

Solving a Continuous Multifacility Location Problem by DC Algorithms

ANUJ BAJAJ¹, BORIS S. MORDUKHOVICH², NGUYEN MAU NAM³, and TUYEN TRAN⁴.

Abstract. The paper presents a new approach to solve multifacility location problems, which is based on mixed integer programming and algorithms for minimizing differences of convex (DC) functions. The main challenges for solving the multifacility location problems under consideration come from their intrinsic discrete, nonconvex, and nondifferentiable nature. We provide a reformulation of these problems as those of continuous optimization and then develop a new DC type algorithm for their solutions involving Nesterov's smoothing. The proposed algorithm is computationally implemented via MATLAB numerical tests on both artificial and real data sets.

Key words. Mixed integer programming, multifacility location, difference of convex functions, Nesterov's smoothing, the DCA

AMS subject classifications. 49J52, 49J53, 90C31

1 Introduction and Problem Formulation

In the 17th century, Pierre de Fermat posed a problem of finding for a point that minimizes the sum of its Euclidean distances to three given points in the plane. The problem was soon solved by Evangelista Torricelli, and it is now known as the *Fermat-Torricelli problem*. This problem and its extended version that involves a finite number of points in higher dimensions are examples of continuous *single facility* location problems. Over the years several generalized models of the Fermat-Torricelli type have been introduced and studied in the literature with practical applications to facility location decisions; see [8, 12, 14, 17, 19, 20] and the references therein. An important feature of single facility location problems and the problems studied in the aforementioned references is that only one *center/server* has to be found to serve a finitely many *demand points/customers*.

However, numerous practical applications lead to formulations of facility location problems in which more than one center must be found to serve a finite number of demand points. Such problems are referred to as *multifacility location problems* (MFLPs). Given a finite number of demand points a_1, \dots, a_n in \mathbb{R}^d , we consider here the facility location in which k centers v_1, \dots, v_k ($1 \leq k \leq n$) in \mathbb{R}^d need to be found to serve these demand points by assigning each of them to its nearest center and minimizing the total distances from the centers to the assigned demand points. In the case where $k = 1$, this problem reduces to the generalized Fermat-Torricelli problem of finding a point that minimizes the sum of the distances to a finite number of given points in \mathbb{R}^d .

¹ Department of Mathematics, Wayne State University, Detroit, Michigan 48202, USA (anuj.bajaj@wayne.edu). Research of this author was partly supported by the USA National Science Foundation under grant DMS-1808978 and by the USA Air Force Office of Scientific Research grant #15RT04.

² Department of Mathematics, Wayne State University, Detroit, Michigan 48202, USA (boris@math.wayne.edu). Research of this author was partly supported by the USA National Science Foundation under grants DMS-1512846 and DMS-1808978, by the USA Air Force Office of Scientific Research grant #15RT04, and by Australian Research Council under grant DP-190100555.

³ Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland, OR 97207, USA (mnn3@pdx.edu). Research of this author was partly supported by the USA National Science Foundation under grant DMS-1716057.

⁴ Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland, OR 97207, USA (tuyen2@pdx.edu).

Let us formulate the problem under consideration in this paper as the following problem of *mixed integer programming* with *nonsmooth* objective functions. It is convenient to use a variable $k \times d$ -matrix \mathbf{V} with v_i as its i th row to store the centers to be found. We also use another variable $k \times n$ -matrix $\mathbf{U} = [u_{i,j}]$ with $u_{i,j} \in \{0, 1\}$ and $\sum_{i=1}^k u_{i,j} = 1$ for $j = 1, \dots, n$ to assign demand points to the centers. The set of all such matrices is denoted by \mathcal{U} . Note that $u_{i,j} = 1$ if the center v_i is assigned to the demand point a_j while $\sum_{i=1}^k u_{i,j} = 1$ means that the demand point a_j is assigned to only one center. Our goal is to solve the constrained optimization problem formulated as follows:

$$\begin{aligned} & \text{minimize } \mathcal{F}(\mathbf{U}, \mathbf{V}) := \sum_{i=1}^k \sum_{j=1}^n u_{i,j}^2 \|a_j - v_i\| \\ & \text{subject to } \mathbf{U} \in \mathcal{U} \text{ and } \mathbf{V} \in \mathbb{R}^{k \times d}. \end{aligned} \quad (1.1)$$

Taking into account that $u_{i,j} \in \{0, 1\}$, it is convenient to use $u_{i,j}^2$ instead of $u_{i,j}$ in the definition of the objective function \mathcal{F} ; See Section 4.

Note that a similarly looking problem was considered by An, Minh and Tao [4] for different purposes. The main difference between our problem (1.1) and the one from [4] is that in [4] the squared Euclidean norm is used instead of the Euclidean norm in our formulation. From the point of applications this difference is significant; namely, using the Euclidean norm allows us to model the total distance in *supply delivery*, while using the squared Euclidean norm is meaningful in *clustering*. Mathematically these two problems are essentially different as well. In addition to the challenging discrete nature and nonconvexity that both problems share, the objective function of our multifacility location problem (1.1) is *non-differentiable* in contrast to [4]. This is yet another serious challenge from both theoretical and algorithmic viewpoints. Observe also that for $k = 1$ our problem becomes the aforementioned generalized Fermat-Torricelli problem that does not have a closed-form solution, while the problem considered in [4] reduces to the standard problem of minimizing the sum of squares of the Euclidean distances to the demand points. The latter has a simple closed form solution given by the mean of the data points.

In this paper we develop the following algorithmic procedure to solve the formulated nonsmooth problem (1.1) of mixed integer programming:

- (i) Employ *Nesterov's smoothing* to approximate the nonsmooth objective function in (1.1) by a family of *smooth* functions, which are represented as *differences of convex* ones.
- (ii) Enclose the obtained smooth discrete problems into constrained problems of *continuous DC optimization* and then approximate them by unconstrained ones while using *penalties*.
- (iii) Solve the latter class of problems by developing an appropriate modification of the algorithm for minimizing differences of convex functions known as the *DCA*.

As a result of all the three steps above, we propose a *new algorithm* for solving the class of multifacility local problems of type (1.1), verify its efficiency and implementation with MATLAB numerical tests on both artificial and real data sets.

Recall that the early developments on the DCA trace back to the work by Tao in 1986 with more recent results presented in [2]–[5], [25], [26], and the bibliographies therein. Nesterov's smoothing technique was introduced in his seminal paper [21] and was further developed and applied in many great publications; see, e.g., [22, 23] for more details and references. The combination of these two important tools provides an effective way to deal with nonconvexity and nondifferentiability in many optimization problems encountered in

facility location, machine learning, compressed sensing, and imaging. It is demonstrated in this paper in solving multifacility location problems of type (1.1).

The rest of this paper is organized as follows. Section 2 contains the basic definitions and some preliminaries, which are systematically employed in the text. In Section 3 we briefly overview two versions of the DCA, discuss their convergence, and present two examples that illustrate their performances.

Section 4 is devoted to applying Nesterov’s smoothing technique to the objective function of the multifacility location problem (1.1) and constructing in this way a smooth approximation of the original problem by a family of DC ones. Further, we reduce the latter smooth DC problems of discrete constrained optimization to unconstrained problems by using an appropriate penalty function method. Finally, the obtained discrete optimization problems are enclosed here into the DC framework of unconstrained continuous optimization.

In Section 5 we proposed, based on the above developments, a new algorithm to solve the multifacility location problem (1.1) by applying the updated version of the DCA taken from Section 3 to the smooth DC problems of continuous optimization constructed in Section 4. The proposed algorithm is implemented in this section to solving several multifacility problems arising in practical modeling. Section 6 summarizes the obtained results and discusses some directions of future research.

2 Basic Definitions and Preliminaries

For the reader’s convenience, in this section we collect those basic definitions and preliminaries, which are largely used throughout the paper; see the books [9, 15, 24] for more details and proofs of the presented results.

Consider the difference of two convex functions $g - h$ on a finite-dimensional space and assume that $g: \mathbb{R}^d \rightarrow \overline{\mathbb{R}} := (-\infty, \infty]$ is extended-real-valued while $h: \mathbb{R}^d \rightarrow \mathbb{R}$ is real-valued on \mathbb{R}^d . Then a general problem of *DC optimization* is defined by:

$$\text{minimize } f(x) := g(x) - h(x), \quad x \in \mathbb{R}^d. \quad (2.1)$$

Note that problem (2.1) is written in the unconstrained format, but—due to the allowed infinite value for g —it actually contains the domain constraint $x \in \text{dom}(g) := \{u \in \mathbb{R}^d \mid g(u) < \infty\}$. Furthermore, the explicit constraints of the type $x \in \Omega$ given by a nonempty convex set $\Omega \subset \mathbb{R}^d$ can be incorporated into the format of (2.1) via the indicator function $\delta_\Omega(x)$ of Ω , which equals 0 for $x \in \Omega$ and ∞ otherwise. The representation $f = g - h$ is called a *DC decomposition* of f . Note that the class of DC functions is fairly large and include many nonconvex functions important in optimization. We refer the reader to the recent book [13] with the commentaries and bibliographies therein for various classes of nonconvex optimization problems that can be represented in the DC framework (2.1).

