A Practical Algorithm for Computing a Subadditive Dual Function for Set Partitioning

Diego Klabjan
Department of Mechanical and Industrial Engineering
University of Illinois at Urbana-Champaign
Urbana, IL
email: klabjan@uiuc.edu

Abstract

Recently, a new algorithm for computing an optimal subadditive dual function to an integer program has been proposed. In this paper we show how to apply the algorithm to the set partitioning problem. We give several enhancements to the algorithm and we report computational experiments. The results show that it is tractable to obtain an optimal subadditive function for small and medium size problems. To the best of our knowledge this is the first work that reports computational experiments on computing an optimal subadditive dual function.

1 Introduction

The linear programming duality has been developed a long time ago and the dual vector of an LP is used in many algorithms and in sensitivity analysis. On the other hand, very little is known about an equivalent notion for integer programming. Gomory (1969) developed subadditive duality for the group problem and Johnson (1973) extended his theory to integer programs. There are very few algorithms for computing an optimal subadditive function (OSF). Burdet and Johnson (1977) and Llewellyn and Ryan (1993) present an algorithm for computing an OSF but they do not report computational results and they do not elaborate on how to use the OSF in developing further algorithms or for sensitivity analysis. Wolsey (1981) addresses sensitivity analysis for integer programs and he shows how to obtain an OSF from the branch-and-bound tree.

Frequently in integer programming (IP) we would like to estimate the change of the objective value if we perturb the right hand side. Sensitivity analysis for IPs is typically done either by considering the dual vector of the LP relaxation or by resolving the problem after changing the right hand side. Is it possible to compute a vector or a function that would measure the change in the objective function of an IP after perturbing the right hand side? A slightly different scenario is when we are given a new variable and we wonder how the optimal objective value changes if this variable is added to the formulation. In many real world problems that are modeled as IPs we would like to obtain alternative optimal solutions. For example, a decision maker wants to select the most robust solution among several optimal solutions. All optimal solutions can be found among the variables with zero reduced cost, which requires an optimal dual function and a suitable definition of reduced cost. In this work we show that by using appropriate subadditive functions, we can perform sensitivity analysis, which gives much better results than using LP dual vectors.

Recently, Klabjan (2003) gives a new family of subadditive functions, called the *generator subadditive functions*. These functions are easy to encode and typically easy to evaluate. He presents several properties of generator subadditive functions and various IP related algorithms that use the generator OSF. He also presents an algorithm that computes a generator OSF. In this paper we show how to adapt the algorithm to the set partitioning problem

$$\min\{cx : Ax = \mathbf{1}, x \text{ integer}\}\,, \tag{1}$$

where $\mathbf{1}$ is a vector with every component equal to 1 and A is an $m \times n$ matrix with 0/1 coefficients. We denote by $a_i \in \mathbb{R}^m$ column i of A and by $a^j \in \mathbb{R}^n$ row j of A. We give several enhancements to the algorithm presented in Klabjan (2003), and we present computational experiments. To the best of our knowledge this paper is the first paper that reports computational results on obtaining an OSF. In addition, we show how to construct a generator OSF from a generator OSF of the preprocessed problem.

In Section 2 we list the preprocessing rules together with three new rules. Algorithmic enhancements for set partitioning are given in Section 3. An interesting row generation procedure is presented in this section and some other implementation details. How to construct a generator OSF from the preprocessed problem is shown in Section 4. Section 5 reports extensive computational experiments. We conclude the introduction with a brief description of the algorithm for computing a generator OSF from Klabjan (2003).

Algorithm for Obtaining a Generator Optimal Subadditive Function

Johnson (1973) showed that

$$\begin{array}{ccc} \min\ cx & & \max\ F(\mathbf{1}) \\ Ax = \mathbf{1} & = & F(a_i) \le c_i \qquad i = 1, \dots, n \\ x \text{ integer} & & F \text{ subadditive}\,, \end{array} \tag{2}$$

where *subadditivity* is defined as $F(d_1 + d_2) \le F(d_1) + F(d_2)$ for every pair of vectors d_1 and d_2 in $\mathbb{R}^m$. We say that function F is *dual feasible* if $F(a_i) \le c_i$ for all $i = 1, \dots, n$, and the value of a subadditive function F is $F(\mathbf{1})$.

For simplicity of exposition, we assume that (1) is feasible (see Klabjan (2003) for treatment of infeasible problems) and we denote the variables indices as $N = \{1, 2, \dots, n\}$. Given a vector $\alpha \in \mathbb{R}^m$, we define a generator subadditive function $F_\alpha : \mathbb{R}^m \to \mathbb{R}$ as

$$\begin{aligned} F_\alpha(d) = \alpha d - \max & \sum_{i \in E} (\alpha a_i - c_i) x_i \\ & A^E x \le d \\ & x \text{ integer}\,, \end{aligned}$$

where $E = \{i \in N : \alpha a_i > c_i\}$ is a generator set and A^E is the submatrix of A corresponding to the columns in E. Similarly we denote c^E. For simplicity of notation we write $H = N \setminus E$ and whenever an ambiguity can occur we write $E(\alpha)$ instead of simply E. It is easy to see that F_α is a dual feasible subadditive function. It is shown in Klabjan (2003) that there exists a generator OSF, i.e. there exists a generator subadditive function F_α with $F_\alpha(\mathbf{1}) = z^{\mathrm{IP}}$, where z^{IP}

is the optimal value to (1). We say that F_α is *optimal over* $S \subseteq N$ if it is a generator OSF to the IP $\min\{c^S x : A^S x = \mathbf{1}, x \text{ integer}\}$.

Next we outline the algorithm that finds an optimal solution to (1) and computes a generator OSF, Algorithm 1. We first reduce the problem size by preprocessing (see Section 2) and next we add clique inequalities. If the LP relaxation with clique inequalities gives an integral solution, then a generator OSF is readily available by selecting α as the optimal dual vector corresponding to constraints $Ax = \mathbf{1}$ (see Klabjan (2003)). The algorithm then proceeds in two stages. In the first stage, consisting of steps 3-10, we find an approximate generator OSF and an optimal primal solution. The second stage (steps 11-29) starts with the subadditive function from the first stage and it computes a generator OSF.

```
1:  Preprocess the problem.
2:  Add clique inequalities.
3:  $U = \{0\}, V = \{e_i : i \in N\}, \alpha =$ optimal dual vector of the LP relaxation,
    $z^{\mathrm{IP}} = -\infty$.
4:  **loop**
5:    Choose a vector $\bar{x} \in V$.
6:    $U = U \cup \{\bar{x}\}, V = V \cup \{\bar{x} + e_i : i \in N$ such that $x_i = 0$, and $y \in$
      $U$ for every $y < \bar{x} + e_i\}$
7:    Update $\alpha$ by solving $D(U, V)$. Let $\pi_0^*$ be the optimal value.
8:    $z^{\mathrm{IP}} = \max\{z^{\mathrm{IP}}, \pi_0^*\}$
9:    If $z^{\mathrm{IP}} = \min\{cx : x \in V, Ax = b\}$, then we have an optimal primal IP
      solution and goto step 11.
10: **end loop**
11: $E = \{i \in N : \alpha a_i \geq c_i\}$ and let $H$ be a subset, with given small cardinality,
    of columns $i$ with the lowest but negative $\alpha a_i - c_i$. Set $S = E \cup H$.
12: Solve (3) with $E := S$ and update $\alpha$ and $E$.
13: $S = S \cup \{i \in N \setminus S : \alpha a_i \leq c_i\}$
14: Further expand $S$.
15: Let $j$ be a column where $\max\{\alpha a_i - c_i : i \in N \setminus S\}$ is attained and set
    $S = S \cup \{j\}, E = E \cup \{j\}$.
16: **loop**
17:   Solve (3) with $A = A^S, S = E \cup H$, and let $\alpha$ be the optimal solution and
      $\eta^*$ the objective value.
18:   **if** $\eta^* = z^{\mathrm{IP}}$ **then**
19:     **if** $S = N$ **then**
20:       $F_\alpha$ is a generator OSF and exit.
21:     **else**
22:       Let $j$ be a column where $\max\{\alpha a_i - c_i : i \in N \setminus S\}$ is attained.
23:       $S = S \cup \{j\}, E = E \cup \{j\}$.
24:     **end if**
25:   **else**
26:     Select a subset $\tilde{E}$ of columns from $S \setminus E$ and set $E = E \cup \tilde{E}$.
27:     $S = S \cup \{i \in N \setminus S : \alpha a_i \leq c_i\}, H = S \setminus E$
28:   **end if**
29: **end loop**
30: Construct a generator OSF of the original problem.
```

