On a class of subadditive duals for the uncapacitated facility location problem

Ehsan Monabbati∗, Hossein Taghizadeh Kakhki∗

∗Department of Mathematics, Ferdowsi University of Mashhad, Azadi Square, Mashhad 91775-1159, Islamic Republic of Iran

Abstract

A family of subadditive functions which contains the optimal solution of subadditive duals for integer programs has been presented by Klabjan (2007). In this paper we present a subadditive dual ascent procedure to find an optimal subadditive dual function based on Klabjan’s generator subadditive function. Then we use the proposed method to solve the so called uncapacitated facility location problem (UFLP). Establishing an explicit formulation for generator subadditive functions in the columns of the coefficient matrix of the UFLP, we derive some variable fixing rules for the problem. Some computational results are also presented.

Keywords: uncapacitated facility location problem, subadditive dual, integer programming, exact methods, variable fixing

1. Introduction

The uncapacitated facility location problem (UFLP) is a well-studied optimization problem. Along with the p-median, p-center, and quadratic assignment problems it is considered to be a “prototype” location problem, and perhaps the most studied one, according to [28]. It can be stated as follows: There are m customers, indexed by \( i \in I := \{1, 2, \ldots, m\} \) which can be served by n potential facilities (servers), indexed by \( j \in J := \{1, 2, \ldots, n\} \). The service cost of serving customer \( i \) by server \( j \) is \( c_{ij} \) (\( \geq 0 \)). Setting up a facility at location \( j \) has a fixed cost of \( f_{j} (> 0) \). Find the minimum cost placement of facilities and assignment of customers to facilities such that each customer is assigned to exactly one server. This problem can be formulated as an integer program by defining \( x_{ij} \) to be the binary variable which is equal to 1 if demand point \( i \) is served by opened facility \( j \), and by defining \( y_{j} \) to be one if facility \( j \) is closed, and zero; otherwise. Adding slack variables \( s_{ij} \), and ignoring the constant term \( \sum_{j\in J} f_{j} \), the integer programming formulation of UFLP can be stated as

∗Corresponding author, Tel: +98-511-8828601, Fax:+98-511-8828609
Email addresses: eh_mn@yahoo.com (Ehsan Monabbati), taghizad@um.ac.ir (Hossein Taghizadeh Kakhki)

Preprint submitted to AMC October 22, 2014
follows:

\[
(UFLP) \quad \min \quad z = \sum_{i \in I, j \in J} c_{ij}x_{ij} - \sum_{j \in J} f_jy_j
\]

s.t.

\[
\sum_{j \in J} x_{ij} = 1, \quad i \in I
\]

\[
x_{ij} + y_j + s_{ij} = 1, \quad i \in I, \quad j \in J
\]

\[
y_j, x_{ij}, s_{ij} \in \{0, 1\}, \quad i \in I, \quad j \in J
\]

There are other mathematical formulations for this NP-hard \cite{28} optimization problem which can be found in many interesting surveys including \cite{28, 9, 29} and \cite{37}. UFLP has several applications in other location or combinatorial optimization problems \cite{28, 9}. In out of location context, it has applications in distribution system design, self-configuration in wireless sensor networks, computational biology and computer vision \cite{30}.

A wide range of solution procedures have been suggested for UFLP in three main areas: approximation algorithms, exact algorithms and heuristics. Due to the vastness of the literature, we concentrate on exact algorithms.

As it mentioned in \cite{31}, to solve most of instances of UFLP, it is better to use a Branch and Bound (BB) method together with a bounding procedure in each node of the enumeration tree. Erlenkotter \cite{11} proposed one of the earlier bounding procedures, called dual-ascent and dual-adjustment, based on the linear programming relaxation of the problem. His methods have been improved later, by some authors \cite{27, 21, 31}. Goldengorin et al. \cite{14} employed the boolean representation of the problem \cite{17}, and proposed some variable fixing rules. They used these rules in the BB framework to exactly solve some medium size UFLP instances. Letchford and Miller \cite{32} solved some extremely large-scale metric instances by joining their aggressive reduction scheme and BB. Hansen et al. \cite{18} used variants of the variable neighborhood search method both for LP relaxation and its dual in a BB algorithm to solve some very large-scale Euclidean instances.

Some bounding procedures are proposed based on Lagrangian relaxation of the constraints \cite{3} or \cite{4}. Galvão and Raggi \cite{13} proposed a 3-stage algorithm in which, they used Lagrangian relaxation bounds by relaxing constraints \cite{3} in second stage. Beasley \cite{2} proposed some Lagrangian-based heuristics for location problems by relaxing constraints \cite{3}. Guignard \cite{15} established a dual-ascent algorithm for solving Lagrangian dual problem obtained by relaxing constraints \cite{4}. Recently, Posta et al. \cite{34} applied a Lagrangian branch and bound as a dual solver, and could solve some unsolved instances of UFLP.

As a general linear integer program, the branch and bound \cite{10}, the branch and cut \cite{7} and variants of decomposition methods are used to solve the problem \cite{1, 36, 20}.

Another approach for solving UFLP, may be to use general integer programming duality; and specifically the subadditive duality. The subadditive dual problem consists of finding a subadditive function, called optimal subadditive function (OSF) which satisfies strong duality and some feasibility conditions.
A discussion of subadditive dual and its properties can be found in [6], [22], [33] and [16]. It is known that subadditive functions are useful for generating facets and valid inequalities (see for example [6]). Unfortunately, constructing the subadditive dual is not easy. Klabjan [25] introduced a subset of subadditive functions which are automatically feasible and contain OSF. He labeled them as generator subadditive functions. These functions are easy to encode and often easy to evaluate. Kalbjan also, established an algorithmic framework to obtain OSF. Using these functions, he established an algorithm for computing OSF for set partitioning problem, and proposed one of the earlier computational results on computing OSF [24].

In this paper, we endeavor to give some computational aspects of generator subadditive functions for computing OSF for the UFLP. In the next section, we briefly recall some properties of generator subadditive functions. Then, we propose a subadditive dual-ascent algorithm for finding OSF for a class of integer programs. In section 3.2, we use the proposed algorithm for solving the UFLP. In section 4, we report some comparisons with a state-of-the-art solver.

2. Subadditive Dual Ascent Method

In this section, we first recall some properties of generator subadditive functions. Then, proving the monotonicity of these functions, we propose a finitely convergent dual-ascent method for solving a class of integer programs.

We begin by some definitions.

**Definition 2.1.** A function $F : \mathbb{R}^m \to \mathbb{R}$ is called *subadditive* on $D \subseteq \mathbb{R}^m$ if for $x, y \in D$ we have $F(x + y) \leq F(x) + F(y)$ with $x + y \in D$.

Consider the following pure integer program (IP):

$$
\begin{align*}
\text{min} & \quad c^T x \\
\text{s.t.} & \quad Ax = b \\
& \quad x \in \mathbb{Z}^n_+ 
\end{align*}
$$

(5)

where $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ and $c \in \mathbb{Z}^n$.

The subadditive dual of IP is defined as:

$$
\max_{F \in \mathcal{F}} \{ F(b) : F(a_{\bullet j}) \leq c_j \},
$$

(6)

where $a_{\bullet j}$ is the $j$th column of $A$ and $\mathcal{F}$ is the set of all subadditive functions $F$ with $F(0) = 0$.

It is known [25] that if $z^{IP}$ is the optimal value of IP [5], then we have

$$
z^{IP} = \max_{F \in \mathcal{F}} \{ F(b) : F(a_{\bullet j}) \leq c_j \}.
$$

(7)

A subadditive function for which the above equality holds is called the optimal subadditive function.
The difficulty in computing a subadditive dual is then how to find \(F\); and in particular an \(F\) that satisfies (7). Klabjan \[25\] provides a partial answer by introducing the concept of a generator subadditive function. Note that from now on, we assume that the coefficient matrix \(A\) in (5) has no negative entries.