Considering a nonempty (may not be convex) set $\Omega \subset \mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, define the *Euclidean projection* of x to Ω by

$$P(x; \Omega) := \{w \in \Omega \mid \|x - w\| = d(x; \Omega)\}, \quad (2.2)$$

where $d(x; \Omega)$ stands for the *Euclidean distance* from x to Ω , i.e.,

$$d(x; \Omega) := \inf \{\|x - w\| \mid w \in \Omega\}. \quad (2.3)$$

Observe that $P(x; \Omega) \neq \emptyset$ for closed sets Ω while being always a singleton if the set Ω is convex.

Given further an extended-real-valued and generally nonconvex function $\varphi: \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$, the *Fenchel conjugate* of φ is defined by

$$\varphi^*(v) := \sup \{ \langle v, x \rangle - \varphi(x) \mid x \in \mathbb{R}^d \}, \quad v \in \mathbb{R}^d.$$

If φ is *proper*, i.e., $\text{dom}(\varphi) \neq \emptyset$, its Fenchel conjugate $\varphi^*: \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$ is automatically convex.

The *subdifferential* of $\varphi: \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$ at $\bar{x} \in \text{dom}(\varphi)$ is the set of subgradients given by

$$\partial\varphi(\bar{x}) := \{ v \in \mathbb{R}^d \mid \langle v, x - \bar{x} \rangle \leq \varphi(x) - \varphi(\bar{x}) \text{ for all } x \in \mathbb{R}^d \}. \quad (2.4)$$

If $\bar{x} \notin \text{dom}(\varphi)$, we let $\partial\varphi(\bar{x}) := \emptyset$. Recall that for functions φ differentiable at \bar{x} with the gradient $\nabla\varphi(\bar{x})$ we have $\partial\varphi(\bar{x}) = \{\nabla\varphi(\bar{x})\}$.

The following proposition gives us a two-sided relationship between the Fenchel conjugates and subgradients of convex functions.

Proposition 2.1 *Let $\varphi: \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$ be a proper, lower semicontinuous, and convex function. Then $v \in \partial\varphi^*(y)$ if and only if*

$$v \in \text{argmin} \{ \varphi(x) - \langle y, x \rangle \mid x \in \mathbb{R}^d \}. \quad (2.5)$$

We have furthermore that $w \in \partial\varphi(x)$ if and only if

$$w \in \text{argmin} \{ \varphi^*(y) - \langle x, y \rangle \mid y \in \mathbb{R}^d \}. \quad (2.6)$$

Proof. To verify the first assertion, suppose that (2.5) is satisfied and then get $0 \in \partial\psi(v)$, where $\psi(x) := \varphi(x) - \langle y, x \rangle$ as $x \in \mathbb{R}^d$. It tells us that

$$0 \in \partial\varphi(v) - y,$$

and hence $y \in \partial\varphi(v)$, which is equivalent to $v \in \partial\varphi^*(y)$ due to the biconjugate relationship $\varphi^{**} = \varphi$ valued under the assumptions made.

In the opposite way, assuming $v \in \partial\varphi^*(y)$ gives us by the proof above that $0 \in \partial\psi(v)$, which clearly yields (2.5) and thus justifies the first assertion.

To verify the second assertion, suppose that (2.6) holds and then get $0 \in \partial\psi(w)$, where $\psi(y) := \varphi^*(y) - \langle x, y \rangle$ as $y \in \mathbb{R}^d$. This clearly implies that

$$0 \in \partial\varphi^*(w) - x,$$

and hence $x \in \partial\varphi^*(w)$, which is equivalent to $w \in \partial\varphi(x)$ due to the biconjugate relationship. The proof of the opposite implication in (2.6) is similar to the one given above. \square

Finally in this section, recall that for a given variable matrix $\mathbf{U} \in \mathbb{R}^{k \times n}$ as in the optimization problem (1.1), the *Frobenius norm* on $\mathbb{R}^{k \times n}$ is defined by

$$\|\mathbf{U}\|_F := \sqrt{\sum_{i=1}^k \sum_{j=1}^n |u_{i,j}|^2}. \quad (2.7)$$

3 Overview of the DCA and Some Examples

In this section we first briefly overview two algorithms of the DCA type to solve DC problems (2.1) while referring the reader to [25, 26] for more details and further developments. Then we present numerical examples illustrating both algorithms.

Algorithm 1: DCA-1.

```

INPUT:  $x_0 \in \mathbb{R}^d$ ,  $N \in \mathbb{N}$ .
for  $l = 1, \dots, N$  do
    Find  $y_{l-1} \in \partial h(x_{l-1})$ .
    Find  $x_l \in \partial g^*(y_{l-1})$ .
end for
OUTPUT:  $x_N$ .

```

Since the convex function $h: \mathbb{R}^d \rightarrow \mathbb{R}$ in (2.1) is real-valued on the whole space \mathbb{R}^d , we always have $\partial h(x) \neq \emptyset$ for all $x \in \mathbb{R}^d$. At the same time, the other convex function $g: \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$ in (2.1) is generally extended-real-valued, and so the subdifferential of its conjugate g^* may be empty. Let us present an efficient condition that excludes this possibility. Recall that a function $g: \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$ is *coercive* if

$$\lim_{\|x\| \rightarrow \infty} \frac{g(x)}{\|x\|} = \infty.$$

Proposition 3.1 *Let $g: \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$ be a proper, lower semicontinuous, and convex function. If in addition g is coercive, then $\partial g^*(v) \neq \emptyset$ for all $v \in \mathbb{R}^d$.*

Proof. Since g is proper, the conjugate function g^* takes values in $(-\infty, \infty]$ being convex on \mathbb{R}^d . Taking into account that g is also lower semicontinuous and invoking the aforementioned biconjugate relationship, we find $w \in \mathbb{R}^d$ and $c \in \mathbb{R}$ such that

$$c + \langle w, x \rangle \leq g(x) \quad \text{for all } x \in \mathbb{R}^d. \quad (3.1)$$

The coercivity property of g ensures the existence of $\eta > 0$ for which

$$\|x\|(\|w\| + 1) \leq g(x) \quad \text{whenever } \|x\| \geq \eta.$$

It follows furthermore that

$$\sup \{ \langle v, x \rangle - g(x) \mid \|x\| \geq \eta \} \leq -\|x\| \quad \text{for any } v \in \mathbb{R}^d.$$

By using (3.1), we arrive at the estimates

$$\sup \{ \langle v, x \rangle - g(x) \mid \|x\| \leq \eta \} \leq \sup \{ \langle v, x \rangle - \langle w, x \rangle - c \mid \|x\| \leq \eta \} < \infty, \quad v \in \mathbb{R}^d.$$

This tells us that $g^*(v) < \infty$, and therefore $\text{dom}(g^*) = \mathbb{R}^d$. Since g^* is a convex function with finite values, it is continuous on \mathbb{R}^d and hence $\partial g^*(v) \neq \emptyset$ for all $v \in \mathbb{R}^d$. \square

To proceed further, recall that a function $\varphi: \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$ is γ -convex with a given modulus $\gamma \geq 0$ if the function $\psi(x) := \varphi(x) - \frac{\gamma}{2}\|x\|^2$ as $x \in \mathbb{R}^d$ is convex on \mathbb{R}^d . If there exists $\gamma > 0$ such that φ is γ -convex, then φ is called *strongly convex* on \mathbb{R}^d .

We also recall that a vector $\bar{x} \in \mathbb{R}^d$ is a *stationary point* of the DC function f from (2.1) if

$$\partial g(\bar{x}) \cap \partial h(\bar{x}) \neq \emptyset.$$

The next result, which can be derived from [25, 26], summarizes some convergence results of the DCA. Deeper studies of the convergence of this algorithm and its generalizations involving the Kurdyka-Lojasiewicz (KL) inequality are given in [2, 3].

Theorem 3.2 *Let f be a DC function taken from (2.1), and let $\{x_l\}$ be an iterative sequence generated by Algorithm 1. The following assertions hold:*

- (a) *The sequence $\{f(x_l)\}$ is always monotone decreasing.*
- (b) *Suppose that f is bounded from below, that g is lower semicontinuous and γ_1 -convex, and that h is γ_2 -convex with $\gamma_1 + \gamma_2 > 0$. If $\{x_l\}$ is bounded, then the limit of any convergent subsequence of $\{x_l\}$ is a stationary point of f .*

In many practical applications of Algorithm 1, for a given DC decomposition of f it is possible to find subgradient vectors from $\partial h(x_l)$ based on available formulas and calculus rules of convex analysis. However, it may not be possible to explicitly calculate an element of $\partial g^*(y_l)$. Such a situation requires either constructing a more suitable DC decomposition of f , or finding $x_{l+1} \in \partial g^*(y_l)$ by using the description of Proposition 2.1. This leads us to the following modified version of the DCA.

Algorithm 2: DCA-2.

```

INPUT:  $x_0 \in \mathbb{R}^d, N \in \mathbb{N}$ 
for  $l = 1, \dots, N$  do
    Find  $y_{l-1} \in \partial h(x_{l-1})$ 
    Find  $x_l$  by solving the problem:
        minimize  $\varphi_l(x) := g(x) - \langle y_{l-1}, x \rangle, x \in \mathbb{R}^d$ .
end for
OUTPUT:  $x_N$ 

```

Let us now present two examples illustrating the performances of Algorithms 1 and 2. The first example concerns a polynomial function of one variable.

