Report in Wirtschaftsmathematik
Nr. 91/2003

Set covering with almost consecutive ones property

Nikolaus Ruf and Anita Schöbel
University of Kaiserslautern
e-mail: ruf,schoebel@mathematik.uni-kl.de

November 11, 2003

Abstract

In this paper we consider set covering problems with a coefficient matrix *almost* having the consecutive ones property, i.e., in many rows of the coefficient matrix, the ones appear consecutively. If this property holds for all rows it is well known that the set covering problem can be solved efficiently. For our case of almost consecutive ones we present a reformulation exploiting the consecutive ones structure to develop bounds and a branching scheme. Our approach has been tested on real-world data as well as on theoretical problem instances.

1 Introduction

Set covering problems belongs to the best studied combinatorial optimization problems, which becomes evident when reading the annotated bibliography [CNS97] or the excellent survey [CFT00] on state-of-the art algorithms.
Among other reasons the interest in set covering problems is due to their large potential of modeling real-world problems such as scheduling, facility

location, or production optimization problems. Unfortunately, the majority of set covering problems arising in practice are very large. For example, in crew scheduling one easily obtains set covering problems with thousands of variables and constraints as it is reported, e.g., in [CFT$^+$97] for railway and in [MS00] for airline crew scheduling problems. Since the set covering problem is NP-hard ([GJ79]) and also difficult from the point of view of theoretical approximation ([LY94]), such large problem instances are hard to solve. This motivates the development of efficient heuristic procedures for solving large-scale problems, see e.g. the Lagrangian-based heuristic of [CFT99].

However, if the coefficient matrix of the set covering problem has the consecutive ones property, (i.e., the ones in each row appear consecutively) the problem can be easily solved. This can be done, e.g., by linear programming methods which is due to the fact that matrices with consecutive ones property are totally unimodular. More sophisticated methods are discussed in [jVW62, NW88, Sch03].

In this paper we propose a new approach for solving large real-world set covering problems. Namely, many practical applications of set covering problems deal with relatively sparse matrices containing many rows of consecutive ones, if the columns are sorted in a way that is often motivated within the application. One example (for which real-world data were available for testing our approach) is the continuous stop location problem which is described in detail in Section 6. Many other examples for coefficient matrices with almost consecutive ones property appear in various practical applications. This gives rise to develop a procedure for solving set covering problems in which the covering matrix almost has the consecutive ones property.

The remainder of the paper is structured as follows. First, we formally introduce the notion of set covering problems, the consecutive ones property and give a first reformulation which will be the basis of the subsequent approach. In Section 3 we derive lower and upper bounds on the problem. Based on these bounds we develop a branch and bound approach in Section 4. In Section 5 we show how the problem size can be reduced using the efficient data structure of Section 2. Section 6 is devoted to our numerical results, and in Section 7 we interpret these results and present a better definition of *almost having the consecutive ones property*.

2 The almost consecutive ones property and a reformulation

We use the following notation to describe set covering problems.

(SCP)

$$\begin{array}{lrcl} \min & cx & & \\ \text{s.t.} & A^{\text{cov}}x & \geq & \underline{1}_M \\ & x & \in & \{0,1\}^N, \end{array} \tag{1}$$

where $\underline{1}_M \in \mathbb{R}^M$ denotes the vector consisting of M ones, $c \in \mathbb{R}^N$ contains the costs of the columns, and A^{cov} is an $M \times N$-matrix with elements $a_{mj} \in \{0,1\}, m = 1, \ldots, M, j = 1, \ldots, N$. We may assume without loss of generality that A^{cov} neither has zero rows nor zero columns and that the costs c_j are positive.
The goal is to find an optimal solution x^*, or equivalently, an optimal set $\mathcal{N}^* \subseteq \mathcal{N} := \{1, \ldots, N\}$ of columns of A^{cov}, where $\mathcal{N}^* = \{n \in \mathcal{N} : x^*_n = 1\}$.

Definition 1 *A matrix A^{cov} has the* **consecutive ones property (C1P)** *if there exists a permutation of its columns such that all rows $m \in \{1, \ldots, M\}$ of the resulting matrix A satisfy the following condition for all $j_1, j_2 \in \{1, \ldots, N\}$:*

$$a_{mj_1} = 1 \text{ and } a_{mj_2} = 1 \Longrightarrow a_{mj} = 1 \text{ for all } j_1 \leq j \leq j_2.$$

If a matrix has the consecutive ones property, the permutation of the columns making the ones appear consecutively can be found by using the algorithm of [BL76, MT98]. This algorithm can be performed in O(MN). Without loss of generality we can therefore assume that a matrix with consecutive ones property is already ordered, i.e. we assume that its ones already appear consecutively in all of its rows. We say that a set covering problem has C1P if its covering matrix A^{cov} has C1P.

For a matrix A^{cov} (not necessarily having the consecutive ones property) we say that a row $\bar{m}$ of a given matrix A^{cov} has the *consecutive ones property*, if the ones appear consecutively in this row, i.e., if for all $j_1, j_2 \in \{1, \ldots, N\}$:

$$a_{\bar{m}j_1} = 1 \text{ and } a_{\bar{m}j_2} = 1 \Longrightarrow a_{\bar{m}j} = 1 \text{ for all } j_1 \leq j \leq j_2.$$

Let us now assume that in the set covering problem the coefficient matrix A^{cov} almost has the consecutive ones property, i.e., that the ones appear consecutively (possibly after permuting the columns) in many rows of A^{cov}. Since set covering problems in which A^{cov} has the consecutive ones property can be solved efficiently the idea is to decompose each "bad" row in which the ones do not appear consecutively into a set of new rows, all of them satisfying the consecutive ones property, and to require that at least one of these rows needs to be covered. In a first attempt, we define:

Definition 2 *Let A^{cov} be a 0-1-matrix with M rows and N columns.*

1. *If A^{cov}_m is a row of A^{cov} let bl_m be its number of blocks of consecutive ones.*
2. *A^{cov} **almost** has the consecutive ones property, if $\sum_{m=1}^{M} bl_m \ll MN$.*

We remark that the condition of the above definition will turn out to be necessary to ensure an efficient behavior of our solution approach, but still there remain instances that cannot be solved in reasonable time by our approach although satisfying the *almost consecutive ones property*. Another criterion to classify well-solvable matrices will be made precise at the end of this paper.

Now consider a zero-one matrix A^{cov} with M rows, such that in rows $1, \ldots, p$ the ones appear consecutively (i.e., $bl_m = 1$ for $m = 1, \ldots, p$), and in rows $p+1, \ldots, M$ we have $bl_m > 1$.
For the ith block of consecutive ones in row m let

- $f_{m,i}$ be the column of the first 1 of block i and
- $l_{m,i}$ be the column of its last 1.