**Definition 2.2.** Consider IP (5). Given \(\alpha \in \mathbb{R}^m\), a generator subadditive function \(F_\alpha : \mathbb{R}^m_+ \rightarrow \mathbb{R}\) is defined in \[25\] as

\[
F_\alpha(d) = \alpha^T d - \max \left( (A^E)^T \alpha - c^E \right)^T x \\
\text{s.t. } A^E x \leq d \\
x \in \mathbb{Z}^{|E|}_+
\]

where \(E = E(\alpha) = \{ j : \alpha^T a_{\bullet j} - c_j > 0 \}\), \(A^E\) contains columns of \(A\) corresponding to \(E\) and \(c^E\) is the corresponding sub-vector of \(c\). Similarly, given \(\beta \in \mathbb{R}^m\), a ray generator subadditive function \(\bar{F}_\beta : \mathbb{R}^n_+ \rightarrow \mathbb{R}\) is defined as

\[
\bar{F}_\beta(d) = \beta^T d - \max \beta^T A^E x \\
\text{s.t. } A^E x \leq d \\
x \in \mathbb{Z}^{|E|}_+
\]

Beltran et. al \[3\] state that the generator subadditive functions are closely related to the new Lagrangian-type relaxation they used, called *semi-Lagrangian relaxation*. They employed semi-Lagrangian relaxation to solve \(p\)-median \[3\] and UFLP \[4\].

The following properties of generator subadditive functions are proved in \[25\].

**Lemma 2.3.** For any \(\alpha\) we have

- \(F_\alpha\) is subadditive and \(F_\alpha(0) = 0\),
- \(F_\alpha(a_{\bullet j}) \leq \alpha^T a_{\bullet j} \leq c_j\), for all \(j \notin E\),
- \(F_\alpha(a_{\bullet j}) \leq c_j\), for all \(j \in E\).

This lemma shows that \(F_\alpha\) is feasible for (6). The strong-duality is also proved for generator subadditive functions \[25\].

**Theorem 2.4.** The following primal-dual relations hold

- If IP (5) is feasible, then there exists a generator subadditive function \(F_\alpha\) that satisfies (7).
- IP (5) is infeasible if and only if there exists a ray generator subadditive function.
Thus, finding an OSF is reduced to finding an optimal generator subadditive function. Klabjan [24] presents a column generation procedure which starts from an initial set \( E \). It then tries to iteratively expand \( E \) by adding some variables having negative reduced cost. We outline this method in Algorithm 1.

**Algorithm 1:** Outline of a column generation algorithm for solving (7)

Start with an initial set \( E \).

repeat
  - Find a vector \( \alpha \) for which \( E = E(\alpha) \).
  - Find a variable \( x_j \) outside of \( E \) with \( c_j - F_\alpha(a_j) < 0 \).
  - Add \( x_j \) to \( E \).
until There exists a feasible solution \( x \) with \( F_\alpha(b) = c^T x \);

According to [24], the time consuming parts of the above algorithm is to compute \( F_\alpha(b) \) and \( c_j - F_\alpha(a_j) \), and, to find an \( \alpha \) for which \( E(\alpha) = E \).

Here, we propose an algorithm that works on the vector \( \alpha \) instead of the set \( E \). Indeed, we first prove that the generator subadditive functions are monotone in general. Note here that a similar result holds for the semi-Lagrangian dual function [3].

**Theorem 2.5.** Consider Klabjan subadditive dual of IP [5]. If \( \alpha_2 \geq \alpha_1 \), then \( F_{\alpha_2}(d) \geq F_{\alpha_1}(d) \).

**Proof.** For each \( \alpha \) let \( x(\alpha) \) be one of the maximizers of the problem \( \max \{ (A^E)^T \alpha - c^E)^T x : A^E x \leq d, x \in \mathbb{Z}^{|E|}_+ \} \).

Since \( \alpha_2 \geq \alpha_1 \), then by definition of \( E(\alpha) \), we have \( E(\alpha_1) \subseteq E(\alpha_2) \) and \( \alpha_1^T a_j - c_j \leq 0 \), \( j \in E(\alpha_2) \setminus E(\alpha_1) \). Therefore

\[
F_{\alpha_2}(d) = \alpha_2^T d - \sum_{j \in E(\alpha_2)} (\alpha_2^T a_j - c_j) x_j(\alpha_2)
\]

\[
= \alpha_1^T d - \sum_{j \in E(\alpha_2)} (\alpha_1^T a_j - c_j) x_j(\alpha_2) + \alpha_2^T d - \alpha_1^T d + \sum_{j \in E(\alpha_2)} \alpha_1^T a_j x_j(\alpha_2) - \sum_{j \in E(\alpha_2)} \alpha_2^T a_j x_j(\alpha_2)
\]

\[
\geq \alpha_1^T d - \sum_{j \in E(\alpha_1)} (\alpha_1^T a_j - c_j) x_j(\alpha_2) + (\alpha_1 - \alpha_2)^T \left( \sum_{j \in E(\alpha_2)} a_j x_j(\alpha_2) - d \right)
\]

\[
\geq F_{\alpha_1}(d).
\]

The last inequality is based on the feasibility of \( x(\alpha_2) \) and \( \alpha_2 \geq \alpha_1 \).

**Corollary 2.6.** If \( \alpha_2 \geq \alpha_1 \) is satisfied strictly in some components and \( F_{\alpha_1} \) is not an optimal generator subadditive function, then \( F_{\alpha_2}(d) > F_{\alpha_1}(d) \).

**Proof.** If \( A^{E(\alpha_2)} x(\alpha_2) = d \) then we have

\[
F_{\alpha_2}(d) = \alpha_2^T (d - A^{E(\alpha_2)} x(\alpha_2)) + c^{E(\alpha_2)} x(\alpha_2) = c^{E(\alpha_2)} x(\alpha_2).
\]

Thus \( x(\alpha_2) \) is optimal for IP and \( F_{\alpha_2} \) is the optimal generator subadditive function. Therefore \( F_{\alpha_2}(d) > F_{\alpha_1}(d) \). Otherwise, if \( A^{E(\alpha_2)} x(\alpha_2) \neq d \) then inequality (12) gives the result. \( \square \)
Therefore, one could iteratively increase the components of $\alpha$ to obtain an optimal generator subadditive function. We summarize the general subadditive dual-ascent procedure in Algorithm 2.

**Algorithm 2:** General subadditive dual-ascent algorithm (GSDA)

Start with an initial vector $\alpha$.

repeat

| Update $\alpha$ by increasing its components. |

until There exists a feasible solution $x$ with $F_\alpha(b) = c^T x$;

The following theorem shows that the above dual-ascent algorithm converges to the optimal generator subadditive function under mild conditions.

**Theorem 2.7.** In Algorithm 2, if the size of subproblem $F_\alpha(b)$ is iteratively increased, then, the algorithm is terminated after finitely many iterations.

**Proof.** We prove that $F_\alpha(b)$ equals to the optimal objective function whenever it includes all variables of the original problem. To this end, consider integer program (5) and let $x^\ast$ be its optimal solution. Suppose $\alpha \in \mathbb{R}^m$ is such that $E(\alpha)$ includes all variables. Let $z(x) = (A^T \alpha - c)^T x$ be the objective function of subproblem $F_\alpha(b)$. Then $x^\ast$ is feasible for subproblem $F_\alpha(b)$ and we have

$$z(x^\ast) = (A^T \alpha - c)^T x^\ast = \alpha^T A x^\ast - c^T x^\ast = \alpha^T b - c^T x^\ast.$$ 

Therefore $F_\alpha(b) \geq \alpha^T b - z(x^\ast) = \alpha^T b - (\alpha^T b - c^T x^\ast) = c^T x^\ast = z_{IP} \geq F_\alpha(b)$. That is $x^\ast$ solves $F_\alpha(b)$.

Since the size of $F_\alpha(b)$ is increased iteratively, then, after finite number of iterations, $F_\alpha(b)$ contains all variables of original problem. Therefore there exists an $x$ such that $F_\alpha(b_{UFLP}) = c^T x$. In this case, the algorithm is terminated at optimal solution, since $c^T x$ is an upper bound and $F_\alpha(b_{UFLP})$ is a lower bound on the optimal objective function. \qed

3. Generator subadditive functions for the UFLP

Our main contribution in this section is twofold: First, we give an explicit formulations for computing $F_\alpha(d)$ where $d$ is a column of the UFLP coefficient matrix. Applications of these formulas in constructing valid inequalities and developing variable fixing rules are also shown with two examples. Next, we use general dual-ascent procedure of preceding section for finding an OSF for the UFLP.