Example 3.3 Consider the function $f: \mathbb{R} \rightarrow \mathbb{R}$ given

$$f(x) := x^4 - 2x^2 + 2x - 3 \text{ for } x \in \mathbb{R}.$$

This function admits the DC representation $f = g - h$ with $g(x) := x^4$ and $h(x) := 2x^2 - 2x + 3$. To minimize f , apply first the *gradient method* with constant stepsize. Calculating the derivative of f is $f'(x) = 4x^3 - 4x + 2$ and picking any starting point $x_0 \in \mathbb{R}$, we get the sequence of iterates

$$x_{l+1} = x_l - t(4x_l^3 - 4x_l + 2) \text{ for } l = 0, 1, \dots$$

constructed by the gradient method with stepsize $t > 0$. The usage of the DC *Algorithm 1* (DCA-1) gives us $y_l = \nabla h(x_l) = 4x_l - 2$ and then $g^*(x) = 3(x/4)^{4/3}$ with $\nabla g^*(x) = (x/4)^{1/3}$. Thus the iterates of DCA-1 are as follows:

$$x_{l+1} = \nabla g^*(y_l) = \left(\frac{y_l}{4}\right)^{1/3} = \left(\frac{4x_l - 2}{4}\right)^{1/3} = \left(\frac{2x_l - 1}{2}\right)^{1/3}, \quad l = 0, 1, \dots$$

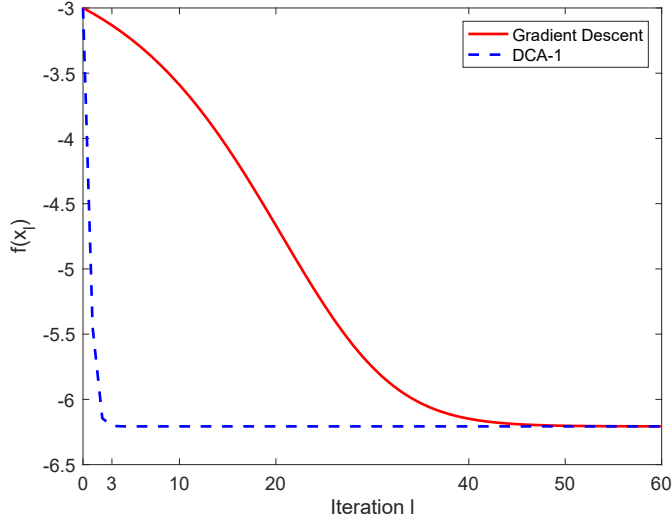


Figure 3.1: Convergence of the gradient method and DCA-1.

Figure 3.1 provides the visualization and comparison between the DCA-1 and the gradient method. It shows that for $x_0 = 0$ and $t = 0.01$ the DCA-1 exhibits much faster convergence.

The next two-dimensional example illustrates the performance of the DCA-2.

Example 3.4 Consider the nonsmooth optimization problem defined by

$$\text{minimize } f(x_1, x_2) := x_1^4 + x_2^2 - 2x_1^2 - |x_2| \text{ over } x = (x_1, x_2) \in \mathbb{R}^2.$$

The graph of the function f is depicted in Figure 3.2. Observe that this function has four global minimizers, which are $(1, 0.5)$, $(1, -0.5)$, $(-1, 0.5)$, and $(-1, -0.5)$. It is easy to see that f admits a DC representation $f = g - h$ with $g(x_1, x_2) := x_1^4 + x_2^2$ and $h(x_1, x_2) := 2x_1^2 + |x_2|$. We get the gradient $\nabla g(x) = [4x_1^3, 2x_2]^T$ and the Hessian

$$\nabla^2 g(x) = \begin{bmatrix} 12x_1^2 & 0 \\ 0 & 2 \end{bmatrix},$$

while an explicit formula to calculate $\partial g^*(y_l)$ is not available. Let us apply the DCA-2 to solve this problem. The subdifferential of h is calculated by

$$\partial h(x) = [4x_1, \text{sign}(x_2)]^T \text{ for any } x = (x_1, x_2) \in \mathbb{R}^2.$$

Having y_l , we proceed with solving the subproblem

$$\text{minimize } \varphi_l(x) := g(x) - \langle y_l, x \rangle \text{ over } x \in \mathbb{R}^2 \quad (3.2)$$

by the classical Newton method with $\nabla^2 \varphi_l(x) = \nabla^2 g(x)$ and observe that the DCA-2 shows its superiority in convergence with different choices of initial points. Figure 3.3 presents the results of computation by using the DCA-2 with the starting point $x_0 = (-2, 2)$ and employing the Newton method with $\varepsilon = 10^{-8}$ to solve subproblem (3.2).

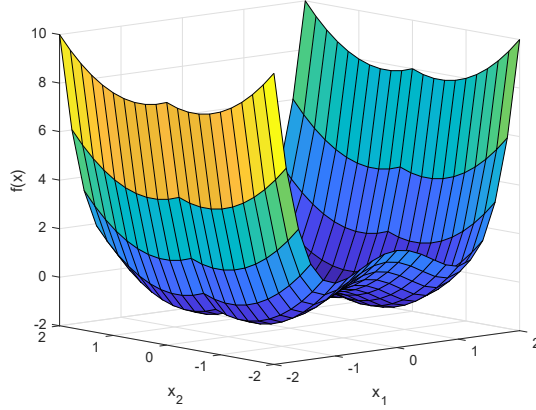


Figure 3.2: Function graph for Example 3.4.

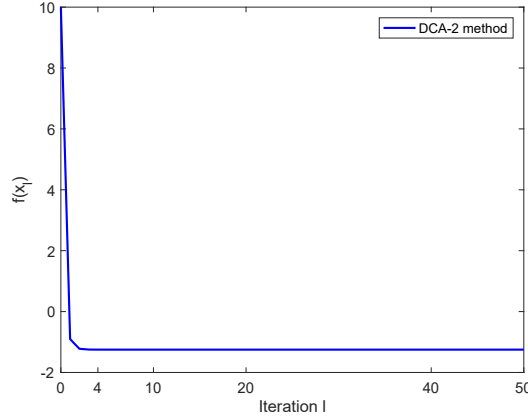


Figure 3.3: Convergence of the DCA-2.

4 Smooth Approximation by Continuous DC Problems

In this section we first employ and further develop Nesterov's smoothing technique for the case of multifacility location problem (1.1). Then we enclose the family of DC mixed integer programs obtained in this way into a class of smooth DC problems of continuous optimization. The suggested procedures are efficiently justified by deriving numerical estimates expressed entirely via the given data of the original problem (1.1).

We begin with recalling the following useful result on Nesterov's smoothing related to the problem under consideration, which is taken from [18, Proposition 2.2].

Proposition 4.1 *Given any $a \in \mathbb{R}^d$ and $\mu > 0$, a Nesterov smoothing approximation of the function $f: \mathbb{R}^d \rightarrow \mathbb{R}$ defined by*

$$f(x) := \|x - a\|, \quad x \in \mathbb{R}^d,$$

admits the smooth DC representation

$$f_\mu(x) := \frac{1}{2\mu} \|x - a\|^2 - \frac{\mu}{2} \left[d\left(\frac{x - a}{\mu}; \mathbb{B}\right) \right]^2.$$

Furthermore, we have the relationships

$$\nabla f_\mu(x) = P\left(\frac{x - a}{\mu}; \mathbb{B}\right) \quad \text{and} \quad f_\mu(x) \leq f(x) \leq f_\mu(x) + \frac{\mu}{2},$$

where $\mathbb{B} \subset \mathbb{R}^d$ is the closed unit ball, and where P stands for the Euclidean projection (2.2).

Using Proposition 4.1 allows us to approximate the objective function \mathcal{F} in (1.1) by a smooth DC function \mathcal{F}_μ as $\mu > 0$ defined as follows:

$$\begin{aligned} \mathcal{F}_\mu(\mathbf{U}, \mathbf{V}) &:= \frac{1}{2\mu} \sum_{i=1}^k \sum_{j=1}^n u_{i,j}^2 \|a_j - v_i\|^2 - \frac{\mu}{2} \sum_{i=1}^k \sum_{j=1}^n u_{i,j}^2 \left[d\left(\frac{a_j - v_i}{\mu}; \mathbb{B}\right) \right]^2 \\ &= \mathcal{G}_\mu(\mathbf{U}, \mathbf{V}) - \mathcal{H}_\mu(\mathbf{U}, \mathbf{V}), \end{aligned}$$

where $\mathcal{G}_\mu, \mathcal{H}_\mu: \mathbb{R}^{k \times n} \times \mathbb{R}^{k \times d} \rightarrow \mathbb{R}$ are given by

$$\begin{aligned} \mathcal{G}_\mu(\mathbf{U}, \mathbf{V}) &:= \frac{1}{2\mu} \sum_{i=1}^k \sum_{j=1}^n u_{i,j}^2 \|a_j - v_i\|^2, \\ \mathcal{H}_\mu(\mathbf{U}, \mathbf{V}) &:= \frac{\mu}{2} \sum_{i=1}^k \sum_{j=1}^n u_{i,j}^2 \left[d\left(\frac{a_j - v_i}{\mu}; \mathbb{B}\right) \right]^2. \end{aligned}$$

This leads us to the construction of the following family of smooth approximations of the main problem (1.1) defined by

$$\begin{aligned} &\text{minimize } \mathcal{F}_\mu(\mathbf{U}, \mathbf{V}) := \mathcal{G}_\mu(\mathbf{U}, \mathbf{V}) - \mathcal{H}_\mu(\mathbf{U}, \mathbf{V}) \quad \text{as } \mu > 0 \\ &\text{subject to } \mathbf{U} \in \mathcal{U} = \Delta^n \cap \{0, 1\}^{k \times n} \quad \text{and} \quad \mathbf{V} \in \mathbb{R}^{k \times d}, \end{aligned} \tag{4.1}$$

where Δ^n is the the n th Cartesian degree of the $(k-1)$ -simplex $\Delta := \{y \in [0, 1]^k \mid \sum_{i=1}^k y_i = 1\}$, which is a subset of \mathbb{R}^k .

