A VARIATIONAL INEQUALITY APPROACH FOR CONstrained MULTIFACILITY WEBER PROBLEM UNDER GAUGE

JIANLIN JIANG AND SHUN ZHANG

Department of Mathematics, College of Science
Nanjing University of Aeronautics and Astronautics
29 Yudao Street, Qinhui District, Nanjing 210016, China

SU ZHANG*

Department of Financial Management, Business School, Nankai University
94 Weijin Road, Nankai District, Tianjin 300071, China

JIE WEN

Department of Mathematics, College of Science
Nanjing University of Aeronautics and Astronautics
29 Yudao Street, Qinhui District, Nanjing 210016, China

(Communicated by Jie Sun)

Abstract. The classical multifacility Weber problem (MFWP) is one of the most important models in facility location. This paper considers more general and practical case of MFWP called constrained multifacility Weber problem (CMFWP), in which the gauge is used to measure distances and locational constraints are imposed to facilities. In particular, we develop a variational inequality approach for solving it. The CMFWP is reformulated into a linear variational inequality, whose special structures lead to new projection-type methods. Global convergence of the projection-type methods is proved under mild assumptions. Some preliminary numerical results are reported which verify the effectiveness of proposed methods.

1. Introduction. The issue of the location of new facilities in modern logistics system is a strategic sense. The past decades have witnessed a fruitful development in the field of location theory as well as its wide applications in diverse fields. For a comprehensive review, see e.g. [6, 18].

The classical multifacility Weber problem (MFWP) is to find the locations of \( m \) new facilities in order to minimize the sum of transportation costs between \( n \)
customers and $m$ facilities and transportation costs between $m$ facilities. More specifically, the mathematical model of MFWP is the following:

\[
\text{MFWP: } \min_{x_i \in \mathbb{R}^2} \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \|x_i - a_j\| + \sum_{i=1}^{m-1} \sum_{l=i+1}^{m} v_{il} \|x_i - x_l\|,
\]

(1)

where

1) $a_j \in \mathbb{R}^2$ is the location of the $j$-th customer, $j = 1, 2, \cdots, n$;
2) $x_i \in \mathbb{R}^2$ is the location of the $i$-th facility to be determined, $i = 1, 2, \cdots, m$;
3) $w_{ij} > 0$ is the weight associated with the transportation cost between the $i$-th facility and the $j$-th customer;
4) $v_{il} > 0$ is the weight associated with the transportation cost between the $i$-th facility and the $l$-th facility;
5) $\| \cdot \|$ is the Euclidean distance.

MFWP is one of the most important models in the area of facility location. Due to its wide applications in operations research, marketing, urban planning etc., MFWP has attracted wide attentions and many authors are devoted to both promoting its theoretical development and presenting effective numerical algorithms, see, e.g. [5, 6, 18, 20, 25].

In the literature, the distances of facility location are usually measured by norms such as $l_p$-norms and block norms. [16, 17] discussed the approximations about the weighted $l_p$-norms based on statistical evidence. [30, 31] investigated the use of block norms to obtain good approximation of actual distances. The symmetry property of the norm assures that the distance from one point to another is always equal to the distance back. Nevertheless, [33] has highlighted the fact that in numerous real situations the symmetry of the distance is violated. The examples include transportation in rush-hour traffic, flight in the presence of wind, navigation in the presence of currents, transportation on an inclined terrain, etc. In spite of the pioneering work of [33], the asymmetric distance has started to attract the researchers’ interest until the recent several decades, and some progress has been made in both its theoretical and computational aspects recently, see e.g. [2, 3, 7, 15, 22, 24].

As a matter of fact, not every point on the plane is possible to locate the new facilities due to some actual situations. For example, we can’t build a hospital or medical center where there is a mountain or a lake. In addition, the location of a facility in health care system can’t be very close to heavily polluting and noisy factories. Therefore, the locations of new facilities are usually restricted to be in some specific regions such as convex subsets of the plane after considering some relevant practical factors.

In this paper we consider the more general case of MFWP with the aim of enhancing its practical applicability. Firstly, the gauge rather than the widely-used symmetric norm is used to measure distance between points. By using gauge as the distance measuring function, both symmetric and asymmetric distances can be discussed in our model. Secondly, we introduce locational constraints on new facilities. By choosing appropriate locational constraints for facilities, e.g. the plane $\mathbb{R}^2$ or the subsets of $\mathbb{R}^2$, both the unconstrained and constrained problems can be taken into consideration in our model.

The rest of this paper is organized as follows. The formulation of general problem is given in Section 2. In Section 3, it is solved by a variational inequality approach.
Specifically, the relationship between the problem and a linear variational inequality (LVI) is built and new projection-type methods are proposed. Then the convergence properties are proved under some mild conditions. Preliminary numerical results are reported in Section 4 to verify the efficiency of proposed projection-type methods. Finally, some conclusions are drawn in the last section.

2. Model formulation. Let $B$ be a compact convex set in $R^2$ and the interior of $B$ contain the origin, then the gauge of $B$ is defined by

$$\gamma(x) = \inf\{\lambda > 0 \mid \frac{x}{\lambda} \in B\}, \quad \forall x \in R^2. \quad (2)$$

$B$ is called the unit ball of $\gamma(\cdot)$ due to

$$B = \{x \in R^2 \mid \gamma(x) \leq 1\}. \quad (3)$$

This definition of a gauge from a compact convex set was first introduced by Minkowski [21]. The gauge $\gamma(\cdot)$ satisfies the following properties:

1) $\gamma(x) \geq 0$ and $\gamma(x) = 0 \iff x = 0$;
2) $\gamma(tx) = t\gamma(x)$ for any $t \geq 0$;
3) $\gamma(x + y) \leq \gamma(x) + \gamma(y)$ for any $x$ and $y \in R^2$.

It follows from 2) and 3) that any gauge $\gamma(x)$ is a convex function of $x$. The distance measuring function can be derived from a gauge by

$$D(x,y) = \gamma(y - x). \quad (4)$$

When $B$ is symmetric around the origin, according to the definition of (2) we have

$$\gamma(-x) = \gamma(x), \quad \forall x \in R^2. \quad (5)$$

It follows that

$$D(x,y) = D(y,x), \quad (6)$$

which means the distance measured by $\gamma(\cdot)$ is symmetric. On the contrary, when $B$ is not symmetric around the origin, (5) does not hold any more and then the distance measured by $\gamma(\cdot)$ is asymmetric. Thus, with different compact convex sets used as unit balls, various gauges (symmetric or asymmetric) can be employed to measure distances in location problems, depending on the requirements of practical applications.

With considering the gauge for measuring distances and the locational constraints on facilities, the constrained multifacility Weber problem (abbreviated as CMFWP) is formulated as follows:

CMFWP: \[
\min \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \gamma(x_i - a_j) + \sum_{i=1}^{m} \sum_{l \neq i, l=1}^{m} v_{il} \gamma(x_i - x_l) \]
\[s.t. \quad x_i \in X_i \subseteq R^2, \quad i = 1, 2, \cdots, m, \quad (7)\]

where

1) $a_j$, $x_i$, $w_{ij}$ and $v_{il}$ are defined as the same as those in (1);
2) $X_i$ denotes the locational constraints for the $i$-th facility and we assume $X_i$, $i = 1, \cdots, m$, are closed convex subsets of $R^2$;
3) $\gamma(\cdot)$ is the gauge generated by a given compact convex set $B \subset R^2$. 

Remark 1. More explanations are required for the CMFWP (7). Firstly, the locational constraint $X_i$ can also be chosen as $R^2$. If all $X_i$, $i = 1, \cdots, m$, are $R^2$, the CMFWP (7) is an unconstrained location problem; otherwise, (7) is a constrained problem. Thus, both constrained and unconstrained cases can be considered in the CMFWP model. Secondly, the CMFWP (7) includes some well-known location problems as its special cases: if $m = 1$, $X = R^2$ and $\gamma(\cdot)$ is the Euclidean distance, (7) is the classical Weber problem; if $m = 1$ and $\gamma(\cdot)$ is the Euclidean distance, (7) is a constrained Weber problem; if all $X_i = R^2$ and $\gamma(\cdot)$ is the Euclidean distance, (7) is the classical multifacility Weber problem (1).

Though the gauge is used and locational constraints are considered, fortunately we have the following proposition for CMFWP which reveals its convexity.