We begin by some notations. Given two-indexed variable $x = (x_{ij})_{i \in I, j \in J}$, we use $x$ instead of $[x_{11}, x_{12}, \cdots, x_{1n}, \cdots, x_{m1}, \cdots, x_{mn}]^T$, when there is no ambiguity. For simplicity we use $[a; b]$ to represent the column vector $[a \ b]$. Since the coefficient matrix has $m + mn$ rows, then, we assume that $\alpha = [u; w] = [u_{i=1,\cdots,m}; w_{ij=1,\cdots,m;j=1,\cdots,n}] \in \mathbb{R}^{m+mn}$, where $u$ and $w$ correspond to constraints (2) and (3), respectively. We denote by $a_{ij}^1$, $a_{ij}^2$, $a_j$ and $b_{UFLP}$ columns of the coefficient matrix corresponding to $x_{ij}$, $s_{ij}$, $y_j$ and the right hand side, respectively. Any column of coefficient
matrix could be partitioned into components corresponding to constraints (2), referred to as “w-part”, and components corresponding to constraints (3), referred to as “u-part”. The structure of the coefficient matrix could be easily determined as follows

\[
\begin{array}{ccc}
  a_{ij}^1 & a_{ij}^1 & a_j \\
  w-part & e_{ij} & 0 \\
  w-part & e_{ij} & \sum_k e_{kj}
\end{array}
\]

where \(e_i\) is the \(i\)th column of identity matrix of proper size.

We have \(\alpha^T a_{ij}^1 = u_i + w_{ij}, \alpha^T a_{ij}^2 = w_{ij}\) and \(\alpha^T a_j = \sum_{i\in I} w_{ij}\). Given \(\alpha\), let \(X(\alpha)\), \(X'(\alpha)\), \(Y(\alpha)\) be the sets of some \(x_{ij}\), \(s_{ij}\) and \(y_j\) indices, respectively, defined as

\[
X(\alpha) = \{(i, j) : \alpha^T a_{ij}^1 - c_{ij} > 0\} = \{(i, j) : u_i + w_{ij} - c_{ij} > 0\},
X'(\alpha) = \{(i, j) : \alpha^T a_{ij}^2 > 0\} = \{(i, j) : w_{ij} > 0\},
Y(\alpha) = \{j : \alpha^T a_j + f_j > 0\} = \left\{ j : \sum_{i\in I} w_{ij} + f_j > 0 \right\}.
\]

(14)

In the following when there is no ambiguity, we use \(X\), \(X'\) and \(Y\), instead of \(X(\alpha)\), \(X'(\alpha)\), and \(Y(\alpha)\), respectively.

For an arbitrary set of ordered pairs \(K \subseteq I \times J\) we define

\[
I(K) = \{ i \in I : (i, j) \in K \text{ for some } j \in J \},
J(K) = \{ j \in J : (i, j) \in K \text{ for some } i \in I \},
I_i(K) = \{ j : (i, j) \in K \}, \ i \in I
J_j(K) = \{ i : (i, j) \in K \}, \ j \in J
\]

Given \(\alpha = [u; w] \in \mathbb{R}^{m+n}\) and \(d = [d^u; d^w] \in \mathbb{R}^{m+n}\), we have

\[
F_\alpha(d) = \alpha^T d - \max \sum_{(k,l) \in X} (\alpha^T a_{kl}^1 - c_{kl}) x_{kl} + \sum_{(k,l) \in X'} (\alpha^T a_{kl}^2) s_{kl} + \sum_{l \in Y} (\alpha^T a_l + f_l) y_l
\]
\[\text{s.t.} \sum_{(k,l) \in X} a_{kl}^1 x_{kl} + \sum_{(k,l) \in X'} a_{kl}^2 s_{kl} + \sum_{l \in Y} a_l y_l \leq d \]
\]

\]

(15)

Denote by \(S(\alpha, d)\), \(z_\alpha(x, y, s)\) the feasible set and the objective function of the max problem in the above relation, then we can write (15) as

\[
F_\alpha(d) = \alpha d - \max \{ z_\alpha(x, y, s) : (x, y, s) \in S(\alpha, d) \}.
\]

(16)

Note that \(S(\alpha, d)\) is the set of all variables \([x; s; y]\) that satisfy
\[
\sum_{l \in J_k(X)} x_{kl} \leq d_{kl}^u, \quad k \in I(X) \tag{17}
\]
\[
x_{kl} + s_{kl} + y_l \leq d_{kl}^u, \quad \text{if } (k, l) \in X \cap X', l \in Y \tag{18}
\]
\[
x_{kl} + s_{kl} \leq d_{kl}^u, \quad \text{if } (k, l) \in X \cap X', l \notin Y \tag{19}
\]
\[
x_{kl} + y_l \leq d_{kl}^u, \quad \text{if } (k, l) \in X \setminus X', l \in Y \tag{20}
\]
\[
x_{kl} \leq d_{kl}^u, \quad \text{if } (k, l) \in (X \setminus X') \cap Y \tag{21}
\]
\[
y_l \leq d_{kl}^u, \quad \text{if } (k, l) \notin X \cup X', l \in Y \tag{24}
\]
\[y_l, x_{kl}, s_{kl} \in \mathbb{Z}_+ \tag{25}\]

3.1. Explicit formulations for columns of coefficient matrix

The simple structure of the columns of UFLP coefficient matrix leads to the following result.

**Lemma 3.1.** Given \(\alpha = [u; w] \in \mathbb{R}^{m+mn}\). Let \(X, X'\) and \(Y\) be defined as (14). Then we have

\[
F_{\alpha}(a_{ij}^1) = \begin{cases} \min \{u_i, c_{ij}\} & \text{if } (i, j) \in X \cup X', \\ u_i + w_{ij} & \text{otherwise} \end{cases} \tag{26}
\]
\[
F_{\alpha}(a_{ij}^2) = \begin{cases} 0 & \text{if } (i, j) \in X', \\ w_{ij} & \text{otherwise} \end{cases} \tag{27}
\]
\[
F_{\alpha}(a_{j}) = \sum_{k \in I} w_{kj} - \max \left\{ \sum_{k \in I} w_{kj} + f_j, \sum_{(k, j) \in X'} w_{kj} \right\} \tag{28}
\]

**Proof.** The proof is based on the fact that when in (16), \(d\) equals to the one of UFLP coefficient columns, then most of variables vanish.

For simplicity, let \(z\) be the value of the objective function in the subproblem \(F_{\alpha}(d)\) in (16) and \(z^*\) be its optimal value. To show (26) observe that if \(d = a_{ij}^1\), then \(d_k^u = 1\) only if \(k = i\); and \(d_k^w = 1\) if \((k, l) = (i, j)\). We consider three cases:

**Case 1.** \((i, j) \in X \cap X'\): Since \((i, j) \in X \cap X'\) then the right hand side of inequalities (18) and (19) could be one when \((k, l) = (i, j)\). Thus we have

\[
\sum_{l \in J_i(X)} x_{il} \leq 1,
\]
\[
x_{ij} + y_j + s_{ij} \leq 1, \quad \text{if } j \in Y
\]
\[
x_{ij} + s_{ij} \leq 1. \quad \text{if } j \notin Y
\]

Therefore variables \(x_{il}, l \in J_i(X), s_{ij}\) and \(y_j\) could be nonzero. We prove that the only potentially nonzero variables are \(x_{ij}\) and \(s_{ij}\).

8
Let \( l \in J_i(X) \) and \( l \neq j \). If \( l \) is also in \( Y \) then from inequality (18) or (20) when \( k = i \) we have \( x_{il} = 0 \). If \( l \not\in Y \) then from (19) or (21) we have \( x_{il} = 0 \). This shows that the only nonzero variable among \( x_{il} \)'s, \( l \in J_i(X) \), is \( x_{ij} \).

In a similar manner it can be shown that all \( s_{kl} \), \( (k,l) \neq (i,j) \), are zero.

Now if there is an index \( i \neq k \in I_j(X) \), then from inequality (18) or (20) (when \( l = j \)) we have \( y_j = 0 \). The remaining case is when there is an index \( k \in I \) such that \( (k,j) \not\in X \). In this case from (22) and (24) for \( l = j \) we have \( y_j = 0 \). Therefore \( y_j \) could be nonzero if \( I_j(X) = \{l\} \) and there is no index \( k \) such that \( (k,j) \not\in X \), that is \( |I| = 1 \); i.e., we have a single demand, which is not what we assume.

