Improved Exact Algorithm for the Capacitated Facility Location Problem on a Line Graph

Edward Gimadi[1,2], Alexandr Shtepa[1], Oxana Tsidulko[1,2]

[1]Department of Mechanics and Mathematics, Novosibirsk State University, Russian Federation
[2]Sobolev Institute of Mathematics of the Siberian Branch of the Russian Academy of Sciences, Russian Federation

26th of August, 2019

Capacitated Facility Location Problem (general case)

Given:

- a set M of possible facility locations ($|M| = m$),
- a set N of clients ($|N| = n$);
- f_i is an opening cost for facility i,
- a_i is a capacity of facility i;
- b_j is an integer demand of client j;
- g_{ij} is a transportation cost of delivering a unit of product from facility i to client j.

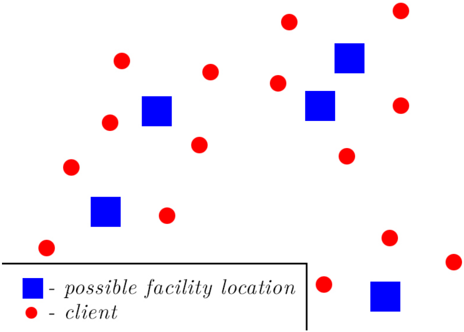
Find: a subset of facilities $M' \subseteq M$ to open such that:

$$\sum_{i \in M'} f_i + \sum_{j \in N} \sum_{i \in M'} b_j g_{ij} x_{ij} \to \min$$

$$\sum_{i \in M'} x_{ij} = 1, \ j \in N,$$

$$\sum_{j \in N} b_j x_{ij} \le a_i, \ i \in M',$$

$$x_{ij} \ge 0$$



Capacitated Facility Location Problem (general case)

Given:

- a set M of possible facility locations ($|M| = m$),
- a set N of clients ($|N| = n$);
- f_i is an opening cost for facility i,
- a_i is a capacity of facility i;
- b_j is an integer demand of client j;
- g_{ij} is a transportation cost of delivering a unit of product from facility i to client j.

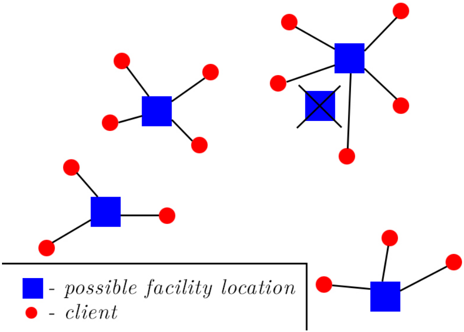
Find: a subset of facilities $M' \subseteq M$ to open such that:

$$\sum_{i\in M'} f_i + \sum_{j\in N} \sum_{i\in M'} b_j g_{ij} x_{ij} \to \min$$

$$\sum_{i\in M'} x_{ij} = 1,\ j \in N,$$

$$\sum_{j\in N} b_j x_{ij} \le a_i,\ i \in M',$$

$$x_{ij} \ge 0$$



Types of Capacitated Facility Location Problem

Metric CFLP

Transportation costs satisfy triangle inequality. (The transportation cost from i to j is defined according to the shortest path distance in network graph).

Types of Capacitated Facility Location Problem

Metric CFLP

Transportation costs satisfy triangle inequality. (The transportation cost from i to j is defined according to the shortest path distance in network graph).

Single allocation CFLP

A demand of a client must be served by only one facility.
For the allocation variables x_{ij}: $x_{ij} \in \{0,1\}$.

Multiple allocation CFLP

A client can be served by multiple facilities simultaneously.
For the allocation variables x_{ij}: $0 \leq x_{ij} \leq 1$.

Types of Capacitated Facility Location Problem

Metric CFLP

Transportation costs satisfy triangle inequality. (The transportation cost from i to j is defined according to the shortest path distance in network graph).

Single allocation CFLP

A demand of a client must be served by only one facility.
For the allocation variables x_{ij}: $x_{ij} \in \{0,1\}$.

Multiple allocation CFLP

A client can be served by multiple facilities simultaneously.
For the allocation variables x_{ij}: $0 \leq x_{ij} \leq 1$.