Observe that for each $\mu > 0$ problem (4.1) is of *discrete optimization*, while our intention is to convert it to a family of problems of *continuous optimization* for which we are going to develop and implement a DCA-based algorithm in Section 5.

The rest of this section is devoted to deriving two results, which justify such a reduction. The first theorem allows us to verify the *existence* of optimal solutions to the constrained optimization problems that appear in this procedure. It is required for having well-posedness of the algorithm construction.

Theorem 4.2 Let $(\bar{\mathbf{U}}, \bar{\mathbf{V}})$ be an optimal solution to problem (4.1). Then for any $\mu > 0$ we have $\bar{\mathbf{V}} \in \mathcal{B}$, where $\mathcal{B} := \prod_{i=1}^k B_i$ is the Cartesian product of the k Euclidean balls B_i centered at $0 \in \mathbb{R}^d$ with radius $r := \sqrt{\sum_{j=1}^n \|a_j\|^2}$ that contain the optimal centers \bar{v}_i for each index $i = 1, \dots, k$.

Proof. We can clearly rewrite the objective function in (4.1) in the form

$$\mathcal{F}_\mu(\mathbf{U}, \mathbf{V}) = \frac{1}{2\mu} \sum_{i=1}^k \sum_{j=1}^n u_{i,j} \|a_j - v_i\|^2 - \frac{\mu}{2} \sum_{i=1}^k \sum_{j=1}^n u_{i,j} \left[d\left(\frac{a_j - v_i}{\mu}; \mathbb{B}\right) \right]^2 \quad (4.2)$$

due to interchangeability between $u_{i,j}^2$ and $u_{i,j}$. Observe that $\mathcal{F}_\mu(\mathbf{U}, \mathbf{V})$ is differentiable on $R^{k \times n} \times \mathbb{R}^{k \times d}$. Employing the classical Fermat rule in (4.1) with respect to V gives us $\nabla_{\mathbf{V}} \mathcal{F}_\mu(\bar{\mathbf{U}}, \bar{\mathbf{V}}) = 0$. To calculate this partial gradient, we need some clarification for the second term in (4.2), which is differentiable as a whole while containing the nonsmooth distance function (2.3). The convexity of the distance function in the setting of (4.2) allows us to apply the subdifferential calculation of convex analysis (see, e.g., [15, Theorem 2.39]) and to combine it with an appropriate chain rule to handle the composition in (4.2). Observe that the distance function square in (4.2) is the composition of the nondecreasing convex function $\varphi(t) := t^2$ on $[0, \infty)$ and the distance function to the ball \mathbb{B} . Thus the chain rule from [15, Corollary 2.62] is applicable. Thus, we can show that $d^2(\cdot; \mathbb{B})$ is differentiable with

$$\nabla d^2(x; \mathbb{B}) = 2[x - P(x; \mathbb{B})] \text{ for } x \in \mathbb{R}^d. \quad (4.3)$$

Using (4.3), we consider the following two cases:

Case 1: $(a_j - \bar{v}_i)/\mu \in \mathbb{B}$ for the fixed indices $i \in \{1, \dots, k\}$ and $j \in \{1, \dots, n\}$. Then

$$\nabla d^2\left(\frac{a_j - \bar{v}_i}{\mu}; \mathbb{B}\right) = \{0\},$$

which gives us

$$\frac{\partial \mathcal{F}_\mu}{\partial v_i}(\bar{\mathbf{U}}, \bar{\mathbf{V}}) = \frac{1}{\mu} \sum_{j=1}^n \bar{u}_{i,j} (\bar{v}_i - a_j), \quad i = 1, \dots, k,$$

for the corresponding partial derivatives of \mathcal{F}_μ .

Case 2: $(a_j - \bar{v}_i)/\mu \notin \mathbb{B}$ for the fixed indices $i \in \{1, \dots, k\}$ and $j \in \{1, \dots, n\}$. In this case we have

$$\begin{aligned} \frac{\partial \mathcal{F}_\mu}{\partial v_i}(\bar{\mathbf{U}}, \bar{\mathbf{V}}) &= \frac{1}{2\mu} \sum_{j=1}^n \bar{u}_{i,j} 2(\bar{v}_i - a_j) + \sum_{j=1}^n \bar{u}_{i,j} \left[\frac{a_j - \bar{v}_i}{\mu} - P\left(\frac{a_j - \bar{v}_i}{\mu}; \mathbb{B}\right) \right] \\ &= \frac{1}{\mu} \sum_{j=1}^n \bar{u}_{i,j} (\bar{v}_i - a_j) + \sum_{j=1}^n \bar{u}_{i,j} \left[\frac{a_j - \bar{v}_i}{\mu} - \left(\frac{a_j - \bar{v}_i}{\|a_j - \bar{v}_i\|} \right) \right] \\ &= \frac{1}{\|a_j - \bar{v}_i\|} \sum_{j=1}^n \bar{u}_{i,j} (\bar{v}_i - a_j). \end{aligned}$$

Thus in both cases above it follows from the stationary condition $\nabla_{\mathbf{V}} \mathcal{F}_\mu(\bar{\mathbf{U}}, \bar{\mathbf{V}}) = 0$ that

$$\bar{v}_i = \frac{\sum_{j=1}^n \bar{u}_{i,j} a_j}{\sum_{j=1}^n \bar{u}_{i,j}} \text{ for all } i = 1, \dots, k,$$

since we have $\sum_{j=1}^n \bar{u}_{i,j} > 0$ due to the nonemptiness of the clusters. Then the classical Cauchy-Schwarz inequality leads us to the estimates

$$\|\bar{v}_i\|^2 \leq \frac{\left(\sum_{j=1}^n \bar{u}_{i,j} a_j\right)^2}{\left(\sum_{j=1}^n \bar{u}_{i,j}\right)^2} \leq \sum_{j=1}^n \|a_j\|^2 := r^2,$$

which therefore verify all the conclusions of this theorem. \square

Our next step is to enclose each discrete optimization problem (4.1) into the corresponding one of *continuous optimization*. For the reader's convenience if no confusion arises, we keep the same notation \mathbf{U} for *all* the $k \times n$ -matrices without the discrete restrictions on their entries. Define now the function $\mathcal{P}: \mathbb{R}^{k \times n} \rightarrow \mathbb{R}$ by

$$\mathcal{P}(\mathbf{U}) := \sum_{i=1}^k \sum_{j=1}^n u_{i,j}(1 - u_{i,j}) \text{ for all } \mathbf{U} \in \mathbb{R}^{k \times n}$$

and observe that this function is concave on $\mathbb{R}^{k \times n}$ with $\mathcal{P}(\mathbf{U}) \geq 0$ whenever $\mathbf{U} \in \Delta^n$. Furthermore, we have the representations

$$\mathcal{U} = \{\mathbf{U} \in \Delta^n \mid \mathcal{P}(\mathbf{U}) = 0\} = \{\mathbf{U} \in \Delta^n \mid \mathcal{P}(\mathbf{U}) \leq 0\} \quad (4.4)$$

for the set of feasible $k \times n$ -matrices \mathcal{U} in the original problem (1.1). Employing further the standard *penalty function* method allows us to eliminate the most involved constraint on \mathbf{U} in (4.4) given by the function \mathcal{P} . Taking the penalty parameter $\alpha > 0$ sufficiently large and using the smoothing parameter $\mu > 0$ sufficiently small, consider the following family of continuous optimization problems:

$$\begin{aligned} & \text{minimize } \mathcal{F}_\mu(\mathbf{U}, \mathbf{V}) + \alpha \mathcal{P}(\mathbf{U}) = \mathcal{G}_\mu(\mathbf{U}, \mathbf{V}) - \mathcal{H}_\mu(\mathbf{U}, \mathbf{V}) + \alpha \mathcal{P}(\mathbf{U}) \\ & \text{subject to } \mathbf{U} \in \Delta^n \text{ and } \mathbf{V} \in \mathcal{B}. \end{aligned} \quad (4.5)$$

Observe that Theorem 4.2 ensures the existence of feasible solutions to problem (4.5) and hence optimal solutions to this problem by the Weierstrass theorem due to the continuity of the objective functions therein and the compactness of the constraints sets Δ^n and \mathcal{B} .