Algorithm 1: The algorithm for computing a generator OSF

Given a fixed E, the generator subadditive function with the largest objective value corresponds to a solution of the LP

$$\begin{aligned} &\max \eta && && \text{(3a)}\\ &\eta + \alpha(A^E x - \mathbf{1}) \le c^E x && \text{for all } A^E x \le \mathbf{1}, x \text{ integer} && \text{(3b)}\\ &\alpha a_i \le c_i && i \in H && \text{(3c)}\\ &(\eta, \alpha) \in \mathbb{R} \times \mathbb{R}^m\,. && && \text{(3d)} \end{aligned}$$

Constraints (3b) capture the objective value of the resulting generator subadditive function, (3c) express that for columns in H we must have $\alpha a_i \le c_i$. The basic idea of the algorithm is to iteratively expand E, i.e. move columns from H to E, and then to update α by solving (3). However, solving this LP directly is computational expensive due to a possible large number of constraints (3b). To circumvent this, we approximate generator subadditive functions (first stage of the algorithm) by considering only a subinclusive subset U of $\{x \in \mathbb{Z}^n_+ : A^E x \le \mathbf{1}\}$. We say that a set $U \subseteq \mathbb{R}^n$ is *subinclusive* if for every $x \in U$ and $y \le x$ it follows $y \in U$.

We now work with functions from $\mathbb{R}^n$ to $\mathbb{R}$. Let $S(x) = \{y \in \mathbb{Z}^n_+ : y \le x\}$. Given a subinclusive $U \subseteq \mathbb{Z}^n_+$ and a vector α we define

$$\pi(x) = \alpha A x - \max_{\substack{y \in U \\ y \in S(x)}} \{(\alpha A - c)y\}\,. \qquad (4)$$

We say that π is dual feasible if $\pi(e_i) \le c_i$ for every $i \in N$. If π is dual feasible and subadditive, and x is feasible to the IP, then

$$\pi(x) \le \sum_{i \in N} \pi(e_i) x_i \le cx\,. \qquad (5)$$

Therefore π provides weak duality. It can be shown that these modified subadditive functions still provides strong duality, Klabjan (2003). If $|U|$ is much smaller than $|\{x \in \mathbb{Z}^n_+ : A^E x \le \mathbf{1}\}|$, then π has the advantage over the generator functions since it is easier to evaluate. On the other hand, it is harder to encode π since we need to store α and U. π does solve the IP but however it does not serve the purpose of the generator subadditive functions (e.g. sensitivity analysis) since it is defined on $\mathbb{R}^n$.

We have relaxed our problem to the problem of solving

$$\begin{aligned} &\max_{\pi} \max_{\substack{x \in \mathbb{Z}^n_+ \\ Ax = b}} \pi(x) && \\ &\pi(e_i) \le c_i && i \in N \qquad (6)\\ &\pi \text{ subadditive}\,. && \end{aligned}$$

This optimization problem is solved by using the same framework of gradually expanding U. We start with $U = \emptyset$ and we gradually enlarge it. After every expansion we recompute α so as to maximize the objective value of π. Given U, we define $V = \{x \in \mathbb{Z}^n_+ : x \notin U, S(x) \setminus \{x\} \subseteq U\}$. π is subadditive if and only if $\alpha A x \le cx$ for every $x \in V$. Given U and V, α that gives the largest dual objective value is an optimal solution to

$$D(U,V) \qquad \begin{aligned} &\max \ \pi_0 && \\ &\pi_0 + \alpha(Ay - \mathbf{1}) \le cy && y \in U \qquad (7)\\ &\alpha A x \le cx && x \in V \qquad (8)\\ &\alpha \text{ unrestricted}, \pi_0 \ \text{ unrestricted}\,. && \end{aligned}$$

(7) capture the objective value and (8) assure that π stays subadditive. In other words, we maximize the dual objective value while maintaining subadditivity.

Steps 3-10 of the algorithm consist of solving (6). Steps 5 and 6 expand U and update V. It is easy to check that V satisfies the definition. In step 7 we update α and steps 8 and 9 update the dual and the primal value, respectively.

Steps 11-13 convert the optimal π into an initial generator subadditive function. The main idea is based on $U \subseteq \{x \in \mathbb{Z}_+^n : A^E x \leq \mathbf{1}\}$. To this end, we define E as in step 11. After step 12, F_α is a generator OSF over a subset S of N. If $S = N$, then we have a generator OSF. In Section 3 we elaborate on step 14. In the remaining steps we gradually extend S. Given E and S, in step 17 we find a generator subadditive function with the largest objective value. If we find a generator OSF over S, in steps 22 and 23 we add a new column to S. Otherwise, in steps 26 and 27 we expand E. Finally, at the end, we convert the generator OSF of the preprocessed problem into a generator OSF of the original problem. We call steps 15-29 the *enlarge generator algorithm*.

The bottleneck of stage 3 is in solving (3). This LP is hard to solve due to the large number of constraints (3b). In Section 3.2 we present a row generation algorithm.

2 Preprocessing

In this section we list the preprocessing rules for set partitioning and we give three new rules. All of the rules are needed in Section 4.

Preprocessing is a powerful technique that can substantially reduce computational times, Savelsbergh (1994). Recent research on applying preprocessing to set partitioning is given by Eso (1999) and Borndorfer (1998). In the former work it is shown that the order in which different preprocessing rules are applied does not influence the produced preprocessed problem, while the latter work has a comprehensive list of preprocessing rules and a probabilistic analysis of their usefulness.

2.1 Preprocessing Rules

For a vector x we denote $\text{supp}(x) = \{i : x_i > 0\}$ and e_i is the ith unit vector. We use the following preprocessing rules, where for two sets S_1 and S_2 the notation $S_1 \oplus S_2$ denotes the symmetric difference, i.e. $S_1 \oplus S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$.

P1 (Duplicate column) If $a_j = a_k, j \neq k$ and $a_j \geq c_k$, then we eliminate column j.

P2 (Dominated row) If $\text{supp}(a^i) \subseteq \text{supp}(a^l), i \neq l$, then we

1) eliminate all columns $j \in \text{supp}(a^l) \setminus \text{supp}(a^i)$,
2) eliminate row i.

P3 (Row clique) If there exist row i and column j such that $j \notin \text{supp}(a^i)$ and $\text{supp}(a_j) \cap \text{supp}(a_k) \neq \emptyset$ for every $k \in \text{supp}(a^i)$, then we eliminate column j.

P4 (Symmetric difference) If there exist rows $i, l, r, i \neq l, i \neq r, l \neq r$ such that $\text{supp}(a^r) \oplus \text{supp}(a^l) \subseteq \text{supp}(a^i)$, then

1) eliminate all columns in $\text{supp}(a^r) \oplus \text{supp}(a^l)$,

2) eliminate row l.

P5 (Row singleton) If there exist row i and column j such that $a^i = e_j$, then we

1) eliminate column j, since $x_j = 1$ in an optimal solution,
2) eliminate all columns k with $\text{supp}(a_k) \cap \text{supp}(a_j) \neq \emptyset$,
3) eliminate all rows $l \in \text{supp}(a_j)$.

P6 (Parallel column) If there exist two rows $i, l, i \neq l$ and two columns $j, k, j \neq k$ such that $a^i \oplus a^l = e_j + e_k$, then

1) eliminate row l,
2) eliminate columns j and k if $\text{supp}(a_j) \cap \text{supp}(a_k) \neq \emptyset$,
3) merge columns j and k into a new column $u, a_u = a_j + a_k$ with cost $c_u = c_j + c_k$, otherwise.

P7 (Column singleton) If there exist a column j and a row i such that $a_j = e_i$ and $(\text{supp}(a_p) \cap \text{supp}(a_q)) \setminus \{i\} \neq \emptyset$ for all $\{p, q\} \in \text{supp}(a^i) \setminus \{j\}$, then

1) change $c_k = c_k - c_j$ for every $k \in \text{supp}(a^i)$ and $z^{\text{IP}} = c_j + \bar{z}^{\text{IP}}$, where $\bar{z}^{\text{IP}}$ is the optimal value of the preprocessed problem,
2) eliminate column j,
3) eliminate row i.

P8 (Dominated by symmetric difference) If there exist rows $i, l, r, i \neq l, i \neq r, l \neq r$ such that $\text{supp}(a^i) \subseteq \text{supp}(a^r) \oplus \text{supp}(a^l)$, then we eliminate all columns in $\text{supp}(a^r) \cap \text{supp}(a^l)$.