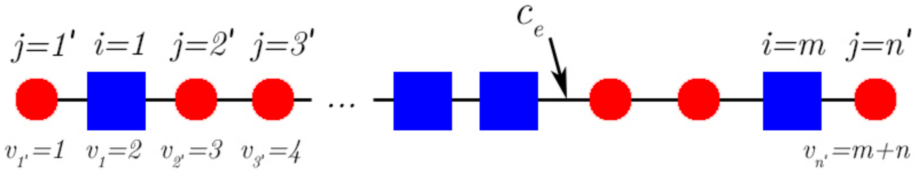
Statement

*All variants of the problem are **NP-hard**.*

Capacitated Facility Location Problem on a Line Graph

Given:

- a line graph $G = (V,E)$, $V = M \uplus N$,
- M is a set of possible facility locations ($|M| = m$),
- N is a set of clients ($|N| = n$);
- f_i is an opening cost for facility i,
- a_i is a capacity of facility i;
- b_j is an integer demand of client j;
- c_e is a cost of transporting a unit of product along edge $e \in E$,
- P_{ij} is a (shortest) path between a facility i at vertex number v_i and a client j at vertex number v_j
- $g_{ij} = \sum_{e \in P_{ij}} c_e$ is a transportation cost of delivering a unit of product from facility i to client j.



Capacitated Facility Location Problem on a Line Graph

Find: which facilities to open such that:

$$\sum_{i \in M} f_i y_i + \sum_{i \in M} \sum_{j \in N} b_j g_{ij} x_{ij} \to \min_{y_i, x_{ij}} \tag{1}$$

$$\sum_{j \in N} b_j x_{ij} \le a_i y_i, \quad i \in M, |M| = m, \tag{2}$$

$$\sum_{i \in M} x_{ij} = 1, \quad j \in N, |N| = n, \tag{3}$$

$$x_{ij} \ge 0, y_i \in \{0; 1\}, \tag{4}$$

where

x_{ij} is a share of the demand of a client j at vertex number v_j served by a facility i at vertex number v_i,

$$y_i = \begin{cases} 1, & \text{if one opens a facility } i \in M, \\ 0, & \text{otherwise.} \end{cases}$$

Capacitated Facility Location Problem on a Line Graph

Find: which facilities to open such that:

$$\sum_{i\in M} f_i y_i + \sum_{i\in M}\sum_{j\in N} b_j g_{ij} x_{ij} \to \min_{y_i, x_{ij}} \tag{1}$$

$$\sum_{j\in N} b_j x_{ij} \le a_i y_i, \quad i \in M, |M| = m, \tag{2}$$

$$\sum_{i\in M} x_{ij} = 1, \quad j \in N, |N| = n, \tag{3}$$

$$x_{ij} \ge 0, y_i \in \{0;1\}, \tag{4}$$

where

x_{ij} is a share of the demand of a client j at vertex number v_j served by a facility i at vertex number v_i,

$$y_i = \begin{cases} 1, & \text{if one opens a facility } i \in M, \\ 0, & \text{otherwise.} \end{cases}$$

Statement

*CFLP is **NP-hard** even on a line graph, since in the case of zero transportation costs and only one client it contains the MINIMIZATION KNAPSACK problem.*

Applications of CFLP on a Line Graph

- *Rest area location*. Cars enter a highway at different points. What is the smallest number of rest areas that are needed along the highway to ensure that each car can access a rest area within a given distance from its point of entry?
- *Transformer location*. A high-voltage power line runs through rural townships. To limit power losses, step-down transformers must be installed within certain distances of the townships. What is the smallest number of transformers required to service all communities?

Our contributions

Known result: [Mirchandani et al., 1996]

The multiple allocation CFLP on a line graph can be solved by a *dynamic programming* pseudopolynomial-time algorithm with running-time

$$O(mB\min\{a_{max}, B\}),$$

where $B = \sum_{j \in N} b_j$ is the total demand and a_{max} is the maximum facility capacity.

Our contributions

Known result: [Mirchandani et al., 1996]

The multiple allocation CFLP on a line graph can be solved by a *dynamic programming* pseudopolynomial-time algorithm with running-time