If \( x_{ij} \) and \( s_{ij} \) nonzero, either \( x_{ij} \) or \( s_{ij} \) is one. If \( x_{ij} = 1 \) then \( z = u_i + w_{ij} - c_{ij} \); Otherwise, \( z = w_{ij} \) and we compute \( z^* \) as \( z^* = \max\{u_i + w_{ij} - c_{ij}, w_{ij}\} \). Therefore \( F_a(a_{ij}^1) = u_i + w_{ij} - z^* = \min\{u_i, c_{ij}\} \).

Case 2. \((i,j) \in X \setminus X' \): In this case only the following inequalities have nonzero right hand sides:

\[
\sum_{l \in J_i(X)} x_{il} \leq 1, \\
x_{ij} + y_j \leq 1, \text{ If } j \in Y \\
x_{ij} \leq 1. \text{ If } j \not\in Y
\]

By omitting the terms corresponding to \( s_{ij} \) from Case 1, we obtain \( z^* = u_i + w_{ij} - c_{ij} \). Since, by definition of \( X \) and \( X' \), \( u_i + w_{ij} - c_{ij} > 0 \) and \( w_{ij} \leq 0 \), then \( c_{ij} \leq u_i \). Thus \( F_a(a_{ij}^1) = u_i + w_{ij} - z^* = c_{ij} = \min\{u_i, c_{ij}\} \).

Case 3. \((i,j) \in X' \setminus X \): Since \( i \not\in X \) then \( k \neq i \) for each \( k \in I(X) \), and from (17) we have \( x_{kl} = 0 \) for all \( (k,l) \in X \). If \( j \in Y \) then we set \( (k,l) = (i,j) \) in (22) and obtain \( s_{ij} + y_j \leq 1 \); Otherwise, if \( j \not\in Y \), (23) implies that \( s_{ij} \leq 1 \). As in Case 1, \( y_j \) could be one only if \( X' = \{(i,j)\} \) and there is no index \( k \) such that \( (k,j) \not\in X \cup X' \) which is impossible. Thus \( z^* = w_{ij} \). On the other hand, from \( w_{ij} > 0 \) and \( u_i + w_{ij} - c_{ij} \leq 0 \) we have \( u_i \leq c_{ij} \). Hence \( F_a(a_{ij}^1) = u_i + w_{ij} - z^* = u_i = \min\{u_i, c_{ij}\} \).

To prove (27) note that if \( d = a_{ij}^2 \) then \( d^w_{kl} = 0 \) for all \( k \in I \), thus the right hand side of inequalities (17) are zero. Consequently all \( x_{kl} \)'s, \( (k,l) \in X \) are zero. Since the \( w \)-part of \( a_{ij}^1 \) and \( a_{ij}^2 \) are equal, then we have cases 1, 2 and 3 with elements of \( X' \) and \( Y \) only. This proves (27).

Finally, to show (28) observe that if \( d = a_j \) then \( d^u = 0_m \) and \( d^w_{kl} = 1 \) if \( l = j \). Again (17) yields \( x_{kl} = 0 \) for all \( (k,l) \in X \). If \( j \in Y \) then from (18) or (22) we have \( s_{kj} + y_j \leq d^w_{kj} = 1 \), and inequalities (20) and (24) imply that \( y_j \leq d^w_{kj} = 1 \). Now if \( y_j = 1 \) then \( s_{kj} = 0 \) for all \( j \), and \( z = \sum_{k \in I} w_{kj} + f_j \); If \( y_j = 0 \), then \( s_{kj} = 1 \) for all \( j \). Thus \( z^* = \max\{\sum_{k \in I} w_{kj} + f_j, \sum_{(k,j) \in X'} w_{kj}\} \). On the other hand, if \( j \not\in Y \), then omitting \( y_j \) from previous argument will yield \( z^* = \sum_{(k,j) \in X'} w_{kj} \). Since \( \sum_{k \in I} w_{kj} + f_j \leq 0 \), then we have \( z^* = \max\{\sum_{k \in I} w_{kj} + f_j, \sum_{(k,j) \in X'} w_{kj}\} \).

\[ \square \]

A direct benefit of the above lemma is its use in obtaining valid inequalities.
It is well known that if $F$ is a subadditive function with $F(0) = 0$, then the inequality
\[ \sum_{j} F(a_{*j})x_j \geq F(b) \quad (29) \]
is a valid inequality for IP (5). Now, if $F = F_\alpha$, for some $\alpha$ then by Lemma 3.1 the left hand side of the corresponding inequality for UFLP is known explicitly.

If $F_\alpha(b_{UFLP})$ is not difficult to compute then one can provide a valid inequality of the following form for UFLP
\[ \sum_{i,j} F_\alpha(a_{1ij})x_{ij} + \sum_{i,j} F_\alpha(a_{2ij})s_{ij} + \sum_{j} F_\alpha(a_j)y_j \geq F_\alpha(b_{UFLP}). \quad (30) \]

In the following example with parameters taken from [12] we demonstrate the use of the above lemma in the construction of a valid inequality.

**Example 3.1.** Consider the UFLP with $m = 8$ customers and $n = 5$ potential servers which has the following cost matrix $C$, and fixed costs vector $f$:
\[
C = \begin{bmatrix}
0 & 90 & 60 & 90 & 50 \\
30 & \infty & 40 & 40 & 0 \\
0 & 50 & 10 & 50 & 10 \\
\infty & 90 & 45 & 30 & 0 \\
10 & 5 & 0 & 15 & 20 \\
\infty & 90 & \infty & 0 & 75 \\
70 & 0 & \infty & 50 & 90 \\
\infty & 45 & 75 & 0 & \infty
\end{bmatrix}, \quad f = [200, 200, 200, 400, 300].
\]

Let $\alpha = [u; w]$ be given as
\[
u = [96, 19, 60, 7, 7, 2, 51]^T, \quad w = \begin{bmatrix}
-116 & -4 & -56 & -26 & -66 \\
-8 & -8 & 10 & -9 & -39 \\
-80 & -9 & -70 & -30 & -70 \\
-9 & 4 & -10 & 3 & -27 \\
7 & -22 & -27 & -9 & 4 \\
6 & 3 & -2 & -27 & 0 \\
-6 & -1 & 0 & 1 & 2 \\
8 & -4 & 2 & -71 & -6
\end{bmatrix}.
\]

Then
\[
Y = \{1, 2, 3, 4, 5\}, \\
X = \{(5, 1), (1, 2), (3, 2), (7, 2), (8, 2)\}, \\
X' = \{(5, 1), (6, 1), (8, 1), (4, 2), (6, 2), (2, 3), (8, 3), (4, 4), (7, 4), (5, 5), (7, 5)\}.
\]

Let $F_1$ and $F_2$ be two $8 \times 5$ matrices with components $F_\alpha(a_{1ij})$ and $F_\alpha(a_{2ij})$, respectively. For example to compute $F_1(1, 1)$, from (26), since $(1, 1) \notin X \cup X'$, then we have $F_\alpha(a_{11}) = \ldots$
u_1 + w_{11} = 96 - 116 = -20. Since (1, 2) \in X \setminus X', then F_1(1, 2) = F_\alpha(a_{12}^\top) = c_{12} = 90. We have

\[ F_1 = \begin{bmatrix} -20 & 90 & 40 & 70 & 30 \\ 11 & 11 & 19 & 10 & -20 \\ -20 & 50 & -10 & 30 & -10 \\ 7 & -15 & -20 & 7 & -20 \\ 7 & 7 & 5 & -20 & 7 \\ -4 & 0 & 2 & 2 & 2 \\ 51 & 45 & 51 & -20 & 45 \end{bmatrix}, \quad F_2 = \begin{bmatrix} -116 & -4 & -56 & -26 & -66 \\ -8 & -8 & 0 & -9 & -39 \\ -80 & -9 & -70 & -30 & -70 \\ -9 & 0 & -10 & 0 & -27 \\ 0 & -22 & -27 & -9 & 0 \\ 0 & 0 & -2 & -27 & 0 \\ -6 & -1 & 0 & 0 & 0 \\ -4 & 0 & -71 & -71 & -6 \end{bmatrix}. \]

Moreover let \( F_3 \in \mathbb{R}^5 \) be a vector with components \( F_\alpha(a_j), j \in J \). Then

\[ F_3 = [-219 \quad -200 \quad -400 \quad -300]^T. \]