P9 (Half parallel column) If there exist rows $i, l, i \neq l$ such that for a column j we have $\text{supp}(a^i) \setminus \text{supp}(a^l) = e_j$, then we eliminate all columns $k \in \text{supp}(a^l)$ such that $\text{supp}(a_j) \cap \text{supp}(a_k) \neq \emptyset$.

Borndorfer (1998) lists rules P1-P6 and some additional rules that we did not implement since they are computationally expensive. Rule P7 is an extension of a rule presented in Borndorfer (1998). To show its correctness note that the condition says that column j is a singleton and $\text{supp}(a^i) \setminus \{j\}$ is a clique in $\tilde{A}$, where $\tilde{A}$ is A without row i. Therefore $\sum_{p \in \text{supp}(a^i) \setminus \{j\}} x_p \leq 1$ is a valid inequity for $\{\tilde{A}x \leq \mathbf{1}, x \text{ integer}\}$ and we can make a variable substitution $x_j = \sum_{p \in \text{supp}(a^i) \setminus \{j\}} x_p$. Since in real-world set partitioning instances there are not many column singletons, it is computationally tractable to check that $\text{supp}(a^i) \setminus \{j\}$ is a clique. To check validity of P8, note that if $x_p = 1$ for a $p \in \text{supp}(a^r) \cap \text{supp}(a^l)$, then $x_q = 0$ for all $q \in \text{supp}(a^r) \oplus \text{supp}(a^l)$ and therefore $a^i x = 0$. Similarly we can show rule P9.

We implemented these rules using the techniques from Eso (1999) and Borndorfer (1998). Before we start with preprocessing, we lexicographically sort all of the columns and then, as suggested by Eso (1999), we maintain this order in every reduction. This technique allows an efficient check for duplicate columns. All other rules are implemented as in Borndorfer (1998). Namely, row cliques are checked only for rows with less than 16 elements and rules P6 and P8 are implemented using the intersection technique.

3 Algorithmic Enhancements to the Second Stage

3.1 Extension Heuristic

In this section we elaborate on the enlarge generator algorithm. Steps 15-29 start with a generator OSF over S, where $S \subseteq N$, and we obtain a generator OSF over N. A fast heuristic, called the *LP based expansion heuristic*, based on linear programming is presented in Klabjan (2003), which further extends S by adjusting α. This heuristic is employed in step 14. The resulting generator function is not necessarily a generator OSF over N but in many instances $N \setminus S$ has only a few columns. Here we give a new procedure that can further decrease $N \setminus S$ and it requires solving an LP over a large number of rows but with polynomial separation.

Let F_α be a generator OSF over S and let $C \subseteq N \setminus S$ be such that the objective value of the LP

$$\begin{aligned} \min \ & \sum_{i \in E}(c_i - F_\alpha(a_i))x_i + \sum_{i \in H}(c_i - \alpha a_i)x_i + \sum_{i \in C}(c_i - \alpha a_i)y_i \\ & A^S x + A^C y = \mathbf{1} \\ & 0 \le x, 0 \le y \end{aligned} \tag{9}$$

is greater or equal to 0. It is shown in Klabjan (2003) that $F_{\alpha+\gamma}$ is a generator OSF over $S \cup C$, where γ is the optimal dual vector of (9). This claim is the cornerstone to the LP based expansion heuristic. Here we give a stronger result for set partitioning.

Assume that $C = \{k\}$. The variables in the set partitioning formulation are integer and therefore we should require in (9) that y_k is integer. Note that if we fix $y_k = 0$, then the optimal solution to (9) is integral. The objective value of (9) can be negative even if we require that x and y be integer. If it is nonnegative, then we can expand S based on the following proposition. Denote

$$\begin{aligned} R = \{(u, \bar{u}, w, \pi, \pi_k) \in \mathbb{R}^m_+ \times \mathbb{R}^m_+ \times \mathbb{R}_+ \times \mathbb{R}^{|S|} \times \mathbb{R} : & \\ \pi - uA^S &\le 0 && (10) \\ \pi - \bar{u}A^S &\le 0 && (11) \\ \mathbf{1}u &\le 1 && (12) \\ \pi_k - \bar{u}a_k + w &\le 0 && (13) \\ \mathbf{1}\bar{u} - w &\le 1\} && (14) \end{aligned}$$

and let $(u^j, \bar{u}^j, w^j, \pi^j, \pi^j_k), j \in J$ be the set of all extreme points of R.

Proposition 1. *Let F_α be a generator OSF over S and $k \in N \setminus S$. Let $d_j = 0$ for $j \in E$ and $d_j = c_j - \alpha a_j$ for every $j \in (S \cup \{k\}) \setminus E$ and assume that*

$$\begin{aligned} d_k + \min \ & dx \\ & A^S x = \mathbf{1} - a_k \\ & x \ge 0 \end{aligned} \tag{15}$$

is greater or equal to 0. Let $(\bar{\alpha}, \bar{\lambda}) \in \mathbb{R}^m \times \mathbb{R}^{|J|}_+$ be an optimal dual vector to

$$\begin{aligned} \min dx + d_k y_k & && (16a) \\ A^S x + a_k y_k &= \mathbf{1} && (16b) \\ \pi^j x + \pi^j_k y_k &\le 1 \qquad j \in J && (16c) \\ x \ge 0, y_k &\ge 0\,, && (16d) \end{aligned}$$

where $\bar{\alpha}$ corresponds to (16b)*. Then $F_{\alpha+\bar{\alpha}}$ is a generator OSF over $S \cup \{k\}$.*

Proof. It is easy to see that from disjunctive programming, see e.g. Wolsey (1998), pp. 130-133, it follows that $\text{conv}\{(x, y_k) \in \mathbb{R}_+^{|S|} \times \{0,1\} : A^S x + a_k y_k \leq \mathbf{1}\} = \{(x, y_k) \in \mathbb{R}_+^{|S|} \times \mathbb{R}_+ : \pi^j x + \pi_k^j y_k \leq 1 \text{ for all } j \in J\}$. Note that by selecting, for any row i, vectors $\pi = a^i, u = \bar{u} = e_i, \bar{w} = 0, \pi_k = a_{ik}$, which are in R, we get precisely the rows of $A^S x + a_k y_k \leq \mathbf{1}$.

By definition of π, we have that (16) is equivalent to

$$\begin{aligned} &\min dx + d_k y_k \\ &A^S x + d_k y_k = \mathbf{1} \\ &x \geq 0, y_k \text{ integer}\,. \end{aligned} \tag{17}$$

If $y_k = 0$ in (17), then the objective value is 0. (By complementary slackness, the optimal IP solution gives an objective value 0 and all the coefficients are nonnegative since F_α is dual feasible). If $y_k = 1$, then the objective value is nonnegative by (15). We conclude that the optimal value of (17) is 0 and this implies that the objective value of (16) is 0.

The dual of (16) is

$$\begin{aligned} &\max \mathbf{1}\tilde{\alpha} - \mathbf{1}\lambda \\ &a_i \tilde{\alpha} - \sum_{j \in J} \pi_i^j \lambda_j \leq d_i \qquad\qquad i \in S \cup \{k\} \\ &\tilde{\alpha} \text{ unrestricted}, \lambda \geq 0\,. \end{aligned} \tag{18}$$

Let $(\bar{\alpha}, \bar{\lambda})$ be an optimal solution. Then $\mathbf{1}\bar{\alpha} = \mathbf{1}\bar{\lambda}$.

For simplicity of notation we denote $\gamma = \alpha + \bar{\alpha}$. We need to show that $\max\{(\gamma A^E - c^E x : A^E x \leq \mathbf{1}, x \text{ integer}\} \leq \mathbf{1}\gamma$, where $E \subseteq S \cup \{k\}$ is the generator set of F_γ. Let x be a integer vector with $A^E x \leq \mathbf{1}$ and let $L = \text{supp}(x) \cap S$.

