Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7, D-14195 Berlin-Dahlem, Germany

Ralf Borndörfer Carlos E. Ferreira Alexander Martin

Matrix Decomposition by Branch-and-Cut

Preprint SC-97-14 (March 1997)

Matrix Decomposition by Branch-and-Cut*

Ralf Borndörfer** Carlos E. Ferreira¶ Alexander Martin**

Abstract. In this paper we investigate whether matrices arising from linear or integer programming problems can be decomposed into so-called *bordered block diagonal form.* More precisely, given some matrix A, we try to assign as many rows as possible to some number β of blocks of size κ such that no two rows assigned to different blocks intersect in a common column. Bordered block diagonal form is desirable because it can guide and speed up the solution process for linear and integer programming problems. We show that various matrices from the `Miplib` can indeed be decomposed into this form by computing optimal decompositions or decompositions with proven quality. These computations are done with a branch-and-cut algorithm based on polyhedral investigations of the matrix decomposition problem.

Keywords. Block Structure of a Sparse Matrix, Matrix Decomposition, Integer Programming, Polyhedral Combinatorics, Cutting Planes.

Mathematics Subject Classification (MSC 1991). 90C10, 65F50

1 Introduction

We consider in this paper the following *matrix decomposition problem.* Given some matrix A, some number β of *blocks* (sets of rows), and some *capacity* κ (maximum block-size); try to assign as many rows as possible to the blocks such that (i) each row is assigned to at most one block, (ii) each block contains at most κ rows, and (iii) no two rows in different blocks have a common non-zero entry in a column. The set of rows that are not assigned to any block is called the *border.* An equivalent statement of the problem in matrix terminology is as follows: Decompose the matrix into *bordered block diagonal form* with β blocks of capacity at most κ, such that the border is of minimal. Figure 1 shows the structure of a 55×55 non-symmetric matrix (left) and an optimal decomposition into four blocks of capacity $\lceil 55/4 \rceil = 14$ (right). (To make the block structure of the decomposition visible, the rows and columns were permuted.) The border consists of one row that could not be assigned to any block.

The matrix decomposition problem fits into the context of reordering matrices to *special forms.* Special forms and *methods* to obtain them are well studied in the literature on computational linear algebra: They can be exploited by solution methods for linear equation systems, see, e.g., Duff, Erisman, and Reid [1986] or Kumar, Grama, Gupta, and Karypis [1994]. The matrices considered there mainly arise from the discretization of partial differential equations, and, recently, from interior point algorithms for linear programs, see Gupta [1996] and Rothberg and Hendrickson [1996]. But all these matrices are symmetric and —to the best of our knowledge— the proposed ordering algorithms only apply to symmetric (or almost symmetric) matrices.

*This work has been supported jointly by the Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and the German Academic Exchange Service (DAAD), program PROBRAL. Responsibility for the contents lies with the authors.

**Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustraße 7, 14195 Berlin, Germany, Fax +49 30/84185-269, Email `[borndoerfer,martin]@zib.de`

¶Universidade de São Paulo, Rua do Matão, 1010, 05508-970 — São Paulo, SP, Brazil, Fax +55 11/814-4135, Email `cef@ime.usp.br`

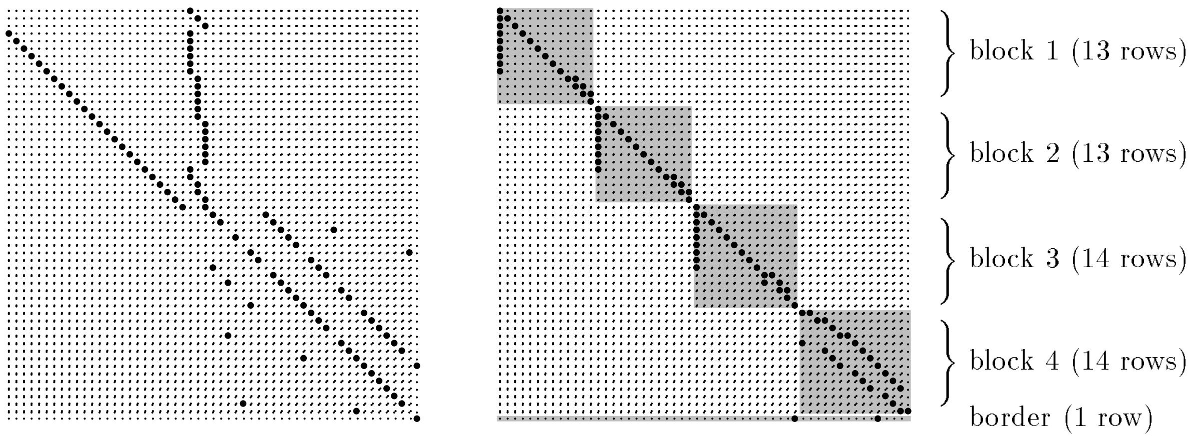

Figure 1: Decomposing a matrix into bordered block diagonal form.