**Proposition 1.** The CMFWP (7) is convex.

**Proof.** Note that all constraints $X_j$, $j = 1, \cdots, m$, are closed convex sets and $\gamma(\cdot)$ is a convex function, it follows that the CMFWP is a convex problem. □

While CMFWP (7) is convex, the nondifferentiability occurs, e.g., when a new facility coincides with another new facility or one customer, which is usually called singularity. This unexpected characteristic of singularity is also shared by MFWP. As a classical and important facility location problem, MFWP has been well studied and many approaches have been suggested for solving it. The first attractive contribution to this aspect, denoted by Miehle algorithm (MA), is due to [20], where an extension of Weiszfeld procedure [32] was proposed for MFWP. Ostresh [23] proved later that the Miehle algorithm is a descent method. Rosen and Xue [26], however, constructed counterexamples showing that Miehle algorithm may stop or converge to one nonoptimal point. To deal with the undesired singularity, another well-known method, hyperboloid approximation procedure (HAP), which is perhaps the most standard method for MFWP, was proposed. The idea of HAP is to introduce a small positive number $\varepsilon$ to the actual objective function to get a smooth approximating problem which is then solved by Miehle algorithm, see e.g. [9, 27]. The global convergence of HAP can be found in Rosen and Xue [27]. Recall that for HAP the larger the value of $\varepsilon$, the faster the convergence to optimal value of approximating function but the worse the accuracy of approximation to the actual nondifferentiable function. Consequently, in the implementation of HAP a large value of $\varepsilon$ can be used initially and then the obtained solution is used as the initial iterate for HAP using a smaller value of $\varepsilon$. The process continues by successively reducing the value of $\varepsilon$ until no significant improvement is possible. HAP with successively reducing the value of $\varepsilon$ is called SHAP throughout this paper. For more details about HAP please refer to [4, 25].

In this paper, a variational inequality (VI) approach, different from the existing approaches for MFWP, is developed for solving CMFWP. Specifically, the relationship of CMFWP and a linear variational inequality (LVI) is first built to show their equivalence, and then numerical methods are proposed to solve the related LVI and thus solve CMFWP. The advantages of the variational inequality approach for CMFWP include the following aspects. Firstly, the variational inequality approach provides a new research framework for CMFWP since the topic of variational inequality has been a mature research area (see, e.g., [10, 11]) and a large number of efficient methods (e.g. the projection-type methods [12, 28] and the proximal point methods [1, 19]) have been proposed for solving VI. Secondly, the related LVI problem usually has a special structure and the variational inequality approach can take
advantages of such structure to develop new efficient VI-based methods under the
algorithmic framework of VI. Thirdly, the global convergence of new methods can
be proved under the analytic framework of VI. Fourthly, the variational inequality
approach is able to provide both the primal and dual solutions of CMFWP simulta-
neously. Fifthly and the most importantly, the variational inequality approach
40
3

Another

In the spirit of the framework of variational inequality approach, new projection-
type methods are proposed in this paper to solve the related LVI. The iterative
scheme of our projection-type methods is simple and easy to implement in com-
putational aspects. The global convergence of projection-type methods is proved.
41
Preliminary numerical experiments compare the projection-type methods with the
well-known methods in facility location for MFWP (1) and numerical methods in
variational inequality topic applicable to CMFWP (7), which verify the effectiveness
of these projection-type methods.

3. The variational inequality approach. This section aims at solving CMFWP
(7) by a variational inequality approach.

3.1. LVI reformulation of CMFWP. For the gauge \( \gamma(\cdot) \) defined by (2), there
exists a dual gauge \( \gamma^d(\cdot) \) defined by

\[
\gamma^d(z) = \max \{ z^T x \mid \gamma(x) \leq 1 \}. \tag{8}
\]

Let \( B^d \) be the unit ball of the dual gauge \( \gamma^d(\cdot) \), which is also convex and compact
and exactly the polar set of \( B \). It is well known that the dual gauge of \( \gamma^d(\cdot) \) is again
\( \gamma(\cdot) \), i.e.,

\[
\gamma(x) = \max \{ z^T x \mid \gamma^d(z) \leq 1 \}, \tag{9}
\]

which can be rewritten as

\[
\gamma(x) = \max_{z \in B^d} z^T x. \tag{10}
\]

For more details about the gauge and dual gauge, as well as their properties, the
readers can be referred to [24].

According to (10), CMFWP (7) is equivalent to the following min-max problem:

\[
\min_{x_i \in X_i} \max_{z_{ij}, z_{il}^l} \sum_{i=1}^{m} \sum_{j=1}^{n} z_{ij}^T (x_i - a_j) + \sum_{i=1}^{m} \sum_{l \neq i, l=1}^{m} z_{il}^l T (x_i - x_l), \tag{11}
\]

where each \( z_{ij} \) is a vector in \( B_{ij}^d = \{ \xi \in R^2 \mid \gamma^d(\xi) \leq w_{ij} \} \) for \( i = 1, \ldots, m \) and
\( j = 1, \ldots, n \); \( z_{il}^l \) is a vector in \( B_{il}^d = \{ \xi \in R^2 \mid \gamma^d(\xi) \leq v_{il} \} \) for \( i = 1, \ldots, m \) and
\( l = 1, \ldots, n \) (\( l \neq i \)).

We first define the following matrix

\[
A_1 = \begin{pmatrix}
I_2 & \cdots & I_2 \\
I_2 & \cdots & I_2 \\
& \ddots & \phantom{I}_I \\
I_2 & \cdots & I_2
\end{pmatrix}^T, \tag{12}
\]

which consists of \( mn \times m \) blocks. Each block of \( A_1 \) is either \( 2 \times 2 \) identity matrix
\((I_2)\) or \( 2 \times 2 \) zero matrix (blank block), and each block column includes \( n \) identity
matrix while others are all zero. We also define matrices \( N_t, t = 1, \ldots, m \), as

\[
N_1 = \begin{pmatrix}
1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1
\end{pmatrix},
\]

\[
N_2 = \begin{pmatrix}
-1 & 1 & 0 & \cdots & 0 \\
1 & -1 & 1 & \cdots & 0 \\
0 & 1 & -1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1
\end{pmatrix},
\]

\[
\vdots, N_m = \begin{pmatrix}
-1 & -1 & \cdots & -1 \\
-1 & 1 & \cdots & 0 \\
1 & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 1 & \cdots & 1
\end{pmatrix}.
\]

Similar to \( A_1 \), each block of \( N_t \) is also either \( 2 \times 2 \) identity matrix or \( 2 \times 2 \) zero matrix. Based on these \( N_t, t = 1, \ldots, m \), we define the following matrix

\[
A_2 = (N_1, N_2, \ldots, N_m)^T.
\]

By introducing the following \( B^d = B^d_1 \times \cdots \times B^d_m \), we can rewrite (11) as a compact form:

\[
\min_{x \in X} \max_{z \in B^d} z^T (Ax - b).
\]

Let \((x^*, z^*)^T \in X \times B^d \) be a solution of (15), then it follows that

\[
z^T (Ax^* - b) \leq z^T (Ax^* - b) \leq z^T (Ax - b), \quad \forall x \in X, z \in B^d,
\]

thus \((x^*, z^*)^T \) is a solution of the following LVI:

\[
\text{Find } x^* \in X, \ z^* \in B^d, \text{ such that } \left\{ \begin{array}{l}
(x-x^*)^T (A^T z^*) \geq 0 \quad \forall x \in X, \\
(z-z^*)^T (-Ax^* + b) \geq 0 \quad \forall z \in B^d.
\end{array} \right.
\]

A compact form of the above LVI is:

\[
\text{LVI: } u^* \in \Omega, \quad (u - u^*)^T (Mu^* + q) \geq 0, \quad \forall u \in \Omega,
\]

where

\[
u = \begin{pmatrix} x \\ z \end{pmatrix}, \quad M = \begin{pmatrix} 0 & AT \\ -A & 0 \end{pmatrix}, \quad q = \begin{pmatrix} 0 \\ b \end{pmatrix} \quad \text{and} \quad \Omega = X \times B^d.
\]

Therefore, the CMFWP (7) is reformulated into the LVI (18)-(19). Note that \( M \) in (19) is a skew-symmetric matrix, then it is positive semi-definite, and thus the LVI (18)-(19) is monotone.

**Remark 2.** It is worth pointing out that the equivalence between CMFWP (7) and LVI (18)-(19) can also be obtained by the duality theory. In fact, the variable \( z_{ij} \) \((z_{ij}^*)\) and the set \( B^d_{ij} \) \((B^d_{ij})\) in (14) are respectively the dual vector and dual ball which satisfy \( z_{ij} \in B^d_{ij} \) \((z_{ij}^* \in B^d_{ij})\).