First assume that either $k \notin E$ or $x_k = 0$. We have

$$\sum_{i \in E} (\bar{\alpha} a_i - d_i) x_i = \sum_{i \in L} (\bar{\alpha} a_i - d_i) \leq \sum_{i \in L} \sum_{j \in J} \pi_i^j \bar{\lambda}_j \tag{19}$$

$$= \sum_{j \in J} \bar{\lambda}_j \sum_{i \in L} \pi_i^j \leq \sum_{j \in J} \bar{\lambda}_j \sum_{i \in L} u^j a_i \tag{20}$$

$$= \sum_{j \in J} \bar{\lambda}_j u^j \sum_{i \in L} a_i \leq \sum_{j \in J} \bar{\lambda}_j u^j \mathbf{1} \tag{21}$$

$$\leq \sum_{j \in J} \bar{\lambda}_j = \mathbf{1}\bar{\lambda} = \mathbf{1}\bar{\alpha}\,, \tag{22}$$

where (19) follows from (18). (20) holds because of nonnegativity of $\bar{\lambda}$, (10), and $L \subseteq S$. Inequality (21) follows from $A^E x \leq \mathbf{1}, \bar{\lambda} \geq 0, u^j \geq 0$, and (22) follows from (12).

Now assume that $k \in E$ and $x_k = 1$. Then we have

$$\sum_{i \in E} (\bar{\alpha} a_i - d_i) x_i = \sum_{i \in L} (\bar{\alpha} a_i - d_i) + \bar{\alpha} a_k - d_k \leq \sum_{j \in J} \bar{\lambda}_j \bar{u}^j \sum_{i \in L} a_i + \sum_{j \in J} \pi_k^j \bar{\lambda}_j \tag{23}$$

$$\leq \sum_{j \in J} \bar{\lambda}_j \bar{u}^j (\mathbf{1} - a_k) + \sum_{j \in J} \bar{\lambda}_j (\bar{u}^j a_k - w^j) \tag{24}$$

$$= \sum_{j \in J} \bar{\lambda}_j (\mathbf{1}\bar{u}^j - w^j) \leq \sum_{j \in J} \bar{\lambda}_j = \mathbf{1}\bar{\alpha}\,, \tag{25}$$

where (23) follows as above by using (11) and (18) for $i = k$. Inequality (24) is obtained from (13) and since $A^E x \leq \mathbf{1}, \bar{\lambda} \geq 0, \bar{u}^j \geq 0$. (25) follows from $\bar{\lambda} \geq 0$ and (14).

We conclude that $\sum_{i \in E}(\bar{\alpha} a_i - d_i)x_i \leq \mathbf{1}\bar{\alpha}$. For any integer x with $A^E x \leq \mathbf{1}$ we have

$$\begin{aligned}(\gamma A^E - c^E)x &= (\alpha A^E - c^E + d^E)x + (\bar{\alpha} A^E - d^E)x \\ &\leq \sum_{i \in E}(\alpha a_i - c_i)x_i + \mathbf{1}\bar{\alpha} \leq \mathbf{1}\alpha - z^{\mathrm{IP}} + \mathbf{1}\bar{\alpha} = \mathbf{1}\gamma - z^{\mathrm{IP}} \ ,\end{aligned}$$

where we use $d_i = 0$ for every $i \in E$, and $F_\alpha(\mathbf{1}) = z^{\mathrm{IP}}$. This shows that $F_\gamma(\mathbf{1}) \geq z^{\mathrm{IP}}$, which completes the proof. □

Note that if the optimal value to (9) is negative for every variable in $N \setminus S$, it can still happen that the optimal value to (15) is nonnegative. In Algorithm 1, step 14, we first apply the LP based extension heuristic and then for each variable i in $N \setminus S$ we solve (15). If the objective value of this LP is nonnegative, we solve (16) by row generation, set $\alpha = \alpha + \bar{\alpha}$, and $S = S \cup \{i\}$. We call this additional step the *enhanced expansion heuristic.*

3.2 Solving (3)

To obtain a generator OSF over S, we need to gradually add columns to E, step 26. Each time we add a column to E, we need to update the generator subaddative function by solving (3), step 17. This LP has a large number of rows due to the large number of feasible solutions to $\{A^E x \leq \mathbf{1}, x \text{ integer}\}$ and therefore it is solved by row generation. Given an optimal solution (η^*, α^*) to (3) with only a subset of rows (3b), we have to find the most violated row by solving the set packing problem

$$z^* = \max\{(\alpha^* A^E - c^E)x : A^E x \leq \mathbf{1}, x \text{ integer}\} \ . \tag{26}$$

If $z^* \leq \mathbf{1}\alpha^* - \eta^*$, then (η^*, α^*) is an optimal solution to (3). Otherwise, we add the constraint $\eta + \alpha(A^E x^* - \mathbf{1}) \leq c^E x^*$ to (3), where x^* is an optimal solution to (26), and we repeat the procedure.

Since in general, (26) is NP-hard, solving this problem at every iteration with a commercial branch-and-cut solver to optimality is too time consuming and therefore we solve (26) approximately. It has been observed in the past that if the separation problem is solved approximately, then it is beneficial to add several rows at once. Therefore we have developed a heuristic that generates several "good" solutions. In the context of set covering, Balas and Carrera (1996) assign to each column several greedy estimates and they generate several set covers by randomly selecting an estimate type. We use their idea to generate initial set packing solutions. For simplicity of notation let $d = \alpha^* A^E - c^E$, let t_i be the number of nonzero elements in column i, and let $E = \{i_1, i_2, \ldots, i_{|E|}\}$. For every column $i_j \in E$ we define the greedy estimates

$$\begin{aligned} g^1_{i_j} &= d_{i_j} & g^2_{i_j} &= \frac{d_{i_j}}{t_{i_j}} & g^3_{i_j} &= \frac{d_{i_j}}{1 + \log t_{i_j}} \\ g^4_{i_j} &= \frac{d_{i_j}}{1 + t_{i_j} \log t_{i_j}} & g^5_{i_j} &= \frac{d_{i_j}}{t^2_{i_j}} & g^6_{i_j} &= \frac{\sqrt{d_{i_j}}}{t^2_{i_j}} \ . \end{aligned}$$

Several random set packing solutions are obtained by repeating 4 times the following procedure. We first choose a random estimate type k, i.e. k is a random number between 1 and 6. Next we greedily find a set packing $\bar{x}$ based on the greedy estimates g^k. We find additional set packing solutions by randomly selecting an element to either add to or remove from the set packing $\bar{x}$. To obtain high cost set packing solutions, we want to remove elements with low cost and add elements with high cost. Let X be a random variable that selects an element from E. We define the probability distribution of X as

$$P[X = i_j] = \begin{cases} d_{i_j}/w & \bar{x}_{i_j} = 0 \\ (M - d_{i_j})/w & \bar{x}_{i_j} = 1 \,, \end{cases}$$

where $M = \max\{d_{i_j} : \bar{x}_{i_j} = 1\}$ and $w = \sum_{i_j:\bar{x}_{i_j}=0} d_{i_j} + \sum_{i_j:\bar{x}_{i_j}=1}(M - d_{i_j})$. Note that the probability distribution depends on $\bar{x}$.

To generate alternative set packing solutions resulting from $\bar{x}$, we iterate the following K times, where K is a parameter. Generate a random number l based on X. If $\bar{x}_l = 1$, then we set $\bar{x}_l = 0$. If $\bar{x}_l = 0$ and $\bar{x} + e_l$ is a feasible solution to (26), then we set $\bar{x}_l = 1$. Every time we find a different $\bar{x}$, we add the corresponding constraint (3b) to the current LP if it is violated.

If this row generation procedure does not find a single violated constraint, then we solve (26) to optimality by a branch-and-cut algorithm.

To further reduce the computational time for solving (3) we use complementary slackness, Klabjan (2003). Recall that in the first stage we obtain an optimal solution x^* to the set partitioning problem. By complementary slackness it follows that in a generator OSF F_α we have $\alpha a_i \geq c_i$ for every column $i \in \text{supp}(x^*)$. Therefore we can assume that the columns in $\text{supp}(x^*)$ are in E. If we include them explicitly in E, then (26) becomes hard to solve, leading to high execution times. Instead we consider some of them only implicitly in E. We denote by $E' \subseteq E$ the set of columns that are included explicitly. In each iteration of the enlarge generator algorithm, let $R = \{i \in \text{supp}(x^*) : \text{supp}(a_i) \cap \text{supp}(a_j) = \emptyset \text{ for every } j \in E'\}$. We maintain the property that $E = E' \cup R$. By definition of R, and since x^* is a set packing vector with $d_i \geq 0$ for every $i \in \text{supp}(x^*)$, it follows that z^* from (26) is equal to $\sum_{i\in R} d_i + \max\{dx : A^{E'}x \leq \mathbf{1}, x \text{ integer}\}$. Thus in separation it suffices to solve $\max\{dx : A^{E'}x \leq \mathbf{1}, x \text{ integer}\}$. Every time we add a new column $j \notin \text{supp}(x^*)$ to E, we need to reduce R and expand E'.