Let us introduce yet another parameter $\rho > 0$ ensuring a DC representation of the objective function in (4.5) as follows:

$$\begin{aligned} \mathcal{F}_\mu(\mathbf{U}, \mathbf{V}) + \alpha \mathcal{P}(\mathbf{U}) &= \frac{\rho}{2} \|\mathbf{U}, \mathbf{V}\|^2 - \left(\frac{\rho}{2} \|\mathbf{U}, \mathbf{V}\|^2 - \mathcal{F}_\mu(\mathbf{U}, \mathbf{V}) - \alpha \mathcal{P}(\mathbf{U}) \right) \\ &= \frac{\rho}{2} \|\mathbf{U}, \mathbf{V}\|^2 - \left(\frac{\rho}{2} \|\mathbf{U}, \mathbf{V}\|^2 - \mathcal{G}_\mu(\mathbf{U}, \mathbf{V}) + \mathcal{H}_\mu(\mathbf{U}, \mathbf{V}) - \alpha \mathcal{P}(\mathbf{U}) \right) \\ &=: \mathcal{G}(\mathbf{U}, \mathbf{V}) - \mathcal{H}(\mathbf{U}, \mathbf{V}), \end{aligned}$$

where the function $\mathcal{G}(\mathbf{U}, \mathbf{V}) := \frac{\rho}{2} \|\mathbf{U}, \mathbf{V}\|^2$ is obviously convex, and

$$\mathcal{H}(\mathbf{U}, \mathbf{V}) := \frac{\rho}{2} \|\mathbf{U}, \mathbf{V}\|^2 - \mathcal{G}_\mu(\mathbf{U}, \mathbf{V}) + \mathcal{H}_\mu(\mathbf{U}, \mathbf{V}) - \alpha \mathcal{P}(\mathbf{U}).$$

Since $\mathcal{H}_\mu(\mathbf{U}, \mathbf{V}) - \alpha \mathcal{P}(\mathbf{U})$ is also convex as $\alpha > 0$, we are going to show that for any given number $\mu > 0$ it is possible to determine the values of the parameter $\rho > 0$ such that the function $\frac{\rho}{2} \|(\mathbf{U}, \mathbf{V})\|^2 - \mathcal{G}_\mu(\mathbf{U}, \mathbf{V})$ is convex under an appropriate choice of ρ . This would yield the convexity of $\mathcal{H}(\mathbf{U}, \mathbf{V})$ and therefore would justify a desired representation of the objective function in (4.5). The following result gives us a precise meaning of this statement, which therefore verify the required reduction of (4.5) to *DC continuous optimization*.

Theorem 4.3 *The function*

$$\mathcal{G}_1(\mathbf{U}, \mathbf{V}) := \frac{\rho}{2} \|(\mathbf{U}, \mathbf{V})\|^2 - \mathcal{G}_\mu(\mathbf{U}, \mathbf{V}) \quad (4.6)$$

is convex on $\Delta^n \times \mathcal{B}$ provided that

$$\rho \geq \frac{n}{2\mu} \left[\left(1 + \frac{1}{n} \xi^2 \right) + \sqrt{\left(1 + \frac{1}{n} \xi^2 \right)^2 + \frac{12}{n} \xi^2} \right], \quad (4.7)$$

where $\xi := r + \max_{1 \leq j \leq n} \|a_j\|$ and $r := \sqrt{\sum_{j=1}^n \|a_j\|^2}$.

Proof. Consider the function $\mathcal{G}_1(\mathbf{U}, \mathbf{V})$ defined in (4.6) for all $(\mathbf{U}, \mathbf{V}) \in \Delta^n \times \mathcal{B}$ and deduce by elementary transformations directly from its construction that

$$\begin{aligned} \mathcal{G}_1(\mathbf{U}, \mathbf{V}) &= \frac{\rho}{2} \|(\mathbf{U}, \mathbf{V})\|^2 - \mathcal{G}_\mu(\mathbf{U}, \mathbf{V}) \\ &= \frac{\rho}{2} \|(\mathbf{U}, \mathbf{V})\|^2 - \frac{1}{2\mu} \sum_{i=1}^k \sum_{j=1}^n u_{i,j}^2 \|a_j - v_i\|^2 \\ &= \frac{\rho}{2} \|\mathbf{U}\|^2 + \frac{\rho}{2} \|\mathbf{V}\|^2 - \frac{1}{2\mu} \sum_{i=1}^k \sum_{j=1}^n u_{i,j}^2 \|a_j - v_i\|^2 \\ &= \frac{\rho}{2} \sum_{i=1}^k \sum_{j=1}^n u_{i,j}^2 + \frac{\rho}{2n} \sum_{i=1}^k \sum_{j=1}^n \|v_i\|^2 - \frac{1}{2\mu} \sum_{i=1}^k \sum_{j=1}^n u_{i,j}^2 \|a_j - v_i\|^2 \\ &= \sum_{i=1}^k \sum_{j=1}^n \frac{\rho}{2} u_{i,j}^2 + \frac{\rho}{2n} \|a_j - v_i\|^2 + \frac{\rho}{n} \langle a_j, v_i \rangle - \frac{\rho}{2n} \|a_j\|^2 - \frac{1}{2\mu} u_{i,j}^2 \|a_j - v_i\|^2. \end{aligned}$$

Next we define the functions $\gamma_{i,j}: \mathbb{R} \times \mathbb{R}^d \rightarrow \mathbb{R}$ for all $i = 1, \dots, k$ and $j = 1, \dots, n$ by

$$\gamma_{i,j}(u_{i,j}, v_i) := \frac{\rho}{2} u_{i,j}^2 + \frac{\rho}{2n} \|a_j - v_i\|^2 - \frac{1}{2\mu} u_{i,j}^2 \|a_j - v_i\|^2 \quad (4.8)$$

and show that each of these functions is convex on the set $\{u_{i,j} \in [0, 1], \|v_i\| \leq r\}$, where $r > 0$ is taken from Theorem 4.2.

To proceed, consider the Hessian matrix of each function in (4.8) given by

$$J_{\gamma_{i,j}}(u_{i,j}, v_i) := \begin{bmatrix} \rho - \frac{1}{\mu} \|a_j - v_i\|^2 & -\frac{2}{\mu} u_{i,j} (v_i - a_j) \\ -\frac{2}{\mu} u_{i,j} (v_i - a_j) & \frac{\rho}{n} - \frac{1}{\mu} u_{i,j}^2 \end{bmatrix}$$

and calculate its determinant $\det(J_{\gamma_{i,j}}(u_{i,j}, v_i))$ by

$$\begin{aligned}\det(J_{\gamma_{i,j}}(u_{i,j}, v_i)) &:= \left(\rho - \frac{1}{\mu}\|a_j - v_i\|^2\right) \left(\frac{\rho}{n} - \frac{1}{\mu}u_{i,j}^2\right) - \frac{4}{\mu^2}u_{i,j}^2(v_i - a_j)^T(v_i - a_j) \\ &= \frac{\rho^2}{n} - \rho\left(\frac{u_{i,j}^2}{\mu} + \frac{1}{n\mu}\|v_i - a_j\|^2\right) - \frac{3u_{i,j}^2}{\mu^2}\|v_i - a_j\|^2.\end{aligned}$$

It follows from the well-known second-order characterization of the convexity that the function $\gamma_{i,j}(u_{i,j}, v_i)$ is convex on $\{u_{i,j} \in [0, 1], \|v_i\| \leq r\}$ if $\det(J_{\gamma_{i,j}}(u_{i,j}, v_i)) \geq 0$. Using [5, Theorem 1] gives us the estimate

$$\det(J_{\gamma_{i,j}}(u_{i,j}, v_i)) \geq \frac{\rho^2}{n} - \rho\left(\frac{1}{\mu} + \frac{1}{n\mu}\|v_i - a_j\|^2\right) - \frac{3}{\mu^2}\|v_i - a_j\|^2.$$

Then we get from the construction of \mathcal{B} in Theorem 4.2 that $0 < \|v_i - a_j\| \leq \|v_i\| + \|a_j\| \leq r + \max_{1 \leq j \leq n} \|a_j\| =: \xi$, and therefore

$$\det(J_{\gamma_{i,j}}(u_{i,j}, v_i)) \geq \frac{\rho^2}{n} - \frac{\rho}{\mu}\left(1 + \frac{1}{n}\xi^2\right) - \frac{3}{\mu^2}\xi^2. \quad (4.9)$$

It allows us to deduce from the aforementioned condition for the convexity of $\gamma_{i,j}(u_{i,j}, v_i)$ that we do have this convexity if ρ satisfies the estimate (4.7). \square

5 Design and Implementation of the Solution Algorithm

Based on the developments presented in the previous sections and using the established smooth DC structure of problem (4.5) with the subsequent ρ -parameterization of the objective function therein as $\mathcal{G}(\mathbf{U}, \mathbf{V}) - \mathcal{H}(\mathbf{U}, \mathbf{V})$, we are now ready to propose and implement a new algorithm for solving this problem involving both DCA-2 and Nesterov's smoothing.

To proceed, let us present the problem under consideration in the equivalent *unconstrained* format by using the infinite penalty via the indicator function:

$$\begin{aligned}\text{minimize} \quad & \frac{\rho}{2}\|(\mathbf{U}, \mathbf{V})\|^2 - \mathcal{H}(\mathbf{U}, \mathbf{V}) + \delta_{\Delta \times \mathcal{B}}(\mathbf{U}, \mathbf{V}) \\ \text{subject to} \quad & (\mathbf{U}, \mathbf{V}) \in \mathbb{R}^{k \times n} \times \mathbb{R}^{k \times d},\end{aligned} \quad (5.10)$$

where \mathcal{B} , Δ , and ρ are taken from Section 4.