The maximization part of \( F_\alpha(b_{\text{UFLP}}) \) takes the following simple form

\[
\begin{align*}
\max & \quad 4x_{51} + 2x_{12} + x_{32} + x_{72} + 2x_{82} + 7s_{51} + 6s_{61} + 8s_{81} + 4s_{42} + 3s_{62} + 10s_{23} + 2s_{83} + 3s_{45} + s_{71} + 4s_{45} + 2s_{75} + 2y_1 + 159y_2 + 47y_3 + 232y_4 + 98y_5 \\
\text{s.t.} & \quad x_{51} \leq 1, \quad x_{12} \leq 1, \quad x_{32} \leq 1 \\
& \quad x_{72} \leq 1, \quad x_{82} \leq 1 \\
& \quad x_{ij} + y_j \leq 1, \quad (i,j) \in X \setminus \{(5,1)\} \\
& \quad s_{ij} + y_j \leq 1, \quad (i,j) \in X’ \setminus \{(5,1)\} \\
& \quad x_{51} + s_{51} + y_i \leq 1 \\
& \quad x_{ij}, s_{ij}, y_j \in \{0,1\} 
\end{align*}
\]

Note that the variable \( s_{ij} \) is appeared exactly in one constraint and its cost coefficient is positive. On the other hand, the problem is in maximization form, thus, the constraints containing \( s_{ij} \) are active at the optimal solution and we can eliminate \( s_{ij} \) from the model; In fact we set

\[ s_{ij} = 1 - y_j, \quad \forall (i,j) \in X \setminus \{(5,1)\}; \quad s_{51} = 1 - x_{51} - y_1. \]

Then we have,

\[
\begin{align*}
\max & \quad -3x_{51} + 2x_{12} + x_{32} + x_{72} + 2x_{82} - 19y_1 + 152y_2 + 35y_3 + 228y_4 + 92y_5 \\
\text{s.t.} & \quad x_{51} + y_1 \leq 1, \quad x_{12} + y_2 \leq 1 \\
& \quad x_{32} + y_2 \leq 1, \quad x_{72} + y_2 \leq 1 \\
& \quad x_{82} + y_2 \leq 1 \\
& \quad x_{ij}, y_j \in \{0,1\} 
\end{align*}
\]

The solution to this problem is easily obtained as \( y_2 = y_3 = y_4 = y_5 = 1, \) all other variables are zero. The objective value is 557. Thus \( F_\alpha(b_{\text{UFLP}}) = \sum_{i,j} u_i + \sum_{i,j} w_{ij} - 557 = -1070 \) which can be used to construct a valid inequality of type (30).
The following variable fixing rule is shown in [23, Theorem 3].

**Lemma 3.2.** Suppose \( \hat{z} \) is an upper bound on the optimal objective value of IP (5). If for some \( k \), \( 0 < \hat{z} - F(b) \leq c_k - F(a_k) \), then there is an optimal solution \( x^* \) with \( x_k^* = 0 \).

The above lemma together with Lemma 3.1 could be used in a variable fixing scheme, as explained in the following example.

**Example 3.2.** Consider the UFLP with \( m = n = 3 \) with the following costs matrix and fixed costs:

\[
C = \begin{bmatrix}
30 & 0 & 0 \\
0 & 30 & 0 \\
0 & 0 & 30
\end{bmatrix}, \quad f = [10 \ 10 \ 10].
\]

Let \( \alpha = [u; w] \) be given as

\[
u = [5 \ 5 \ 5]^T, \quad w = \begin{bmatrix}
10 & 0 & 0 \\
0 & 10 & 0 \\
0 & 0 & 10
\end{bmatrix}.
\]

Consider the upper bound \( \hat{z} = -10 \) which is provided by the feasible solution

\[
x = \begin{bmatrix}
1 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}, \quad y = [0 \ 1 \ 0].
\]

Let us denote the reduced costs of variables \( x, s \) and \( y \) by \( R_x, R_s \) and \( R_y \), respectively. Then, using Lemma 3.1 we have

\[
R_x = \begin{bmatrix}
25 & 0 & 0 \\
0 & 25 & 0 \\
0 & 0 & 25
\end{bmatrix}, \quad R_s = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}, \quad R_y = [0 \ 0 \ 0].
\]

We have \( F_\alpha(b_{UFLP}) = -15 \). Since \( \hat{z} - F_\alpha(b_{UFLP}) = 5 < 25 \), then, by Lemma 3.2, the components of \( R_x \) greater than 5 could be fixed at zero. Thus, at the optimal solution, \( x \) takes the following form:

\[
x = \begin{bmatrix}
0 & \ast & \ast \\
\ast & 0 & \ast \\
\ast & \ast & 0
\end{bmatrix}.
\]

When a good upper bound \( \hat{z} \) is at hand, then one can fix some \( x_{ij} \) and \( s_{ij} \) to zero using Lemma 3.1. This could be used in determining or bounding close centers.

**Lemma 3.3** (Variable fixing rules for the UFLP). Suppose \( \alpha = [u; w] \) and let \( \hat{z}_{UFLP} \) be an upper bound on the optimal objective value, and assume that \( \tau = \hat{z}_{UFLP} - F_\alpha(b_{UFLP}) > 0 \). Then the following statements hold:

(i) If \( \tau < \min\{-F_\alpha(a_{ij}^2), c_{ij} - F_\alpha(a_{ij})\} \), for some \( i \), then, center \( j \) should be closed.

(ii) Suppose the UFLP has a unique optimal solution, \( [x^*; s^*; y^*] \). Then, \( \sum_{j \in J} y_j^* \geq \max_{i \in I}\{|U_i|-1\} \), where \( U_i = \{j : \tau < -F_\alpha(a_{ij}^2)\} \).
Proof. (i) Since \( \tau < -F_\alpha(a^2_{ij}) \) and \( \tau < c_{ij} - F_\alpha(a^1_{ij}) \), then, by Lemma 3.2, there is an optimal solution \([x^*; s^*; y^*] \) with \( s^*_{ij} = x^*_{ij} = 0 \). Therefore \( y^*_j = 1 - x^*_{ij} - s^*_{ij} = 1 \).

(ii) By Lemma 3.2, we have \( s^*_{ij} = 0 \) for all \( i \in I \) and \( j \in U_i \). Thus, \( x^*_{ij} = 1 - y^*_j \). On the other hand, \( \sum_{j \in J} x^*_{ij} = 1 \), thus

\[
1 = \sum_{j \in J} x^*_{ij} \geq \sum_{j \in U_i} x^*_{ij} = \sum_{j \in U_i} (1 - y^*_j) = |U_i| - \sum_{j \in U_i} y^*_j \geq |U_i| - \sum_{j \in J} y^*_j.
\]

Therefore \( \sum_{j \in J} y^*_j \geq |U_i| - 1, i \in I \). Consequently, the number of closed centers could be bounded by \( \sum_{j \in J} y^*_j \geq \max_{i \in I} \{|U_i| - 1\} \).

These rules are used in the algorithm of the next section.

3.2. GSAD for the UFLP

As an application of generator subadditive functions for solving general integer programming problems, we use the algorithm of section 2 to solve the UFLP.

In what follows, we separately demonstrate main parts of subadditive dual-ascent procedure applied for the UFLP.

3.2.1. Subproblems

The structure of subproblems plays an important role in Algorithm 2 since this algorithm needs to compute \( F_\alpha(b) \) in each iteration. The following lemma shows that the subproblem \( F_\alpha(b_{UFLP}) \) is not computationally easier than a UFLP. Thus solving \( F_\alpha(b_{UFLP}) \) may be useful only when the corresponding UFLP is smaller than the original problem.

Lemma 3.4. The integer program (16) is equivalent to a UFLP, when \( d = b_{UFLP} \).