$$O(mB\min\{a_{max}, B\}),$$

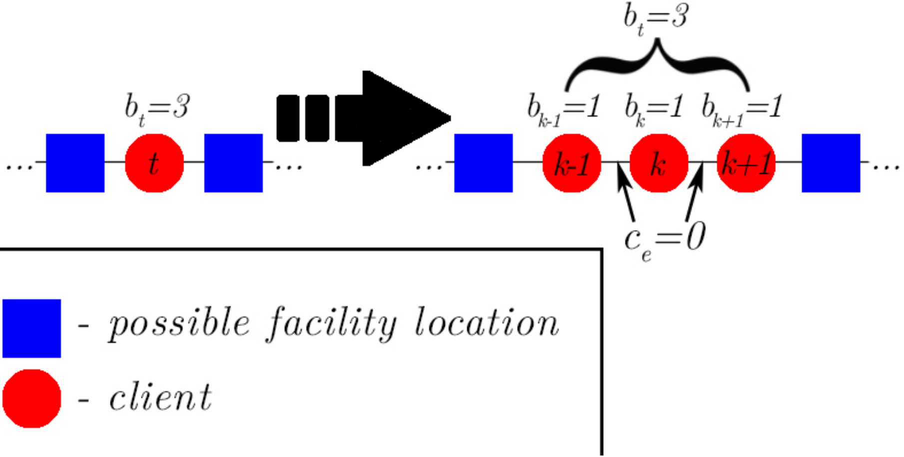
where $B = \sum_{j \in N} b_j$ is the total demand and a_{max} is the maximum facility capacity.

We present 2 modifications of this algorithm:

1. **First modification:** using binary heap, we improve time complexity to $O(mB\log(\min\{a_{max}, B\}))$.
2. **Second modification:** using algorithm from [Aggarwal et al., 1987], we improve time complexity to $O(mB)$.

Reduction to CFLP with unit demands:

The algorithm from [Mirchandani et al., 1996] starts by reducing the multiple allocation CFLP with n clients to the multiple allocation CFLP with $B = \sum_{j \in N} b_j$ clients, each of unit demand.



Notation

For each facility i let ℓ_i and r_i be the lowest and the highest client indices such that facility i has enough capacity to serve all the clients of the segments $[\ell_i, v_i]$ and $[v_i, r_i]$, respectively.

Notation

For each facility i let ℓ_i and r_i be the lowest and the highest client indices such that facility i has enough capacity to serve all the clients of the segments $[\ell_i, v_i]$ and $[v_i, r_i]$, respectively.

Remark

A facility of unbounded capacity can be considered as a facility of capacity B. Let $\widetilde{a}_i = \min\{a_i, B\}$ be the revised facility capacities, $i = 1, \ldots, m$.

Notation

Let $w_i(k,j)$, $k < j$, *be the total transportation costs required to serve all the clients of the segment* $(k,j]$ *from the facility* i: $w_i(k,j) = \sum_{t=k}^{j} g_{it} b_t$.

Notation

Let $w_i(k,j)$*,* $k<j$*, be the total transportation costs required to serve all the clients of the segment* $(k,j]$ *from the facility* i*:* $w_i(k,j)=\sum_{t=k}^{j} g_{it} b_t$*.*

Remark

Using data structures that can be precomputed in time $O(B+m)$*, the values* $w_i(k,j)$ *can be found in constant time for any given* $1 \le i \le m$*,* $1 \le k < j \le B$ *by the following formula.*

$$w_i(k,j)=\begin{cases} D(j)-D(k)-d(i)(v_j-v_k), & \text{if } v_i \le v_k < v_j, \\ D(k)+D(j)-2D(i)-d(i)(v_k+v_j-2v_i), & \text{if } v_k < v_i < v_j, \\ D(k)-D(j)+d(i)(v_j-v_k), & \text{if } v_k < v_j \le v_i, \end{cases} \tag{5}$$

where the partial sums $d(t)=\sum_{j=1}^{t} c_{(j-1,j)}$ *and* $D(t)=\sum_{j=1}^{t} d(j)$ *for all* $t=1,\dots,B+m$ *can be computed recursively in total time* $O(B+m)$*.*

Notation

Let $w_i(k,j)$, $k<j$, *be the total transportation costs required to serve all the clients of the segment* $(k,j]$ *from the facility* i*:* $w_i(k,j)=\sum_{t=k}^{j} g_{it}b_t$.

Remark

Using data structures that can be precomputed in time $O(B+m)$*, the values* $w_i(k,j)$ *can be found in constant time for any given* $1 \le i \le m$, $1 \le k < j \le B$ *by the following formula.*

$$w_i(k,j)=\begin{cases} D(j)-D(k)-d(i)(v_j-v_k), & \text{if } v_i \le v_k < v_j,\\ D(k)+D(j)-2D(i)-d(i)(v_k+v_j-2v_i), & \text{if } v_k < v_i < v_j,\\ D(k)-D(j)+d(i)(v_j-v_k), & \text{if } v_k < v_j \le v_i, \end{cases} \tag{5}$$

where the partial sums $d(t)=\sum_{j=1}^{t} c_{(j-1,j)}$ *and* $D(t)=\sum_{j=1}^{t} d(j)$ *for all* $t = 1,\ldots,B+m$ *can be computed recursively in total time* $O(B+m)$.

