalities

Let S be a cover with respect to some knapsack k, let $l \in M \setminus \{k\}$ and $T \subseteq N \setminus S$.

The inequality

$$\sum_{i\in S}(x_{ik}+x_{il})+\sum_{i\in T}x_{il}\leq |S|+|T|-1$$

is the extended cover inequality corresponding to S, T, k and l. As shown in [**?**], it is valid for the polytope MK if and only if $T\cup\{i\}$ is a cover with respect to l for all $i\in S$. We have been implementing two (heuristic) procedures for the extension of some minimal cover S by some set T which we briefly explain now.

The first routine chooses a cover S (with respect to some knapsack k) that is stored in the pool. For every $l\in M\setminus\{k\}$ we proceed as follows. Initialize T by setting $T:=N\setminus S$. Iteratively, we remove from T the item i with minimal value $\frac{x_{il}}{f_i}$ until the condition $f(T)\leq F_l$ holds. Finally, we check whether $T\cup\{s_{min}\}$ defines a cover with respect to k where $s_{min}\in S$ is an element in S mith minimal weight $f_{s_{min}}$. In case this is true, we lift the extended cover inequality corresponding to S, T, k and l and check for a possible violation.

At this point, let us note that the criterion how to delete items from the initial set T is of very heuristic nature. The idea is that an item i with heigh weight and small value x_{il} is very unlikely contained in the set T belonging to a violated extended cover inequality. Second, the condition "$T\cup\{s_{min}\}$ defines a cover" guarantees that the resulting inequality is valid, since $f(T\cup\{s\})\geq f(T\cup\{s_{min}\})>F_k$ for all $s\in S$. One property of this algorithm is that the result it produces strongly depends on the covers stored in the pool. Another way to generated extended cover inequalities is to start from the scratch and build up a cover S and a set $T\setminus S$ based on the information given by the fractional solution x'. This is, roughly speaking, the outline of the subsequent algorithm.

Extended Cover Heuristic 2
Choose two knapsacks $k,l\in M$, $k\neq l$ and solve the linear program:
min $\sum_{i\in N}(1.0-x'_{ik}-x'_{il})s_i$
s.t. $\sum_{i\in N}f_is_i>F_k$,
$0\leq s_i\leq 1$ for all $i\in N$.
Set $S:=\{i\in N\mid s_i>0\}$.
Reduce S to a minimal cover.
Let s_{min} be an item in S with minimum weight.
Solve the linear program:
min $\sum_{i\in N\setminus S}(1.0-x'_{il})t_i$
s.t. $\sum_{i\in N\setminus S}f_it_i>F_l-f_{s_{min}}$, Set $T:=\{i\in N\setminus S\mid t_i>0\}$.
$0\leq t_i\leq 1$ for all $i\in N\setminus S$.
Reduce T so that $T\cup\{s_{min}\}$ is a minimal cover.
Lift the extended cover inequality corresponding to S, T, k and l.

We have performed several experiments to evaluate both algorithms. It turned out that in most cases the latter procedure finds more violated inequalities than does the first one. Nevertheless, we included both routines in our cutting plane algorithm in order to find as many extended cover inequalities as possible.

Separation of Heterogeneous Two-Cover Inequalities

Before sketching the ideas how to find violated heterogeneous two-cover inequalities, let us recall the definition of this class. Suppose, two knapsacks $k, l \in M, k \neq l$ are given. Moreover, assume that $S \subseteq N$ is a cover with respect to k and $G \subseteq N \setminus S$ is some set of items. The inequality

$$\sum_{i \in S} x_{ik} + \sum_{i \in S \cup G} (|S| - 1)x_{il} \leq |S|(|S| - 1)$$

is called *heterogeneous two-cover inequality* corresponding to S, G, k and l. It is valid for the multiple knapsack polytope MK if and only if for all $\tilde{G} \subseteq G$ and $\tilde{S} \subseteq S, |\tilde{G}| = |\tilde{S}| \geq 1$, the set $S \setminus \tilde{S} \cup \tilde{G}$ is a cover with respect to knapsack l. The algorithm we designed and implemented for the separation of this class of inequalities proceeds in a two stage process. First a cover S with respect to knapsack k is generated by solving the same linear program as in the Extended Cover Heuristic 2. Thereafter, a set $G \subseteq N \setminus S$ is generated by successively adding elements of $N \setminus S$ to G as long as the following condition is satisfied. The condition requires that every subset $T \subseteq S \cup G$, $|T| = |S|$ and $T \cap G \neq \emptyset$ is a cover with respect to l. This test can be performed efficiently by maintaining a set T_{min} with the $|S| - 1$ smallest elements in $S \cup G$ and updating this set T_{min} for each new item added to G. Moreover, the condition guarantees that the resulting inequality is valid for MK. If the procedure succeeds in finding a violated heterogeneous two-cover inequality, a lifting step is performed. This issue is discussed in the next subsection.