This means, that

$$a_{mj} = \begin{cases} 1 & \text{if there exists } i \in \{1, \ldots, bl_m\} \text{ such that } f_{m,i} \leq j \leq l_{m,i} \\ 0 & \text{otherwise.} \end{cases}$$

We remark that we can save a consecutive ones matrix in $\mathrm{O}(M)$ space and consequently, a matrix with almost consecutive ones property in almost linear space. We will henceforth use this data structure to save problem instances of (SCP) with almost C1P.

Consider a row A^{cov}_m of A^{cov} with $bl_m > 1$. We replace A^{cov}_m by bl_m rows,

$$B_{m,1}, B_{m,2}, \ldots, B_{m,bl_m}$$

each of them containing only one single block of row A_m, i.e., we define the jth element of row $B_{m,i}$ as

$$(B_{m,i})_j = \begin{cases} 1 & \text{if } f_{m,i} \leq j \leq l_{m,i} \\ 0 & \text{otherwise.} \end{cases}$$

The set covering problem

(SCP)

$$\begin{array}{rrcl} \min & cx & & \\ s.t. & A_m^{\text{cov}} x & \geq & 1 \quad \text{for } m = 1, \ldots, M \\ & x & \in & \{0,1\}^N \end{array}$$

can hence be reformulated as **(SCP')**

$$\begin{array}{rrcl} \min & cx & & \\ s.t. & A_m^{\text{cov}} x & \geq & 1 \quad \text{for } m = 1, \ldots, p \\ & B_{m,i} x & \geq & y_{m,i} \quad \text{for } m = p+1, \ldots, M, i = 1, \ldots, bl_m \\ & \sum_{i=1}^{bl_m} y_{m,i} & \geq & 1 \quad \text{for } m = p+1, \ldots, M \\ & y_{m,i} & \in & \{0,1\} \quad \text{for } m = p+1, \ldots, M, i = 1, \ldots, bl_m \\ & x & \in & \{0,1\}^N. \end{array}$$

Lemma 1 *(SCP) and (SCP') are equivalent.*

Proof:

(SCP) $\Longrightarrow$ (SCP'): Let x be a feasible solution of (SCP). Since $A_m^{\text{cov}} x \geq 1$ for all $m = 1, \ldots, M$ there exists (at least) one block $i = \mathfrak{l}(m)$ of row m such that $B_{m,i} x \geq 1$. Defining

$$y_{m,i} = \begin{cases} 1 & \text{if } i = \mathfrak{l}(m) \\ 0 & \text{otherwise} \end{cases}$$

yields $B_{m,i} x \geq y_{m,i}$ and $\sum_{i=1}^{bl_m} y_{m,i} \geq y_{m,\mathfrak{l}(m)} = 1$, hence (x, y) is feasible for (SCP') with the same objective value.

(SCP') $\Longrightarrow$ (SCP): On the other hand, each feasible solution of (SCP') satisfies $A_m^{\text{cov}} x \geq 1$ for $m = 1, \ldots, p$, while for $m = p+1, \ldots, M$ we know that

$$\sum_{i=1}^{bl_m} y_{m,i} \geq 1$$

and hence there exists (at least) one $i = \mathfrak{l}(m)$ for each row m with $y_{m,\mathfrak{l}(m)} = 1$. From this we conclude

$$B_{m,\mathfrak{l}(m)}x \geq y_{m,\mathfrak{l}(m)} = 1,$$

i.e., x covers block $l = \mathfrak{l}(m)$ of row m. This finally yields $A_m^{\text{cov}}x \geq 1$ also for $m = p+1, \ldots, M$. Together, $A^{\text{cov}}x \geq 1$, hence x is feasible for (SCP) with the same objective value.

QED

It is more convenient to rewrite (SCP') in matrix form. To this end, we define

- the matrix A as the first p rows of A^{cov},
- $bl = \sum_{m=p+1}^{M} bl_m$ as the total number of blocks in the "bad" rows of A^{cov}, i.e., in rows of A^{cov} without consecutive ones property,
- I as the $bl \times bl$ identity matrix,
- B as the matrix containing the bl rows $B_{m,i}$, $m = p+1, \ldots, M$,$i = 1, \ldots, bl_m$ and
- C as a matrix with $M-p$ rows and bl columns, with elements

$$c_{ij} = \begin{cases} 1 & \text{if } \sum_{m=p+1}^{p+i-1} bl_m < j \leq \sum_{m=p+1}^{p+i} bl_m \\ 0 & \text{otherwise.} \end{cases}$$

In the following we will use the next — equivalent — formulation of (SCP'):

(SCP')

$$\begin{array}{rrrcl} \min & cx & & & \\ \text{s.t.} & Ax & & \geq & \underline{1}_p \\ & Bx & -Iy & \geq & \underline{0}_{bl} \\ & & Cy & \geq & \underline{1}_{M-p} \\ & x & & \in & \{0,1\}^N, \\ & & y & \in & \{0,1\}^{bl}. \end{array} \tag{2}$$

The constraint $Cy \geq \underline{1}_{M-p}$ makes sure that at least one block of each row A_m^{cov} with $m \geq p+1$ is covered.

Note that all three matrices A,B, and C have the consecutive ones property. Unfortunately, the coefficient matrix of (SCP') does not have the consecutive ones property, and also is not totally unimodular in general, such that non-integer basic solutions may exist.

3 Deriving lower and upper bounds

Our reformulation (SCP') suggests simple bounds for the optimal solution. A lower bound is obtained by relaxing all constraints that contain variables $y_{m,i}$. This can be interpreted as simply forgetting about the rows which destroy the consecutive ones property of the matrix, i.e., we do not require them to be covered. The corresponding IP is the following set covering problem with C1P

(SCPl)

$$\begin{array}{rrcl} \min & cx & & \\ \text{s.t.} & Ax & \geq & \underline{1}_p \\ & x & \in & \{0,1\}^N. \end{array}$$

Lemma 2 *Each optimal solution of (SCPl) is a lower bound on (SCP').*

<u>Proof:</u> Since A only contains a part of the rows of A^{cov} (SCPl) is a relaxation of (SCP), and the result follows by Lemma 1.