This strategy reduces the computational time for solving (3) on average by 50%. In early iterations, when E' is small, the reduction is larger. As E' grows, more columns have to be moved from R to E' and the benefit gradually disappears.

4 Reconstructing the Generator OSF

At the beginning of the algorithm for finding a generator OSF we reduced the problem by preprocessing (step 1). Let

$$\min\{\bar{c}x : \bar{A}x = \mathbf{1}, x \text{ integer}\} \tag{27}$$

be the preprocessed problem. We denote by $\bar{N} = \{1, \ldots, \bar{n}\}, \bar{M} = \{1, \ldots, \bar{m}\}$ the set of all the variables and rows, respectively. A column j and a row i of $\bar{A}$ are denoted by $\bar{a}_j$ and $\bar{a}^i$, respectively. In steps 2-29 the algorithm computes a generator OSF $F_{\bar{\alpha}}$ for this preprocessed problem. Next we show how to construct a generator OSF of the original problem from $F_{\bar{\alpha}}$ (step 30 in the algorithm).

Suppose that (27) is obtained from (1) after a single application of a rule P1-P9. We need to show how to obtain a generator OSF to (1) from a generator OSF $F_{\bar{\alpha}}$ to (27). It suffices to consider only a single application of a rule since the argument can be repeated. For most of the rules the construction is based on the following proposition, where E and H are defined with respect to $\bar{\alpha}$.

Proposition 2. *Let r be a row of $\bar{A}$ and let $\bar{a}_{\bar{n}+1} \in \{0,1\}^{\bar{m}}$ be a new column with cost $\bar{c}_{\bar{n}+1}$ such that $r \notin supp(\bar{a}_{\bar{n}+1})$ and $\sum_{i \in supp(\bar{a}^r)} x_i + x_{\bar{n}+1} \leq 1$ is a valid inequality for $conv\{\bar{A}x + \bar{a}_{\bar{n}+1}x_{\bar{n}+1} \leq \mathbf{1} : (x, x_{\bar{n}+1}) \in \mathbb{Z}_+^{\bar{n}+1}\}$. Then the LP*

$$\begin{aligned}
&\min \mathbf{1}y && && \text{(28a)}\\
&\bar{a}_i y \leq 0 && i \in E \setminus supp(\bar{a}^r) && \text{(28b)}\\
&\bar{a}_i y \leq \bar{c}_i - \bar{\alpha}\bar{a}_i && i \in H \setminus supp(\bar{a}^r) && \text{(28c)}\\
&(\bar{a}_i - \mathbf{1})y \leq 0 && i \in E \cap supp(\bar{a}^r) && \text{(28d)}\\
&(\bar{a}_i - \mathbf{1})y \leq \bar{c}_i - \bar{\alpha}\bar{a}_i && i \in H \cap supp(\bar{a}^r) && \text{(28e)}\\
&(\bar{a}_{\bar{n}+1} - \mathbf{1})y \leq \bar{c}_{\bar{n}+1} - \bar{\alpha}\bar{a}_{\bar{n}+1} && && \text{(28f)}\\
&\mathbf{1}y \geq 0 && && \text{(28g)}
\end{aligned}$$

has an optimal solution y^* and $F_{\bar{\alpha}+y^*}$ is a generator OSF for $\min\{\bar{c}x + \bar{c}_{\bar{n}+1}x_{\bar{n}+1} : \bar{A}x + \bar{a}_{\bar{n}+1}x_{\bar{n}+1} = \mathbf{1} : (x, x_{\bar{n}+1}) \in \mathbb{Z}_+^{\bar{n}+1}\}$.

Proof. Consider the LP

$$\min \sum_{i \in H \cup \{\bar{n}+1\}} (\bar{c}_i - \bar{\alpha}\bar{a}_i)x_i \tag{29}$$

$$\begin{aligned}
\bar{A}x + \bar{a}_{\bar{n}+1}x_{\bar{n}+1} &= \mathbf{1}\\
\bar{a}^r x + x_{\bar{n}+1} &\leq 1 \qquad (30)\\
x \geq 0, x_{\bar{n}+1} &\geq 0\,.
\end{aligned}$$

Since $r \notin \text{supp}(\bar{a}_{\bar{n}+1})$ and $\bar{a}^r x = 1$ for any $(x, x_{\bar{n}+1})$ that is feasible to this LP, it follows that $x_{\bar{n}+1} = 0$. Note that the objective coefficients in (29) are nonnegative, except possibly for the coefficient corresponding to $x_{\bar{n}+1}$. The optimal IP solution to (27) gives a solution of value 0 and therefore the objective value of this LP is 0. If $(\bar{y}, \bar{u})$ is an optimal dual vector, where $\bar{u}$ corresponds to (30), then $\mathbf{1}\bar{y} = \bar{u} \geq 0$. Now it is easy to see that $\bar{y}$ is feasible to (28) and therefore (28) clearly has an optimal solution.

Let y^* be an optimal solution to (28). First note that since y^* satisfies (28c), we have $E(\bar{\alpha} + y^*) \subseteq E \cup \text{supp}(\bar{a}^r) \cup \{\bar{n}+1\}$. Let $\bar{z} = \max\{(\bar{\alpha}\bar{A}^E - \bar{c})x : \bar{A}^E x \leq b, x \in \mathbb{Z}_+^{|E|}\}$. Consider $S \subseteq E(\bar{\alpha}+y^*) \subseteq E \cup \text{supp}(\bar{a}^r) \cup \{\bar{n}+1\} \subseteq \bar{N} \cup \{\bar{n}+1\}$ such that $\sum_{i \in S} \bar{a}_i \leq \mathbf{1}$ and let $C = S \cap (\text{supp}(\bar{a}^r) \cup \{\bar{n}+1\})$. Since $\sum_{i \in \text{supp}(\bar{a}^r)} x_i + x_{\bar{n}+1} \leq 1$ is a valid inequality to $\text{conv}\{Ax + \bar{a}_{\bar{n}+1}x_{\bar{n}+1} \leq \mathbf{1} : (x, x_{\bar{n}+1}) \in \mathbb{Z}_+^{\bar{n}+1}\}$, it follows that $|C| \leq 1$.

Suppose first that $C \subseteq E$. Then we have

$$\begin{aligned}
\sum_{i \in S}(\bar{\alpha}\bar{a}_i + y^*\bar{a}_i - \bar{c}_i) &= \sum_{i \in S \setminus C}(\bar{\alpha}\bar{a}_i + y^*\bar{a}_i - \bar{c}_i) + \sum_{i \in C}(\bar{\alpha}\bar{a}_i + y^*\bar{a}_i - \bar{c}_i)\\
&\leq \sum_{i \in S \setminus C}(\bar{\alpha}\bar{a}_i - \bar{c}_i) + \sum_{i \in C}(\bar{\alpha}\bar{a}_i - \bar{c}_i) + \mathbf{1}y^*\\
&= \sum_{i \in S}(\bar{\alpha}\bar{a}_i - \bar{c}_i) + \mathbf{1}y^* \leq \bar{z} + \mathbf{1}y^*
\end{aligned}$$

since from (28b) it follows $y^*\bar{a}_i \le 0$ for $i \in S \setminus C$, and $\bar{\alpha}\bar{a}_i + y^*\bar{a}_i - \bar{c}_i \le \bar{\alpha}\bar{a}_i - \bar{c}_i + y^*\mathbf{1}$ from (28d) for $i \in C$.

Let now $C \subseteq (H \cap \operatorname{supp}(\bar{a}^r)) \cup \{\bar{n}+1\}$. Then from either (28e) or (28f) it follows that

$$\bar{\alpha}\bar{a}_i + y^*\bar{a}_i - \bar{c}_i \le \bar{\alpha}\bar{a}_i - \bar{c}_i + y^*\mathbf{1} + \bar{c}_i - \bar{\alpha}\bar{a}_i = \mathbf{1}y^*$$

for $i \in C$. Therefore from (28c) and the above inequality we obtain

$$\sum_{i \in S}(\bar{\alpha}\bar{a}_i + y^*\bar{a}_i - \bar{c}_i) \le \sum_{i \in S \setminus C}(\bar{\alpha}\bar{a}_i - \bar{c}_i) + \mathbf{1}y^* \le \bar{z} + \mathbf{1}y^* \, .$$

This shows that $F_{\bar{\alpha}+y^*}(\mathbf{1}) \ge \bar{\alpha}\mathbf{1} - \bar{z}$. Since $\min\{\bar{c}x + \bar{c}_{\bar{n}+1}x_{\bar{n}+1} : \bar{A}x + \bar{a}_{\bar{n}+1}x_{\bar{n}+1} = \mathbf{1} : (x, x_{\bar{n}+1}) \in \mathbb{Z}_+^{\bar{n}+1}\} = \min\{\bar{c}x : \bar{A}x = \mathbf{1} : x \in \mathbb{Z}_+^{\bar{n}}\}$, it follows that $F_{\bar{\alpha}+y^*}$ is the desired generator OSF. Note that the value of any dual feasible subadditive function provides a lower bound on z^{IP}. □

Now we show how to obtain a generator OSF for (1) given a generator OSF $F_{\bar{\alpha}}$ of (27) and a single application of a rule P1-P9 . For each rule we use the notation that appears in the description of the rule (see Section 2).