**Remark 3.** The norms, especially \( l_1, l_2, l_\infty \)-norm and block norms, are frequently used to measure distances in the literature, see, e.g., [16, 30, 31]. It is noted that the gauge used in this paper is an extension of norms. When the gauge \( \gamma(\cdot) \) is chosen as \( l_1, l_2, l_\infty \)-norm, the dual gauge \( \gamma^d(\cdot) \) will be \( l_\infty, l_2, l_1 \)-norm, respectively; and when \( \gamma(\cdot) \) is a block norm, its dual is also a block norm.
3.2. **New projection-type methods.** Among numerous effective numerical algorithms for solving VI, especially LVI, is the projection-type method. The attractive characteristics of the projection-type method, e.g. simpleness and effectiveness, have motivated further development on VI, see e.g. [12, 13, 29]. In this paper we aim to propose new projection-type methods for solving the LVI (18)-(19).

For a given vector \( v \in \mathbb{R}^n \) and a nonempty closed convex set \( \Pi \), the projection of \( v \) onto \( \Pi \) is

\[
P_\Pi(v) = \arg\min_{u \in \Pi} \{ \| u - v \|_2 \}.
\]

It is well known (see e.g. [8]) that for any \( \beta > 0 \), \( u^* \) is a solution of the LVI (18)-(19) if and only if

\[
u^* = P_\Omega[u^* - \beta(Mu^* + q)].
\]

Then the residual function, which measures the magnitude of \( u \) away from the solution set, is defined as

\[
e(u, \beta) = u - P_\Omega[u - \beta(Mu + q)].
\]

We propose a new projection-type method for solving the LVI (18)-(19) as follows.

**Algorithm 1.** The new projection-type method 1 for LVI (18)-(19)

Given an initial iterate \( u^0 = (x^0, z^0) \). For \( k = 0, 1, 2 \cdots \), do:

**Step 1.** Generate \( \tilde{u}^k = (\tilde{x}^k, \tilde{z}^k) \):

\[
\tilde{z}^k = P_{B^d}(z^k + \frac{A x^k - b}{s}),
\]

\[
\tilde{x}^k = P_X(x^k - \frac{A^T \tilde{z}^k}{r}).
\]

**Step 2.** Generate the new iterate \( u^{k+1} = (x^{k+1}, z^{k+1}) \):

\[
u^{k+1} = u^k - \alpha W(u^k - \tilde{u}^k).
\]

**Step 3.** If \( \| x^{k+1} - x^k \| \geq \varepsilon, k = k + 1 \) and go to Step 1; otherwise, stop.

\( W \) is an invertible matrix, \( r, s \) and \( \alpha \) are positive constants. They satisfy some mild conditions so as to guarantee the convergence, which will be discussed later. The \( \varepsilon \) is positive and used as the stopping criterion parameter.

**Remark 4.** Existing algorithms available for solving the LVI (18)-(19) include those presented in [12, 13], etc. For example, applying the well-known projection-contraction method in [13] for the LVI (18)-(19) results in the following procedure to generate new iterate:

\[
z^{k+1} = P_{B^d}[z^k + \gamma \alpha_k [AP_X(x^k - A^T z^k) - b]],
\]

\[
x^{k+1} = P_X[x^k - \gamma \alpha_k A^T P_{B^d}(z^k + Ax^k - b)],
\]

where

\[
\alpha_k = \|e(u^k, 1)\|^2/\|e(u^k, 1) + M^T e(u^k, 1)\|^2 \quad \text{and} \quad \gamma \in (0, 2).
\]

Instead of applying directly existing algorithms, we present the simple method (22)-(24). At (23), the latest available information \( \tilde{z}^k \) is used to compute \( x^{k+1} \); while in the computation of \( x^{k+1} \) in (26), \( z^{k+1} \) is not used. This idea inheriting from the Gauss-Seidel method for solving system of nonlinear equations contributes greatly
to the improvement and effectiveness of Algorithm 1, which will be verified by the numerical results later.

Next we will analyze the convergence of Algorithm 1. Assume that the solution set of the LVI (18)-(19) is nonempty and \( u^* = \begin{pmatrix} x^* \\ z^* \end{pmatrix} \) is a solution of the LVI (18)-(19).

Rewrite \( \tilde{u}^k = \begin{pmatrix} \tilde{x}^k \\ \tilde{z}^k \end{pmatrix} \) in Step 1 of Algorithm 1 as
\[
\tilde{x}^k = Px \left( \tilde{x}^k - \frac{A^T \tilde{z}^k + r(\tilde{x}^k - x^k)}{r} \right),
\]
\[
\tilde{z}^k = P_{Ba} \left( \tilde{z}^k - \frac{-Ax^k + b + s(\tilde{z}^k - z^k)}{s} \right).
\]

Equivalently,
\[
(x - \tilde{z}^k)^T \left( A^T \tilde{z}^k + r(\tilde{x}^k - x^k) \right) \geq 0, \quad \forall x \in X.
\]
\[
(z - \tilde{z}^k)^T \left( -Ax^k + b + A(\tilde{x}^k - x^k) + s(\tilde{z}^k - z^k) \right) \geq 0, \quad \forall z \in B^d.
\]

Let the matrix \( Q \) and the mapping \( F(u) \) defined as
\[
Q = \begin{pmatrix} rI & 0 \\ A & sI \end{pmatrix}
\]
and
\[
F(u) = Mu + q = \begin{pmatrix} A^T z \\ -Ax + b \end{pmatrix},
\]
then \( \{\tilde{u}^k\} \) satisfies
\[
(u - \tilde{u}^k)^T \{ F(\tilde{u}^k) + Q(\tilde{u}^k - u^k) \} \geq 0, \quad \forall u \in \Omega.
\]

**Theorem 3.1.** Let \( W \) be an invertible matrix and \( r, s \) and \( \alpha \) be positive constants such that \( H := QW^{-1} \succ 0 \) and \( G := Q^T + Q - \alpha W^T HW \succ 0 \), then the sequences \( \{u^k\} \) and \( \{\tilde{u}^k\} \) generated by Algorithm 1 satisfies
\[
\|u^{k+1} - u^*\|^2_H \leq \|u^k - u^*\|^2_H - \alpha \|u^k - \tilde{u}^k\|^2_G.
\]

**Proof.** Setting \( u = u^* \) in (29), then we have
\[
(u^* - \tilde{u}^k)^T \{ F(\tilde{u}^k) + Q(\tilde{u}^k - u^k) \} \geq 0.
\]

It follows that
\[
(\tilde{u}^k - u^*)^T Q(u^k - \tilde{u}^k) \geq (\tilde{u}^k - u^*)^T F(\tilde{u}^k).
\]

Since \( F(u) = Mu + q \) is monotone, we have
\[
(\tilde{u}^k - u^*)^T F(\tilde{u}^k) \geq (\tilde{u}^k - u^*)^T F(u^*).
\]

Combining (32) with (33), we get
\[
(\tilde{u}^k - u^*)^T Q(u^k - \tilde{u}^k) \geq (\tilde{u}^k - u^*)^T F(u^*).
\]

Setting \( u = \tilde{u}^k \) in (18), then it follows from (34) that
\[
(\tilde{u}^k - u^*)^T Q(u^k - \tilde{u}^k) \geq 0,
\]
and therefore the following inequality is true
\[
(u^k - u^*)^T Q(u^k - \tilde{u}^k) \geq (u^k - \tilde{u}^k)^T Q(u^k - \tilde{u}^k).
\]
By a simple manipulation, we have
\[
\|u^k - u^*\|^2_H - \|u^{k+1} - u^*\|^2_H
= \|u^k - u^*\|^2_H - \|(u^k - u^* - \alpha W(u^k - \tilde{u}^k))\|^2_H
= 2\alpha(u^k - u^*)^T HW(u^k - \tilde{u}^k) - \alpha^2(u^k - \tilde{u}^k)^T W^T HW(u^k - \tilde{u}^k)
= 2\alpha(u^k - u^*)^T Q(u^k - \tilde{u}^k) - \alpha^2(u^k - \tilde{u}^k)^T W^T HW(u^k - \tilde{u}^k)
\geq 2\alpha(u^k - \tilde{u}^k)^T Q(u^k - \tilde{u}^k) - \alpha^2(u^k - \tilde{u}^k)^T W^T HW(u^k - \tilde{u}^k)
= \alpha(u^k - \tilde{u}^k)^T (Q^T + Q - \alpha W^T HW)(u^k - \tilde{u}^k)
= \alpha(u^k - \tilde{u}^k)^T G(u^k - \tilde{u}^k)
= \alpha\|u^k - \tilde{u}^k\|^2_H,
\] (37)
where the inequality is according to (36). Thus, this theorem follows from (37) directly and the proof is complete.

In fact, the condition of Theorem 3.1 is easy to be satisfied. The following proposition provides one possibility for the choice of matrixes and parameters.

**Proposition 2.** When \( W = \begin{pmatrix} I & 0 \\ A & I \end{pmatrix}, 0 < \alpha < 2 \) and \( rs > \frac{\|A^T A\|}{(2-\alpha)^2} \), the condition of Theorem 3.1 is satisfied.