QED

Since the coefficient matrix of (SCPl) has the consecutive ones property, solutions may be calculated efficiently. However, we can tighten the lower bound as follows. To this end, consider the dual of the LP-relaxation of (SCP'), given by

(Dual-SCP')

$$\begin{array}{rccccccl} \max & \underline{1}_p\eta_A & & & + & \underline{1}_{bl}\eta_C & & \\ s.t. & A^T\eta_A & + & B^T\eta_B & & & \leq & c \\ & & - & \eta_B & + & C^T\eta_C & \leq & 0 \\ & \eta_A & , & \eta_B & , & \eta_C & \geq & 0. \end{array}$$

We easily obtain a bound by solving (Dual-SCP') and rounding as follows.

Lemma 3 *Let $\eta' = (\eta'_A, \eta'_B, \eta'_C)$ be feasible for (Dual-SCP'), then*

$$f^l := \lceil \underline{1}_p \eta'_A + \underline{1}_{bl} \eta'_C \rceil$$

is a lower bound for (SCP').

<u>Proof:</u> By the well known duality results for linear programs, the expression $\underline{1}_p \eta'_A + \underline{1}_{bl} \eta'_C$ is a lower bound for the LP-relaxation of (SCP'), and thus for the problem itself. The integrality requirements allow for rounding up.

QED

Now suppose that an optimal solution x^l of (SCPl) is known. Let η^*_A be the corresponding dual optimal solution, i.e., belonging to problem

(A)

$$\begin{array}{rcl} \max \underline{1}\eta_A & & \\ \text{s.t. } A^T\eta_A & \leq & c \\ \eta_A & \geq & 0. \end{array}$$

Then, $\eta := (\eta^*_A, \underline{0}, \underline{0})$ is feasible for the dual of the LP-relaxation of (SCP') and hence a lower bound according to Lemma 3. It can be improved by performing a limited number of simplex pivots on (Dual-SCP') starting from η. We do not suggest to solve to optimality, as this may be too costly if the initial solution is far from optimal.

Now we turn our attention to the calculation of a upper bounds. We again start with the formulation (SCP') (see page 5). Fixing $y_{m,i} = 1$ for all $m \in \mathcal{M}$ and all $i = 1, \ldots, bl_m$ again results in set covering problem with consecutive ones property. Moreover, it yields a feasible solution to the original problem and thus an upper bound. This strategy requires that each row m which can be covered by more than one block *must* be covered by at least one column in each block. The solution found is hence feasible but will in general have more columns selected than necessary. Formally, this solution is found by solving

(SCPu)

$$\begin{array}{lrcl} \min & cx & & \\ \text{s.t.} & Ax & \geq & \underline{1}_p \\ & Bx & \geq & \underline{1}_{bl} \\ & x & \in & \{0,1\}^N \end{array}$$

Lemma 4 *Each feasible solution of (SCPu) is an upper bound on (SCP').*

Proof: Let x^u be a feasible solution of (SCPu). Defining $y = \underline{1}_{bl}$ yields a feasible solution (x^u, y) of (SCP'), hence cx^u is an upper bound on the optimal objective value. QED

A straightforward idea to improve this bound is, *not* to require that all rows of B are covered, but select only one of them for each original row m.

Definition 3 *Let $\mathfrak{l} : \{p+1, \ldots, M\} \to \mathcal{N}$ be a mapping selecting a block $i = \mathfrak{l}(m)$ for each row $m \in \{p+1, \ldots, M\}$. We call the mapping $\mathfrak{l}$ feasible if*

$$1 \leq \mathfrak{l}(m) \leq bl_m$$

for all $m = p+1, \ldots, M$. We also write $\mathfrak{l} \subseteq \{p+1, \ldots, M\} \times \mathcal{N}$ to specify $\mathfrak{l}$.

Now let $\mathfrak{l}$ be a feasible mapping and consider the following set covering problem with C1P.

(SCPu($\mathfrak{l}$))

$$\begin{array}{lrcl} \min & cx & & \\ \text{s.t.} & Ax & \geq & \underline{1}_p \\ & B_{m,\mathfrak{l}(m)}x & \geq & 1 \ \text{ for all } p+1, \ldots, M \\ & x & \in & \{0,1\}^N \end{array}$$

By solving (SCPu($\mathfrak{l}$)) we can derive an upper bound on (SCP') which is better than the best bound obtained by solving (SCPu) as follows.

Lemma 5 *Let x^* be the optimal solution of (SCP) and $\mathfrak{l}$ be a feasible mapping.*

1. *Each feasible solution x of (SCPu($\mathfrak{l}$)) satisfies $cx \geq cx^*$.*

2. *If x^u is an optimal solution of (SCPu), and $x^{u(\mathfrak{l})}$ an optimal solution of (SCPu($\mathfrak{l}$)) we have*
$$cx^* \leq cx^{u(\mathfrak{l})} \leq cx^u.$$

Proof:

1. We define for $m = p+1, \ldots, M$

$$y_{m,i} = \begin{cases} 1 & \text{if } i = \mathfrak{l}(m) \\ 0 & \text{otherwise} \end{cases}$$

to obtain a feasible solution (x, y) of (SCP') (and hence a feasible solution x of (SCP)) with the same objective value as (SCPu($\mathfrak{l}$)).

2. $cx^* \leq cx^{u(\mathfrak{l})}$ directly follows from part 1 of this lemma, while $cx^{u(\mathfrak{l})} \leq cx^u$ holds since (SCPu($\mathfrak{l}$)) is a relaxation of (SCPu).

QED

Next, we introduce a heuristic for (SCP') that works by choosing a good mapping $\mathfrak{l}(m)$ for the formulation (SCPu($\mathfrak{l}$)). It is based on a cost argument, i.e., for each row we choose the cheapest block that can be used to cover it:

Heuristic 1: Cost-Heuristic

Input: A^{cov}, b, c.

Output: `A feasible solution` x `of (SCP).`

Step 1. `Obtain matrices` A `and` B `of (SCP').`

Step 2. `For` $m = p+1, \ldots, M$:
`Assign` $\mathfrak{l}(m) = i$ `if` $c_j = \min_{j' : a_{mj'}=1} c_{j'}$ `and` $f_{m,i} \leq j \leq l_{m,i}$.

Step 3. `Let` x, y `be the solution of (SCPu(`$\mathfrak{l}$`)).`

Step 4. `Output:` x.

Note that it is also a reasonable strategy to choose $\mathfrak{l}(m)$ based on the probability that the chosen column can be used to cover many other rows, i.e., we change Step 2 in Heuristic 1 to

Step 2'. `For` $m = p+1, \ldots, M$:
`Assign` $\mathfrak{l}(m) = i$ `if` $|\text{cover}(j)| = \max_{j' : a_{mj'}=1} |\text{cover}(j')|$ `and` $f_{m,i} \leq j \leq l_{m,i}$.