3.3 Lifting and Complementing Inequalities

The concept of lifting and complementing inequalities is a very important issue in solving practical problem instances via a polyhedral based approach. The idea is to study subpolytopes obtained by fixing the value of some variables to zero or one. The inequalities found for the subpolytopes must then be "translated" back to the original space. In other words, the coefficients of the fixed variables in the inequality must be determined. We call *lifting* the operation of calculating the coefficients of those variables that are fixed to zero in the subproblem. If some variable was fixed to one, we refer to the operation of calculating this coefficient as *complementing*. In the following we will discuss both operations in more detail.

Lifting Inequalities

Given an index set N, a subset $S \subseteq N$ and a polytope $P \subseteq [0,1]^N$ with 0/1 vertices. Moreover, suppose that $a^T x \leq \alpha$ is a valid inequality for the polytope $P \cap \{x \in \mathbb{R}^N \mid x_i = 0 \text{ for all } i \in N \setminus S\}$. We say that an inequality $\overline{a}^T x \leq \overline{\alpha}$ is a *lifting* of $a^T x \leq \alpha$ if $\overline{a}_i = a_i$ for all $i \in S$, $\overline{\alpha} = \alpha$ and $\overline{a}^T x \leq \overline{\alpha}$ is valid for P. The coefficients $\overline{a}_k$ ($k \in N \setminus S$) are called *lifting coefficients*. In principle, lifting some inequality can be solved via the follwing procedure ([**?**]).

> Initialize $\overline{a}_i = a_i$ for all $i \in S$.
> Choose a sequence of the variables in $N \setminus S$ (w.l.o.g. we assume that these variables are indexed by $\{1, \ldots, |N \setminus S|\}$).
> For $k = 1, \ldots, |N \setminus S|$ calculate
>
> $$\begin{array}{rlll} \gamma_k := & \max & \overline{a}^T x & \\ & s.t. & x \in P \cap \{x \in \mathbb{R}^N \mid x_k = 1, x_i = 0 \text{ if } |N \setminus S| \geq i > k\}; & \\ & & x_i \in \{0,1\}, \text{ for all } i \in S \cup \{1, \ldots, k-1\}. & \\ \overline{a}_k := & \alpha - \gamma_k. & & \end{array}$$

At this point a few comments are appropriate. First, note that the lifted inequality depends on the ordering of the variables in $N \setminus S$. Second, if the inequality $a^T x \leq \alpha$ is facet-defining, then the lifted inequality is facet-defining as well. Finally, let us point out that the computation of γ_k is $\mathcal{NP}$-hard, in general. However, if P is the single knapsack polytope, Zemel showed that the coefficients γ_k (and hence $\overline{a}_k$) can be computed in polynomial time ([**?**]). His idea is to "dualize" the problem, slightly modify it and solve this modified program by applying dynamic programming techniques. Due to the particular structure of the resulting optimization problem the running time of the dynamic program is bounded by a polynomial in the size of the input data.

We have been implementing this idea and applied it to the classes of minimal cover and $(1,d)$-configuration inequalities.

For the joint inequalities we cannot calculate the exact lifting coefficients efficiently. Instead, we just determine an upper bound, u_k say, on the value γ_k and determine the (approximate) lifting coefficient by setting $\overline{a}_k := \alpha - u_k$. This approach guarantees validity of the generated inequality. In case of heterogeneous two-cover- and extended cover inequalities we solve the linear relaxation of the program given in the procedure above and define u_k as the objective function value rounded down. In order to lift multiple cover inequalities we proceed in a slightly different way. Roughly speaking, lifting of multiple cover inequalities is heuristically performed by lifting a cover inequality. Suppose, $S \subseteq N$ is a multiple

cover with respect to $M' \subseteq M$. With the multiple cover inequality corresponding to S and M' we associate a cover inequality $\sum_{i \in S} y_i \leq |S| - 1$ which is valid for the single knapsack polytope $SK(N, f, F(M'))$. The latter inequality is now lifted by using Zemel's procedure. Let $b^T y \leq |S| - 1$ denote this lifted inequality and define $\overline{a}_{ik} := b_i$ for all $k \in M'$ and $i \in N$. Then, it is easy to see that the inequality $\overline{a}^T x \leq |S| - 1$ is a lifted multiple cover inequality corresponding to S and M' which is valid for the multiple knapsack polytope MK.