We first explicitly compute the gradient of the convex function $\mathcal{H}(\mathbf{U}, \mathbf{V})$ in (5.10). Denoting

$$\begin{aligned}[\mathcal{Y}, \mathcal{Z}] &:= \nabla \mathcal{H}(\mathbf{U}, \mathbf{V}) = \nabla \left(\frac{\rho}{2}\|(\mathbf{U}, \mathbf{V})\|^2 - \frac{1}{2\mu} \sum_{i=1}^k \sum_{j=1}^n u_{i,j}^2 \|a_j - v_i\|^2 \right. \\ &\quad \left. + \frac{\mu}{2} \sum_{i=1}^k \sum_{j=1}^n u_{i,j}^2 \left[d\left(\frac{a_j - v_i}{\mu}; \mathbb{B}\right) \right]^2 - \alpha \sum_{i=1}^k \sum_{j=1}^n u_{i,j}(1 - u_{i,j}) \right),\end{aligned}$$

we have $\mathcal{Y} = \nabla \mathcal{H}_{\mathbf{U}}(\mathbf{U}, \mathbf{V})$ and $\mathcal{Z} = \nabla \mathcal{H}_{\mathbf{V}}(\mathbf{U}, \mathbf{V})$. Thus for each $i = 1, \dots, k$ and $j = 1, \dots, n$ the (j, i) -entry of the matrix \mathcal{Y} and the i th row of the matrix \mathcal{Z} are

$$\begin{aligned}\mathcal{Y}_{j,i} &:= \rho u_{i,j} - \frac{u_{i,j}}{\mu} \|a_j - v_i\|^2 + \mu u_{i,j} \left[d\left(\frac{a_j - v_i}{\mu}; \mathbb{B}\right) \right]^2 + 2\alpha u_{i,j} - \alpha, \\ \mathcal{Z}_i &:= \rho v_i - \frac{1}{\mu} \sum_{j=1}^n u_{i,j}^2 (v_i - a_j) - \sum_{j=1}^n u_{i,j}^2 \left[\frac{a_j - v_i}{\mu} - P\left(\frac{a_j - v_i}{\mu}; \mathbb{B}\right) \right],\end{aligned}$$

respectively. Let us now describe the proposed algorithm for solving the DC program (5.10) and hence the original problem (1.1) of multifacility location. The symbols $\mathcal{Y}_{[j,:]}^{l-1}$ and \mathcal{Z}_i^{l-1} in this description represents the j th row of the matrix \mathcal{Y} and the i th row of the matrix \mathcal{Z} at the l th iteration, respectively. Accordingly we use the symbols $\mathbf{U}_{[:,j]}^l$ and \mathbf{V}_i^l . Recall also that the Frobenius norm of the matrices in this algorithm is defined in (2.7).

Algorithm 3: Solving Multifacility Location Problems.

```

INPUT:  $\mathbf{X}$  (the dataset),  $\mathbf{V}^0$  (initial centers), ClusterNum (number of clusters),  $\mu > 0$ ,
 $\beta$  (scaling parameter)  $> 0$ ,  $N \in \mathbb{N}$ 
INITIALIZATION:  $\mathbf{U}^0$ ,  $\varepsilon > 0$ ,  $\mu_f$  (minimum threshold for  $\mu$ )  $> 0$ ,  $\alpha > 0$ ,  $\rho > 0$ ,
tol (tolerance parameter) = 1
while tol  $> \varepsilon$  and  $\mu > \mu_f$ 
for  $l = 1, 2, \dots, N$ 
  For  $1 \leq i \leq k$  and  $1 \leq j \leq n$  compute
    
$$\mathcal{Y}_{j,i}^{l-1} := \rho u_{i,j}^{l-1} - \frac{u_{i,j}^{l-1}}{\mu} \|a_j - v_i^{l-1}\|^2 + \mu u_{i,j}^{l-1} \left[ d\left(\frac{a_j - v_i^{l-1}}{\mu}; \mathbb{B}\right) \right]^2 + 2\alpha u_{i,j}^{l-1} - \alpha,$$

    
$$\mathcal{Z}_i^{l-1} := \rho v_i^{l-1} - \frac{1}{\mu} \sum_{j=1}^n (u_{i,j}^{l-1})^2 (v_i^{l-1} - a_j) - \sum_{j=1}^n (u_{i,j}^{l-1})^2 \left[ \frac{a_j - v_i^{l-1}}{\mu} - P\left(\frac{a_j - v_i^{l-1}}{\mu}; \mathbb{B}\right) \right].$$

  For  $1 \leq i \leq k$  and  $1 \leq j \leq n$  compute
    
$$\mathbf{U}_{[:,j]}^l := P\left(\frac{\mathcal{Y}_{[j,:]}^{l-1}}{\rho}; \Delta\right),$$

    
$$\mathbf{V}_i^l := P\left(\frac{\mathcal{Z}_i^{l-1}}{\rho}; B_i\right) = \begin{cases} \frac{\mathcal{Z}_i^{l-1}}{\rho} & \text{if } \|\mathcal{Z}_i^{l-1}\| \leq \rho r, \\ \frac{\rho \mathcal{Z}_i^{l-1}}{\|\mathcal{Z}_i^{l-1}\|} & \text{if } \|\mathcal{Z}_i^{l-1}\| > \rho r. \end{cases}$$

  end for
  UPDATE:
    tol :=  $\|[\mathbf{U}^l, \mathbf{V}^l] - [\mathbf{U}^{l-1}, \mathbf{V}^{l-1}]\|_F$ 
     $\mu := \beta \mu$ .
end while
OUTPUT:  $[\mathbf{U}^N, \mathbf{V}^N]$ .

```

Next we employ Algorithm 3 to solve several multifacility location problems of some practical meaning. By trial and error we verify that the values chosen for μ determine the performance of the algorithm for each data set. It can be seen that very small values of the smoothing parameter μ may prevent the algorithm from clustering, and thus we gradually decrease these values. This is done via multiplying μ by some number $0 < \beta < 1$ and stopping when $\mu < \mu_f$. Note also that in the implementation of our algorithm we use the standard approach of choosing \mathbf{U}^0 by computing the distance between the point in question and each group center \mathbf{V}^0 and then by classifying this point to be in the group whose center is the closest to it by assigning the value of 1, while otherwise we assign the value of 0.

Let us now present several numerical examples, where we compute the optimal centers by using Algorithm 3 via MATLAB calculations. Fix in what follows the values of $\mu = 0.5$, $\beta = 0.85$, $\varepsilon = 10^{-6}$, $\mu_f = 10^{-6}$, $\alpha = 30$, and $\rho = 30$ unless otherwise stated. The objective

function is the *total distance* from the centers to the assigned data point. Note that this choice of the objective function seems to be natural from practical aspects in, e.g., airline and other transportation industries, where the goal is to reach the destination via the best possible route available. This reflects minimizing the transportation cost.

In the following examples we implement the standard k -means algorithm in MATLAB using the in-built function `kmeans()`.

Example 5.1 Let us consider a data set with 14 entries in \mathbb{R}^2 given by

$$\mathbf{X} := \begin{bmatrix} 0 & 2 & 7 & 2 & 3 & 6 & 5 & 8 & 8 & 9 & 1 & 7 & 0 & 0 \\ 3 & 2 & 1 & 4 & 3 & 2 & 3 & 1 & 3 & 2 & 1 & 4 & 4 & 1 \end{bmatrix}^T$$

with the initial data defined by

$$\mathbf{V}^0 := \begin{bmatrix} 7.1429 & 2.2857 \\ 1.1429 & 2.5714 \end{bmatrix} \text{ is obtained from the } k\text{-means algorithm; see Table 1,}$$

ClusterNum := 2.

Employing Algorithm 3, we obtain the optimal centers as depicted in Table 1 and Figure 5.1.

Table 1: Comparison between Algorithm 3 and k -means

Method	Optimal Center (\mathbf{V}^N)	Cost Function
k -means	$\begin{bmatrix} 7.1429 & 2.2857 \\ 1.1429 & 2.5714 \end{bmatrix}$	22.1637
Algorithm 3	$\begin{bmatrix} 7.2220 & 2.1802 \\ 1.1886 & 2.5069 \end{bmatrix}$	22.1352

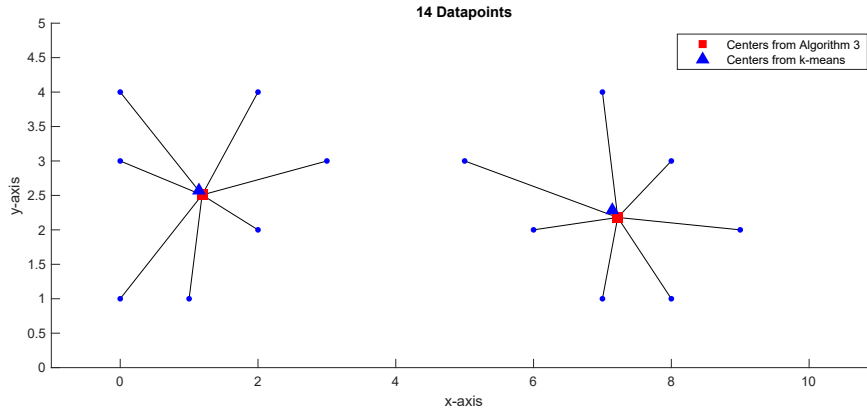


Figure 5.1: *MFLP with 14 demand points and 2 centers.*

Table 1 shows that the proposed Algorithm 3 is marginally better for the given data in comparison to the classical k -means approach in terms of the objective function.