1. (Duplicate column) The same generator subadditive function can be used.

2. (Dominated row) Define

$$\tilde{\alpha}_j = \begin{cases} \bar{\alpha}_j & j \in \bar{N} - \{i\} \\ \bar{\alpha}_i/2 & j = i \text{ or } j = l \, . \end{cases} \tag{31}$$

Then it is easy to see that $F_{\tilde{\alpha}}$ is a generator OSF for $\min\{\bar{c}x : \begin{bmatrix} \bar{A} \\ \bar{a}^i \end{bmatrix} x = \mathbf{1}, x \in \mathbb{Z}_+^{\bar{n}}\}$, where $\bar{a}^i$ is a duplicate row to l in $\bar{A}$. The eliminated columns from $\operatorname{supp}(a^l) \setminus \operatorname{supp}(a^i)$ do not intersect $\bar{a}^i$ and we can apply Proposition 2 for every eliminated column. For the first column we use $\bar{\alpha} = \tilde{\alpha}$ and for every subsequent column we use $\bar{\alpha}$ obtained after adding the previous column.

3. (Row clique) We can directly apply Proposition 2.

4. (Symmetric difference) Define

$$\tilde{\alpha}_p = \begin{cases} \bar{\alpha}_p & p \in \bar{N} \setminus \{r\} \\ \bar{\alpha}_i/2 & p = r \text{ or } p = l \, . \end{cases}$$

As above $F_{\tilde{\alpha}}$ is a generator OSF for $\min\{\bar{c}x : \begin{bmatrix} \bar{A} \\ \bar{a}^l \end{bmatrix} x = \mathbf{1}, x \in \mathbb{Z}_+^{\bar{n}}\}$, where $\bar{a}^l$ is a duplicate row to r in $\bar{A}$. Denote by $S_1 = \{j \in N : j \in \operatorname{supp}(a^r) \setminus \operatorname{supp}(a^l)\}$ and by $S_2 = \{j \in N : j \in \operatorname{supp}(a^l) \setminus \operatorname{supp}(a^r)\}$. Now we first introduce columns from S_1 by applying Proposition 2 with row l and starting with $\bar{\alpha} = \tilde{\alpha}$. Then for columns in S_2 we apply Proposition 2 with row r.

5. (Row singleton) Let

$$M = \min\{\frac{c_p - \sum_{v \in \operatorname{supp}(a_p), v \le \bar{m}} \bar{\alpha}_v}{|\operatorname{supp}(a_j)|} : \operatorname{supp}(a_p) \cap \operatorname{supp}(a_j) \ne \emptyset, p \ne j\} \, . \tag{32}$$

Note that the minimum is taken over all columns that have been removed except column j. Define

$$\alpha_q = \begin{cases} \bar{\alpha}_q & q \notin \operatorname{supp}(a_j) \\ M & q \in \operatorname{supp}(a^j) \setminus \{i\} \\ c_j - M(|\operatorname{supp}(a_j)| - 1) & q = i\,. \end{cases}$$

For $p = 1, \ldots, \bar{n}$ we have $\alpha a_p = \bar{\alpha}\bar{a}_p$. By the definition of M, for every column p with $\operatorname{supp}(a_p) \cap \operatorname{supp}(a_j) \neq \emptyset, p \neq j$ it follows

$$\begin{aligned} \alpha a_p &= \sum_{v \in \operatorname{supp}(a_p), v \leq \bar{m}} \bar{\alpha}_v + M|\operatorname{supp}(a_p) \cap \operatorname{supp}(a_j)| \\ &\leq \sum_{v \in \operatorname{supp}(a_p), v \leq \bar{m}} \bar{\alpha}_v + M|\operatorname{supp}(a_j)| \leq c_p\,. \end{aligned}$$

For $p = j$ we have $\alpha a_p = c_p$ by the definition of α. This shows that the generator sets of $F_{\bar{\alpha}}$ and F_α are identical and since $\alpha\mathbf{1} = \bar{\alpha}\mathbf{1} + c_j$, it follows that $F_\alpha(\mathbf{1}) = F_{\bar{\alpha}}(\mathbf{1}) + c_j$.

6. (Parallel column)

Case 1.) Let $\operatorname{supp}(a_j) \cap \operatorname{supp}(a_k) \neq \emptyset$. Define $\tilde{\alpha}_q$ as $\bar{\alpha}_q$ if $q \in \bar{N} \setminus \{i, l\}$ and as $\bar{\alpha}_i/2$ for $q = i, q = l$. By applying Proposition 2 twice we get the desired generator OSF.

Case 2.) Let $\operatorname{supp}(a_j) \cap \operatorname{supp}(a_k) = \emptyset$ and $u \in H$. Without loss of generality we assume that $j \in \operatorname{supp}(a^i)$. Then

$$\bar{\alpha}\bar{a}_u = \bar{\alpha}(a_j - e_i) + \bar{\alpha}(a_k - e_l) + \bar{\alpha}_i \leq \bar{c}_u = c_j + c_k\,. \tag{33}$$

Define

$$\alpha_q = \begin{cases} \bar{\alpha}_q & q \neq i, q \neq l \\ c_j - \bar{\alpha}(a_j - e_i) & q = i \\ \bar{\alpha}_i - c_j + \bar{\alpha}(a_j - e_i) & q = l\,. \end{cases}$$

Since $\alpha_i + \alpha_l = \bar{\alpha}_i$, we have $\alpha a_p = \bar{\alpha}\bar{a}_p$ for all $p \in \bar{N}$. We also have

$$\begin{aligned} \alpha a_j &= \alpha(a_j - e_i) + \alpha_i = \bar{\alpha}(a_j - e_i) + \alpha_i = c_j, \\ \alpha a_k &= \alpha(a_k - e_l) + \alpha_l = \bar{\alpha}(a_k - e_l) + \bar{\alpha}_i - c_j + \bar{\alpha}(a_j - e_i) \leq c_k\,, \end{aligned}$$

where the last inequality follows from (33). This shows that the generator sets of $F_{\bar{\alpha}}$ and F_α are identical. Since $\alpha_i + \alpha_l = \bar{\alpha}_i$, it is easy to see that F_α is a generator OSF.

Case 3.) Let $\operatorname{supp}(a_j) \cap \operatorname{supp}(a_k) = \emptyset$ and $u \in E$. Without loss of generality we assume that $j \in \operatorname{supp}(a^i)$. Let $K = \bar{\alpha}(a_j - e_i) + \bar{\alpha}(a_k - e_i) + \bar{\alpha}_i - c_j - c_k > 0$. Next we define

$$\alpha_q = \begin{cases} \bar{\alpha}_q & q \neq i, q \neq l \\ \bar{\alpha}_i - c_k + \bar{\alpha}(a_j - e_i) & q = i \\ \bar{\alpha}_i - c_j + \bar{\alpha}(a_k - e_i) & q = l\,. \end{cases}$$

Note that $\alpha_i + \alpha_l = \bar{\alpha}_i + K$. We need to show that $\max\{(\alpha A^E - c^E) : A^E x \leq \mathbf{1}, x \text{ integer}\} \leq K + \max\{(\bar{\alpha}A^E - c^E) : A^E x \leq \mathbf{1}, x \text{ integer}\}$. If $S \subseteq E$ such that $\sum_{p \in S} a_p \leq \mathbf{1}$, then this can be easily shown by a case analysis if $\sum_{p \in S} a_p$ covers either i or l. Since $\alpha_i + \alpha_l = \bar{\alpha}_i + K$, it follows that F_α is a generator OSF.

4. (Column singleton) Define $\alpha_p = \bar{\alpha}_p$ for all $p \in \bar{N}$ and $\alpha_i = c_j$. It is easy to see that F_α has the desired property.

5. (Dominated by symmetric difference) We use Proposition 2 with row i.