Complementing Inequalities

To our knowledge the idea of complementing inequalities (lifting the complemented variables) was first suggested in [**?**]. In the following we will briefly discuss the algorithms we designed and implemented for complementing the inequalities introduced in section 2.

Given an index set N, a subset $S \subseteq N$ and a polytope $P \subseteq [0,1]^N$ with 0/1 vertices. Moreover, suppose that $a^T x \leq \alpha$ is a valid inequality for the polytope $P \cap \{x \in \mathbb{R}^N \mid x_i = 1 \text{ for all } i \in N \setminus S\}$. We say that an inequality $\overline{a}^T x \leq \overline{\alpha}$ is a *complementing* of $a^T x \leq \alpha$ if $\overline{a}_i = a_i$ for all $i \in S$, $\overline{\alpha} \geq \alpha$ and $\overline{a}^T x \leq \overline{\alpha}$ is valid for P. The coefficients $\overline{a}_k$ ($k \in N \setminus S$) are called *complementing coefficients*. Similarly to the previous subsection, a general procedure can be derived for the complementing of some inequality $a^T x \leq \alpha$ which is valid for the polytope $P \cap \{x \in \mathbb{R}^N \mid x_i = 1 \text{ for all } i \in N \setminus S\}$.

Initialize $\overline{a}_i = a_i$ for all $i \in S$ and set $\overline{\alpha} := \alpha$.
Choose a sequence of the coefficients in $N \setminus S$ (w.l.o.g. we assume that the variables are indexed by $\{1, \ldots, |N \setminus S|\}$).
For $k = 1, \ldots, |N \setminus S|$ calculate

$$
\begin{array}{lll}
\mu_k := & \max & \overline{a}^T x \\
& s.t. & x \in P \cap \{x \in \mathbb{R}^N \mid x_k = 0, x_i = 1 \text{ for all } |N \setminus S| \geq i > k\}; \\
& & x_i \in \{0,1\}, \text{ for all } i \in S \cup \{1, \ldots, k-1\}; \\
\overline{a}_k := & \mu_k - \overline{\alpha}. & \\
\overline{\alpha} := & \mu_k. &
\end{array}
$$

By using similar techniques as described in [**?**] we now show that, if P is the single knapsack polytope, the complementing coefficients for minimal cover and (1,d)-configuration inequalities can be computed in polynomial time.

Let $k \in \{1, \ldots, |N \setminus S|\}$ be given and suppose that $\overline{a}^T x \leq \overline{\alpha}$ defines a facet of the polytope $SK(S \cup \{1, \ldots, k-1\}, f, F - \sum_{i=k}^{|N \setminus S|} f_i)$. In order to obtain a facet-defining inequality of the polytope $SK(S \cup \{1, \ldots, k\}, f, F - \sum_{i=k+1}^{|N \setminus S|} f_i)$ the

value μ_k must be determined.

$$\begin{array}{rll} \mu_k = & \max & \overline{a}^T x \\ & s.t. & \sum_{j \in S \cup \{1,\ldots,k-1\}} f_j x_j \leq F - \sum_{i=k+1}^{|N \setminus S|} f_i, \\ & & x_j \in \{0,1\}, \text{ for all } j \in S \cup \{1,\ldots,k-1\}. \end{array}$$

In order to solve this problem we consider, for $\mu \in \mathbb{N}$, the following minimization problem (cf. [**?**]).

$$\begin{array}{rll} d(\mu) = & \min & \sum_{j \in S \cup \{1,\ldots,k-1\}} f_j x_j \\ & s.t. & \overline{a}^T x \geq \mu \\ & & x_j \in \{0,1\}, \text{ for all } j \in S \cup \{1,\ldots,k-1\}. \end{array}$$

The nice relationship between the two problems is that μ_k can be expressed in terms of the second problem, namely $\mu_k = \max\{\mu \mid d(\mu) \leq F - \sum_{i=k+1}^{|N \setminus S|} f_i\}$. Using dynamic programming techniques the problem of computing $\max\{\mu \mid d(\mu) \leq F - \sum_{i=k+1}^{|N \setminus S|} f_i\}$ can be solved in time complexity $O(|N|\mu_k)$. Hence, if there exists an upper bound for μ_k that is polynomial in $|N|$, then μ_k itself can be computed in polynomial time. A trivial upper bound for μ_k is given by

$$\mu_k \leq \sum_{i \in S} \overline{a}_i + \overline{a}_1 + \ldots + \overline{a}_{k-1}.$$