Next we construct a feasible solution and an upper bound by combining the lower bound obtained from (SCPl) with the cost-based heuristic (Algorithm 1). To this end, determine the set of rows which are not covered by x^l, i.e., define $\mathcal{M}^0 = \{m : A_m^{\text{cov}} x^l = 0\}$ and choose $\mathfrak{l}(m)$ according to Algorithm 1 for all $m \in \mathcal{M}^0$. Note that x^l is an optimal solution of (SCP) if $\mathcal{M}^0 = \emptyset$. We solve the reduced set covering problem

(Red-SCP($\mathfrak{l}$))

$$\begin{array}{rrcl} \min & cx & & \\ \text{s.t.} & B_{m,\mathfrak{l}(m)}x & \geq & 1 \text{ for all } m \in \mathcal{M}^0 \\ & x & \in & \{0,1\}^N \end{array}$$

and let $\tilde{x}$ be an optimal solution. We obtain the following result.

Lemma 6 *Let x^* be an optimal solution of (SCP). Furthermore, let x^l be an optimal solution of (SCPl) and $\tilde{x}$ be an optimal solution of (Red-SCP($\mathfrak{l}$)). Then x given via*

$$x_n = \max\{x_n^l, \tilde{x}_n\}, \; n = 1, \ldots, N$$

is feasible for (SCP), and in particular $cx \geq cx^*$.

Proof: Let $\mathcal{M}^0 = \{m : A_m^{\text{cov}} x^l = 0\}$. Then, for all $m \notin \mathcal{M}^0$ we have that

$$A_m^{\text{cov}} x \geq A_m^{\text{cov}} x^l \geq 1,$$

hence these rows are covered by x. Now take $m \in \mathcal{M}^0$. We obtain

$$A_m^{\text{cov}} x \geq A_m^{\text{cov}} \tilde{x} \geq B_{m,\mathfrak{l}(m)} \tilde{x} \geq 1.$$

Together, $A^{\text{cov}} x \geq \underline{1}_M$ and the result follows.

QED

The following algorithm contains upper and lower bound computation both based on a solution x^l of (SCPl).

Algorithm 2: Upper and lower bound for (SCP)

Input: `data of (SCP), iteration limit` k.

Output: `Feasible solution` x^u `and lower bound` f^l `on (SCP).`

Step 1: `Solve (SCPl) with optimal solution` x^l `and dual solution` η^*_A.

Step 2: `Perform` k `simplex iterations on` $(Dual{-}SCP')$ `starting from the feasible solution` $\eta = (\eta^*_A, \underline{0}, \underline{0})$. `Let` $\eta'_A, \eta'_B, \eta'_C$ `be the result.`

Step 3: `For all` $m \in \mathcal{M}^0 := \{m | A_m x^l = 0\}$ `find` $\mathfrak{l}(m)$ `as in Heuristic 1.`
`If` $\mathcal{M}^0 = \emptyset$ `stop:` $x^u = x^l$ `is optimal solution.`

Step 4: `Solve (Red-SCP(`$\mathfrak{l}$`)) with respect to` $\mathcal{M}^0$ `and` $\mathfrak{l}$.
`Let` $\tilde{x}$ `be the solution.`

Step 5: `Define for all` $n \in \mathcal{N}$: $x^u_n = \max\{x^l_n, \tilde{x}_n\}$.

Step 6: `Output:` x^u `and` $f^l = \lceil \underline{1}_p \eta'_A + \underline{1}_{bl} \eta'_C \rceil$.

---

4 Branch and bound approach

For solving set covering problems with almost C1P we propose a branch and bound algorithm based on the equivalence of (SCP) and (SCP'). The idea is to consider a row m of the original covering matrix A^{cov} (for $p < m \leq M$) in each layer of the branch and bound tree and iteratively select one of the $y_{m,i}$ variables in (SCP') and set it to one. This means, the corresponding row $B_{m,i}$ can be added to matrix A in (SCP') while all other rows $B_{m,i'}$ with $i' \neq i$ can be deleted from B.
Consider an instance of (SCP'). For a set of rows $\mathcal{M}^{\text{fix}} \subseteq \{p+1, \ldots, M\}$ with $bl_m > 1$ for all $m \in \mathcal{M}^{\text{fix}}$ and a feasible mapping $\mathfrak{l}$ on $\mathcal{M}^{\text{fix}}$ we define $\mathrm{P}(\mathcal{M}^{\text{fix}}, \mathfrak{l})$ as the new problem instance of (SCP') in which the variables $y_{m,\mathfrak{l}(m)}$ are fixed to 1 for all $m \in \mathcal{M}^{\text{fix}}$.
Using the notation $\mathcal{M}^C = \{p+1, \ldots, M\} \setminus \mathcal{M}^{\text{fix}}$ we get

$\mathbf{P}(\mathcal{M}^{\text{fix}}, \mathfrak{l})$

$$\begin{array}{lrcl}
\min & cx & & \\
\text{s.t.} & Ax & \geq & \underline{1}_p \\
& B_{m,\mathfrak{l}(m)} x & \geq & 1 \text{ for all } m \in \mathcal{M}^{\text{fix}} \\
& B_{m,i} x & \geq & y_{m,i} \;\; \text{for } m \in \mathcal{M}^C, i = 1, \ldots, bl_m \\
& \sum_{i=1}^{bl_m} y_{m,i} & \geq & 1 \;\; \text{for } m \in \mathcal{M}^C \\
& y_{m,i} & \in & \{0,1\} \;\; \text{for } m \in \mathcal{M}^C, i = 1, \ldots, bl_m \\
& x & \in & \{0,1\}^N.
\end{array}$$

Lemma 7 *Let x^* be an optimal solution of (SCP), and let $x^{\mathcal{M}^{\text{fix}},\mathfrak{l}}, y^{\mathcal{M}^{\text{fix}},\mathfrak{l}}$ be an optimal solution of $P(\mathcal{M}^{\text{fix}},\mathfrak{l})$. Then*

1. *$cx^* \leq cx^{\mathcal{M}^{\text{fix}},\mathfrak{l}}$.*
2. *For each fixed $\mathcal{M}^{\text{fix}} \subseteq \{p+1,\ldots,M\}$ we have*

$$cx^* = \min_{\mathfrak{l}\text{ feasible}} cx^{\mathcal{M}^{\text{fix}},\mathfrak{l}}.$$

Proof:

1. Extend $x^{\mathcal{M}^{\text{fix}},\mathfrak{l}}, y^{\mathcal{M}^{\text{fix}},\mathfrak{l}}$ to a feasible solution of (SCP') by defining
$$y_{m,i} = \begin{cases} 1 & \text{if } i = \mathfrak{l}(m) \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } m \in \mathcal{M}^{\text{fix}}.$$
2. Let x^*, y^* be an optimal solution of (SCP'). Then for all $m \in \{p+1,\ldots,M\}$ there exists some i such that $y_{m,i} = 1$. Define $\mathfrak{l}(m) = i$ for all $m \in \mathcal{M}^{\text{fix}}$ and let $y^{\mathcal{M}^{\text{fix}}}$ be the vector y^*, restricted to the components of $\mathcal{M}^{\text{fix}}$. This means, $x^*, y^{\mathcal{M}^{\text{fix}}}$ is feasible for P($\mathcal{M}^{\text{fix}},\mathfrak{l}$) and consequently,
$$cx^{\mathcal{M}^{\text{fix}},\mathfrak{l}} \leq cx^*$$
From part 1 we already know $cx^* \leq cx^{\mathcal{M}^{\text{fix}},\mathfrak{l}}$, hence equality is attained.

QED

The following observations are the basis for the branch and bound approach.

- $P(\emptyset,\emptyset)$ =(SCP').
- Fixing $y_{m,i} = 1$ in $P(\mathcal{M}^{\text{fix}},\mathfrak{l})$ for some $m \in \mathcal{M}^C$ and for some $1 \leq i \leq bl_m$ leads to $P(\mathcal{M}^{\text{fix}} \cup \{m\}, \mathfrak{l} \cup \{(m,i)\})$
- The coefficient matrix of $P(\{p+1,\ldots,M\},\mathfrak{l})$ has the consecutive ones property and the problem can hence be solved efficiently, e.g., by an adapted network simplex approach as described in detail in [Ruf02].

Thus, by fixing iteratively variables $y_{m,i}$ we always obtain subproblems of the same type, and in each iteration the number of rows m with $bl_m = 1$ increases (yielding a larger matrix A with consecutive ones property) while the number of "bad" rows m with $bl_m > 1$ decreases. Hence, we get closer to the consecutive ones property in each step.
The branch and bound algorithm can finally be stated as follows.

Algorithm 3: Branch and bound for (SCP)

Input: A^{cov}, b, c, and accuracy ϵ.

Output: Feasible solution x of (SCP), such that $cx - cx^* \leq \epsilon cx^*$,
if x^* is the optimal objective value.

Step 1: Initialize best known upper bound $f^u := \infty$,
best known solution $x := \underline{1}$,
and set of problems to be investigated $List := \{P(\emptyset, \emptyset)\}$.

Step 2: While $List \neq \emptyset$ **do**

Step 3: Select problem $P = P(\mathcal{M}^{\text{fix}}, \mathfrak{l}) \in List$ and reduce its size according to Section 5.

Step 4: For P, calculate lower bound f_P^l, upper bound f_P^u, and corresponding feasible solution x_P with Algorithm 2.

Step 5: If $f^u > f_P^u$ **then** update $f^u := f_P^u$, $x := x_P$.

Step 6: If $f^u > (1+\epsilon)f_P^l$ **then** select row $m \in \mathcal{M}^C$ and update $List := List \cup \{P(\mathcal{M}^{\text{fix}} \cup \{m\}, \mathfrak{l} \cup \{(m,i)\}) | i = 1, \ldots, bl_m\}$.

Step 7: $List := List \setminus \{P\}$.

5 Reducing the size of the problem

Before attempting to solve a set covering problem it is advisable to try and reduce its size. The well-known reduction rules of [TR73] (see also [NW88]) can be modified slightly to account for the special data structure used for storing the instance of a set covering problem with almost C1P property. Recall that for each row m of the original covering matrix A^{cov} we only have to store the first and the last column $f_{m,i}$ and $l_{m,i}$ of each block i.

Lemma 8 *Let* $m, m_1 \in \{1, \ldots, M\}$.

1. *If $bl_m = 0$, the problem is infeasible.*
2. *If $bl_m = 1$ and $f_{m,1} = l_{m,1}$, all feasible solutions x of (SCP) satisfy $x_{f_{m,1}} = 1$.*
3. *If $bl_m = 1$ and there exists $i_1 \in \{1, \ldots, bl_{m_1}\}$ such that*

$$f_{m_1,i_1} \leq f_{m,1} \leq l_{m,1} \leq l_{m_1,i_1},$$

it is sufficient to consider (SCP) without row $A^{\text{cov}}_{m_1}$.

The first two rules are trivial to check and apply, and the third can be efficiently implemented for matrices with C1P. As has been shown in [Sch03], all possible reductions according to rule 3 can be performed in $O(NlogN)$ time for an $N \times N$ C1P-matrix. In our case, only in the first p rows of A^{cov} the ones appear consecutively, i.e., each of them only has one block of ones between $f_{k,1}$ and $l_{k,1}$. Applying the reduction procedure leads to the *strictly monotone form* of the first rows of A^{cov}, i.e.,

$$f_{1,1} < f_{2,1} < \ldots < f_{k,1} \text{ and } l_{1,1} < l_{2,1} < \ldots < l_{k,1}$$

reducing the size of the matrix A to $k \leq p$ rows. Still missing from the list of rules in Lemma 8 is the usual column reduction criterion ([TR73]) for set covering problems with non-unit costs. While there is no obvious reformulation of this rule in terms of the special data structure $f_{m,i}, l_{m,i}$ of (SCP), it still allows us to limit optimization to column sets of the following type:

Definition 4 *Let $S \subset \mathcal{N}$. For each row $m \in \mathcal{M}$ and block of ones $i \in \{1, \ldots, bl_m\}$, set $c_{m,i}(S) = \min\{c_j | f_{m,i} \leq j \leq l_{m,i} \wedge j \in S\}$, as well as*

$$j^{\min}_{m,i}(S) := \min\{j \in S | f_{m,i} \leq j \leq l_{m,i} \wedge c_{j^{\min}_{m,i}(S)} = c_{m,i}(S)\}$$

as the leftmost column in which the minimum is attained, and

$$j^{\max}_{m,i}(S) := \max\{j \in S | f_{m,i} \leq j \leq l_{m,i} \wedge c_{j^{\max}_{m,i}(S)} = c_{m,i}(S)\}$$

as the rightmost column containing the minimum.