**Proof.** When \( W = \begin{pmatrix} I & 0 \\ A & I \end{pmatrix} \), \( W^{-1} = \begin{pmatrix} rI & 0 \\ 0 & sI \end{pmatrix} \) and thus \( H = QW^{-1} = \begin{pmatrix} rI & 0 \\ 0 & sI \end{pmatrix} \). Notice that
\[
G = Q^T + Q - \alpha W^T HW
= \begin{pmatrix} rI & 0 \\ A & sI \end{pmatrix} + \begin{pmatrix} rI & A^T \\ 0 & sI \end{pmatrix} - \alpha \begin{pmatrix} I & \frac{A^T}{s} \\ 0 & sI \end{pmatrix} \begin{pmatrix} rI & 0 \\ 0 & sI \end{pmatrix} \begin{pmatrix} I & 0 \\ A & sI \end{pmatrix}
= \begin{pmatrix} (2-\alpha)rI - \frac{r}{s} A^T A & (1-\alpha)A^T \\ (1-\alpha)A & (2-\alpha)sI \end{pmatrix}
= \begin{pmatrix} (2-\alpha) \begin{pmatrix} rI - \frac{\alpha}{(2-\alpha)s} A^T A & \frac{1-\alpha}{2-\alpha} A^T \\ \frac{1-\alpha}{2-\alpha} A & sI \end{pmatrix} \end{pmatrix}.
\]

Multiplying this matrix with an invertible matrix and its transpose on both sides, we obtain
\[
(2-\alpha) \begin{pmatrix} I & -\frac{1-\alpha}{(2-\alpha)s} A^T \\ 0 & I \end{pmatrix} \begin{pmatrix} rI - \frac{\alpha}{(2-\alpha)s} A^T A & \frac{1-\alpha}{2-\alpha} A^T \\ \frac{1-\alpha}{2-\alpha} A & sI \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}
= \begin{pmatrix} (2-\alpha) \begin{pmatrix} rI - \frac{\alpha}{(2-\alpha)s} A^T A & \frac{1-\alpha}{s(2-\alpha)} A^T A \\ 0 & sI \end{pmatrix} \end{pmatrix}
= \begin{pmatrix} (2-\alpha) \begin{pmatrix} rI - \frac{A^T A}{s(2-\alpha)^2} & 0 \\ 0 & sI \end{pmatrix} \end{pmatrix}.
\]

Thus \( G \) and \( (2-\alpha) \begin{pmatrix} rI - \frac{A^T A}{s(2-\alpha)^2} & 0 \\ 0 & sI \end{pmatrix} \) share the same property of semidefinite. Obviously when \( 0 < \alpha < 2 \) and \( rs > \frac{\|A^T A\|}{(2-\alpha)^2} \), \( G \) is ensured. Thus, the condition of Theorem 3.1 is satisfied. \( \square \)
Remark 5. For a special case of $\alpha = 1$, the condition of $rs > \|A^T A\|$ becomes $rs > \|A^T A\|$ to satisfy Theorem 3.1.

Theorem 3.2. Let $\{u^k\}$ and $\{\tilde{u}^k\}$ be the sequences generated by Algorithm 1 which satisfies the condition of Theorem 3.1. Then we have
1) The sequence $\{u^k\}$ is bounded.
2) $\lim_{k \to \infty} \|u^k - \tilde{u}^k\| = 0$.
3) The sequence $\{u^k\}$ converges to a solution of the LVI (18)-(19).

Proof. The first assertion follows from (30) directly.

From the inequality (30) we get
\[
\sum_{k=0}^{\infty} \alpha \|u^k - \tilde{u}^k\|^2_G \leq \|u^0 - u^*\|^2_H,
\]
and thus we have
\[
\lim_{k \to \infty} \|u^k - \tilde{u}^k\|^2_G = 0.
\]
Consequently,
\[
\lim_{k \to \infty} \|u^k - \tilde{u}^k\| = 0,
\]
and the second assertion is thus proved.

Substituting (40) into (29)
\[
\lim_{k \to \infty} (u - \tilde{u}^k)^T \{\beta F(\tilde{u}^k)\} \geq 0, \quad \forall u \in \Omega.
\]
According to the first and second assertion we can conclude that the sequence $\{\tilde{u}^k\}$ is also bounded. Let $\tilde{u}^\infty$ be a cluster point of $\{\tilde{u}^k\}$ and the subsequence $\{\tilde{u}^{kj}\}$ converges to $\tilde{u}^\infty$. Then it follows from (41) that
\[
\lim_{j \to \infty} (u - \tilde{u}^{kj})^T \{\beta F(\tilde{u}^{kj})\} \geq 0, \quad \forall u \in \Omega,
\]
and consequently,
\[
(u - \tilde{u}^\infty)^T \{\beta F(\tilde{u}^\infty)\} \geq 0, \quad \forall u \in \Omega.
\]
This means that $\tilde{u}^\infty$ is a solution of the LVI (18)-(19).

Because
\[
\|u^{kj} - \tilde{u}^\infty\| \leq \|u^{kj} - \tilde{u}^{kj}\| + \|\tilde{u}^{kj} - \tilde{u}^\infty\|,
\]
we have
\[
\lim_{j \to \infty} \|u^{kj} - \tilde{u}^\infty\| = 0.
\]
Let $u^* = \tilde{u}^\infty$ in (30), then we can get
\[
\|u^{k+1} - \tilde{u}^\infty\| \leq \|u^k - \tilde{u}^\infty\|,
\]
which means that $\|u^k - \tilde{u}^\infty\|$ is monotone decreasing. Combining it with (44), the sequence $\{u^k\}$ converges to the solution point $\tilde{u}^\infty$.

In Step 1 of Algorithm 1, we firstly compute $\hat{z}^k$ and then update $\tilde{x}^k$ using the information of obtained $\hat{z}^k$. Alternatively, we can compute $\hat{z}^k$ firstly and then update $\tilde{z}^k$. Then another projection-type method Algorithm 2 is proposed as follows.

Algorithm 2. The new projection-type method 2 for LVI (18)-(19)

Given an initial iterate $u^0 = \left( \begin{array}{c} x_0^0 \\ z_0^0 \end{array} \right)$. For $k = 0, 1, 2 \cdots$, do:
Step 1. Generate $\tilde{u}^k = \left( \tilde{x}^k \quad \tilde{z}^k \right)$:

$$\tilde{x}^k = P_X \left( x^k - \frac{A^T \bar{z}^k}{r} \right),$$  \quad (46)

$$\tilde{z}^k = P_{B^d} \left( z^k + \frac{A \bar{z}^k - b}{s} \right).$$  \quad (47)

Step 2. Generate the new iterate $u^{k+1} = \left( x^{k+1} \quad z^{k+1} \right)$:

$$u^{k+1} = u^k - \alpha W (u^k - \tilde{u}^k).$$  \quad (48)

Step 3. If $\|x^{k+1} - x^k\| \geq \varepsilon, k = k + 1$ and go to Step 1; otherwise, stop.

$W$ is an invertible matrix, $r$, $s$ and $\alpha$ are positive constants. The $\varepsilon$ is positive and used as the stopping criterion parameter.

Noted that $\tilde{u}^k = \left( \tilde{x}^k \quad \tilde{z}^k \right)$ in Step 1 of Algorithm 2 is equivalent to

$$(x - \tilde{x}^k)^T \left( A^T \tilde{z}^k + r(\tilde{x}^k - x^k) - A^T (\tilde{z}^k - z^k) \right) \geq 0, \forall x \in X.$$  \quad (49)

$$(z - \tilde{z}^k)^T \left( -A \tilde{z}^k + b + s(\tilde{z}^k - z^k) \right) \geq 0, \forall z \in B^d.$$  \quad (50)

Let $Q$ and $F(u)$ defined as

$$Q = \begin{bmatrix} rI & -A^T \\ 0 & sI \end{bmatrix} \quad \text{and} \quad F(u) = Mu + q = \begin{bmatrix} A^T z \\ -Ax + b \end{bmatrix},$$

then $\{\tilde{u}^k\}$ satisfies

$$(u - \tilde{u}^k)^T \{F(\tilde{u}^k) + Q(\tilde{u}^k - u^k)\} \geq 0, \forall u \in \Omega,$$  \quad (51)

which is the same as (29) except for different $Q$. Similar to Theorem 3.1 and Theorem 3.2, we can prove the convergence of Algorithm 2 under similar mild conditions. Specifically, when $W = \begin{bmatrix} I & -A^T \\ 0 & I \end{bmatrix}$, $0 < \alpha < 2$ and $rs > \frac{\|A^T A\|}{(2 - \alpha)^2}$ (for a special case, $\alpha = 1$ and $rs > \|A^T A\|$), the condition is satisfied and Algorithm 2 is convergent to a solution of LVI (18)-(19).