Since $\overline{a}_i = \mu_i - \mu_{i-1}$ for all $i = 1, \ldots, k-1$ (with $\mu_0 = \overline{\alpha}$) and $\overline{a}_i = a_i$ for all $i \in S$, we obtain

$$\mu_k \leq \sum_{i \in S} \overline{a}_i + \mu_{k-1} - \overline{\alpha}.$$

Solving this recursion formula yields $\mu_k \leq k(\sum_{i \in S} \overline{a}_i - \overline{\alpha})$. Moreover, in the case of minimal cover and (1,d)-configuration inequalities, $\sum_{i \in S} \overline{a}_i - \overline{\alpha}$ is bounded by $|N|$. Finally, k is bounded by $|N|$ and, consequently, $|N|^2$ is an upper bound for μ_k which is, of course, polynomial in the encoding length of the input.

In order to complement joint inequalities, this type of approach does not work any more. Hence, we determine approximate complementing coefficients by applying the same techniques as in the case of lifting (see previous subsection). Besides this we have been implementing an additional complementing procedure for extended cover inequalities. This issue is discussed next. For the ease of exposition let us assume that just one variable x_{jr} is fixed to one. Moreover, assume that $\sum_{i \in S} x_{ik} + \sum_{i \in S \cup T} x_{il} \leq |S| + |T| - 1$ is an extended cover inequality which is valid for $MK \cap \{x_{jr} = 1\}$. This implies, in particular, that S is a cover with respect to k and that $T \cup \{i\}$ is a cover with respect to l for all $i \in S$. Depending on r we distinguish the following cases:

- $r \neq k, r \neq l$: Then, the complementing coefficient corresponding to variable x_{jr} is 0 and the extended cover inequality $\sum_{i \in S} x_{ik} + \sum_{i \in S \cup T} x_{il} \leq |S| + |T| - 1$ is valid for MK.

- $r = k$: In this case the relation $\sum_{i \in S} f_i > F_k - f_j$ holds and hence, $S \cup \{j\}$ is a cover with respect to k. If, in addition, $T \cup \{j\}$ is a cover for knapsack l, we add j to S, i.e., $S := S \cup \{j\}$ and the extended cover inequality $\sum_{i \in S} x_{ik} + \sum_{i \in S \cup T} x_{il} \leq |S| + |T| - 1$ is valid for MK. Otherwise, the coefficient corresponding to variable x_{jk} is set to 1 and the coefficient corresponding to variable x_{jl} is set to 0. This yields the inequality $\sum_{i \in S \cup \{j\}} x_{ik} + \sum_{i \in S \cup T} x_{il} \leq |S| + |T| - 1$ which is valid for MK, but no longer an extended cover inequality.
- $r = l$: The conditions above imply that $T \cup \{i\}$ is a cover with respect to l for all $i \in S$, i.e., $f(T) + f_i > F_l - f_j$. Therefore, we can append j to the set T, i.e., $T := T \cup \{j\}$ and the inequality $\sum_{i \in S} x_{ik} + \sum_{i \in S \cup T} x_{il} \leq |S| + |T| - 1$ is the extended cover inequality corresponding to S, T, k and l. It is valid for MK, as one can easily convince oneself.

In fact, this complementing procedure for extended cover inequalities can be generalized to the case when more than one variable is fixed to one.

Let us conclude this section with some remarks on the interplay between lifting, complementing and separation routines. Suppose, $x' \in \mathbb{R}^{N \times M}$ is a fractional solution that is obtained during the run of the cutting plane algorithm. In our implementation all variables that are close to zero or one will be fixed temporarily. More precisely, there is a threshold parameter, t say, and a variable x_{ik} is fixed if $x'_{ik} < t$ or $x'_{ik} > 1 - t$ holds. Initially, t is set to 0.05. This way, we reduce the size of the original problem instance drastically. Thereafter, the separation routines are called in order to find violated inequalities for the subpolytope $P := MK \cap \{x_{ik} = 0, \text{ if } x'_{ik} < t, x_{ik} = 1, \text{ if } x'_{ik} > 1 - t\}$. If the separation algorithms succeed in finding violated inequalities for P, the appropriate complementing and lifting procedures are applied. Table 2 shows the number of violated inequalities that are obtained by fixing variables to one (as described above), calling the separation routines, applying lifting and complementing the corresponding inequalities. For more details on the problem instances we refer to section 5.

4 Further Implementation Details

At the end of the introduction we have already discussed the general outline of a cutting plane based algorithm. In order to make such an algorithm effective and applicable for the solution of practical problem instances many "little" tricks and problem dependent ideas must be implemented. In this section we briefly report on some of these issues.