We are interested in matrices arising from *(mixed) integer programs* (MIPs). Such matrices are not symmetric. Bordered block diagonal form can help to solve MIPs in several ways: First, to speed up the linear algebra required to solve the LP-relaxation. Second, to improve the polyhedral description of the set of feasible points of a MIP: Given a block decomposition and taking one constraint from each block plus an according number from the border results in the structure of a generalized assignment or multiple knapsack problem (see Gottlieb and Rao [1990] and Ferreira, Martin, and Weismantel [1996]) whose facets are valid inequalities for the MIP under consideration.

In this paper we develop a branch-and-cut algorithm for the matrix decomposition problem. This gives us a tool at hand that in principle obtains an *optimal* bordered block diagonal form to evaluate whether this special matrix structure has potential to help in solving MIPs.

2 Integer Programming Formulation and Related Problems

Consider an instance (A, β, κ) of the matrix decomposition problem where $A \in \mathbb{R}^{m \times n}$ is some real matrix, $\beta \in \mathbb{N}$ is the number of blocks and $\kappa \in \mathbb{N}$ is the block capacity. We introduce for each row $i = 1, \ldots, m$ and block $b = 1, \ldots, \beta$ a binary variable x_i^b that has value 1 if row i is assigned to block b and 0 otherwise. Then the matrix decomposition problem (A, β, κ) can be stated as the following 0/1 linear program:

$$
\text{(IP)} \qquad
\begin{array}{lll}
\max & \displaystyle\sum_{i=1}^{m}\sum_{b=1}^{\beta} x_i^b & \\
\text{(i)} & \displaystyle\sum_{b=1}^{\beta} x_i^b \le 1, & \text{for } i = 1, \ldots, m; \\
\text{(ii)} & \displaystyle\sum_{i=1}^{m} x_i^b \le \kappa, & \text{for } b = 1, \ldots, \beta; \\
\text{(iii)} & x_i^b + x_j^{b'} \le 1, & \text{for } b, b' = 1, \ldots, \beta,\ b \ne b' \text{ and} \\
 & & \text{for } i, j = 1, \ldots, m,\ i \ne j \text{ such that} \\
 & & a_{ik} \ne 0 \ne a_{jk} \text{ for some } k \in \{1, \ldots, n\}; \\
\text{(iv)} & 0 \le x_i^b \le 1, & \text{for } i = 1, \ldots, m,\ b = 1, \ldots, \beta; \\
\text{(v)} & x_i^b \text{ integer}, & \text{for } i = 1, \ldots, m,\ b = 1, \ldots, \beta.
\end{array}
$$

The feasible solutions of (IP) correspond exactly to the block decompositions of the matrix A into β blocks of capacity κ and we will thus not distinguish them. We may assume without loss of generality $\beta \leq m$, because no more than m rows will be assigned, that the block capacity is at least one ($\kappa \geq 1$), and that we have at least two blocks ($\beta \geq 2$).

Different matrices A can give rise to the same integer program. This can be seen by considering the *(column) intersection graph* $G(A)$ of an $m \times n$-matrix A as introduced by Padberg [1973]. $G(A)$'s node set is $\{1, \ldots, n\}$ and there is an edge ij if columns i and j of A have a common non-zero entry in some row. Applying this concept to the transposed matrix A^T, we obtain the *row intersection graph* $G(A^T)$ of A. The edges of $G(A^T)$ give rise to the inequalities (IP) (iii) and we have that for fixed β and κ two matrices A and A' have the same row intersection graph if and only if the corresponding integer programs (IP) are equal.

The matrix decomposition problem is related to the *set packing* problem in two ways. First, matrix decomposition is a generalization of set packing, because feasible solutions (stable sets) of some set packing problem $\max\{\mathbb{1}^T x \mid Ax \leq \mathbb{1}, x \in \{0,1\}^n\}$, $A \in \{0,1\}^{m \times n}$, correspond to solutions of the matrix decomposition problem $(A^T, m, 1)$ of the same objective value and vice versa. This shows that the matrix decomposition problem is $\mathcal{NP}$-hard. Second, we obtain a *set packing relaxation* of the matrix decomposition problem by deleting the block capacity constraints (ii) from the formulation (IP). This means that all inequalities that are valid for this relaxation are also valid for the matrix decomposition problem. There is also a connection to *set covering* via complementing variables. For the case of two blocks, the corresponding formulation has been used by Nicoloso and Nobili [1992] for the solution of the *matrix equipartition problem.* The matrix equipartition problem is the matrix decomposition problem for $\beta = 2$ and $\kappa = \lfloor m/2 \rfloor$ with the additional constraint that the two blocks of the decomposition must have equal size.