The following example shows that the DCA and the k -means may result both different clusters and cluster centers.

Example 5.2 Let \mathbf{X} be 10 data points in \mathbb{R}^2 given by

$$\mathbf{X} := \begin{bmatrix} 1.90 & 1.76 & 2.32 & 2.31 & 1.14 & 5.02 & 5.74 & 2.25 & 4.71 & 3.17 \\ 0.97 & 0.84 & 1.63 & 2.09 & 2.11 & 3.02 & 3.84 & 3.47 & 3.60 & 4.96 \end{bmatrix}^T.$$

With the initial data defined by taking two random data points from the data set

$$\mathbf{V}^0 := \begin{bmatrix} 1.90 & 0.97 \\ 3.17 & 4.96 \end{bmatrix} \text{ chosen randomly,}$$

ClusterNum := 2.

We obtain the optimal centers as outlined in Table 2.

Table 2: Comparison between Algorithm 3 and k -means

Method	Optimal Center (\mathbf{V}^N)	Cost Function
k -means	$\begin{bmatrix} 3.3943 & 2.2843 \\ 2.1867 & 3.5133 \end{bmatrix}$	16.5669
Algorithm 3	$\begin{bmatrix} 1.9995 & 1.4757 \\ 4.7185 & 3.5838 \end{bmatrix}$	9.0994

Observe from Table 2 that the proposed Algorithm 3 is better for the given data in comparison to the standard k -means algorithm. In addition, our approach gives a better approximation for the optimal center as shown in Figure 5.2, which coincides with the results of the Fermat-Torricelli problem; see [16]. It also yields a different set of clusters compared to the standard k -means algorithm as shown in Figures 5.3 and 5.4, respectively.

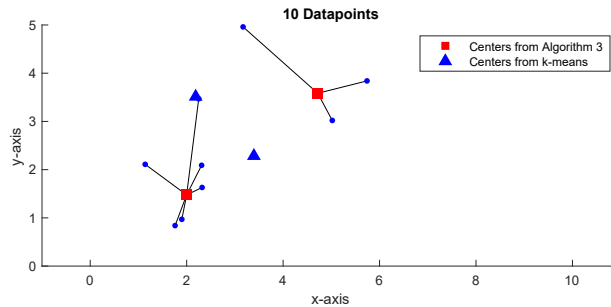


Figure 5.2: *MFLP with 10 demand points and 2 centers.*

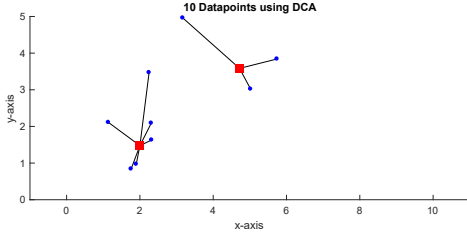


Figure 5.3: *Clusters from Algorithm 3*

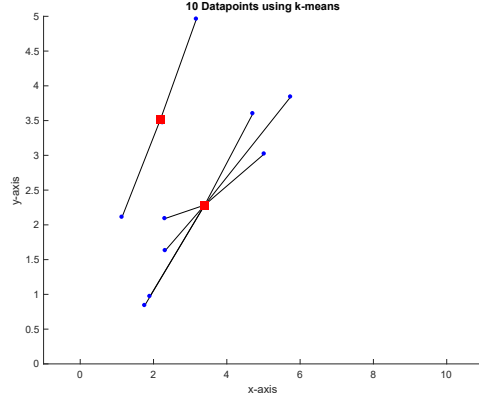


Figure 5.4: *Clusters from k-means*

Note that a drawback in employing the random approach to choose the initial cluster \mathbf{V}^0 in Example 5.2 is the need of having prior knowledge about the data. Typically it may not be plausible to extract such an information from large unpredictable real life datasets.

In the next example we choose the initial cluster by the process of *random selection* and see its effect on the optimal centers. Then the results obtained in this way by Algorithm 3 are compared with those computed by the *k-means* approach.

Example 5.3 Let \mathbf{X} be 200 standard normally distributed random datapoints in \mathbb{R}^2 , and let the initial data be given by

$$\begin{aligned} \mathbf{V}^0 &:= \text{randomly permuting and selecting 2 rows of } \mathbf{X}, \\ \text{ClusterNum} &:= 2. \end{aligned}$$

We obtain the optimal centers as outlined in Table 3.

Table 3: Comparison between Algorithm 3 and *k-means*

Method	Optimal Center (\mathbf{V}^N)	Cost Function
<i>k-means</i>	$\begin{bmatrix} 2.1016 & 1.2320 \\ -1.3060 & -1.0047 \end{bmatrix}$	403.3966
Algorithm 3	$\begin{bmatrix} 1.4902 & 0.7406 \\ -1.3464 & -1.0716 \end{bmatrix}$	401.7506

Observe from Table 3 that the proposed Algorithm 3 is better for the given data in comparison to the standard *k-means* approach. In addition, our approach gives a better approximation for the optimal center as shown in Figure 5.5.

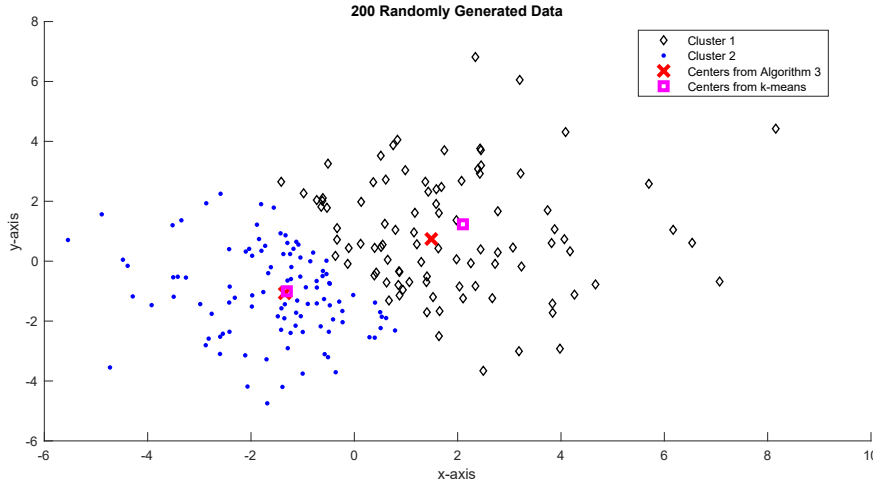


Figure 5.5: *MFLP with 200 demand points and 2 centers.*

Note that a real-life data may not be as efficiently clustered as in Example 5.3. Thus a suitable selection of the initial cluster \mathbf{V}^0 is vital for the convergence of the DCA based algorithms. In the next Example 5.4 we select \mathbf{V}^0 in Algorithm 3 by using the standard k -means method. The results achieved by our Algorithm 3 are again compared with those obtained by using the k -means approach.

Example 5.4 Consider the dataset \mathbf{X} consisting of the latitudes and longitudes of 50 most populous cities in the USA⁵ with

$$\mathbf{V}^0 := \begin{bmatrix} -80.9222 & 37.9882 \\ -97.8273 & 35.3241 \\ -118.3121 & 36.9535 \end{bmatrix} \text{ is obtained from the } k\text{-means algorithm; see Table 4,}$$

ClusterNum := 3.

By using Algorithm 3 we obtain the following optimal centers as given in Table 4.

Table 4: Comparison between Algorithm 3 (combined with k -means) and standard k -means

Method	Optimal Center (\mathbf{V}^N)	Cost Function
k -means	$\begin{bmatrix} -80.9222 & 37.9882 \\ -97.8273 & 35.3241 \\ -118.3121 & 36.9535 \end{bmatrix}$	288.8348
Algorithm 3 (combined with k -means)	$\begin{bmatrix} -81.0970 & 38.3092 \\ -97.4138 & 35.3383 \\ -119.3112 & 36.5410 \end{bmatrix}$	286.6523

We see that Algorithm 3 (combined with k -means) in which the initial cluster \mathbf{V}^0 is selected by using k -means method performs better in comparison to the standard k -means approach (Table 4). Moreover, it gives us optimal centers as depicted in Figure 5.6.