Problem	# viol. ineq.
cl2 0% red.	60
cl2 1% red.	127
cl2 2% red.	109
cl2 3% red.	123
cl2 4% red.	157
dm1 36.75% red.	14
dm1 36.8% red.	14
dm2 27 % red.	42
dm2 28 % red.	47
dm2 29 % red.	50
dm2 30 % red.	15

Table 2: Separation in subproblems using complementing procedure.

For solving the linear programs we use the CPLEX Callable Library [**?**], a very fast and robust LP solver written and supported by R. E. Bixby. The initial linear program in our algorithm is composed of the knapsack constraints, the SOS constraints and the trivial inequalities $0 \leq x_{ik} \leq 1$, for all $i \in N$, $k \in M$. For the practical applications we have in mind all items must be assigned to the knapsacks, i. e., the SOS constraints must be satisfied with equality. Thus, instead of using the SOS constraints we factually add the corresponding equalities $\sum_{k \in M} x_{ik} = 1$ $(i \in N)$ to the initial linear program. From now on, we will call a solution *feasible* for the multiple knapsack problem only if all items are assigned to the knapsacks, i. e., if all SOS constraints are satisfied with equality.

One main aspect in the design of a cutting plane algorithm is to keep the actual linear program of moderate size. To this end, we eliminate inequalities from the LP that are not tight at the current linear programming solution x'. In each iteration we call all separation algorithms discussed in the previous section, but we control the number of violated inequalities added to the LP by a parameter. If we find more violated inequalities than specified by this parameter, we take the most "violated" inequalities, i. e., the inequalities $a^T x \leq \alpha$ whose value $a^T x' - \alpha$ is smallest. Of course, we make sure that no redundant inequalities are added to the linear program.

An important issue in our algorithm is the fixing of variables by reduced costs. Let z' be the objective function value of the current LP solution, z^* be an upper bound on the value of the optimum solution, $x' = (x'_{ik})$ the current LP solution and $d = (d_{ik})$ the corresponding reduced cost vector. Then, each non basic

variable x_{ik} with $x'_{ik} = 0$ and $z^* - z' \leq d_{ik}$ can be fixed to zero. Similarly, each non basic variable x_{ik} with $x'_{ik} = 1$ and $z^* - z' \leq -d_{ik}$ can be fixed to one. Moreover, further variables can be fixed by logical implications, for example, if some variable x_{ik} is fixed to one by the reduced cost criterion, then all variables x_{il}, $l \in M \setminus \{k\}$, can be fixed to zero. Table **??** shows for some problem instances the number of variables (in percentages) that are fixed by this procedure.

Problem	Variables fixed	
cl2 0% red.	2433	(22.32%)
cl2 1% red.	3039	(27.89%)
cl2 2% red.	2184	(20.04%)
cl2 3% red.	2933	(26.91%)
cl2 4% red.	3293	(30.22%)
dm1 36.75% red.	89	(8.79%)
dm1 36.8% red.	89	(8.79%)
dm2 27 % red.	1544	(33.33%)
dm2 28 % red.	1538	(33.20%)
dm2 29 % red.	3384	(73.05%)
dm2 30 % red.	1346	(29.05%)

Table 3: Variables fixed by reduced costs.

Clearly, we cannot guarantee that our cutting plane algorithms finds an integer optimum solution, since a complete description of the multiple knapsack polytope is not known and exact separation algorithms for the known classes of inequalities are not at hand. Due to this fact, we have also implemented several LP-based heuristics in order to obtain good upper bounds on the value of the optimal solution. These heuristics are based on rounding the current linear programming solution x'.

In the first heuristic we proceed as follows. We set $F'_k := F_k$ for all $k \in M$ and determine the item i whose value $\max\{x'_{ik} \mid k \in M \text{ and } f_i \leq F'_k\}$ is maximum. Let i^* be the "maximum" item and k^* the corresponding knapsack. We assign i^* to knapsack k^* and update the value F'_{k^*} by setting $F'_{k^*} := F'_{k^*} - f_{i^*}$. We continue this way until a feasible solution is found (i. e. all items are assigned) or no further item can be assigned.

The second heuristic differs from the first one by selecting the node that is assigned next, randomly. More precisely, we determine a random sequence of the items, and, according to this sequence, we compute $\max\{x'_{ik} \mid k \in M$ and

$f_i \leq F'_k\}$. If there exists no more feasible knapsack for item i we stop. Otherwise, we assign item i to a knapsack k^* where the maximum value is attained, update F'_{k^*} accordingly and continue.