3 Polyhedral Investigations

Associated to the IP-formulation (IP) of the matrix decomposition problem is the polytope

$$P(A, \beta, \kappa) := \text{conv}\,\{x \in \mathbb{R}^{m \times \beta} \mid x \text{ satisfies (IP) (i) to (v)}\}, \tag{1}$$

given by the convex hull of all block decompositions. We study in this section the structure of $P(A, \beta, \kappa)$ to derive classes of valid and facet defining inequalities for later use as cutting planes.

3.1 Remark

(a) *$P(A, \beta, \kappa)$ is full dimensional.*

(b) *The non-negativity inequalities $x_i^b \geq 0$ are facet defining for all $i = 1, \ldots, m$ and all $b = 1, \ldots, \beta$.*

(c) *All facet defining inequalities $a^T x \leq \alpha$ that are not non-negativity constraints satisfy $a \geq 0$ and $\alpha > 0$.*

Facet-defining inequalities have an interesting property. Consider some vector $x \in \mathbb{R}^{m \times \beta}$, some *permutation σ of the blocks* $\{1, \ldots, \beta\}$, and define the vector $\bar{x} \in \mathbb{R}^{m \times \beta}$ by $\bar{x}_i^b := x_i^{\sigma(b)}$, for $i = 1, \ldots, m, b = 1, \ldots, \beta$. We will use in the sequel the symbol $\sigma(x)$ to denote the vector $\bar{x}$ that arises from x by applying the block permutation σ. Then $\sigma(x) = \bar{x}$ is a feasible block decomposition if and only if x is. This means that the matrix decomposition problem is dual degenerate (has multiple optima). It also implies that $a^T x \leq b$ is a facet of $P(A, \beta, \kappa)$ if and only if its block-wise permutation $\sigma(a)^T x \leq b$ is. Dual degeneracy and the large number of permutable facets cause difficulties in our branch-and-cut algorithm and we will have to control the number of cuts generated and to handle stalling of the objective value.

The next two subsections list the results of our polyhedral investigations. We distinguish between inequalities $a^T x \leq b$ that are *invariant under block permutations* and have the same coefficients $a_i^b = a_i^{b'}$ for all blocks $b \neq b'$ and rows i, and *block-discernible inequalities* that distinguish different blocks. Most block-discernible inequalities will be inherited from the stable set relaxation of the matrix decomposition problem, while the block-invariant constraints are related to an "aggregated" version of the problem. In both subsections we want to assume $\kappa \geq 2$, because otherwise the matrix decomposition problem is a (special) set packing problem.

3.1 Block-Discernible Inequalities

We saw in Section 2 that we obtain a set packing relaxation of the matrix decomposition problem by dropping the block capacity constraints (ii) from the integer program (IP). The column intersection graph associated to the matrix $\mathrm{IP}_{(\mathrm{i}),(\mathrm{iii})}$ formed by the left-hand sides of the constraints (IP) (i) and (iii) has the set of possible row assignments $\{1, \ldots, m\} \times \{1, \ldots, \beta\}$ as its node set. A (conflict) edge exists between two assignments (i, b) and (j, b'), if rows i and j cannot be simultaneously assigned to the blocks b and b', i. e., either if $i = j$ and $b \neq b'$ or if $i \neq j$, $b \neq b'$, and rows i and j have a common non-zero entry in some column of A. We want to call this graph the *conflict graph* associated to the matrix decomposition problem (A, β, κ) and denote it by $G_c(A, \beta)$. In formulas: $G_c(A, \beta) = G(\mathrm{IP}_{(\mathrm{i}),(\mathrm{iii})})$. This graph allows us to interpret the inequality classes (i) and (iii) of (IP) as *clique inequalities* of the set packing relaxation corresponding to the matrix decomposition problem as also introduced by Padberg [1973].

3.2 Theorem (Clique) *Let $G_c(A, \beta) = (V, E)$ and $Q \subseteq V$. The inequality $\sum_{(i,b) \in Q} x_i^b \leq 1$ is valid for $P(A, \beta, \kappa)$ if and only if Q is a clique in $G_c(A, \beta)$. It is facet defining if and only if Q is a maximal clique in $G_c(A, \beta)$.*

The separation problem for clique inequalities is a maximum-weight clique problem and thus $\mathcal{NP}$-hard, see Garey and Johnson [1979]. But the subclass of *two-partition inequalities*

$$\sum_{b \in B} x_i^b + \sum_{b' \notin B} x_j^{b'} \leq 1,$$

defined for all sets of blocks $B \subseteq \{1, \ldots, \beta\}$ and all pairs of rows i, j, can be separated efficiently.

Along the same lines, the matrix decomposition polytope also inherits all other inequalities of its set packing relaxation, e.g., the *cycle inequalities*, see again Padberg [1973], and of the corresponding set covering problem obtained by complementing variables. Not inherited from these related polytopes are the *block capacity constraints*.