Remark

All the values r_i *and* ℓ_i *for* $i=1,\ldots,m$ *can be found in time* $O(m+B)$.

Dynamic Programming Algorithm

Algorithm from [Mirchandani et al., 1996].

Let $S(i,j)$ be the optimum value of a subproblem in which the first j clients on the line are optimally served by a subset of the first i facilities.

For all $i = 1,\dots,m,\ j = 1,\dots,B$

$$S(i,j) = \begin{cases} \min\left\{S(i-1,j), f_i + \min\limits_{\max\{j-\widetilde{a}_i,\ell_i\}\le k\le j}\{ \quad S(i-1,k)+w_i(k,j)\}\right\}, & \text{if } \ell_i \le j \le r_i, \\ S(i-1,j), & \text{otherwise.} \end{cases} \tag{6}$$

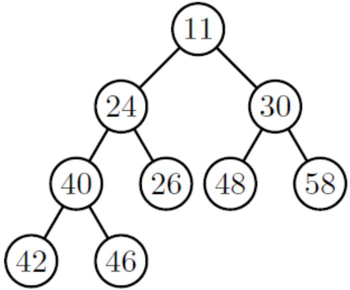
Time complexity: $O(mB\min\{a_{max},B\})$.

The First Modification: Using Binary Heap

Definition

A minimum binary heap is a complete binary tree, in which the value of each node is greater than or equal to the value of its parent, with the minimum-value element at the root.

If q is the number of nodes in a binary heap, then each of the operations: deleting an element, adding a new element and restoring the shape property of a heap can be done in $O(\log q)$ time, while finding the minimum element takes $O(1)$ time.



Theorem

The multiple allocation CFLP on a line graph can be solved using binary heap in $O(mB\log(\min\{a_{max}, B\}))$ *time.*

The Second Modification: $O(mB)$ Time Algorithm

$$S(i,j) = \begin{cases} \min\left\{S(i-1,j), f_i + \min\limits_{\max\{j-\widetilde{a}_i,\ell_i\}\le k\le j}\{\quad S(i-1,k)+w_i(k,j)\}\right\}, & \\ & \text{if } \ell_i \le j \le r_i, \\ S(i-1,j), & \text{otherwise.} \end{cases}$$

The Second Modification: $O(mB)$ Time Algorithm

$$S(i,j) = \begin{cases} \min\left\{S(i-1,j), f_i + \min\limits_{\max\{j-\widetilde{a}_i,\ell_i\}\le k\le j}\{ \quad S(i-1,k)+w_i(k,j)\}\right\}, & \\ & \text{if } \ell_i \le j \le r_i, \\ S(i-1,j), & \text{otherwise.} \end{cases}$$

Consider the i-th row of table S. To compute element $S(i,j)$ one needs to find $\min_{1\le k\le B} A_i(k,j)$, where

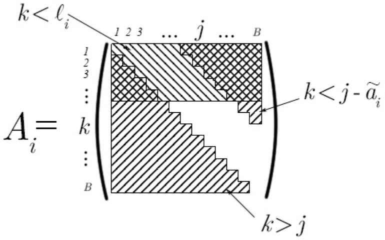
$$A_i(k,j) = \begin{cases} S(i-1,k)+w_i(k,j), & \text{if } \max\{j-\widetilde{a}_i,\ \ell_i\} \le k \le j, \\ \infty, & \text{otherwise,} \end{cases} \qquad (7)$$

The Second Modification: $O(mB)$ Time Algorithm

$$S(i,j) = \begin{cases} \min\Big\{S(i-1,j), f_i + \min\limits_{\max\{j-\widetilde{a}_i,\ell_i\}\le k\le j}\{ \quad S(i-1,k)+w_i(k,j)\}\Big\}, \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } \ell_i \le j \le r_i, \\ S(i-1,j), \qquad\qquad\qquad\qquad\qquad \text{otherwise.} \end{cases}$$

Consider the i-th row of table S. To compute element $S(i,j)$ one needs to find $\min_{1\le k\le B} A_i(k,j)$, where

$$A_i(k,j) = \begin{cases} S(i-1,k)+w_i(k,j), & \text{if } \max\{j-\widetilde{a}_i,\ \ell_i\} \le k \le j, \\ \infty, & \text{otherwise,} \end{cases} \tag{7}$$



We will show how to find minimum element in each column of A_i in $O(B)$ time.