In the third heuristic, we interpret each vector $(x'_{ik})_{k\in M}$, for $i \in N$, as a probability distribution, i. e., we toss a dial that assigns item i to knapsack k with probability x'_{ik}. The sequence according to which the items are chosen is again randomly determined. This procedure is performed several times depending on a parameter that is a multiple of the number of fractional variables.

It turns out that none of these heuristics is superior to the others. In most examples, the first two procedures are more successful at the very beginning, whereas the third one performs better in the sequel.

Moreover, we have implemented an improvement heuristic that is based on the ideas of [**?**]. The procedure applied to our problem is described in the following.

Improvement heuristic.
Initialize z' with z, where $z \in \{0,1\}^{N\times M}$ is a given feasible solution.
For $p := 1$ to *number-of-passes*
    Initialize z^p with z'.
    Set $F'_k := F_k - \sum_{i\in N} f_i z^p_{ik}$ for all $k \in M$.
    All items are supposed to be "unlocked".
    While $\{i \in N \mid i$ unlocked, $z^p_{ik} = 0$ and $f_i \leq F'_k$ for some $k \in M\} \neq \emptyset$
        Determine an unlocked item i^* and a knapsack k^* with $z^p_{i^*k^*} = 0$
        such that $c_{i^*k^*} = \min\{c_{ik} \mid i$ is unlocked, $z^p_{ik} = 0$ and $f_i \leq F'_k\}$.
        Move item i^* to knapsack k^*.
        Update z^p and F'.
        Lock item i^* (in order to avoid cycling).
    Let z^* be the best solution found during this pass.
    If $c^T z^* < c^T z'$, set $z' := z^*$.
    Else return z' and STOP.
Return z'.

It turns out that these three primal heuristics together with the improvement heuristic perform quite well and even find the optimal solution in many cases (see next section).

Finally, let us note that we have embedded our cutting plane algorithms in an enumeration scheme. Here, if we find no further violated inequality and the current linear programming solution is not integer, we select a variable that is

closest to 0.5 and create two subproblems, one where the selected variable is set to zero, and one where this variable is set to one. The resulting branching tree is worked out in depth first search manner.

5 Computational Results

In this section we report on computational experiences with our cutting plane based algorithm. We have tested the algorithm on multiple knapsack problem instances arising in the design of main frame computers and in the layout of electronic circuits.

Let us first focus on the class of problems that arise in the design of main frame computers. Table **??** summarizes the data. (Instances coming from this application are abbreviated by dm in the tables.) Column 2 and 3 give the number of items and knapsacks, respectively, whereas in columns 4 and 5 the total weight of the items and the total sum of the capacities are shown.

Problem	$\|N\|$	$\|M\|$	$\sum_{i \in N} f_i$	$\sum_{k \in M} F_k$
dm1	257	4	83827	132704
dm2	772	6	284608	423972

Table 4: Description of the examples dm.

Table **??** presents the solutions obtained by our algorithm. Neither individual nor joint inequalities where necessary to obtain the objective function value of an optimal solution, which was already found by the primal heuristics in the first iteration. The CPU Times are in seconds obtained on a Sun Sparc IPX.

Problem	Opt. Sol.	Ind. Ineq.	Joint Ineq.	CPU Time
dm1	236250	0	0	1.97
dm2	81120	0	0	6.50

Table 5: Computational results for examples dm.

The fact that both problem instances are trivial is not surprising, since the total sum of the knapsack capacities is much bigger than the sum of the weights of the items. The reason for that is that after assigning the items to the devices the

nets must be connected by wires which requires a certain amount of space. The real amount of space that is necessary for connecting the wires is not available in advance. Thus, the numbers in Column 5 of Table ?? are only a rough estimate. A usual procedure in practice is to start with some initial capacities of the devices and try to find a solution that assigns the modules to the devices and connects the nets by wires. If this succeeds, the capacities of the devices are reduced, the whole problem is solved again, and it is continued in this way until no further area reduction is possible. In fact, one of the main goals in the design of main frame computers is to reduce the available amount of space as far as possible. So, from a practical point of view a very interesting question is how far the capacities of the devices can be reduced at most. We followed this question and iteratively reduced the total amount of the knapsack capacities.

Problem	Red.	Opt. Sol.	Ind. ineq.	Joint ineq.	CPU
dm1	36.75%	236250	18	15	9.72
dm1	36.8%	236250	18	15	9.42
dm2	27%	81134	23	40	10:20.80
dm2	28%	81176	27	55	14:36.73
dm2	29%	81204	25	58	17:21.80
dm2	30%	81302	33	15	41:13.93

Table 6: Solutions of the reduced instances of dm examples.