3.3 Theorem (Block Capacity) *The block capacity constraint*

$$\sum_{i=1}^{m} x_i^b \leq \kappa$$

is facet defining for $P(A, \beta, \kappa)$ if and only if $|\gamma(i)| \leq m - \kappa$ holds for every row i (where $\gamma(i)$ denotes all nodes adjacent to i in $G(A^T)$).

3.2 Block-Invariant Inequalities

Consider for each block decomposition x the "aggregated" vector

$$z(x) := \left(\sum_{b=1}^{\beta} x_1^b, \ldots, \sum_{b=1}^{\beta} x_m^b \right) \in \mathbb{R}^m.$$

$z(x)$ only records whether the matrix rows are assigned to some block or not, but no longer to which block. From a polyhedral point of view, the aggregated block decompositions give rise to an "aggregated" version of the block decomposition polytope

$$P_z(A, \beta, \kappa) := \text{conv}\, \{z \in \mathbb{R}^m : \text{ there is } x \in P(A, \beta, \kappa) \text{ with } z = z(x)\}.$$

The aggregated polytope is interesting because any valid inequality $\sum_{i=1}^m a_i z_i \leq \alpha$ for $P_z(A, \beta, \kappa)$ can be "expanded" into an inequality $\sum_{i=1}^m a_i \sum_{b=1}^\beta x_i^b \leq \alpha$ that is valid for $P(A, \beta, \kappa)$. All inequalities in this subsection are of this type. Obviously, the expansion process yields inequalities that are invariant under block permutation. A first example are the *z-cover inequalities.*

3.4 Theorem (z-Cover) *Let $G(A^T) = (V, E)$ and let $W \subseteq V$ be a set of rows of cardinality $\kappa + 1$. Then, the z-cover inequality*

$$\sum_{i \in W} \sum_{b=1}^{\beta} x_i^b \leq \kappa$$

is valid for $P(A, \beta, \kappa)$ if and only if $(W, E(W))$ is connected (where $E(W)$ denotes all edges with both endpoints in W). It is facet defining for $P(A, \beta, \kappa)$ if and only if for each row $i \notin W$ the graph $(W \cup \{i\}, E(W \cup \{i\}))$ has an articulation point different from i.

The separation problem to find a tree of size $\kappa + 1$ of maximum node weight is $\mathcal{NP}$-hard, see Ehrgott [1992]. The z-cover inequalities are induced by trees and one can generalize them for subgraphs of higher connectivity, but we skip this possibility here and just mention the related *z-clique inequalities.*

3.5 Theorem (z-Clique) *If Q is a clique in $G(A^T)$, then the z-clique inequality*

$$\sum_{i \in Q} \sum_{b=1}^{\beta} x_i^b \leq \kappa$$

is valid for $P(A, \beta, \kappa)$. It is facet-defining if and only if $|Q| \geq \kappa + 1$ and for each row $i \notin Q$ there exists a set of rows $R(i) \subseteq Q$, $|R(i)| = \kappa$, such that i is not adjacent in $G(A^T)$ to any node in $R(i)$.

The z-clique separation problem is again a max-clique problem and thus $\mathcal{NP}$-hard. In our implementation we check easily detectable special cases like the following so-called *big-edge inequalities*

$$\sum_{i \in \text{supp}(A_{\cdot j})} \sum_{b=1}^{\beta} x_i^b \leq \kappa,$$

for all blocks b, where $A_{\cdot j}$ denotes the j-th column of A and $\text{supp}(A_{\cdot j})$ its non-zero row indices.

Another way to generalize the z-cover inequalities is by looking at node induced subgraphs that consist of several components. This idea gives rise to the class of *bin-packing inequalities.* Consider a set of rows W that induces a subgraph of $G(A^T) = (V, E)$. Suppose $(W, E(W))$ consists of l connected components of sizes (in terms of nodes) $a_1, \ldots, a_l$. We can then associate a *bin-packing problem* with $(W, E(W))$, β, and κ in the following way: there are l items of sizes $a_1, \ldots, a_l$, and β bins of capacity κ each. The problem is to put all the items into the bins such that no bin holds items of a total size that exceeds the capacity κ. If this is not possible, we can derive a valid inequality for $P(A, \beta, \kappa)$.

3.6 Theorem *Let $G(A^T) = (V, E)$ and $W \subseteq V$ be some subset of rows. If the bin packing problem associated to $(W, E(W))$, β, and κ has no solution, the bin packing inequality*

$$\sum_{i \in W} \sum_{b=1}^{\beta} x_i^b \leq |W| - 1$$

is valid for $P(A, \beta, \kappa)$.

We do not know any reasonable conditions that characterize when the bin packing inequalities are facet defining. Bin-packing separation is $\mathcal{NP}$-hard, see Garey and Johnson [1979]. The last *star inequality* that we present in this section is special in the sense that it is the only one with non-0/1 coefficients. It was designed to deal with linking rows with many neighbors.