The **left-hand reduced column set given S** *is now defined as*

$$L(S) := S \cap \bigcup_{m=1}^{M} \bigcup_{i=1}^{bl_m} \{f_{m,i}, \ldots, j^{\min}_{m,i}(S)\},$$

and the **right-hand reduced column set given S** *as*

$$R(S) := S \cap \bigcup_{m=1}^{M} \bigcup_{i=1}^{bl_m} \{j_{m,i}^{\max}(S), \ldots, l_{m,i}\}.$$

The above can be used to construct a column set sufficient for optimization:

Lemma 9

1. *To find an optimal solution of (SCP), it is only necessary to consider those columns of A^{cov} with index in $R(L(\mathcal{N}))$.*
2. $R(L(\mathcal{N})) = L(R(L(\mathcal{N}))) = R(R(L(\mathcal{N})))$.

Proof:

1. It is sufficient to show that using only columns in $L(\mathcal{N})$ for optimization yields an optimal solution, since everything else works analogously.

   Assume that $L(\mathcal{N}) \neq \mathcal{N}$, let $j_0 \in \mathcal{N} \setminus L(\mathcal{N})$, and let m_0, i_0 such that

   $$f_{m_0,i_0} = \max\{f_{m,i} : m \in \mathcal{M}, i \in \{1, \ldots, bl_m\} \text{ and } f_{m,i} \leq j_0 \leq l_{m,i}\}.$$

   Since $j_0 \notin L(\mathcal{N})$, we have

   $$f_{m_0,i_0} \leq j_{m_0,i_0}^{\min}(\mathcal{N}) < j_0.$$

   By choice of m_0, it is clear that the column $j_{m_0,i_0}^{\min}(\mathcal{N})$ of A^{cov} contains a 1 in each row where column j_0 has a 1. And by Definition 4, its cost coefficient is less than or equal to c_{j_0}. But these two arguments together form the usual column reduction criterion for set covering problems with non-unit costs, i.e. an optimal solution for (SCP) can be found without considering column j_0.

2. Let $S_0 := R(L(\mathcal{N}))$. Observe that $R(S) = R(R(S))$ for any $S \subset \mathcal{N}$ by construction, so in particular $S_0 = R(S_0)$. Assume that $S_0 \neq L(S_0)$ and let $j_0 \in S_0 \setminus L(S_0)$. Define

   $$m_0 := \arg\max_{m \in \mathcal{M}} \{f_{m,i} | f_{m,i} \leq j_0 \leq l_{m,i}, i \in \{1, \ldots, bl_m\}\}.$$

   If i_0 is the block index such that $f_{m_0,i_0} \leq j_0 \leq l_{m_0,i_0}$, $j_0 \notin L(S_0)$ requires

   $$f_{m_0,i_0} \leq j_{m_0,i_0}^{\min}(S_0) < j_0.$$

As $S_0 \subset \mathcal{N}$, we also have $j^{\min}_{m_0,i_0}(S_0) \geq j^{\min}_{m_0,i_0}(\mathcal{N})$, i.e.

$$f_{m_0,i_0} \leq j^{\min}_{m_0,i_0}(\mathcal{N}) < j_0.$$

But the choice of m_0 implies that $j^{\min}_{m_0,i_0}(\mathcal{N}) \geq j^{\min}_{m,i_0}(\mathcal{N})$ for all pairs (m, i) such that $f_{m,i} \leq j_0 \leq l_{m,i}$, and thus $j_0 \notin L(\mathcal{N})$, a contradiction to $j_0 \in S_0 \subset L(\mathcal{N})$.

QED

Note that the results of Lemma 9 apply analogously to the set $L(R(\mathcal{N}))$. Since in general $R(L(\mathcal{N})) \neq L(R(\mathcal{N}))$, a reduction heuristic based on the above should determine both sets and choose the smaller one for optimization. Part 2 of the lemma shows that further applications of the procedure are futile. Constructing the sets and choosing the smaller one can easily be implemented with a time complexity of $O(MN)$, i.e. linear in matrix size, whereas the implementation of the classical column reduction criterion due to [TR73] usually needs $\mathrm{O}(N^2M)$.
The outlined procedure is a heuristic in the sense that it will in general not remove all columns possible. In fact, it does not even consider dominating columns with non-minimal cost. Nevertheless, the impact on the real-world problem is satisfactory, as shown in Section 6.

6 Numerical results

As mentioned in the introduction, the main purpose of our branch and bound algorithm was to solve a *stop location problem* provided by Deutsche Bahn. The goal in the stop location problem is to cover a given set of demand points by new stops along the track system. A demand point is *covered* if the distance to its closest stop is smaller than a given *covering radius* r. After deriving the finite dominating set of candidate locations for new stops via the method in [SHLW02, Sch02], the problem can be formulated as (SCP) with the following interpretation.

- Each row of A^{cov} corresponds to a demand point.
- Each column of A^{cov} corresponds to a candidate location for a new train station on the existing network of tracks.

- $A^{\mathrm{cov}}_{i,j} = 1$ if and only if candidate location j is at most at distance r from demand point i.
- The costs c are the traffic loads at the candidate locations as a measure of the negative effects of the new stops, i.e. passengers sitting in the train and waiting while the train halts.

We refer to [SHLW02, Sch02] for details. An optimal solution of the problem corresponds to placing new train stations such that all demand points are covered and the negative effect of the new stops is minimized. The covering radius r is given as 2 km, but we also generated problem instances R_r for values of $r = 1, 3, 5, 10$ km, using the same sets of demand points and candidate locations. It can be shown that the covering matrix A^{cov} has the consecutive ones property if the network consists of a straight line rail track only, see [SHLW02, Sch02]. In our real world data, constellations of demand points and candidate locations which result in submatrices violating this property are rare. Thus, the test problems almost have the consecutive ones property, as can be seen in the Table 1 (which will be described in detail below). In particular, the more rows of A^{cov} already have the consecutive ones property, the fewer branchings are required by the algorithm in worst case. Consecutive block minimization is NP-hard [GJ79], so we recommend using some kind of sorting heuristic to improve the structure of the matrix (see e.g. [Ruf02, OR03]).