Proof. Since \( s_{kl} \) appears once in the constraints and the problem is a maximization problem, then, in the optimal solution, constraints containing \( s_{kl} \), i.e. (18), (19) and (22), are satisfied as an equality. Using these equalities to eliminate \( s_{kl} \)’s we obtain the following form after some simplifications

\[
\begin{align*}
\max & \quad \sum_{(k,l) \in X} t'_{kl}x_{kl} + \sum_{l \in Y} v'_l y_l \\
\text{s.t.} & \quad \sum_{l \in J_k(x)} x_{kl} \leq 1, \quad k \in I', \\
& \quad x_{kl} + y_l \leq 1, \quad k \in I', l \in Y', \\
& \quad x_{kl}, y_l \in \{0, 1\}
\end{align*}
\]

where \( t'_{kl} \) and \( v'_l \) are the corresponding objective costs and \( I' \subseteq I(X) \) and \( Y' \subseteq Y \) are properly chosen. Adding the slack variables \( z_k \in \{0, 1\}, k \in I' \) gives a problem equivalent to a UFLP, which is known as facility location problem with outliers [8].

\[\square\]
3.2.2. Updating $\alpha$

The main step of Algorithm 2 is to update $\alpha$ by increasing its components.

The following lemma shows that, in UFLP case, $\alpha^{WD} = [u^{WD}; w^{WD}]$ can not be improved by fixing the $u$ or the $w$ part of $\alpha$ alone.

In the following with some abuse of notation by $X$ ($X'$) we mean $X(\alpha^{WD} + \alpha')$ ($X'(\alpha^{WD} + \alpha')$) depending on the context.

Lemma 3.5. Let $\alpha^{WD} = [u^{WD}; w^{WD}]$ be any optimal solution for the weak-dual of UFLP and $\alpha' = [u'; w']$. Then

1. If $w' = 0$ then $F_{\alpha^{WD} + \alpha'}(b_{UFLP}) \leq F_{\alpha^{WD}}(b_{UFLP})$, that is changing only $u$-part of the

   $\alpha^{WD}$ does not improve $F_{\alpha^{WD}}(b_{UFLP})$.

2. If $u' = 0$ and $w'$ is such that $\sum_{i \in I} w'_{ij} \leq 0$, $j \in J$, then $F_{\alpha^{WD} + \alpha'}(b_{UFLP}) \leq F_{\alpha^{WD}}(b_{UFLP})$.

Proof. Let $w' = 0$ then $\sum_{k \in I} w'_{kl} = 0$. Since $\alpha^{WD}$ is weak-dual feasible then $Y(\alpha^{WD} + \alpha') = \emptyset$ and $F_{\alpha^{WD} + \alpha'}(b_{UFLP})$ has the following form

$$
\max \quad z(x) = \sum_{(i,j) \in X} ((u_i^{WD} + u'_i) + (w_{ij}^{WD} + w'_{ij}) - c_{ij})x_{ij}
$$

s.t.

$$
\sum_{l \in J_i(X)} x_{il} \leq 1, \quad i \in I(X)
$$

$$
x_{ij} \leq 1, \quad (i, j) \in X
$$

$$
x_{ij} \in \mathbb{Z}_+
$$

Therefore

$$
z^* = \max_{z \text{ satisfies (31), (32)}} z(x) = \max_{i \in I(X)} \max_{j \in J_i(X)} \{u_i^{WD} + u'_i + w_{ij}^{WD} - c_{ij}\} = \sum_{i \in I(X)} u'_i + \sum_{i \in I(X)} \max_{j \in J_i(X)} \{u_i^{WD} + w_{ij}^0 - c_{ij}\}.
$$

We prove that the second term in the above relation is zero.

Let $i \in I(X)$. Since $\alpha^{WD}$ is optimal for the weak-dual problem, then there is at least an index $j_0 \in J$ such that $u_i^{WD} + w_{ij_0}^{WD} - c_{ij_0} = 0$ (if not, by increasing $u_i^{WD}$ we can get a better solution for the weak-dual problem). On the other hand, by definition of $I(X)$ we have $u'_i > 0$. Therefore

$$(u_i^{WD} + u'_i) + (w_{ij_0}^{WD} + w'_{ij_0}) - c_{ij_0} = (u_i^{WD} + w_{ij_0}^{WD} - c_{ij_0}) + u'_i > 0,$$

that is $j_0 \in J_i(X)$. This shows $\max_{j \in J_i(X)} \{u_i^{WD} + w_{ij}^0 - c_{ij}\} = 0$.

Now

$$
F_{\alpha^{WD} + \alpha'}(b_{UFLP}) - F_{\alpha^{WD}}(b_{UFLP}) = \sum_{i \in I} (u_i^{WD} + u'_i) + \sum_{(i,j)} w_{ij}^{WD}
$$

$$
- \sum_{i \in I(X)} u'_i - \sum_{i \in I} u_i^{WD} - \sum_{(i,j)} w_{ij}^{WD} = \sum_{i \in I(X)} u'_i \leq 0
$$

14
The last inequality is a consequence of the definition of the set $X$.

To prove the second part of the result observe that the max part in $F_{\alpha WD + \alpha'}(b_{UFLP})$ has a nonnegative value, $z^*$. Since $\sum_{(i,j)} w'_{ij} \leq 0$, then

$$F_{\alpha WD + \alpha'}(b_{UFLP}) - F_{\alpha WD}(b_{UFLP}) = \sum_{i \in I} u_i^{WD} + \sum_{(i,j)} (w_{ij}^{WD} + w'_{ij}) - z^*$$

$$- \sum_{i \in I} w_i^{WD} - \sum_{(i,j)} w_{ij}^{WD} = \sum_{(i,j)} w'_{ij} - z^* \leq 0.$$

In view of the above lemma, it seems that all components of $\alpha$ should be increased. In what follows we discuss the effect of $\alpha$ on the size of subproblems.

Consider the following weak-dual of UFLP (1)–(4)

$$\max \sum_{i \in I} u_i + \sum_{i \in I, j \in J} w_{ij} \quad (33)$$

$$\text{s.t. } u_i + w_{ij} \leq c_{ij}, \quad i \in I, j \in J, \quad (34)$$

$$\sum_{i \in I} w_{ij} \leq -f_j, \quad j \in J, \quad (35)$$

$$w_{ij} \leq 0, \quad i \in I, j \in J. \quad (36)$$

If $\alpha = [u; w]$ is a feasible weak-dual solution to the above problem, then $Y(\alpha) = X'(\alpha) = X(\alpha) = \emptyset$. Thus $F_{\alpha}(b_{UFLP}) = \alpha^T b_{UFLP}$. Therefore, the best generator subadditive function corresponding to weak-dual feasible solutions is the weak-dual optimal solution, $\alpha^{WD} = [u^{WD}; w^{WD}]$. Consequently, a vector $\alpha$ better than $\alpha^{WD}$ should violate some weak-dual constraints.

Increasing all components of $\alpha^{WD}$ causes some violations in constraints (34), (35) and (36) of weak-dual. Violation of constraints (36) puts some $s_{ij}$'s in to subproblem. As we mentioned in Lemma 3.4 this does not alter the size of the subproblem.

The following lemma shows that we can assume that constraints (35) for all $j \in J$, are violated and, therefore constraints (34) are the only constraints that matters for the size of subproblems.

**Lemma 3.6.** Weak-dual problem (33)–(36) has an optimal solution with $\sum_{i \in I} w_{ij} + f_j = 0$ for all $j \in J$.

**Proof.** Using the changing variable $y'_j = 1 - y_j$, linear programming relaxation of UFLP and its dual could be written as

$$(P) \quad \min \sum_{i \in I, j \in J} c_{ij} x_{ij} + \sum_{j \in J} f_j y'_j \quad (D) \quad \max \sum_{i \in I} u_i$$

$$\text{s.t. } \sum_{j \in J} x_{ij} = 1, i \in I$$

$$x_{ij}, y'_j \geq 0, i \in I, j \in J \quad \text{s.t. } u_i + w_{ij} \leq c_{ij}, i \in I, j \in J$$

$$w_{ij} \leq 0, u_i \text{ free}, i \in I, j \in J$$
Let \((u^*, w^*)\) be an optimal solution of (D). Define \((\bar{u}, \bar{w})\) as follows:

\[
\bar{u} = u^*,
\]
\[
\bar{w}_{ij} = w^*_{ij} - \left( \sum_{i \in I} w^*_{ij} + f_j \right), j \in J
\]
\[
\bar{w}_{ij} = w^*_{ij}, i \in I, j \in J, i \neq 1.
\]

Then \((\bar{u}, \bar{w})\) is feasible for weak-dual (34)–(36) and we have

\[
\sum_{i \in I} \bar{u}_i + \sum_{i \in I, j \in J} \bar{w}_{ij} = \sum_{i \in I} \bar{u}_i - \sum_{j \in J} f_j
\]
\[
= \text{optimal value of (D)} - \sum_{j \in J} f_j
\]
\[
= \text{optimal value of (P)} - \sum_{j \in J} f_j
\]
\[
= \text{optimal value of LP relaxation of UFLP}
\]
\[
= \text{optimal value of weak-dual (33)–(36)}.
\]

Now, for \(i \in I, j \in J\) let \(t_{ij}^{WD} = u^{WD}_i + w^{WD}_{ij} - c_{ij}\). Suppose \(t_{ij}^{new}\) is the infeasibility of constraints (34) after increasing all components of \(\alpha^{WD}\) by a positive scalar \(\delta\), i.e. \(t_{ij}^{new} = t_{ij}^{WD} + 2\delta\). By definition of \(X(\alpha)\), positive components of \(t_{ij}^{new}\) add variables \(x_{ij}\) to the subproblem. Therefore one can control the dimension of the subproblems by choosing appropriate value for \(\delta\) such that at most a predetermined fraction, \(\rho\), of \(t_{ij}^{new}\) components are positive.