3.7 Theorem (Star) *Let $G(A^T) = (V, E)$ and consider some row $i \in V$ with $|\gamma(i)| > \kappa$. Then the star inequality*

$$(|\gamma(i)| - \kappa + 1) \sum_{b=1}^{\beta} x_i^b + \sum_{j \in \gamma(i)} \sum_{b=1}^{\beta} x_j^b \leq |\gamma(i)|$$

is valid for $P(A, \beta, \kappa)$.

4 A Branch-And-Cut Algorithm

The polyhedral investigations of the last section form the basis for the implementation of a branch-and-cut algorithm for the solution of the matrix decomposition problem. This section describes two of the problem specific ingredients of this code: A non-standard anti-stalling mechanism and our primal matrix decomposition heuristics. For the sake of brevity we can not describe or separation algorithms, LP-management, problem reduction, and searchtree management here, but it goes without saying that a careful implementation of each of these components is crucial for the overall performance of the algorithm.

4.1 Tie-Breaking Inequalities

In addition to all the classical types of cutting planes that we described in Section 3 we use a number of *"tie-breaking" inequalities* to cut off decompositions that are identical up to block permutations or give rise to multiple optima for other reasons as a means to counter dual degeneracy and stalling. These inequalities are in general not valid for $P(A, \beta, \kappa)$, but for at least one optimal solution. The most simple kind of these cuts are the *permutation inequalities*

$$\sum_{i=1}^{m} x_i^b \leq \sum_{i=1}^{m} x_i^{b+1}, \quad b = 1, \ldots, \beta - 1,$$

stating that blocks with higher indices are of larger size. Another idea that we use to eliminate multiple optima is based on the concept of *row preference*. We say that row i is *preferred* to row j or, in symbols, $i \prec j$ if

$$\gamma(i) \subseteq \gamma(j)$$

with respect to the row intersection graph $G(A^T)$. We may in this situation not know whether or not row i or j can be assigned to a block in some optimal solution, but we can say that for any decomposition x with $z(x)_j = 1$, say $x_j^b = 1$, either $z(x)_i = 1$ or we can get a feasible decomposition $x' = x - e_j^b + e_i^b$ with the same number of rows assigned. In this sense, row i is more attractive than row j. If we break ties on row preference by indices (i.e, $i \prec j \iff \gamma(i) \subsetneq$

$\gamma(j) \vee (\gamma(i) = \gamma(j) \wedge i < j))$, row preferences induce a partial order. We force them in our code by adding all non-transitive *row preference inequalities*

$$\sum_{b=1}^{\beta} x_i^b \geq \sum_{b=1}^{\beta} x_j^b \quad \text{for } (i,j) \text{ with } i \prec j.$$

4.2 Heuristics

Our heuristics fall into three groups: "primal" methods that iteratively fix block assignments, "dual" methods that iteratively exclude assignments, and an improvement method. All of them respect variable fixings at the nodes of the searchtree to increase the probability of finding different solutions. Applied at the root node where (at least initially) no variables are fixed, our methods can be seen as LP-based or pure combinatorial heuristics for the matrix decomposition problem. The heuristics are called after each individual LP.

The *primal methods* consist of a *greedy algorithm* and a *bin-packing heuristic*, both are LP-based. The greedy algorithm orders the x_i^b variables according to increasing x-value. The rows are assigned greedily to the blocks in this order. The bin-packing heuristic starts by determining a set of nodes W that will be assigned to the blocks and used to set up a corresponding bin-packing problem. In order to find a better decomposition than the currently best known with, say, $z^\star$ rows assigned, W should be of cardinality at least $z^\star + 1$ and therefore we take the $z^\star + 1$ rows with the largest $z(x)$-values to be the members of W. The corresponding bin-packing problem is set up and solved with a dynamic program with a time bound.

The *dual methods* also respect variable fixings, but are not LP-based. The idea behind them is not to assign rows to blocks, but to iteratively eliminate assignments of "bad" rows. Suppose that a decision was made to assign certain rows (assigned rows) to certain blocks, to exclude other rows from assignment (unassigned rows), while for the remaining rows a decision has yet to be made (free rows). Removing the unassigned nodes from the row intersection graph $G(A^T)$ leaves us with a number of connected components. Both variants of the dual method will break up the components that are larger than the block capacity κ by unassigning free rows until no more such components exist. At this point, a simple first-fit decreasing heuristic is called to solve the corresponding bin-packing problem. The two variants differ in the choice of the next bad row to remove. Variant I chooses the free row with the largest degree with respect to $G(A^T)$, variant II the least preferable free row.

Our *improvement heuristic* is a variation of a local search technique presented by Fiduccia and Mattheyses [1982]. Given some block decomposition, it performs a sequence of local exchange steps each of the following type. Some assigned row is chosen to be made unassigned opening up possibilities to assign its unassigned neighbors. These assignments are checked and feasible assignments are executed.