4. Numerical results. This section reports some numerical results to verify the computational efficiency of the variational inequality approach. We will implement the proposed Algorithm 1 and 2 to solve a number of CMFWPs including unconstrained problems and constrained problems and also including problems with symmetric distances and asymmetric distances. To demonstrate the efficiency of Algorithm 1 and 2, in the first subsection we compare Algorithm 1 and 2 with the well-known methods in facility location area for MFWP and in the second subsection we compare them with the well-known methods in variational inequalities. For simplicity, we choose $\alpha = 1$, $r = 2\sqrt{t}\|A^T A\|$ and $s = \frac{1}{2}\sqrt{t}\|A^T A\|$ with $t = 1.01$ in the following implementations of Algorithm 1 and 2. All the programming codes were written by MATLAB 7.8 and were run on a Dell laptop (Ceron processor 1.86GHz).
4.1. Algorithm 1 and 2 for MFWP. Miehle algorithm (MA) and hyperboloid approximation procedure (HAP) are perhaps the most standard methods for solving MFWP. This paper provides a variational inequality framework to solve CMFWP and under the new framework Algorithm 1 and 2 are then proposed. Note that MFWP, as a classical model in facility location, is a special case of CMFWP. Hence, the variational inequality framework and Algorithm 1 and 2 are applicable to solving MFWP. In the following, we compare Algorithm 1 and 2 with MA and HAP for solving MFWP and show that even for such a special problem the new framework and the proposed Algorithm 1 and 2 are efficient.

First we compare Algorithm 1 and 2 with MA. The following two examples were constructed in [26].

**E1**: Example 1 in [26]

\[ n = 4, m = 2; \ a_1 = (-4, 1), \ a_2 = (4, 1), \ a_3 = (-4, -1), \ a_4 = (4, -1); \ w_{11} = 10, \ w_{12} = w_{13} = 1, \ w_{21} = w_{22} = w_{24} = 10; \ v_{12} = \frac{-1 + \sqrt{1 - 4c}}{2}, \]

\[ b = \frac{50}{\sqrt{26}} - \frac{40}{\sqrt{5}} + \frac{3}{\sqrt{10}} + \frac{32}{\sqrt{370}}, \ c = -\frac{3200}{\sqrt{130}} + \frac{32}{\sqrt{370}}; \ x_1^0 = (1, 0), \ x_2^0 = (2, 0). \]

**E2**: Example 2 in [26]

\[ n = 4, m = 2; \ a_1 = (-4, 1), \ a_2 = (4, 1), \ a_3 = (-4, -1), \ a_4 = (4, -1); \ w_{11} = w_{13} = 10, \ w_{12} = w_{14} = 1, \ w_{21} = w_{23} = 1, \ w_{22} = w_{24} = 10; \ v_{12} = 100; \ x_1^0 = (1, 0), \ x_2^0 = (2, 0). \]

It has been shown in [26] that the iterate sequences generated by MA stop at a nonoptimal point of E1. This means that the convergence of MA to the optimal solution is not guaranteed.

We further construct the following two examples.

**E1’**: E1 with initial iterate \(x_1^0\) and \(x_2^0\) randomly chosen as different customers.

**E2’**: E2 with initial iterate \(x_1^0\) and \(x_2^0\) randomly chosen as different customers.

As the location of customers are chosen as initial iterates in E1’ and E2’, the undesirable singularity will occur. The two examples are generated to show the difference of the performance of Algorithm 1 and 2 with MA in case of singularity.

Note that the location of customers, the weights and the initial iterates are particularly chosen in E1, E2, E1’ and E2’. For well justification of the efficiency of Algorithm 1 and 2, we construct the following randomly generated problems.

**E3**: E1 (E2) but all customers a\(j\) and the initial iterate \(x_1^0, x_2^0\) are randomly generated in \([-250, 250]^2\), \(w_{ij}\) and \(v_{il}\) are randomly generated in \([1, 100]\).

Note that MA is a Jacobi-like method because the new locations of different facilities change at the same time at each iteration. That is to say, \(x_i^{k+1}\) is a function of \(x_1^k, \ldots, x_{i-1}^k, x_{i+1}^k, \ldots, x_m^k\). An alternative, however, may potentially accelerate the convergence of MA by using the new information as it becomes available, i.e., \(x_i^{k+1}\) is obtained from \(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_{i+1}^{k+1}, x_m^k\). This idea inherits from the Gauss-Seidel method and thus such alternative is called as Gauss-Seidel-like Miehle algorithm (GMA).

We solve the five examples by MA, GMA, Algorithm 1 and Algorithm 2 and the numerical results are reported in Table 1, where “CPU” means the computing time, “Iter.” gives the number of iterations and “Obj.” reports the objective functional values. The stopping criterion is chosen as

\[ \|x^{k+1} - x^k\| < 10^{-6}. \]

Each example is solved by four methods with the same initial iterate \(x_1^0, x_2^0\). The initial \(z_{ij}^0\) and \(z_{ij}^{0'}\) in Algorithm 1 and 2 are randomly chosen in \(\{\xi \in R^2 ||\xi|| \leq w_{ij}\}\).
and \( \{ \xi \in \mathbb{R}^2 \mid \| \xi \| \leq v_{il} \} \), respectively. For \( E1, E2, E1' \) and \( E2' \), each example is solved by Algorithm 1 and 2 one hundred times with randomly generated \( z_{il}^0 \) and \( z_{il}^0 \), and the average numerical results are reported. For \( E3 \), we test it for one hundred scenes with each scene solved by four methods and report the average results.

Table 1. MA, GMA, Algorithm 1 and Algorithm 2 for MFWP

| Problem | CPU    | Iter. | Obj.    | CPU    | Iter. | Obj.    | CPU    | Iter. | Obj.    |
|---------|--------|-------|---------|--------|-------|---------|--------|-------|---------|
| E1      | 0.0003 | 2     | 182.23  | 0.0004 | 9     | 182.43  | 0.0147 | 152.44 | 157.52  |
| E2      | 0.0109 | 206   | 182.19  | 0.0006 | 5     | 182.68  | 0.0149 | 158.58 | 159.96  |
| E1'     | /      | /     | 0.0005  | /      | /     | 0.0002  | /      | /     | 0.0145  |
| E2'     | /      | /     | 0.0147  | /      | /     | 0.0141  | /      | /     | 0.0142  |
| E3      | 0.1949 | 3733.40 | 63737.39 | 0.0050 | 85.86 | 63515.97 | 0.0206 | 63515.97 | 63515.97 |

For \( E1 \), MA achieves a point, \( (1.50591, 0.00000) \) and \( (1.50591, 0.00000) \), in the first iteration and then stops at it from the second iteration. However, it is not the optimal solution of \( E1 \), which is \( (0,0) \) and \( (0,0) \). When GMA is applied to solve \( E1 \), the point obtained by GMA, \( (1.65749,0) \) and \( (1.65749,0) \), is also not the optimal one. In fact, this point is even worse than that from MA. While Algorithm 1 and 2 need more iterations for \( E1 \), they can get their optimal solution and provide the optimal objective functional value. For \( E2 \), MA needs the most number of iterations among four methods. With the adoption of the Gauss-Seidel idea, GMA needs the least number of iterations. When we compare the points obtained by four methods, Algorithm 1 and 2 provide the optimal solution but the other two methods do not.

For \( E1' \) and \( E2' \), the undesirable singularity occurs. Thus, MA and GMA can not proceed the iterations and then their numerical results are labeled as ‘/’ in Table 1. Algorithm 1 and 2, however, are able to deal with the singularity and in such case Algorithm 1 and 2 can still get the optimal solution, which is consistent with the analytic results.

For the randomly generated \( E3 \), it is found that the average computing time and number of iterations of GMA are the least. Nevertheless, the solution obtained by GMA is the worst among four methods. MA needs much more computing time and number of iterations than the other methods, but the point obtained by it is still not the optimal one of \( E3 \). Compared to MA and GMA, Algorithm 1 and 2 can get the optimal solution of \( E3 \) within reasonable computing time and number of iterations.

Next we will compare Algorithm 1 and 2 with HAP. An alternative of HAP using the new available information was proposed in [27]. This alternative is called Gauss-Seidel-like HAP (GHAP). The parameter \( \varepsilon \) is chosen as 50, 5, 0.5, 0.05, respectively, and we report the numerical results of HAP and GHAP under different values of \( \varepsilon \). SHAP successively reduces the value of \( \varepsilon \) during its implementation to accelerate the convergence of HAP under small value of \( \varepsilon \). This technique is also applicable to GHAP and GHAP with such technique is called SGHAP in this paper. For
SHAP and SGHAP, the $\varepsilon$ is reduced from 50 to 0.05 with its value multiplied by 0.1 each time. For convenience, we call HAP, GHAP, SHAP and SGHAP as HAP-type methods.