Algorithm 3 presents the subadditive dual ascent procedure for UFLP. This procedure iteratively increases \(\alpha\) and solves \(F_\alpha(b_{UFLP})\). We use variable fixing rules of Lemma 3.3 in each iteration to reduce the size of subproblems. It required an upper bound on the optimal objective value. To do so, the open centers provided by the solution of \(F_\alpha(b_{UFLP})\)
is completed by assigning each demand to its nearest open center.

**Algorithm 3:** Subadditive Dual-Ascent algorithm (SDA)

1. Initialize \( \alpha = [u; w] \) to a feasible solution of weak-dual of UFLP. A good choice is a weak-dual optimal solution satisfied in Lemma 3.6.
2. Choose \( \rho \) such that \( \rho > \frac{Z}{mn} \), where \( Z \) is the number of components for which \( u_i + w_{ij} - c_{ij} = 0 \).
3. repeat
   4. Find \( \delta > 0 \) such that at most 100\( \rho \) percent of infeasibilities of constraints (34), \( t^{new} \), are positive.
   5. Set \( \alpha \leftarrow \alpha + \delta \).
   6. Find optimal solution \( x(\alpha), y(\alpha) \) of \( F_\alpha(b_{UFLP}) \).
   7. Compute the upper bound \( \hat{z} \) by completing \( y(\alpha) \).
   8. Update \( \rho \): \( \rho \leftarrow 1.1 \rho \).
9. until \( \hat{z} - F_\alpha(b_{UFLP}) = 0 \);

By the update rule of \( \rho \), the size of subproblem \( F_\alpha(b_{UFLP}) \) is strictly increased in each iteration, thus the algorithm is convergent by Theorem 2.7.

4. Computational Results

In this section we present numerical results of running Algorithm 3 on five classes of UFLP benchmark instances. All instances are taken from UFLLIB [19]. CPLEX mixed-integer solver, with default settings, is used to solve subproblems within MATLAB code. All runs are performed on an Intel® Core 2 Duo CPU 2.00 GHz, with 3 GB of RAM.

To produce an initial weak-dual feasible solution in Algorithm 3 for problems with up to 100 clients we have used CPLEX and for larger problems we have used a MATLAB implementation of the fast ascent procedure of Letchford and Miller [31].

We summarize the specifications of the test instances in Table 1. For all instances we initialize \( \rho \) as \( \rho = \theta \frac{Z}{mn} \) where \( \theta \) is depend on the number of potential facilities, \( m \), as follows

\[
\theta = \begin{cases} 
1.1 & m \leq 100 \\
1.07 & 100 < m \leq 500 \\
1.05 & m > 500 
\end{cases}
\]

In what follows, we briefly report the details of computational results.

1. **Bilde-Krarup** The results are presented in Table 2 for this collection. In this table column 1 is the running time of the Algorithm 3 which is referred to as “SDA”. The running time contains the initialization time of \( \alpha \). Column 2 is the running time of the CPLEX. The last column is the number of intermediate subproblems. All reported numbers are averages of the instances in each class.

It is observed that in most problems running time of Algorithm 3 is greater than CPLEX. However one may accept this inefficiency at the cost of performing sensitivity analysis. In fact at the optimal generator subadditive function we can compute reduced costs using Lemma 3.1 Then it is possible to find a range of service costs for which the optimal solution does not change.
Table 1: Summary of the specifications of the test instances.

<table>
<thead>
<tr>
<th>Class name</th>
<th>Subclass</th>
<th>Size</th>
<th>No. of instances</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bilde-Krarup</td>
<td>B</td>
<td>50 × 100</td>
<td>10</td>
<td>[5]</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>50 × 100</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D*</td>
<td>30 × 80</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E*</td>
<td>50 × 100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Galvão-Raggi</td>
<td>GR50</td>
<td>50 × 50</td>
<td>10</td>
<td>[13]</td>
</tr>
<tr>
<td></td>
<td>GR100</td>
<td>100 × 100</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GR150</td>
<td>150 × 150</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GR200</td>
<td>200 × 200</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Janáček and Buzna</td>
<td>K90-411</td>
<td>411 × 457</td>
<td>9</td>
<td>[21]</td>
</tr>
<tr>
<td></td>
<td>K90-457</td>
<td>457 × 457</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>k-median</td>
<td>MED150*</td>
<td>150 × 150</td>
<td>3</td>
<td>[35]</td>
</tr>
<tr>
<td></td>
<td>MED200*</td>
<td>200 × 200</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MED250*</td>
<td>250 × 250</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MED500*</td>
<td>500 × 500</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MED1000*</td>
<td>1000 × 1000</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>ORLIB</td>
<td>cap70</td>
<td>16 × 50</td>
<td>4</td>
<td>[2]</td>
</tr>
<tr>
<td></td>
<td>cap100</td>
<td>25 × 50</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cap130</td>
<td>50 × 50</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>capa</td>
<td>100 × 1000</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>capb</td>
<td>100 × 1000</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>capc</td>
<td>100 × 1000</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Kochetov-Ivanenko</td>
<td>BPC</td>
<td>128 × 128</td>
<td>32</td>
<td>[26]</td>
</tr>
<tr>
<td></td>
<td>CB</td>
<td>144 × 144</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FPP11</td>
<td>133 × 133</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FPP17</td>
<td>307 × 307</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
Table 2: Results for Bilde-Krarup instances.

<table>
<thead>
<tr>
<th>Subclass</th>
<th>SDA (CPU Time s)</th>
<th>CPLEX (CPU Time s)</th>
<th>Subproblems</th>
<th>Subclass</th>
<th>SDA (CPU Time s)</th>
<th>CPLEX (CPU Time s)</th>
<th>Subproblems</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1.978</td>
<td>1.642</td>
<td>2.1</td>
<td>D10</td>
<td>5.525</td>
<td>2.416</td>
<td>2.7</td>
</tr>
<tr>
<td>C</td>
<td>9.194</td>
<td>9.031</td>
<td>2.9</td>
<td>E1</td>
<td>9.840</td>
<td>9.422</td>
<td>3.0</td>
</tr>
<tr>
<td>D1</td>
<td>3.847</td>
<td>3.844</td>
<td>2.8</td>
<td>E2</td>
<td>18.30</td>
<td>17.84</td>
<td>3.3</td>
</tr>
<tr>
<td>D2</td>
<td>4.919</td>
<td>4.445</td>
<td>3.0</td>
<td>E3</td>
<td>20.14</td>
<td>16.33</td>
<td>3.2</td>
</tr>
<tr>
<td>D3</td>
<td>5.428</td>
<td>3.870</td>
<td>3.0</td>
<td>E4</td>
<td>20.97</td>
<td>15.49</td>
<td>3.1</td>
</tr>
<tr>
<td>D4</td>
<td>5.810</td>
<td>3.415</td>
<td>3.0</td>
<td>E5</td>
<td>20.55</td>
<td>13.12</td>
<td>3.2</td>
</tr>
<tr>
<td>D5</td>
<td>5.326</td>
<td>2.745</td>
<td>3.1</td>
<td>E6</td>
<td>23.25</td>
<td>13.45</td>
<td>3.1</td>
</tr>
<tr>
<td>D6</td>
<td>7.372</td>
<td>3.058</td>
<td>3.1</td>
<td>E7</td>
<td>23.90</td>
<td>14.14</td>
<td>3.0</td>
</tr>
<tr>
<td>D7</td>
<td>5.604</td>
<td>2.770</td>
<td>3.0</td>
<td>E8</td>
<td>27.03</td>
<td>13.45</td>
<td>3.1</td>
</tr>
<tr>
<td>D8</td>
<td>5.873</td>
<td>2.722</td>
<td>2.9</td>
<td>E9</td>
<td>24.19</td>
<td>15.53</td>
<td>3.0</td>
</tr>
<tr>
<td>D9</td>
<td>5.809</td>
<td>2.563</td>
<td>2.9</td>
<td>E10</td>
<td>22.82</td>
<td>12.41</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Table 3: Results for Galvão-Raggi and Janáček-Buzna instances.