Table ?? summarizes our results. A reduction of 36.85% in example dm1 and a reduction of 33% in example dm2 leads to infeasibility, since the total available knapsack capacity is less than the sum of the weights of the items. Column 2 shows the amount of reduction, columns 3 to 6 present the value of the optimal solution, the number of individual and joint inequalities found by our separation algorithms and the total CPU Time (min:sec). All problem instances in Table ?? are solved to optimality without branching.

Two problems could not be solved to optimality by our algorithm even when branching was applied. Table ?? shows the results for these two examples. The gap we obtain after running our algorithm 500 minutes of CPU time is presented in the last column. We see that the lower and upper bound are very close.

Let us now turn to the examples arising in the layout of electronic circuits. Table ?? summarizes the data. (Here, the instances are abbreviated by the symbol cl in the tables.) The ratio between the total weight of the items and the total available knapsack capacity is much closer to one than it is the case for the

Problem	Red.	Lower bound	Upper bound	Gap (%)
dm2	31%	81482	81498	0.0196
dm2	32%	81728	81736	0.0097

Table 7: Difficult dm instances.

instances in Table **??**. One might expect that these instances are more difficult than the examples dm with 0% area reduction.

Problem	$\|N\|$	$\|M\|$	$\sum_{i\in N} f_i$	$\sum_{k\in M} F_k$
cl1	2292	16	9522	10000
cl2	681	16	2571	2704
cl3	2669	16	6762	7104
cl4	1021	16	4031	4240
cl5	68	16	260	288
cl6	6112	16	25392	26672

Table 8: Description of the examples cl.

However, the results are very similar. The first lower bound provides already the value of the optimal solution in almost all examples. A possible explanation for this fact is that in these examples the weights of the items are similar and that there are many items with small weights and identical objective function value for all knapsacks. In all cases for which the value of the first LP is already equal to the value of the optimal solution our primal heuristics find an optimal solution in the first iteration of the algorithm. The only nontrivial example is cl2, where indeed individual and joint inequalities were necessary to find the optimal solution. In Table **??** we show the value of an optimal solution, the number of individual and joint inequalities found by our separation algorithms and the total CPU Time (min:sec).

Here, as in the design of main frame computers an interesting problem is to determine the minimum area for which the problems still have a feasible solution. We created some problem instances by reducing the total amount of the capacities for the knapsacks until the total weight of the items exceeds the total available knapsack capacity. Even here, we solve all reduced examples, except the reduced instances of cl2, in the first iteration, a very astonishing fact. The reduced instances of cl2 are much more difficult. In all but one example the first lower bound does not give the optimal objective function value. To solve these

Problem	Opt. Solution	Ind. ineq.	Joint ineq.	CPU Time
cl1	2292	0	0	3:56.30
cl2	939.99	145	60	23:52.40
cl3	2669	0	0	3:25.18
cl4	1021	0	0	4:16.65
cl5	472	0	0	1.80
cl6	6112	0	0	27:03.75

Table 9: Solutions of the examples cl.

problems to optimality not only individual inequalities but also joint inequalities were necessary. Table **??** presents the results. (Note that a reduction of 5 % leads to infeasibility, since in this case the total sum of the knapsack capacities is less than the total weight of the items.)

Problem	Red.	Opt. Sol.	Ind. ineq.	Joint ineq.	CPU Time
cl2	1%	946.99	318	143	21:12.10
cl2	2%	946.99	126	135	22:12.25
cl2	3%	960.99	162	146	22:56.90
cl2	4%	967.99	196	185	16:06.03

Table 10: Solutions of the reduced instances of cl2.

An interesting peculiarity of all practical problem instances is that the first lower bound is quite close to the value of the optimal solution. The reason for this is that, for many items $i \in N$, the objective function coefficients c_{ik}, $k \in M$, are similar. However, the gap between the first lower bound and the upper bound after the first iteration is larger by far. Only when individual and especially when joint inequalities are added the linear programming solution provides structural information such that the primal heuristics find good upper bounds or even an optimal solution. This fact can also be observed by running random examples.