5 Computational Results

In this section we report on computational experiences with our branch-and-cut algorithm and examine whether matrices arising from *mixed integer programs* can be decomposed into (bordered) block diagonal form. Our aim is to find answers to two complexes of *questions*. First, we would like to evaluate our branch-and-cut approach: What are the limits in terms of the size of the matrices that we can solve with our algorithm? What is the quality of the cuts, do they provide a reasonable solution guarantee? Second, we want to discuss our concept of decomposition into bordered block diagonal form. Do the test instances have this structure or are most integer programming matrices not decomposable in this way?

The *algorithm* is implemented in `C` and consists of about 36,000 lines of code. The test runs were performed on a Sun Ultra Sparc 1 Model 170E and we used a time limit of $1,800$ CPU seconds. The LPs were solved with the `CPLEX 4.0` dual simplex algorithm using steepest edge pricing, see CPLEX [1995].

Our *test set*[1] consists of matrices of mixed integer programs taken from the `Miplib`[2] and preprocessed with the presolver of the general purpose MIP-solver `SIP` that is currently under development at the Konrad-Zuse-Zentrum. There are two applications for decompositions of mixed integer programming matrices. First, to speed up the linear algebra in a simplex-type LP-solver for use within a branch-and-cut algorithm for general MIPs. Second, a decomposition of the constraint matrix of a MIP can be useful to *tighten* its LP-relaxations within such an algorithm. The structure of the decomposed matrix is that of a multiple knapsack or general assignment problem, and inequalities known for the associated polytopes (see Gottlieb and Rao [1990], Ferreira, Martin, and Weismantel [1996]) are valid for the MIP under consideration. The first interesting case in this context are two blocks and we set $\beta := 2$. We used $\kappa := \frac{(\#rows) \cdot 1.05}{2}$ rounded up as the block capacity, which allows a deviation of 10% of the actual block sizes in the decomposition.

Table 1 reports the results of our computational experiments for all instances with up to 400 rows. The format is as follows: Column 1 provides the name of the problem, Columns 2 to 4 contain the number of rows, columns and non-zeros of the matrix to be decomposed. The next 5 columns give the number of cuts found by the algorithm: Initial cuts (*Init*), i.e., block assignment and capacity, big-edge, star, and tie-breaking cuts, z-cover (*Cov*), two-partition (*2part*), the sum of the number of bin-packing, cycle, clique, and z-clique cuts (*BCC*), and the cuts separated from the pool (*pool*). Column *Nod* gives the number of branch-and-bound nodes, *Iter* the number of LPs. The next four columns give solution values reported in terms of the number of rows in the border, because it is easier to see whether the matrix could be decomposed into block diagonal form: *Lb* gives the global lower bound, *Ub* the value of the best solution found by heuristic *He* (*G*, *D1*, *D2* and *B* stand for the *greedy*, *dual (variant I and II),* and *bin-packing* heuristic, *I* for a succeeding call to the *improvement heuristic*, $*$ for an integral LP solution) after *No* many nodes (1 means it was found in the root node, 0 means that preprocessing solved the problem). The remaining columns show timings: column *Tot* gives the total running time in CPU seconds, *Cm* the percentage of the total time spent in LP-management, *LP* the same for the time spent in the LP-solver, *Sep* for separation, and *Heu* for heuristics.

The *results* are as follows. The problems up to 90 rows are easy. The range of 90–200 rows is most interesting: The problems here are already difficult, but because of the combinatorial structure and not because of sheer size. The problems with more than 200 rows are large-scale. The algorithm solves very few LPs within the given time limit and we can decompose only a couple of easy large instances, but it is worth noticing that a significant number of these exists: The difficulty of matrix decomposition problems depends as much on the structure of the matrix as on the number of rows, columns, or non-zeros.

Let us first investigate our branch-and-cut approach. We observe that we solve only few problems at the root node (only 12 out of 51), and that the number of cuts is very large. The reason for this is basically the symmetry of the problem, as can be seen from the pool separation column (*Pool*) that counts, in particular, all violated tie-breaking cuts. Since $\beta = 2$ leads to large block capacities, it is difficult to separate inequalities that have a combinatorially restrictive support of this size, see column *BCC*. The quality of the cuts is in our opinion reasonable, see columns *Nod* and *Iter*. It is true, however, that the lower bound improves fast at first while stalling occurs in later stages of the computation although still large numbers of cuts are found and the problem is finished by branch-and-bound.