6. (Half parallel column) We use Proposition 2 with row i.

If (27) is the final preprocessed problem obtained after applying rules P1-P9 several times and in several rounds, then we can apply the above procedures several times to construct a generator OSF for (1). We can apply in turn these procedures since each one of them yields a generator subadditive function for the induced set partitioning problem. Every application of a preprocessing rule requires at most solving an LP and therefore the entire procedure is not computationally intensive.

Example. Consider the following set partitioning problem.

$$\bar{c} = (2, -0.5, -2, 3, 4, -3, -1)$$

$$\bar{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The objective value of the LP relaxation is -3 and the optimal IP solution is $x_2 = x_3 = 1$ with $z^{\text{IP}} = -2.5$. $\bar{\alpha} = (-3, -1, 0.5, 2.5)$ yields a generator OSF with $E = \{3, 6, 7\}, \bar{\alpha}\bar{A} - \bar{c} = (-4.5, 0, 1.5, -1.5, -1, 1.5, 1)$.

We show how to construct the generator OSF based on the presented technique for the following two cases.

$$c_1 = (2, -0.5, -2, 3, 4, -3, -1, -5) \qquad c_2 = (2, 2, -0.5, -2, 3, 4, -3, -1, 3)$$

$$A_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

First consider $\min\{c_1 x : A_1 x = \mathbf{1}, x \in \mathbb{Z}^8_+\}$. Row 5 is dominated by row 4. By rule P2 we can remove row 5 and all the columns that are in $\text{supp}(a_5) \setminus \text{supp}(a_4)$, i.e. column 8. The resulting problem is $\min\{\bar{c}x : \bar{A}x = \mathbf{1}, x \in \mathbb{Z}^7_+\}$ with a generator OSF $F_{\bar{\alpha}}$. To obtain a generator OSF for the set partitioning problem given by c_1 and A_1, first define $\tilde{\alpha} = (-3, -1, 0.5, 1.25, 1.25)$ as indicated by (31). We have $H \setminus \text{supp}(a_4) = \{1\}, H \cap \text{supp}(a_4) = \{2, 4, 5\}, E \setminus \text{supp}(a_4) = \{3\}$ and $E \cap \text{supp}(a_4) = \{6, 7\}$. Next we solve (28), which in this case reads

$$\min y1 + y2 + y3 + y4 + y5$$

$$\begin{array}{rrr} y_1 + y_3 \leq 4.5 & -y_2 - y_3 \leq 0 & y_2 + y_3 \leq 0 \\ -y_1 - y_3 \leq 1.5 & -y_1 - y_2 \leq 1 & -y_4 \leq -2.75 \\ -y_3 \leq 0 & -y_2 \leq 0 & y1 + y2 + y3 + y4 + y5 \geq 0\,. \end{array}$$

An optimal solution is $y^* = (-1, 0, 0, 2.75, -1.75)$. Therefore a generator OSF for $\min\{c_1 x : A_1 x = \mathbf{1}, x \in \mathbb{Z}^8_+\}$ is F_α, where $\alpha = \bar{\alpha} + y^* = (-4, -1, 0.5, 4, -0.5)$ and $E = \{3, 6, 7\}$.

Now consider $\min\{c_2x : A_2x = \mathbf{1}, x \in \mathbb{Z}^9_+\}$. Row 6 is a singleton and therefore we fix $x_9 = 1$ and reduce the problem to $\min\{\bar{c}x : \bar{A}x = \mathbf{1}, x \in \mathbb{Z}^7_+\}$. M given by (32) is $M = \min\{4.5/2, -1/2\} = -0.5$, where the minimum is taken over columns 1 and 8. A generator OSF F_α is then given by $\alpha = (-3, -1, 0.5, 2.5, -0.5, 3.5)$ and $E = \{4, 7, 8\}$. Note that in this case the objective value increases by 3 and therefore $\mathbf{1}\bar{\alpha} + 3 = \mathbf{1}\alpha$. □

5 Computational Experiments

The computational experiments were carried out on the set partitioning instances used by Hoffman and Padberg (1993) and Eso (1999). They were performed on an IBM Thinkpad 570 with a 333 MHz Pentium III processor and 196 MB of main memory. We used Visual Studio C++ version 6.0 and ILOG CPLEX 6.5 as the linear programming solver.

The computational results for the instances from Eso (1999) are presented in Table 1 and in Table 2 we show the results for the instances from Hoffman and Padberg (1993). Instances with an integral solution to the LP relaxation are omitted since F_{y^*} is a generator OSF for these instances, where y^* is an optimal dual solution to the LP relaxation. Among the Padberg-Hoffman instances, due to the low physical memory on the notebook, we were not able to solve instances `nw05`, `nw17`, and `us01`. In addition, `nw16` is omitted since it is solved by preprocessing. Only 3 instances from Eso are solved by CPLEX and therefore all remaining `sp` instances are not presented. The `sp11` instance is infeasible and our algorithm finds an unbounded LP (3). All the times are CPU times in seconds. The column "$|N \setminus S|$" shows the cardinality of $N \setminus S$ before we apply the enlarge generator algorithm, i.e. just before step 15, and the last column gives the CPU time for solving a problem by the branch-and-cut solver CPLEX. The "?" sign denotes that we exceeded the time limit of 2 hours.

The problem `aa04` is hard and we were not able to finish stage 1 in the given time limit. For 5 other problems we were able only to solve stage 1 and the expansion heuristic but the enlarge generator algorithm exceeded the time limit. Note that only for the problems with positive number in the "$|N \setminus S|$" column we had to apply the enlarge generator algorithm. It takes a substantially amount of time to apply the enlarge generator algorithm (e.g. `kl01`). The problem `aa05` is interesting since we have a generator OSF over all columns but one and yet it takes more than 2 hours to find a generator OSF over all of the columns.

An important fact from these two tables is the observation that $|E|$ is relatively low for all of the instances. Even instances with more than 10,000 columns have the cardinality of E less than 300. This is encouraging for applying the algorithms for large-scale IPs that are presented in Klabjan (2003).

The overall computational time, which is the sum of the preprocessing time and the time for solving the 2 stages, is acceptable for a methodology that reveals much more information about an IP instance than just an optimal IP solution. It is unreasonable to expect that the computational times would be lower than branch-and-cut computational times since the latter algorithm finds only a primal optimal solution. Nevertheless, these computational results show that it is doable to obtain an optimal subadditive dual function.

Table 3 shows the breakdown of the computational times of the enlarge generator algorithm. The second column shows the number of iterations of this part of the algorithm, i.e. how many times we solve (3), and the third column shows the average execution time for solving this LP. The enlarge generator algorithm

	size		preprocessing			time				CPLEX
name	rows	cols	rows	cols	time	stage 1	stage 2	$\mid E \mid$	$\|N \setminus S\|$	time
sp11	104	2775	63	519	47	438	0	53	0	240
sp2	173	3686	150	3528	59	39	0	223	0	5
sp3	111	1668	73	967	26	3	0	97	0	18

Table 1: Instances from Eso (1999)

	size		preprocessing			time				CPLEX
name	rows	cols	rows	cols	time	stage 1	stage 2	$\mid E \mid$	$\|N \setminus S\|$	time
aa01	823	8904	605	7531	32	3803	?	?	95	723
aa03	825	8627	537	6695	30	25	171	142	0	88
aa04	426	7195	342	6123	17	?	?	?	?	908
aa05	801	8308	521	6236	21	106	?	?	1	97
aa06	646	7292	486	5858	23	52	?	?	5	39
kl01	55	7479	47	5915	49	12	336	100	13	14
kl02	71	36699	69	16542	8	240	?	?	60	118
nw03	59	43749	53	38958	6	29	0	19	0	48
nw04	36	87482	35	46189	16	5757	?	?	998	378
nw06	50	6774	37	5964	45	10	0	58	0	8
nw11	39	8820	28	6436	28	4	0	3	0	6
nw13	51	16043	48	10900	4	9	0	4	0	18
nw18	124	10757	81	7861	90	100	0	292	0	13
nw20	22	685	22	536	0	0	0	13	0	0
nw21	25	577	25	421	0	0	0	4	0	0
nw22	23	619	23	521	0	0	0	7	0	0
nw23	19	711	15	416	0	4	0	52	0	2
nw24	19	1366	19	926	2	1	0	52	0	1
nw25	20	1217	20	844	0	0	0	9	0	0
nw26	23	771	21	468	1	0	0	7	0	1
nw27	22	1355	22	817	3	0	0	4	0	0
nw28	18	1210	18	582	3	0	0	10	0	0
nw29	18	2540	18	2034	6	3	19	50	4	1
nw30	26	2653	26	1877	22	1	0	27	0	1
nw31	26	2662	26	1728	14	1	0	13	0	1
nw32	19	294	17	250	0	2	0	28	0	1
nw33	23	3068	23	2308	26	2	5	16	1	1
nw34	20	899	20	718	2	0	0	3	0	1
nw35	23	1709	23	1191	7	0	0	3	0	1
nw36	20	1783	20	1244	15	5	14	67	12	0
nw37	19	770	19	639	0	0	0	3	0	0
nw38	23	1220	20	722	9	0	0	6	0	1
nw39	25	677	25	565	1	0	0	9	0	0
nw40	19	404	19	336	0	0	0	11	0	0
nw41	17	197	17	177	0	0	0	4	0	0
nw42	23	1079	20	791	4	1	0	30	0	0
nw43	18	1072	17	982	0	0	0	3	0	0
us02	100	13635	44	5197	289	247	0	287	0	17
us04	163	28016	95	4080	53	3	0	25	0	68