Table **??** provides typical results for random problems. The first two columns give the number of items and the number of knapsacks. The weights of the items are randomly chosen from the interval $[5, 300]$, the objective function coefficients are random numbers in the interval $[1, 1000]$, and the knapsack capacities are randomly computed such that $\sum_{k \in M} F_k \leq \alpha \sum_{i \in N} f_i$, where α is a random number chosen from $[1.05, 1.3]$. We have created four different problems with the same

$\|N\|$	$\|M\|$	Average gap after first LP	Average gap after ind. ineq.		Average gap after joint ineq.	
50	4	561.0	231.4	(41.2%)	36.6	(6.5%)
100	4	265.4	164.0	(61.8%)	78.6	(29.6%)
150	4	355.2	149.2	(42.0%)	114.2	(32.3%)
200	4	335.4	117.0	(34.9%)	75.2	(22.4%)
300	4	332.0	112.4	(33.9%)	100.6	(30.3%)
400	4	210.5	154.8	(73.5%)	90.0	(42.8%)
500	4	171.0	41.5	(24.3%)	38.2	(22.4%)

Table 11: Random examples.

number of knapsacks and items. The numbers in column 3 to 5 show the average over the absolute values of the gaps between the upper bounds and the lower bounds after the first iteration, after no further violated individual inequalities were found, and after no further violated joint inequalities were found. The number in brackets give the percentual improvement of the gap. Although we cannot solve most of these random examples to optimality without branching, the results confirm that the gap between the lower and an upper bound is substantially decreased by using individual and joint inequalities.

6 Conclusions

In this paper we have developed a cutting plane based algorithm for the multiple knapsack problem. In particular, we have discussed the separation problem for several classes of valid and facet-defining inequalities. Our algorithm was tested on problem instances arising in the design of main frame computers and in the layout of electronic circuits. Almost all practical examples are solved to optimality without branching. These results confirm the practical use of the inequalities presented in [**?**] and indicate that the separation procedures perform quite well. This impression is also supported by applying our cutting plane algorithm to random examples.

References

[B75] E. Balas, "Facets of the Knapsack Polytope", *Mathematical Program-*

ming 8, 146 - 164 (1975).

[Bo91] E. A. Boyd,"A Pseudo-Polynomial Network Flow Formulation for Exact Knapsack Separation", *Networks* 22, 503 - 514 (1992).

[CPLEX92] *Using the CPLEX Callable Library and CPLEX Mixed Integer Library*, CPLEX Optimization (1992).

[CJP83] H. Crowder, E. L. Johnson and M. W. Padberg, "Solving Large-Scale Zero-One Linear Programming Problems", *Operations Research* 31, 803 - 844 (1983).

[FGKKMW93] C. E. Ferreira, M. Grötschel, S. Kiefl, C. Krispenz, A. Martin and R. Weismantel, "Some Integer Programs Arising in the Design of Main Frame Computers", appears in *ZOR* 37 (1993).

[FM82] C. M. Fiduccia, R. M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions", *Proc. 19th. DAC*, 175 - 181 (1982).

[FMW93] C. E. Ferreira, A. Martin and R. Weismantel, "Facets for the Multiple Knapsack Polytope", Konrad-Zuse-Zentrum für Informationstechnik Berlin *Preprint* SC 93-04 (1993).

[GR90] E. S. Gottlieb e M. R. Rao, "The Generalized Assignment Problem: Valid Inequalities and Facets", *Mathematical Programming* 46, 31 - 52 (1990).

[HJP75] P. L. Hammer, E. L. Johnson and U. N. Peled, "Facets of Regular 0-1 Polytopes", *Mathematical Programming* 8, 179 - 206 (1975).

[IK75] O. H. Ibarra and C. E. Kim, "Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems", *Journal of ACM* 22, 463 - 468 (1975).

[K72] R. M. Karp, "Reducibility among Combinatorial Problems", in R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer computations*, Plenum Press, New York, 85 - 103 (1972).

[MT91] S. Martello and P. Toth, *Knapsack Problems - Algorithms and Computer Implementations*, John Wiley and Sons, New York (1991).

[P75] M. W. Padberg, "A Note on 0-1 Programming", *Operations Research* 23, 833 - 837 (1975).

[P80] M. W. Padberg, "(1,k)-Configurations and Facets for Packing Problems", *Mathematical Programming* 18, 94 - 99 (1980).

[W75] L. A. Wolsey, "Faces of Linear Inequalities in 0-1 Variables", *Mathematical Programming* 8, 165 - 178 (1975).

[W90] L. A. Wolsey, "Valid Inequalities for 0-1 Knapsacks and MIPS with Generalised Upper Bound Constraints", *Discrete Applied Mathematics* 29, 251 - 261 (1990).

[Z86] E. Zemel, "On the Computational Complexity of Facets for the Knapsack Problem", *Working Paper* 713, The Center for Mathematical Studies in Economics and Management Sciences, Northwestern University (1986).