<table>
<thead>
<tr>
<th>Subclass</th>
<th>SDA (CPU Time s)</th>
<th>CPLEX (CPU Time s)</th>
<th>Subproblems</th>
<th>Subclass</th>
<th>SDA (CPU Time s)</th>
<th>CPLEX (CPU Time s)</th>
<th>Subproblems</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR50</td>
<td>0.021</td>
<td>0.069</td>
<td>1</td>
<td>K90-411-8</td>
<td>14.06</td>
<td>11.81</td>
<td>12</td>
</tr>
<tr>
<td>GR70</td>
<td>0.056</td>
<td>0.112</td>
<td>1</td>
<td>K90-411-9</td>
<td>9.037</td>
<td>40.83</td>
<td>9</td>
</tr>
<tr>
<td>GR100</td>
<td>0.070</td>
<td>0.189</td>
<td>1</td>
<td>K90-457-1</td>
<td>2.699</td>
<td>out of mem.</td>
<td>4</td>
</tr>
<tr>
<td>GR150</td>
<td>0.131</td>
<td>0.443</td>
<td>1</td>
<td>K90-457-2</td>
<td>70.41</td>
<td>out of mem.</td>
<td>12</td>
</tr>
<tr>
<td>GR200</td>
<td>0.189</td>
<td>1.102</td>
<td>1</td>
<td>K90-457-3</td>
<td>6.948</td>
<td>out of mem.</td>
<td>10</td>
</tr>
<tr>
<td>K90-411-1</td>
<td>1.936</td>
<td>11.84</td>
<td>5</td>
<td>K90-457-4</td>
<td>39.74</td>
<td>out of mem.</td>
<td>8</td>
</tr>
<tr>
<td>K90-411-2</td>
<td>2.208</td>
<td>11.81</td>
<td>7</td>
<td>K90-457-5</td>
<td>8.905</td>
<td>out of mem.</td>
<td>9</td>
</tr>
<tr>
<td>K90-411-3</td>
<td>3.505</td>
<td>11.78</td>
<td>8</td>
<td>K90-457-6</td>
<td>17.13</td>
<td>out of mem.</td>
<td>11</td>
</tr>
<tr>
<td>K90-411-4</td>
<td>4.022</td>
<td>11.70</td>
<td>8</td>
<td>K90-457-7</td>
<td>6.827</td>
<td>out of mem.</td>
<td>9</td>
</tr>
<tr>
<td>K90-411-5</td>
<td>5.501</td>
<td>11.95</td>
<td>9</td>
<td>K90-457-8</td>
<td>4.846</td>
<td>out of mem.</td>
<td>8</td>
</tr>
<tr>
<td>K90-411-6</td>
<td>4.700</td>
<td>11.97</td>
<td>7</td>
<td>K90-457-9</td>
<td>4.096</td>
<td>out of mem.</td>
<td>7</td>
</tr>
</tbody>
</table>

2. Galvão-Raggi and Janáček-Buzna Table 3 shows the results for these classes. It seems that GR instances are easy both for CPLEX and SDA. Furthermore, SDA finds optimal solutions quicker than CPLEX. In most instances of K90-411, SDA outperforms CPLEX. Larger problems of this type, K90-457, have become too large for CPLEX in our machine.

3. k-median The results are summarized in Table 4. It can be seen that for these problems, except for type I, SDA provides optimal solutions in reasonable times.

4. ORLIB As can be seen in Table 5, CPLEX reasonably solves large instances. Smaller problems could be solved efficiently with both SDA and CPLEX.

5. Kochetov-Ivanenko Table 6 shows the considerable results for these instances. As can be observed, Algorithm 3 solves more challenging subclass FPP17 in one iteration, whereas CPLEX could not do this in more than ten times that time.
Table 4: Results for $k$-median instances.

<table>
<thead>
<tr>
<th>Subclass</th>
<th>CPU Time (s) SDA</th>
<th>Aver. No. of Subproblems</th>
<th>CPU Time (s) CPLEX</th>
<th>Aver. No. of Subproblems</th>
</tr>
</thead>
<tbody>
<tr>
<td>MED150I</td>
<td>2.423</td>
<td>2.281</td>
<td>5</td>
<td>1.569</td>
</tr>
<tr>
<td>MED150II</td>
<td>1.138</td>
<td>5.016</td>
<td>4</td>
<td>5.569</td>
</tr>
<tr>
<td>MED150III</td>
<td>0.533</td>
<td>2.250</td>
<td>1</td>
<td>12.40</td>
</tr>
<tr>
<td>MED200I</td>
<td>5.822</td>
<td>5.203</td>
<td>6</td>
<td>8.996</td>
</tr>
<tr>
<td>MED200II</td>
<td>2.000</td>
<td>5.032</td>
<td>5</td>
<td>out of mem.</td>
</tr>
<tr>
<td>MED200III</td>
<td>0.9805</td>
<td>5.078</td>
<td>1</td>
<td>105.5</td>
</tr>
<tr>
<td>MED250I</td>
<td>14.66</td>
<td>9.578</td>
<td>7</td>
<td>54.51</td>
</tr>
<tr>
<td>MED250II</td>
<td>3.265</td>
<td>9.235</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Results for ORLIB instances.

<table>
<thead>
<tr>
<th>Subclass</th>
<th>CPU Time (s) SDA</th>
<th>Aver. No. of Subproblems</th>
</tr>
</thead>
<tbody>
<tr>
<td>cap70</td>
<td>0.024</td>
<td>0.062</td>
</tr>
<tr>
<td>cap100</td>
<td>0.027</td>
<td>0.062</td>
</tr>
<tr>
<td>cap130</td>
<td>0.042</td>
<td>0.097</td>
</tr>
<tr>
<td>capa</td>
<td>59.74</td>
<td>21.17</td>
</tr>
<tr>
<td>capb</td>
<td>38.73</td>
<td>23.11</td>
</tr>
<tr>
<td>capc</td>
<td>145.4</td>
<td>91.64</td>
</tr>
</tbody>
</table>

Table 6: Results for Kochetov-Ivanenko collection. The averages for FPP11 and FPP17 instances are over 30 problems.

<table>
<thead>
<tr>
<th>Subclass</th>
<th>CPU Time (s) SDA</th>
<th>Aver. No. of Subproblems</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPC</td>
<td>0.330</td>
<td>2.003</td>
</tr>
<tr>
<td>CB</td>
<td>0.383</td>
<td>2.603</td>
</tr>
<tr>
<td>FPP11</td>
<td>1.477</td>
<td>32.97</td>
</tr>
<tr>
<td>FPP17</td>
<td>34.15</td>
<td>&gt; 400</td>
</tr>
</tbody>
</table>
5. Conclusions

In this paper some theoretical and computational properties of Klabjan generator subadditive functions for the UFLP are considered. We showed that each subproblem in computing $F_\alpha(b_{UFLP})$ is again a UFLP. We also proposed explicit formulations for $F_\alpha(d)$ where $d$ is a column of UFLP coefficient matrix. These formulas were used to develop variable fixing strategies and to find reduced costs, which can then be used in sensitivity analysis.

Using the generator subadditive functions we presented an exact subadditive dual ascent procedure to solve UFLP to optimality. This procedure could be used for general integer programs with nonnegative coefficient matrix.

Finally, we tested the proposed algorithm on five groups of UFLP instances. Our computational experiments showed that the algorithm provides the optimal generator subadditive function, and for most of the problems considered, performs improvements over the state of the art solver.

References