We test MFWP with $n = 500$ and $m = 2, 4, 6, 8, 10$. All customers are randomly generated in $[-250, 250]^2$; the weights between customers and facilities are randomly chosen in $[1, 5]$ and the weights between facilities are chosen as 100. The stopping criterion is set as $\|x^{k+1} - x^k\| < 10^{-4}$. The initial iterate is randomly generated in $[-250, 250]^2$. For each pair $(n, m)$, we solve 100 randomly generated MFWPs by HAP, GHAP, SHAP, SGHAP, Algorithm 1 and Algorithm 2 using the same randomly generated customers and initial iterates. The numerical results are reported in Table 2, where “CPU” and “Iter.” are as the same as those in Table 1 and “Dobj.” reports the difference of objective functional values between Algorithm 1 and other methods. Actually the value of “Dobj.” is the functional value of other methods minus that of Algorithm 1. Therefore, if this value is positive, the point obtained by other method is worse than that by Algorithm 1; otherwise, it is better. It is clear that the larger the value of “Dobj.”, the worse the point obtained by the method.

As well known for HAP, the larger the value of $\varepsilon$, the faster the convergence of HAP but the worse the accuracy of approximation to MFWP, which can be also illustrated in Column 3-6 of Table 2. The same conclusion can be easily drawn for GHAP. Table 2 also reveals that GHAP can accelerate the convergence of HAP with the adoption of Gauss-Seidel idea. The difference between the values of “Dobj.” of HAP and GHAP is quite small for the same $\varepsilon$, which indicates that the points obtained by two methods are almost the same. However, the workload of GHAP is nearly half of HAP for the same $\varepsilon$.

Note that the value of $\varepsilon$ in SHAP is successively reduced from 50 to 0.05. It implies that the points obtained by SHAP are highly possible to be identical to those obtained by HAP under $\varepsilon=0.05$. This can be verified by the almost same values of “Dobj.” in Column 6 and Column 11. However, we find that the workload of SHAP is much less than that of HAP. Thus, SHAP can improve the efficiency of HAP greatly. A similar conclusion is also true for GHAP and SGHAP, indicated by Column 10 and Column 12 of Table 2.

According to the last two columns in Table 2, it can be seen that the performances of Algorithm 1 and 2 are quite similar. This founding is consistent with our expectation, since the difference between Algorithm 1 and Algorithm 2 is just different computation order of $\tilde{x}^k$ and $\tilde{z}^k$ in Step 1.

When we compare Algorithm 1 and 2 with four HAP-type methods, it is easy to observe that Algorithm 1 and 2 are preferable to HAP-type methods especially for large-scale problems. Firstly, we compare them from the quality of obtained points. We have known that the points obtained by Algorithm 1 and 2 are the optimal solution of MFWP. However, the points obtained by HAP-type methods are not the optimal solution, which is indicated by the positive values of “Dobj.”. Secondly, we compare six methods from the computing workload. It is clear that the workloads of Algorithm 1 and 2 are comparable with HAP-type methods especially for large-scale MFWP. When $m$ is small, the workloads of Algorithm 1 and 2 are greater than HAP-type methods except HAP under $\varepsilon=0.05$. However, with the value of $m$ increasing, the increment speed of workload of HAP-type methods is much faster than Algorithm 1 and 2. When $m$ is equal to 10, the workloads of
Table 2. HAP, GHAP, SHAP, SGHAP, Algorithm 1 and Algorithm 2 for MFWP

| m | HAP | | | | | | SHAP | SGHAP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | ε=50 | ε=5 | ε=0.5 | ε=0.05 | ε=50 | ε=5 | ε=0.5 | ε=0.05 | ε=50 | ε=5 | ε=0.5 | ε=0.05 |
| 2 | CPU | 0.21 | 0.44 | 1.08 | 2.75 | 0.15 | 0.27 | 0.60 | 1.53 | 0.87 | 0.80 | 1.38 | 1.32 |
| 2 | Iter. | 48.52 | 101.78 | 247.60 | 630.34 | 33.88 | 61.28 | 138.00 | 348.72 | 262.84 | 243.48 | 272.78 | 272.84 |
| 2 | Dobj. | 82.2151 | 28.6441 | 9.0927 | 2.6080 | 82.2156 | 28.6446 | 9.0932 | 2.6083 | 2.6082 | 2.6052 | 0.0000 | 0.0002 |
| 4 | CPU | 0.98 | 2.48 | 6.60 | 17.89 | 0.58 | 1.33 | 3.45 | 9.26 | 2.61 | 2.33 | 2.15 | 2.08 |
| 4 | Iter. | 107.70 | 270.20 | 720.00 | 1936.20 | 63.50 | 145.90 | 376.10 | 1003.40 | 378.30 | 337.80 | 168.90 | 168.90 |
| 4 | Dobj. | 283.4897 | 94.6138 | 30.0266 | 9.0773 | 283.4903 | 94.6144 | 30.0271 | 9.0767 | 9.0713 | 9.0701 | 0.0000 | 0.0005 |
| 6 | CPU | 2.26 | 5.95 | 16.08 | 43.94 | 1.28 | 3.17 | 8.52 | 23.22 | 5.71 | 4.00 | 2.82 | 2.73 |
| 6 | Iter. | 164.36 | 431.30 | 1167.66 | 3145.08 | 92.64 | 230.90 | 616.96 | 1664.60 | 541.94 | 382.48 | 126.82 | 126.84 |
| 6 | Dobj. | 475.4137 | 155.9050 | 49.3255 | 15.0577 | 475.4124 | 155.9035 | 49.3238 | 15.0528 | 15.0468 | 15.0480 | 0.0000 | -0.0003 |
| 8 | CPU | 3.80 | 10.14 | 27.60 | 74.81 | 2.11 | 5.44 | 14.72 | 39.90 | 8.44 | 5.98 | 3.20 | 3.12 |
| 8 | Iter. | 216.50 | 578.56 | 1573.84 | 4245.06 | 119.98 | 309.62 | 836.70 | 2262.78 | 638.06 | 453.98 | 101.70 | 101.74 |
| 8 | Dobj. | 698.8811 | 227.5549 | 72.0128 | 22.1839 | 698.8829 | 227.5570 | 72.0143 | 22.1790 | 22.1858 | 22.1714 | 0.0000 | 0.0021 |
| 10 | CPU | 6.08 | 16.40 | 44.55 | 120.77 | 3.34 | 8.77 | 23.73 | 64.28 | 14.02 | 9.34 | 4.27 | 4.14 |
| 10 | Iter. | 274.60 | 740.54 | 2012.20 | 5414.64 | 150.90 | 396.42 | 1072.20 | 2899.56 | 840.34 | 563.34 | 97.68 | 97.80 |
| 10 | Dobj. | 925.6176 | 299.2628 | 94.3543 | 28.8958 | 925.6183 | 299.2637 | 94.3537 | 28.8839 | 28.8930 | 28.8793 | 0.0000 | -0.0123 |
HAP-type methods have exceeded Algorithm 1 and 2 except HAP under $\varepsilon=50$ and GHAP under $\varepsilon=50$ (note that the approximation accuracy of HAP under $\varepsilon=50$ and GHAP under $\varepsilon=50$ is low). In addition, we also expect to decrease $\varepsilon$ of HAP-type methods to improve the quality of obtained points. However, it is found during our experiments that though the quality of obtained points is improved, the computing time and number of iterations increase drastically with smaller $\varepsilon$.

Thus, with the comparison of Algorithm 1 and 2 with some well-known methods in facility location area, their efficiency is verified.

4.2. Algorithm 1 and 2 for CMFWP. In this subsection we will apply Algorithm 1 and 2 to solve CMFWP (7). Note that Algorithm 1 and 2 are proposed for CMFWP under the variational inequality framework by solving LVI (18)-(19). There exist other methods for LVI, and thus they are also applicable to solving CMFWP. To verify the efficiency of Algorithm 1 and 2, we compare Algorithm 1 and 2 with some well-known methods for LVI to solve CMFWP. The first method to be compared is the projection and contraction (PC) method presented in [13] and the second one is the customized proximal point algorithm (PPA) proposed in [14]. In the customized PPA, when the metric proximal parameter $G$ is chosen as

$$
G = \begin{pmatrix}
    rI & A^T \\
    A & sI
  \end{pmatrix},
$$

it results in the following iterative scheme (denoted as PPA1)

\[
\begin{align*}
  z^k &= P_{B^s}[z^k + \frac{Ax^k - b}{s}], \\
  x^k &= P_X[x^k - \frac{2A^T z^k - A^T x^k}{r}], \\
  u^{k+1} &= u^k - \gamma(u^k - \tilde{u}^k).
\end{align*}
\]

Alternatively, we can also choose another metric proximal parameter $G$

$$
G = \begin{pmatrix}
    rI & -A^T \\
    -A & sI
  \end{pmatrix}.
$$

It results in the following scheme (denoted as PPA2)

\[
\begin{align*}
  \hat{x}^k &= P_X[x^k - \frac{A^T z^k}{r}], \\
  \hat{z}^k &= P_{B^s}[z^k + \frac{2A\hat{x}^k - Ax^k - b}{s}], \\
  u^{k+1} &= u^k - \gamma(u^k - \tilde{u}^k).
\end{align*}
\]

The convergence of PPA1 and PPA2 under the condition of $rs > \|A^T A\|$ and $\gamma \in (0, 2)$ can be found in [14].