Table 2: Instances from Hoffman and Padberg (1993)

is time consuming due to the large number of iteration and a relatively high execution time for row generation. We believe there is room for improvement especially in decreasing the number of iterations but it requires better strategies to select elements from H that are included to E (see Klabjan (2003) for the existing strategy used in the implementation). The last two columns break down the time for solving a single LP (3). The next to last column shows the average number of the separation heuristic calls per LP solving and the last column shows the number of violated rows that are on average found in the separation procedure.

	LP (3) iterations		separation	
name	total no.	average time	total per iter.	average no. added rows
kl01	69	3.6	11	6.3
nw29	18	0.7	6	4.2
nw33	8	0.1	8	1.5
nw36	24	0.5	3.5	3.1

Table 3: Solving (3)

The effects of the expansion heuristics are shown in Table 4. The second column shows the cardinality of $N \setminus S$ after step 13 and the last column shows the same cardinality after step 14. The remaining column shows the cardinality after applying the LP based expansion heuristic. We see that the expansion heuristic significantly reduces $|N \setminus S|$ and it is effective for all of the problems. The enhanced expansion heuristic reduces this size even further but the impact of this heuristic is not so impressive.

name	$\vert N \setminus S\vert$ after stage 1	$\vert N \setminus S\vert$ after LP expansion	$\vert N \setminus S\vert$ after enhanced expansion
kl01	166	18	13
nw29	40	6	4
nw33	3	1	1
nw36	193	13	12
aa01	1875	109	95
aa03	2845	2	0
aa05	254	5	1
aa06	2201	8	5
nw04	7730	998	998

Table 4: Expansion heuristics

5.1 Obtaining all Optimal Solutions

Given a generator OSF F_α, by complementary slackness, all optimal solutions to the IP are found only among the columns i with $\alpha a_i \geq c_i$. These columns have zero reduced cost, which we define for a column i as $c_i - F_\alpha(a_i)$. Given all such columns, we find all optimal solutions to (1) by a naive methodology of enumerating all of them, i.e. solving several IPs. For selected instances we present the computational results in Table 5. x^* denotes the optimal primal solution obtained by Algorithm 1. From the second column we observe that we do not have many columns with zero reduced cost and therefore enumerating all optimal solutions on 0 reduced cost columns is acceptable, which is confirmed by the execution times given in the last column. The fourth column shows the number of columns that are at 1 in all optimal solutions. We observe that larger instances have alternative optimal solutions, which is important for robustness. A decision maker can select the best optimal solution based on alternative objective criteria.

name	no. 0 reduced cost cols	no. cols at 1 in x^*	no. cols at 1 in all	no. optimal solutions	time in seconds
nw36	8	4	4	1	0
nw33	13	5	5	1	0
nw29	12	4	4	1	0
nw04	17	9	9	1	0
kl01	42	13	32	6	2
aa06	312	93	95	2	17
aa05	449	101	103	2	30
aa03	403	102	102	1	28
aa01	255	101	101	1	15

Table 5: All optimal solutions

5.2 Sensitivity Analysis

Given a generator OSF, we can perform sensitivity analysis. We have performed two computational experiments.

In the first one we generated random columns with negative reduced cost. The generated columns reflect the structure of the instances, e.g. the cost is of the same order and the number of nonzeros per column is also of the same order as those of the existing columns in the constraint matrix. On average, among 200,000 randomly generated columns only 10% had negative reduced cost and therefore we know beforehand that the addition of the remaining columns does not decrease the IP objective value. We added, one at the time, each generated column with negative reduced cost to the IP and solve it to find the new optimal value, which is clearly at most z^{IP}. The computational results are presented in Table 6. The objective improvement shows the average ratio $(z^{\mathrm{IP}}-z)/z^{\mathrm{IP}}$, where z is the IP objective value to (1) after adding a column with the negative reduced cost. The number of improvements represents the percentage of the negative reduced cost columns that actually improve the objective value. Note that because of degeneracy after adding such a column the objective value might not decrease. The last two columns show the improvements if the absolute value of the reduced cost is between 25% and 50% of z^{IP} and the remaining two columns if it is between 0% and 25%. With the exception of `nw18`, the problems are degenerate since only approximately 2% of the columns with negative reduced cost actually decrease the objective value. On the other hand, the objective improvements are substantial. With the exception of `nw06`, the objective value improvements are larger when the reduced cost is smaller.

In the second experiment we considered changing right hand sides. If F_α is a generator OSF to (1), then for any row i the value $F_\alpha(\mathbf{1} - e_i)$ gives a lower bound on $\bar{z}_i = \min\{cx : Ax = \mathbf{1} - e_i, x \text{ integer}\}$. This holds since F_α is a subadditive dual feasible function to this IP. We compare this bound with the

	[0, 25%]		[25%, 50%]	
	improvement		improvement	
name	obj. value	count	obj. value	count
nw18	4.70%	5.5%	15.0%	99%
nw06	5.10%	1.0%	3.50%	2.0%
nw24	11.5%	2.5%	13.1%	3.5%
us02	3.30%	1.5%	7.80%	1.5%
kl01	12.5%	3.0%	24.0%	0.5%

Table 6: Addition of negative reduced cost columns

lower bound given by the optimal dual vector to the LP relaxation of (1). Table 7 shows the average over all rows i of values $(\bar{z}_i - z^{\text{LP}})/\bar{z}_i$, where z^{LP} denotes the corresponding lower bound. Except for the `nw24` instance, the bound obtained by the generator OSF is substantially better. Note that $F_\alpha(\mathbf{1} - e_i)$ does not necessarily produce a better lower bound, which is reflected in instance `nw24`.

name	F_α	LP relaxation
nw18	2.7%	17.2%
nw13	0.7%	4.20%
nw24	5.8%	4.80%
us04	2.5%	4.80%

Table 7: Changing right hand sides

References

Balas, E. and Carrera, M. (1996). A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research*, **44**, 875–890.

Borndorfer, R. (1998). *Aspects of Set Packing, Partitioning, and Covering.* Ph.D. thesis, Technical University of Berlin.

Burdet, C. and Johnson, E. (1977). A subadditive approach to solve integer programs. *Annals of Discrete Mathematics*, **1**, 117–144.

Eso, M. (1999). *Parallel Branch and Cut for Set Partitioning.* Ph.D. thesis, Cornell University.

Gomory, R. (1969). Some polyhedra related to combinatorial problems. *Linear Algebra and Applications*, **2**, 451–558.

Hoffman, K. and Padberg, M. (1993). Solving airline crew scheduling problems by branch-and-cut. *Management Science*, **39**, 657–682.

Johnson, E. (1973). Cyclic groups, cutting planes and shortest path. In T. Hu and S. Robinson, editors, *Mathematical Programming*, pages 185–211. Academic Press.

Klabjan, D. (2003). A new subadditive approach to integer programming. Technical report, University of Illinois at Urbana-Champaign. Available from http://netfiles.uiuc.edu/klabjan/white_papers.htm.

Llewellyn, D. and Ryan, J. (1993). A primal dual integer programming algorithm. *Discrete Applied Mathematics*, **45**, 261–275.

Savelsbergh, M. (1994). Preprocessing and probing for mixed integer programming problems. *ORSA Journal on Computing*, **6**, 445–454.

Wolsey, L. (1981). Integer programming duality: Price functions and sensitivity analysis. *Mathematical Programming*, **20**, 173–195.

Wolsey, L. (1998). *Integer Programming*. John Wiley & Sons.