[1] Available at URL `ftp://ftp.zib.de/pub/mp-testdata/madlib/index.html`

[2] Available at URL `http://www.caam.rice.edu:80/~bixby/miplib/miplib.html`

Name	Sizes			Cuts					B&B		Best Solutions				Time				
	rows	col	nz	Init	Cov	2part	BCC	Pool	Nod	Iter	Lb	Ub	He	No	Cm	LP	Sep	Heu	Tot
mod008	6	319	1243	23	0	0	0	0	1	1	3	**3**	IG	1	0%	6%	6%	0%	0.1
stein9_r	13	9	45	50	115	64	1	23	16	34	7	**7**	IG	1	6%	34%	24%	6%	0.3
p0040	13	40	70	16	0	0	0	0	1	1	3	**3**	*	1	0%	0%	40%	0%	0.1
p0033	15	32	97	41	23	3	0	1	1	3	4	**4**	D1	1	0%	50%	16%	0%	0.1
gt1	15	46	92	19	262	87	0	61	13	33	6	**6**	D1	1	4%	47%	26%	4%	0.4
flugpl	16	16	40	46	16	0	1	0	1	2	1	**1**	ID2	1	0%	25%	0%	0%	0.0
bm23	20	27	478	79	0	0	0	0	1	1	10	**10**	IG	1	0%	22%	0%	11%	0.1
enigma_r	21	100	289	40	922	216	0	315	28	83	10	**10**	IG	1	7%	41%	28%	14%	2.0
air01	23	771	4215	88	0	0	0	0	1	1	3	**3**	IG	1	0%	7%	0%	0%	0.4
rgn_r	24	180	460	64	724	163	1	324	13	47	5	**5**	IG	1	12%	48%	18%	10%	1.2
pipex	25	48	192	46	421	180	0	201	13	33	9	**9**	IG	1	12%	42%	23%	7%	0.8
lseu	28	88	308	68	98	43	0	19	1	5	7	**7**	D1	1	8%	30%	43%	4%	0.2
gt2	28	173	346	39	1333	340	0	290	28	86	11	**11**	IG	1	8%	46%	28%	8%	2.8
sentoy_r	30	60	1800	119	0	0	0	0	1	1	15	**15**	IG	1	0%	18%	12%	0%	0.2
stein15_r	36	15	120	142	17654	13145	0	26682	5458	7595	18	**18**	IG	1	18%	27%	22%	16%	90.1
misc02	43	55	405	153	2455	1065	0	1612	34	110	15	**15**	IG	9	6%	59%	18%	11%	9.2
sample2_r	45	64	140	81	976	171	1	331	7	29	4	**4**	D1	1	6%	61%	21%	6%	2.0
air02	50	6774	61555	270	4	0	34	0	1	3	22	**22**	IG	1	0%	6%	18%	0%	6.0
misc01	54	79	729	218	4556	2762	0	4861	367	581	24	**24**	IG	17	7%	42%	29%	14%	31.4
mod013	62	96	192	254	1810	108	1	722	13	43	6	**6**	D1	1	6%	68%	17%	3%	5.1
mod014	74	86	172	246	148	0	1	9	1	3	2	**2**	D1	1	5%	32%	35%	8%	0.3
lp4l	85	1086	4677	277	23233	16941	0	45877	1129	1779	35	**35**	IG	487	4%	13%	27%	52%	481.2
bell5	87	101	257	171	2418	473	1	800	13	38	4	**4**	ID1	4	5%	67%	15%	9%	10.6
p0291	92	103	373	365	2943	366	0	757	19	54	7	**7**	D1	1	7%	52%	21%	15%	10.7
misc03	96	154	2023	360	50431	32949	1	85357	12745	15609	43	44	IG	9	5%	11%	39%	38%	1800.0
l152lav	97	1989	9922	308	32271	23459	0	71535	1525	2276	36	**36**	IG	1102	3%	16%	24%	53%	754.5
khb05250	100	1299	2598	196	12634	1917	1	3421	73	227	25	**25**	IG	1	3%	25%	42%	25%	89.0
harp2_r	100	1373	2598	116	250	158	0	3	1	4	17	**17**	D1	1	0%	2%	94%	0%	20.4
bell4	101	114	293	191	7162	1041	1	2157	49	145	5	**5**	IB	4	6%	55%	17%	17%	31.7
bell3a	107	121	311	203	2718	492	1	802	10	32	4	**4**	B	6	5%	63%	16%	12%	14.5
bell3b	107	121	311	203	2718	492	1	802	10	32	4	**4**	B	6	5%	63%	15%	13%	14.6
p0201	113	195	1677	200	9230	2390	1	5757	46	128	21	**21**	IG	24	3%	64%	12%	19%	125.4
stein27_r	118	27	378	353	280101	82635	3	382512	1753	3874	32	56	IG	1	12%	44%	34%	6%	1800.1
air03	124	10757	91028	830	21108	9064	0	24479	130	377	49	**49**	IG	13	1%	57%	27%	12%	1146.7
p0808a	136	240	480	392	14298	1463	1	4711	49	175	8	**8**	D1	1	6%	62%	16%	13%	83.2
mod010	146	2655	11203	453	53084	33036	1	118368	727	1170	47	59	IG	28	2%	24%	25%	46%	1802.5
blend2	169	319	1279	515	0	0	0	0	4	4	10	**10**	IG	1	2%	57%	22%	1%	1.7
noswot	182	127	732	745	84957	34285	2	533391	1006	1646	10	15	IG	10	10%	47%	33%	6%	1800.4
10teams	210	1600	9600	210	67719	25710	1	127827	130	418	43	90	D1	1	3%	58%	30%	6%	1804.0
misc07	224	254	8589	894	10476	8516	0	13110	118	119	57	93	IG	11	0%	23%	22%	52%	1836.5
vpm1	234	378	749	906	18959	446	1	4060	34	117	7	**7**	IG	1	3%	80%	9%	5%	322.7
vpm2	234	378	917	906	18959	446	1	4060	34	117	7	**7**	IG	1	3%	79%	9%	6%	317.4
p0808aCUTS	246	240	839	612	15096	1731	1	3058	22	84	8	**8**	D1	1	4%	65%	12%	17%	231.7
p0548	257	477	1522	523	94934	20509	1	94842	175	462	25	49	IG	35	5%	69%	12%	12%	1800.1
misc05	266	131	2873	581	16094	4604	0	5545	43	80	28	116	IG	13	1%	89%	5%	3%	1835.0
modglob	289	420	966	865	75843	1685	1	29153	148	422	8	**8**	IG	18	3%	74%	11%	9%	1512.6
gams	291	556	2431	1622	41190	11	0	6897	169	295	15	21	B	12	1%	84%	9%	3%	1801.1
fiber	297	1232	2644	593	32222	7410	1	19253	31	123	9	22	ID1	3	2%	86%	4%	6%	1807.5
p0282	305	202	1428	609	56425	17488	1	52114	91	252	23	36	D1	1	3%	71%	9%	14%	1803.2
stein45_r	331	45	1034	992	25305	11011	0	22779	28	88	11	154	IG	2	1%	89%	5%	2%	1820.3
qnet1_o	369	1454	4040	699	34759	9401	1	19450	37	111	12	28	D1	1	2%	74%	6%	15%	1812.7
$\sum$	6154	37226	240760	17992	1139054	368476	63	1718328	26349	38953	795	1223		1844	4%	58%	18%	17%	28844.5