⁵Available at https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population

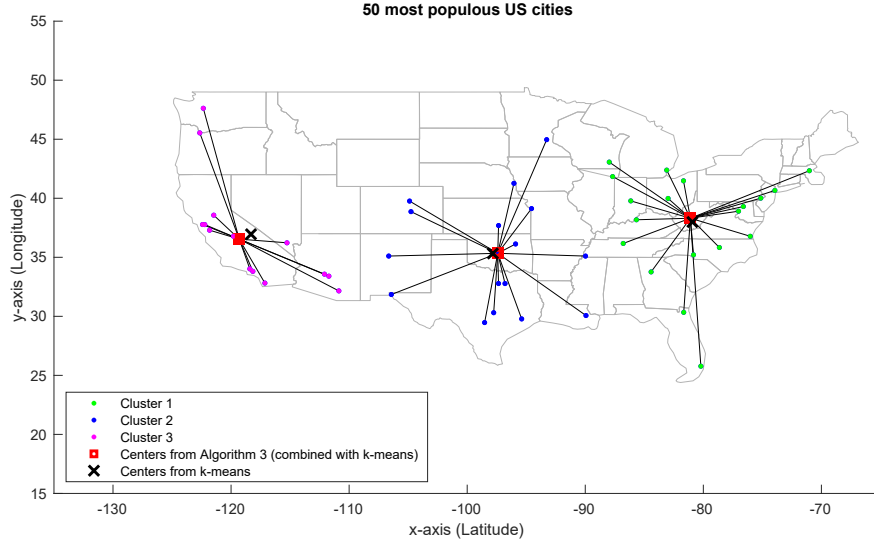


Figure 5.6: *MFLP with 50 demand points and 3 centers.*

In the next example we efficiently solve yet another multifacility location problem by using Algorithm 3.

Example 5.5 Consider the dataset \mathbf{X} in \mathbb{R}^2 that consists of the latitudes and longitudes of 988 US cities [27] with

$$\mathbf{V}^0 := \begin{bmatrix} -89.6747 & 41.1726 \\ -88.4834 & 30.2475 \\ -118.4471 & 35.2843 \\ -75.7890 & 40.0329 \\ -114.1897 & 43.0798 \end{bmatrix} \text{ is obtained from the } k\text{-means algorithm,}$$

ClusterNum := 5.

The optimal centers illustrated in Figure 5.7 are given by

$$\mathbf{V}^N := \begin{bmatrix} -88.2248 & 40.7241 \\ -88.0154 & 30.7041 \\ -120.0735 & 33.1941 \\ -74.5328 & 38.6996 \\ -113.2341 & 42.3969 \end{bmatrix}.$$

The total transportation cost in this problem is 5089.5150.

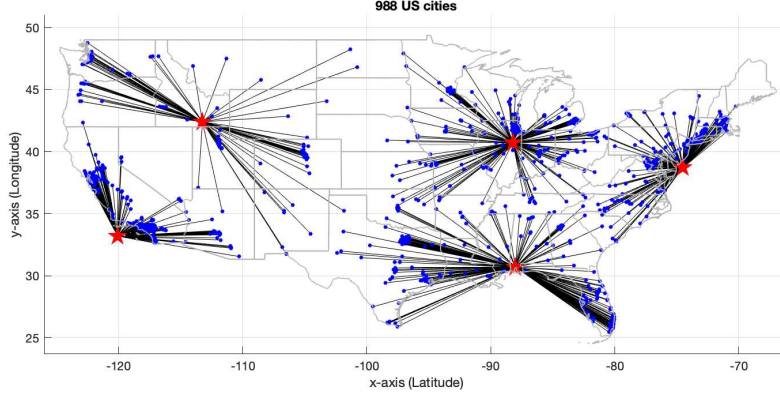


Figure 5.7: *MFLP with 988 demand points and 5 centers.*

In the last example presented in this section we efficiently solve a higher dimensional multifacility location problem by using Algorithm 3 and compare it's value of the cost function with the standard k -means algorithm.

Example 5.6 Let \mathbf{X} in \mathbb{R}^{13} be the *wine dataset* from the UCI Machine Learning Repository [29] consisting of 178 demand points. We apply Algorithm 3 with

\mathbf{V}^0 is obtained from the k -means algorithm,
ClusterNum := 3.

The total costs using Algorithm 3 and the k -means algorithm are obtained in Table 5 showing that former algorithm is better than the latter.

Table 5: Cost comparison between Algorithm 3 (combined with k -means) and k -means

Method	Cost Function
k -means	16556
Algorithm 3 (combined with k -means)	16460

6 Concluding Remarks

In this paper we develop a new algorithm to solve a class of multifacility location problems. Its implementation exhibits a better approximation compared to the classical k -means approach. This is demonstrated by a series of examples dealing with two-dimensional problems with nonnegative weights. Thus the verification and implementation of the proposed algorithm for real-life multifacility location problems in higher dimensions with arbitrary weights is a central direction of our future work. In addition, refining the initial cluster selection and the stopping criterion is an important area to explore.

Acknowledgements. We would like to thank the two anonymous referees for taking their

valuable time to read and give us their invaluable comments that help improve the content and the presentation of the paper.

References

- [1] L.T.H. An, M.T. Belghiti, and P.D. Tao, *A new efficient algorithm based on DC programming and DCA for clustering*, J. Glob. Optim. 37 (2007), pp. 593–608.
- [2] N.T. An and N.M. Nam, *Convergence analysis of a proximal point algorithm for minimizing differences of functions*, Optim. 66 (2017), pp. 129–147.
- [3] L.T.H. An, H.V. Ngai, and P.D. Tao, *Convergence analysis of difference-of-convex algorithm with subanalytic data*, J. Optim. Theory Appl. 179 (2018), pp. 103–126.
- [4] L.T.H. An, L.H. Minh, and P.D. Tao, *New and efficient DCA based algorithms for minimum sum-of-squares clustering*, Pattern Recogn. 47 (2014), pp. 388–401.
- [5] L.T.H. An and P.D. Tao, *Minimum sum-of-squares clustering by DC programming and DCA*, Int. Conf. Intell. Comp. (2009), pp. 327–340.
- [6] J. Brimberg, *The Fermat-Weber location problem revisited*, Math. Program. 71 (1995), pp. 71–76.
- [7] Z. Drezner, *On the convergence of the generalized Weiszfeld algorithm*, Ann. Oper. Res. 167 (2009), pp. 327–336.
- [8] T. Jahn, Y.S. Kupitz, H. Martini, and C. Richter, *Minsum location extended to gauges and to convex sets*, J. Optim. Theory Appl. 166 (2015), pp. 711–746.
- [9] J.-B. Hiriart-Urruty and C. Lemaréchal, *Fundamentals of Convex Analysis*, Springer, Berlin, 2001.
- [10] H.W. Kuhn, *A note on Fermat-Torricelli problem*, Math. Program. 4 (1973), pp. 98–107.
- [11] Y.S. Kupitz and H. Martini, *Geometric aspects of the generalized Fermat-Torricelli problem*, Bolyai Soc. Math. Stud. 6 (1997), pp. 55–128.
- [12] H. Martini, K.J. Swanepoel, and G. Weiss, *The Fermat-Torricelli problem in normed planes and spaces*, J. Optim. Theory Appl. 115 (2002), pp. 283–314.
- [13] B.S. Mordukhovich, *Variational Analysis and Applications*, Springer, Cham, Switzerland, 2018.
- [14] B.S. Mordukhovich and N.M. Nam, *Applications of variational analysis to a generalized Fermat-Torricelli problem*, J. Optim. Theory Appl. 148 (2011), pp. 431–454.
- [15] B.S. Mordukhovich and N.M. Nam, *An Easy Path to Convex Analysis and Applications*, Morgan & Claypool Publishers, San Rafael, 2014.
- [16] B.S. Mordukhovich and N.M. Nam, *The Fermat-Torricelli problem and Weiszfeld’s algorithm in the light of convex analysis*, J. Appl. Numer. Optim. 1 (2019), 205–215.

- [17] N.M. Nam, N.T. An, R.B. Rector, and J. Sun, *Nonsmooth algorithms and Nesterov's smoothing technique for generalized Fermat-Torricelli problems*, SIAM J. Optim. 24 (2014), pp. 1815–1839.
- [18] N.M. Nam, W. Geremew, S. Reynolds, and T. Tran, *Nesterov's smoothing technique and minimizing differences of convex functions for hierarchical clustering*, Optim. Lett. 12 (2018), pp. 455–473.
- [19] N.M. Nam and N. Hoang, *A generalized Sylvester problem and a generalized Fermat-Torricelli problem*, J. Convex Anal. 20 (2013), pp. 669–687.
- [20] N.M. Nam, R.B. Rector, and D. Giles, *Minimizing differences of convex functions with applications to facility location and clustering*, J. Optim. Theory Appl. 173 (2017), pp. 255–278.
- [21] Yu. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$* , Soviet Math. Dokl. 269 (1983), pp. 543–547.
- [22] Yu. Nesterov, *Smooth minimization of nonsmooth functions*, Math. Program. 103 (2005), pp. 127–152.
- [23] Yu. Nesterov, *Lectures on Convex Optimization*, 2nd edition, Springer, Cham, Switzerland, 2018.
- [24] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.
- [25] P.D. Tao and L.T.H. An, *Convex analysis approach to D.C. programming: theory, algorithms and applications*, Acta Math. Vietnam. 22 (1997), pp. 289–355.
- [26] P.D. Tao and L.T.H. An, *A D.C. optimization algorithm for solving the trust-region subproblem*, SIAM J. Optim. 8 (1998), pp. 476–505.
- [27] United States Cities Database, *Simple Maps: Geographic Data Products*, 2017, <http://simplemaps.com/data/us-cities>.
- [28] E. Weiszfeld, *Sur le point pour lequel la somme des distances de n points donnés est minimum*, Tôhoku Math. J. 43 (1937), pp. 355–386.
- [29] UCI Machine Learning Repository, <https://archive.ics.uci.edu/ml/datasets/wine>.