The parameters of different methods are given as follows.

- PC method: $\gamma = 1.9$;
- PPA1 and PPA2: $r = p \cdot \sqrt{t \|A^T A\|}, s = \frac{2}{p} \cdot \sqrt{t \|A^T A\|}, t = 1.01, p = 2$; $\gamma = 1.2$

We first test CMFWP under Euclidean distances with $n = 50, 100, 200, 500, 1000$ and $m = 2, 4, 6, 8, 10$. All customers are randomly generated in $[-250, 250]^2$; the weights between customers and facilities are randomly chosen in $[1, 3]$ and the weights between facilities are chosen in $[1, 5]$. The locational constraints are set as $X = \{x_i \in R^2 \ | \ |x_i - c_i| \leq r_i\}$ where $c_i$ is randomly generated in $[-250, 250]^2$. 
and $r_i$ is randomly chosen in $[10, 15]$, $i = 1, \cdots, m$. The initial iterate is randomly generated in $[-250, 250]^2$ and the stopping criterion is given as

$$\|x^{k+1} - x^k\| < 10^{-4}.$$ 

For each given pair $(n, m)$, we test the CMFWPs for 100 scenes with each scene solved by five methods using the same randomly generated initial iterates. Table 3 reports the average computing time in seconds (CPU) and number of iterations (Iter.) of five methods to solve the CMFWPs under Euclidean distances (symmetric distances).

Table 3. Numerical results for CMFWP under Euclidean distances

| $n$ | $m$ | CPU | Iter. | PC method | CPU | Iter. | PPA1 | CPU | Iter. | PPA2 | CPU | Iter. | Algorithm 1 | CPU | Iter. | Algorithm 2 | CPU | Iter. |
|-----|-----|-----|-------|-----------|-----|-------|------|-----|-------|------|-----|-------|-------------|-----|-------|--------------|-----|-------|
| 50  |   2 | 0.034 | 34.45 | 0.018 | 31.32 | 0.018 | 32.93 | 0.018 | 29.84 | 0.015 | 30.34 |
|     | 4   | 0.093 | 40.32 | 0.041 | 36.60 | 0.046 | 37.78 | 0.044 | 36.39 | 0.043 | 37.11 |
|     | 6   | 0.197 | 45.92 | 0.071 | 39.23 | 0.072 | 40.92 | 0.068 | 37.82 | 0.063 | 38.08 |
|     | 8   | 0.314 | 49.93 | 0.096 | 41.63 | 0.100 | 43.87 | 0.093 | 41.91 | 0.090 | 41.76 |
|     | 10  | 0.499 | 53.08 | 0.144 | 47.04 | 0.151 | 49.52 | 0.141 | 45.92 | 0.137 | 45.72 |
| 100 |   2 | 0.066 | 17.27 | 0.027 | 25.87 | 0.028 | 27.21 | 0.024 | 22.44 | 0.022 | 22.65 |
|     | 4   | 0.155 | 24.74 | 0.006 | 31.24 | 0.065 | 32.14 | 0.057 | 27.51 | 0.054 | 28.48 |
|     | 6   | 0.277 | 26.42 | 0.099 | 33.51 | 0.113 | 35.92 | 0.097 | 31.42 | 0.098 | 31.99 |
|     | 8   | 0.570 | 29.63 | 0.135 | 35.44 | 0.169 | 38.68 | 0.147 | 34.51 | 0.147 | 34.82 |
|     | 10  | 0.830 | 30.46 | 0.221 | 38.73 | 0.238 | 40.56 | 0.221 | 37.22 | 0.219 | 37.81 |
| 200 |   2 | 0.098 | 19.98 | 0.054 | 26.81 | 0.059 | 29.40 | 0.051 | 26.68 | 0.052 | 27.49 |
|     | 4   | 0.370 | 22.05 | 0.121 | 28.88 | 0.126 | 30.43 | 0.116 | 28.23 | 0.111 | 28.13 |
|     | 6   | 0.825 | 23.80 | 0.216 | 33.26 | 0.243 | 37.04 | 0.208 | 31.44 | 0.202 | 31.59 |
|     | 8   | 1.574 | 24.02 | 0.333 | 35.37 | 0.344 | 36.59 | 0.299 | 31.36 | 0.293 | 32.24 |
|     | 10  | 2.391 | 25.59 | 0.523 | 40.48 | 0.557 | 42.38 | 0.500 | 37.77 | 0.489 | 38.27 |
| 500 |   2 | 0.241 | 9.62  | 0.150 | 25.82 | 0.156 | 27.45 | 0.121 | 21.26 | 0.118 | 21.82 |
|     | 4   | 1.131 | 11.73 | 0.090 | 37.29 | 0.505 | 38.14 | 0.388 | 29.03 | 0.371 | 28.75 |
|     | 6   | 3.335 | 16.14 | 0.884 | 38.74 | 0.993 | 42.53 | 0.770 | 32.16 | 0.727 | 32.02 |
|     | 8   | 6.072 | 17.01 | 1.281 | 39.89 | 1.358 | 41.56 | 1.057 | 33.45 | 1.006 | 33.16 |
|     | 10  | 9.968 | 18.62 | 2.138 | 48.18 | 2.266 | 50.49 | 1.653 | 36.62 | 1.646 | 37.34 |
| 1000|   2 | 0.716 | 7.78  | 0.386 | 28.34 | 0.414 | 29.86 | 0.321 | 22.47 | 0.320 | 23.51 |
|     | 4   | 3.306 | 9.75  | 1.443 | 45.45 | 1.484 | 45.77 | 1.214 | 37.59 | 1.251 | 38.65 |
|     | 6   | 8.109 | 10.88 | 2.879 | 49.66 | 3.096 | 51.44 | 2.108 | 35.86 | 2.132 | 36.42 |
|     | 8   | 20.743 | 13.95 | 4.956 | 55.43 | 6.192 | 56.06 | 3.942 | 42.44 | 3.642 | 40.36 |
|     | 10  | 37.590 | 17.52 | 7.712 | 59.18 | 7.914 | 59.85 | 5.819 | 43.43 | 5.893 | 45.80 |

According to Table 3, five methods can solve the randomly generated CMFWPs under Euclidean distance and with locational constraints efficiently. It is shown that the proposed Algorithm 1 and 2 are preferable to the PPA1 and PPA2 for solving the CMFWPs of all scales. This can be illustrated roughly as follows. For PPA1 and PPA2, while the iterative scheme of their first variable ($\hat{x}^k$ for PPA1 and $\tilde{x}^k$ for PPA2) is simple to implement, the iteration of their second variable is complex, which will result in the unbalance of the execution of two variables. However, during the implementation of Algorithm 1 and 2, the iterative scheme of both variables, $\hat{x}^k$ and $\tilde{x}^k$, of Algorithm 1 and 2 is not only quite simple but also balanced and thus its performance is improved. In addition, it is also found that Algorithm 1 and 2 are much preferable to PC method due to using the latest available information $\tilde{x}^k$ to compute $\tilde{x}^k$, which powerfully verifies the conclusion in Remark 4 in computational sense.

Since Algorithm 1 and 2 can solve the CMFWP whose distances are measured by asymmetric gauge, we test some randomly generated CMFWPs under gauge with
different scales. The asymmetric gauge $\gamma(\cdot)$ is generated with the unit ball set as

$$(x - \frac{1}{2})^2 + y^2 \leq \frac{1}{2}.$$ 

To verify the efficiency of Algorithm 1 and 2 for CMFWP under gauge, we also compare Algorithm 1 and 2 with the other three methods simultaneously. The performance of five methods is reported in Table 4.