The algorithm was tested on other instances as well, to get an idea of the class of problems it can solve in reasonable time. First, we applied it to the unit-cost set covering problems arising from the incidence matrices of Steiner triple systems. These problems were introduced by Fulkerson, Nemhauser, and Trotter in 1974 [BL76] (see also [MT98]), who suggest using them as test cases for set covering algorithms. This is motivated by the fact that they are hard to solve despite their relatively small size. Note also that each row of such a matrix has only 3 non-zero entries, i.e., the problem instances almost have the consecutive ones property according to the initial definition. The algorithm was applied to the instances with 27, 45, 81, 135, and 243 columns, referred to as $STS_{27}, \ldots, STS_{243}$. For a specialized algorithm that can solve up to STS_{81}, see [MS94].
More tests were done using randomly generated problem instances of small size (100×100). To highlight the differences between sparse matrices and

Table 1: Algorithmic performance

#	(SCP) before Reduction Cols	Rows* Total	Rows* Split	Max. Blocks	(SCP') after Red. Cols	Rows*	Initial lower bound	Gap	Solution Subp.	Time**	Value†	Opt?
Real-world problem instances												
R_{1km}	3,145	757	7	2	707	4	856,943	0.0 %	1	0 s	856,943	yes
R_{2km}	3,145	1,196	148	3	889	80	1,005,524	1.3 %	9,841	338 s	1,005,524	yes
R_{3km}	3,145	1,419	311	5	886	194	901,077	1.7 %	57,681	1 h	905,195	no
R_{5km}	3,145	1,123	329	5	593	199	505,052	2.2 %	4,860	94 s	505,052	yes
R_{10km}	3,145	275	66	7	165	29	139,370	0.1 %	2	0 s	139,370	yes
Problems based on STS												
STS_{27}	27	117	110	3	27	325	0	19	2,499,475	1 h	18 (18)	no
STS_{45}	45	330	318	3	45	946	0	37	1,139,677	1 h	32 (30)	no
STS_{81}	81	1,080	1,053	3	81	3,160	3	62	422,587	1 h	64 (61)	no
STS_{135}	135	3,015	3,015	3	135	8,911	0	111	138,370	1 h	109 (104)	no
STS_{243}	243	9,801	9,720	3	243	29,161	3	208	62	1 h	211 (202)	no
Random instances with 1-5 blocks per row (15 % density)												
A_1	100	100	86	5	95	157	320	35	44	0 s	320	yes
A_2	100	100	77	5	92	69	550	56	23	0 s	552	yes
A_3	100	100	83	5	92	158	411	71	105	0 s	411	yes
A_4	100	100	82	5	88	118	540	81	133	0 s	540	yes
A_5	100	100	77	5	98	121	532	60	35	0 s	532	yes
Random instances with 3% density												
B_1	100	100	84	9	96	152	623	1131	121	0 s	1385	yes
B_2	100	100	76	8	95	129	1277	171	251	1 s	1277	yes
B_3	100	100	84	8	92	189	1482	97	222	1 s	1482	yes
B_4	100	100	81	8	92	163	1384	225	2430	9 s	1384	yes
B_5	100	100	79	6	92	164	1449	118	214	0 s	1449	yes
Random instances with 5% density												
C_1	100	100	95	10	100	344	316	921	243,522	1,571 s	939	yes
C_2	100	100	98	9	100	457	770	273	580,303	1 h	1,008	no
C_3	100	100	98	11	99	391	152	985	464,253	1 h	868	no
C_4	100	100	95	10	100	350	180	1,001	443,992	1 h	987	no
C_5	100	100	97	11	100	435	99	1,153	390,353	1 h	898	no

* The number of rows before reduction refers to A^{cov} of (SCP), while the number after refers to A and B of (SCP'), i.e. they are not directly comparable.
** Algorithm implemented in C++ on a PC with 2.2 GHz processor; all values obtained with 1 h time limit using at most 1000 simplex pivots per iteration to improve the lower bound.
† For the STS_i, values in brackets are the solutions found in [MS94]. All but the last one are known to be optimal.

those with almost consecutive ones property, we give results for three sets of random problems, all with randomized non-unit costs:

1. In problems $A_1, \ldots, A_5$ we generated matrices with 1 to 5 blocks of consecutive ones per row, each consisting of 1 to 9 ones. This results in an average density of ones of 15 %, i.e., the matrices are not sparse but almost have the C1P.

2. Instances $B_1, \ldots, B_5$ were generated with a probability of 3 % for any given entry to be one. This results in matrices which are both sparse and almost have the C1P.

3. Finally, $C_1, \ldots, C_5$ are similar to the B_i, but with a density of 5 %.

Table 1 shows how our algorithm performed on these instances (where we set a time limit of one hour running time). It lists the following information:

Before Reduction: Here we list the number of columns (**Cols**) and rows (**Rows**), where **Total** refers to the total number of rows and **Split** contains the number of rows with more than one block in the original formulation (SCP). We also listed the maximal number of blocks **Max. Blocks** appearing in the original data.

After Red.: The number of columns and rows are listed again after applying Lemmas 9 and 8. Note that the formulation used here is (SCP'), i.e., each split row is decomposed into a row for each block.

Initial: We further list the lower bound **LB** found for the first subproblem, which is a global lower bound, and the difference **Gap** between the initial upper and lower bound. In case of the real world problems, it is expressed as a fraction of the lower bound.

Solution: Here, **Subp.** contains the number of subproblem instances solved by the algorithm within our time limit of one hour, **Time** lists the total running time of the algorithm, and **Value** refers to the best solution value found. Finally, **Opt?** states, whether optimality was recognized. Note that some of the STS problems where solved optimal although the algorithm did not terminate within the time limit. For this class of problems, we also listed the best known values in brackets in the **Value** column.

The following observations should be mentioned:

1. The real-world problem instances are reduced markedly through preprocessing, and all except the one with 3 km cover radius can be solved to optimality within the time limit. However, in all cases where optimality is established, the initial lower bound is equal to the optimal objective value. Since the bound is valid for all derived subproblems, the algorithm terminates as soon as a corresponding feasible solution is found. Judging from the problem with $r = 3$ km, the number of required branchings is not yet satisfactory if the initial bound is not tight.

   Still, the initial duality gap is so small for all problems that even a single iteration of the algorithm appears to be a good heuristic.

2. The performance on the STS_i problems is not satisfactory. Initial reduction has little effect, and the algorithm requires far too many iterations, although the initial solutions are fairly close to the optima. This result prompts a revision of the notion of almost having the C1P for a matrix in Section 7.

3. The randomized instances illustrate that sparsity and almost having the C1P are indeed two different things.

Note that in the case of hard unit-cost problems like STS_i, the performance of the algorithm as a heuristic can be improved by assigning costs from a large range to cut down on the number of subproblems which need to be investigated. The resulting solutions of the new weighted problem are feasible for the original problem with unit-costs and yield a good approximation of the problem in a considerably smaller running time. For example, assigning the Fibonacci series as costs to STS_{27} results in a problem instance which can be solved to optimality in less than 20 minutes by our branch and bound algorithm. The resulting solution needs 19 columns instead of the optimal 18 for the unit cost problem.