Table 1: Decomposing matrices of mixed integer programs.

How decomposable are the MIP-matrices? We see that not all, but many of the larger problems can be brought into bordered block diagonal form (the small ones can not). Of course, there are also exceptions like the `air`-problems which were expected to be not decomposable. Anyway, there seems to be potential for the multiple-knapsack approach and further research in this direction, especially because there are only very few classes of cutting planes known for general MIPs.

Summary and Conclusions

We have shown in this paper that it is possible to decompose typical mixed integer programming matrices up to 200 and more rows to proven optimality using a cutting plane approach based on polyhedral investigations of the matrix decomposition problem. It turned out that many MIPs can be brought into bordered block diagonal form with two blocks. We think that these results show a significant potential for methods that can exploit this structure to solve general MIPs.

References

CPLEX (1995). *Using the CPLEX Callable Library.* CPLEX Optimization, Inc., Suite 279, 930 Tahoe Blvd., Bldg 802, Incline Village, NV 89451, USA. Informations available at URL: http://www.cplex.com/.

Duff, I., Erisman, A., and Reid, J. (1986). *Direct Methods for Sparse Matrices.* Oxford University Press.

Ehrgott, M. (1992). Optimierungsprobleme in Graphen unter Kardinalitätsrestriktionen. Master's thesis, Universität Kaiserslautern, Department of Mathematics.

Ferreira, C. E., Martin, A., and Weismantel, R. (1996). Solving multiple knapsack problems by cutting planes. *SIAM Journal on Optimization*, 6:858 – 877.

Fiduccia, C. and Mattheyses, R. (1982). A linear-time heuristic for improving network partitions. *Proc. 19th. DAC*, pages 175–181.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York.

Gottlieb, E. and Rao, M. (1990). The generalized assignment problem: Valid inequalities and facets. *Mathematical Programming*, 46:31–52.

Gupta, A. (1996). Fast and effective algorithms for graph partitioning and sparse matrix ordering. Technical Report Research Report RC 20496, IBM T. J. Watson Research Center, Yorktown Heights.

Kumar, V., Grama, A., Gupta, A., and Karypis, G. (1994). *Introduction to Parallel Computing.* The Benjamin/Cummings Publishing Company, Inc.

Nicoloso, S. and Nobili, P. (1992). A set covering formulation of the matrix equipartition problem. In Kall, P., editor, *System Modelling and Optimization, Proceedings of the 15th IFIP conference, Zürich, September 1991*, pages 189–198. Springer Verlag, Berlin, Heidelberg, New York.

Padberg, M. (1973). On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215.

Rothberg, E. and Hendrickson, B. (1996). Sparse matrix ordering methods for interior point linear programming. Technical Report Technical Report SAND96-0475J, Sandia National Laboratories.