### Table 4. Numerical results for CMFWP under gauge

| $n$ | $m$ | PC method | PPA1 | PPA2 | Algorithm 1 | Algorithm 2 |
|-----|-----|-----------|------|------|-------------|-------------|
|     |     | CPU | Iter. | CPU | Iter. | CPU | Iter. | CPU | Iter. |
| 50  | 2   | 0.07 | 58.93 | 0.03 | 52.66 | 0.03 | 53.87 | 0.02 | 39.33 |
|     | 4   | 0.22 | 77.85 | 0.08 | 60.53 | 0.06 | 61.71 | 0.06 | 49.14 |
|     | 6   | 0.48 | 86.40 | 0.14 | 65.16 | 0.14 | 68.42 | 0.11 | 54.47 |
|     | 8   | 0.86 | 101.61| 0.23 | 76.47 | 0.22 | 76.93 | 0.18 | 64.13 |
| 100 | 2   | 0.13 | 47.82 | 0.05 | 38.50 | 0.05 | 39.37 | 0.03 | 28.84 |
|     | 4   | 0.42 | 56.39 | 0.13 | 48.53 | 0.13 | 48.98 | 0.10 | 37.98 |
|     | 6   | 0.78 | 58.80 | 0.23 | 57.14 | 0.23 | 57.93 | 0.17 | 43.07 |
|     | 8   | 1.92 | 86.76 | 0.35 | 62.63 | 0.36 | 66.90 | 0.29 | 52.46 |
| 200 | 2   | 0.15 | 21.38 | 0.07 | 29.11 | 0.08 | 30.09 | 0.05 | 21.32 |
|     | 4   | 0.64 | 31.07 | 0.20 | 37.14 | 0.20 | 37.92 | 0.15 | 27.76 |
|     | 6   | 1.86 | 33.58 | 0.38 | 45.09 | 0.39 | 45.95 | 0.29 | 34.45 |
|     | 8   | 2.79 | 37.18 | 0.62 | 50.46 | 0.65 | 53.94 | 0.47 | 38.48 |
| 500 | 2   | 0.28 | 9.53  | 0.16 | 23.66 | 0.19 | 29.26 | 0.13 | 20.84 |
|     | 4   | 1.69 | 16.28 | 0.57 | 37.75 | 0.62 | 40.80 | 0.42 | 27.21 |
|     | 6   | 6.26 | 22.22 | 1.07 | 41.50 | 1.04 | 40.41 | 0.81 | 31.00 |
|     | 8   | 9.96 | 27.75 | 1.82 | 43.97 | 2.06 | 48.70 | 1.52 | 35.96 |
| 1000| 2   | 0.70 | 7.49  | 0.37 | 23.81 | 0.39 | 24.30 | 0.31 | 19.23 |
|     | 4   | 3.46 | 8.06  | 1.38 | 32.74 | 1.45 | 33.15 | 0.98 | 22.62 |
|     | 6   | 7.44 | 9.23  | 2.45 | 37.53 | 2.72 | 40.88 | 1.88 | 28.29 |
|     | 8   | 14.15| 10.25 | 4.17 | 41.70 | 4.38 | 42.95 | 2.98 | 32.12 |
|     | 10  | 39.91| 13.78 | 8.83 | 51.66 | 9.15 | 51.74 | 6.62 | 38.16 |

It is easy to observe that the proposed Algorithm 1 and 2 are efficient for CMFWP under gauge and they outperform PPA1, PPA2 and PC method for the randomly generated asymmetric problem. In comparison of Table 3 and Table 4, however, we found that for five methods due to the asymmetric property of gauge the computing time and number of iterations for the CMFWP under asymmetric gauge are much greater than those for CMFWP under Euclidean distances.

### 5. Conclusion

This paper investigates the constrained multifacility Weber problem. The locational constraints on facilities are considered and the gauge is employed to measure the distances in this problem. In this paper, a variational inequality approach is developed for solving CMFWP. With the consideration of special structures of reformulated linear variational inequality, we propose new projection-type methods. The convergence of proposed methods is proved under mild assumption. Preliminary numerical results justify the preferable merits of the proposed algorithms.

How to solve other facility location problems, such as multi-source location problem, by the approach of variational inequality deserves further and more extensive investigation in our future research.
Acknowledgments. The authors thank the editor and the anonymous referees for their careful reading of the paper and valuable comments.

REFERENCES

[1] R. S. Burachik and A. N. Iusem, A generalized proximal point algorithm for the variational inequality problem in a Hilbert space, SIAM J. Optim., 8 (1998), 197–216.
[2] E. Carrizosa, E. Conde, M. Muñoz and J. Puerto, Simpson points in planar problems with locational constraints - the polyhedral gauge case, Math. Oper. Res., 22 (1997), 297–300.
[3] M. Cera, J. A. Mesa, F. A. Ortega and F. Plastria, Locating a central hunter on the plane, J. Optim. Theory Appl., 136 (2008), 155–166.
[4] F. Daneshzand and R. Shoeleh, Multifacility location problem, in Facility Location: Concepts, Models, Algorithms and Case Studies, R.Z. Farahani and M. Hekmatfar (eds.), Springer, Berlin, 2009, 69–92.
[5] Z. Drezner, Facility Location: A Survey of Applications and Methods, Springer, New York, 1995.
[6] Z. Drezner and H. W. Hamacher, Facility Location: Applications and Theory, Springer-Verlag, Berlin, 2002.
[7] B. S. He, A new method for a class of linear variational inequalities, Math. Program., 66 (1994), 137–144.
[8] B. S. He, A modified projection and contraction method for a class of linear complementarity problems, J. Comput. Math., 14 (1996), 54–63.
[9] B. S. He, X. M. Yuan and W. X. Zhang, A customized proximal point algorithm for convex minimization with linear constraints, Comput. Optim. Appl., 56 (2013), 559–572.
[10] I. N. Katz and S. R. Vogl, A Weiszfeld algorithm for the solution of an asymmetric extension of the generalized Fermat location problem, Comput. Math. Appl., 59 (2010), 399–410.
[11] R. F. Love and J. G. Morris, Mathematical models of road travel distances, Manag. Sci., 25 (1979), 130–139.
[12] R. F. Love and J. G. Morris, On estimating road distances by mathematical functions, Euro. J. Oper. Res., 36 (1988), 251–253.
[13] R. F. Love, J. G. Morris and G. O. Wesolowsky, Facilities Location: Models and Methods, North-Holland, Amsterdam, 1988.
[14] B. Martinet, Regularisation d’inéquations variationnelles par approximations successives, Revue Francaise d’Automatique et Informatique Recherche Opérationnelle, 4 (1970), 154–158.
[15] W. Miehle, Link length minimization in networks, Oper. Res., 6 (1958), 232–243.
[16] H. Minkowski, Theorie der konvezen Körper, Gesammelte Abhandlungen, Teubner, Berlin, 1911.
[17] S. Nickel, Restricted center problems under polyhedral gauges, Euro. J. Oper. Res., 104 (1998), 343–357.
[18] L. M. Ostresh, The multifacility location problem: Applications and descent theorems, J. Regional Sci., 17 (1977), 409–419.
[19] F. Plastria, Asymmetric distances, semidirected networks and majority in Fermat-Weber problems, Ann. Oper. Res., 167 (2009), 121–155.
[20] F. Plastria, The Weiszfeld algorithm: proof, amendments, and extensions, in Foundations of Location Analysis, H.A. Eiselt and V. Marianov (eds.), Springer, US, 155 (2011), 357–389.
[21] J. B. Rosen and G. L. Xue, On the convergence of Miehle’s algorithm for the Euclidean multifacility location problem, Oper. Res., 40 (1992), 188–191.
[28] M. V. Solodov and P. Tseng, Modified projection-type methods for monotone variational inequalities, *SIAM J. Control Optim.*, 34 (1996), 1814–1830.

[29] H. Uzawa, Iterative methods for concave programming, in *Studies in Linear and Nonlinear Programming*, K.J. Arrow, L. Hurwicz and H. Uzawa (eds.), Stanford University Press, Stanford, 1958, 154–165.

[30] J. E. Ward and R. E. Wendell, A new norm for measuring distance which yields linear location problems, *Oper. Res.*, 28 (1980), 836–844.

[31] J. E. Ward and R. E. Wendell, Using block norms for location modelling, *Oper. Res.*, 33 (1985), 1074–1090.

[32] E. Weiszfeld, Sur le point pour lequel la somme des distances de n points donnés est minimum, *Tohoku Math. J.*, 43 (1937), 355–386.

[33] C. Witzgall, Optimal location of a central facility, mathematical models and concepts, Report 8388, National Bureau of Standards, Washington DC, US, 1964.

Received February 2016; revised May 2017.

E-mail address: jiangjianlin@nuaa.edu.cn
E-mail address: nuaazhangs@126.com
E-mail address: zhangsu@nankai.edu.cn
E-mail address: wenjie@nuaa.edu.cn