7 Extensions

As we have seen in Section 6, the initial definition of a matrix with almost consecutive ones property includes the instances based on Steiner triple systems, where the algorithm generates too many subproblems to be efficient.

Table 2: An added criterion for almost C1P

#	$log_2(T)$	$2N$	**Opt?**	#	$log_2(T)$	$2N$	**Opt?**
Real-world instances				*Random instances (1-5 blocks per row)*			
R_{1km}	10.0	1414	Yes	A_1	151.7	190	Yes
R_{2km}	164.6	1778	Yes	A_2	141.8	184	Yes
R_{3km}	366.3	1772	No	A_3	141.5	184	Yes
R_{5km}	418.5	1186	Yes	A_4	146.1	176	Yes
R_{10km}	104.4	330	Yes	A_5	136.0	196	Yes
STS-based instances				*Random instances (3% density)*			
STS_{27}	174.5	54	No	B_1	144.1	195	Yes
STS_{45}	500.6	90	No	B_2	151.0	190	Yes
STS_{81}	1663.8	162	No	B_3	149.5	184	Yes
STS_{135}	4711.8	270	No	B_4	137.2	184	Yes
STS_{243}	15372.3	486	No	B_5	133.7	184	Yes
				Random instances (5% density)			
				C_1	209.1	200	Yes
				C_2	230.6	200	No
				C_3	217.3	198	No
				C_4	216.0	200	No
				C_5	221.6	200	No

Thus, it seems necessary to include a limit on the worst-case number of subproblems in a more appropriate definition of a matrix almost having the C1P:

Definition 5 *For (SCP) as in 2 determine*

$$T := (M - p + 1) \prod_{m=p+1}^{M} bl_m.$$

The matrix A^{cov} almost has the consecutive ones property for computational purposes if for sufficiently small constant $c > 0$ holds

$$\log_2(T) \leq cN.$$

The above is motivated by the following lemma, and the fact that 2^N is the complexity of solving the problem by total enumeration, i.e., the condition is equivalent to

$$T \leq (2^N)^c.$$

Lemma 10 *T is an upper bound on the number of subproblems $P = P(\mathcal{M}^{fix}, \mathfrak{l})$ processed by Algorithm 3.*

Proof: To find an exact solution, the accuracy for algorithm 3 is set to $\epsilon = 0$, i.e., Step 6 generates new subproblems if $f^u > f^l_P$. Note that the bounds always coincide if $\mathcal{M}^{fix} = \{p+1, \ldots, M\}$, as both the problem (SCPu($\mathfrak{l}$)) and its dual (A) have the C1P. Therefore, in worst case, each subproblem P gives rise to bl_m new problems, where m is the row selected in Step 6, until $\mathcal{M}^{fix} = \{p+1, \ldots, M\}$. We can assume w.l.o.g. that the rows are selected in ascending order. Since the first subproblem has $\mathcal{M}^{fix} = \emptyset$ and each newly generated problem adds one element to the set, the maximal number of subproblems is less than or equal to

$$\sum_{k=0}^{M-p} \prod_{m=p+1}^{p+k} bl_m \leq (M-p+1) \prod_{m=p+1}^{M} bl_m = T.$$

QED

Note that T, like the notion of almost C1P itself, depends heavily on the order of the columns chosen for A^{cov}. Even worse, the criterion is influenced by the ratio of rows to columns in the matrix, which can change drastically during preprocessing. Thus, the extended definition should be treated as a rule of thumb only. Still, Table 2 shows that choosing $c = 2$ classifies the problems of Section 6 properly.

Another field of research is motivated by results in [Ruf02] showing that some sparse matrices can be transformed to almost have the C1P via column permutation. As the criterion favors matrices with more columns than rows, it would be better to deal with the dual problem based on $(A^{cov})^T$ if $M \gg N$. Another case where the dual is of interest are set covering problems where the covering matrix is "almost" an interval matrix, i.e. $(A^{cov})^T$ almost has the C1P. The algorithm as given cannot deal with the resulting set packing problems, but since such problems are as easy as set covering for totally unimodular matrices, a generalization of the algorithm to include set packing would be of interest.

References

[BL76] K.S. Booth and G.S. Lueker. Testing for the consecutive ones property, interval graphs and graph planarity using pq-tree algorithms. *Journal of Computing Systems Science*, 13(3):335–379, 1976.

[CFT+97] A. Caprara, M. Fischetti, P. Toth, D. Vigo, and P.L. Guida. Algorithms for railway crew management. *Mathematical Programming*, 79, 1997.

[CFT99] A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. *Operations Research*, 47(5):730–743, 1999.

[CFT00] A. Caprara, M. Fischetti, and P. Toth. Algorithms for the set covering problem. *Annals of Operations Research*, 98:353–371, 2000.

[CNS97] S. Ceria, P. Nobili, and A. Sassano. Set covering problem. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated bibliographies in combinatorial optimization*, pages 415–428. J. Wiley and Sons, 1997.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[jVW62] A. F. jun. Veinott and H.M. Wagner. Optimal capacity scheduling. *Operations Research*, 10:518–532, 1962.

[LY94] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960–981, 1994.

[MS94] C. Mannino and A. Sassano. Solving hard set covering problems. Technical Report 14, Department of Information and System Science, University of Rome "La Sapienza", 1994.

[MS00] E. Marchiori and A. Steenbeek. An evolutionary algorithm for large set covering problems with application to airline crew scheduling. In *EvoWorkshops*, pages 367–381. 2000.

[MT98] J. Meidanis and G.P. Telles. On the consecutive ones property. *Discrete Applied Mathematics*, 88:325–354, 1998.

[NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.

[OR03] M. Oswald and G. Reinelt. The weighted consecutive ones problem for a fixed number of rows or columns. *Operations Research Letters*, 2003. to appear.

[Ruf02] N. Ruf. Locating train stations: Set covering problems with "almost c1p" matrices. Master's thesis, University of Kaiserslautern, 2002.

[Sch02] A. Schöbel. Customer-oriented optimization in public transportation, 2002. Habilitationsschrift, Universität Kaiserslautern.

[Sch03] A. Schöbel. Set covering problems with consecutive ones property. Technical report, Universität Kaiserslautern, 2003. working paper.

[SHLW02] A. Schöbel, H.W. Hamacher, A. Liebers, and D. Wagner. The continuous stop location problem in public transportation. Technical report, Universität Kaiserslautern, 2002. Report in Wirtschaftsmathematik Nr. 81/2001.

[TR73] C. Toregas and C. ReVelle. Binary logic solutions to a class of location problems. *Geographical Analysis*, 5:145–155, 1973.