The Second Modification: $O(mB)$ Time Algorithm

Definition

An $\alpha \times \beta$-matrix A with real entries is **monotone in columns**, if for every pair of columns with indices $j_0 < j_1$, it holds that $i(j_0) \leq i(j_1)$, where $i(j)$ is the smallest row index i, such that element $A(i, j)$ equals to the minimum value in the j-th column of A. Matrix A is said to be **totally monotone in columns**, if every 2×2 submatrix of A is monotone.

The Second Modification: $O(mB)$ Time Algorithm

Definition

An $\alpha \times \beta$-matrix A with real entries is **monotone in columns**, if for every pair of columns with indices $j_0 < j_1$, it holds that $i(j_0) \leq i(j_1)$, where $i(j)$ is the smallest row index i, such that element $A(i,j)$ equals to the minimum value in the j-th column of A. Matrix A is said to be **totally monotone in columns**, if every 2×2 submatrix of A is monotone.

Statement [Aggarwal et al., 1987].

Let an $\alpha \times \beta$-matrix A be totally monotone in columns. There exists an algorithm [Aggarwal et al., 1987] that finds the minimum entry in each column of A in $O(\alpha + \beta)$ time.

The Second Modification: $O(mB)$ Time Algorithm

Definition

An $\alpha \times \beta$-matrix A with real entries is **monotone in columns**, if for every pair of columns with indices $j_0 < j_1$, it holds that $i(j_0) \leq i(j_1)$, where $i(j)$ is the smallest row index i, such that element $A(i,j)$ equals to the minimum value in the j-th column of A. Matrix A is said to be **totally monotone in columns**, if every 2×2 submatrix of A is monotone.

Statement [Aggarwal et al., 1987].

Let an $\alpha \times \beta$-matrix A be totally monotone in columns. There exists an algorithm [Aggarwal et al., 1987] that finds the minimum entry in each column of A in $O(\alpha + \beta)$ time.

Lemma 1.

For each $1 \leq i \leq m$, the $B \times B$-matrix A_i defined by (7) is totally monotone in columns.

Theorem

The multiple allocation CFLP on a line graph can be solved in $O(mB)$ time.

Proof.
An improved exact algorithm for the multiple allocation CFLP on a line works as follows.

1. We reduce the multiple allocation CFLP to the multiple allocation CFLP with unit demands as in [Mirchandani et al., 1996].

Theorem

The multiple allocation CFLP on a line graph can be solved in $O(mB)$ time.

Proof.

An improved exact algorithm for the multiple allocation CFLP on a line works as follows.

1. We reduce the multiple allocation CFLP to the multiple allocation CFLP with unit demands as in [Mirchandani et al., 1996].
2. We compute the sums from Remark 1 in $O(m+B)$ time and the values l_i, r_i for $1 \le i \le m$, so that we could further calculate any element $w_i(k, j)$ in $O(1)$ time.

Theorem

The multiple allocation CFLP on a line graph can be solved in $O(mB)$ time.

Proof.
An improved exact algorithm for the multiple allocation CFLP on a line works as follows.

1. We reduce the multiple allocation CFLP to the multiple allocation CFLP with unit demands as in [Mirchandani et al., 1996].
2. We compute the sums from Remark 1 in $O(m+B)$ time and the values l_i, r_i for $1 \leq i \leq m$, so that we could further calculate any element $w_i(k, j)$ in $O(1)$ time.
3. For each $i = 1, \ldots, m$ we compute the i-th row of table S defined by (6) as follows:

Theorem

The multiple allocation CFLP on a line graph can be solved in $O(mB)$ time.

Proof.

An improved exact algorithm for the multiple allocation CFLP on a line works as follows.

1. We reduce the multiple allocation CFLP to the multiple allocation CFLP with unit demands as in [Mirchandani et al., 1996].
2. We compute the sums from Remark 1 in $O(m+B)$ time and the values l_i, r_i for $1 \le i \le m$, so that we could further calculate any element $w_i(k, j)$ in $O(1)$ time.
3. For each $i = 1, \ldots, m$ we compute the i-th row of table S defined by (6) as follows:
4. In a $B \times B$-matrix A_i defined as (7), which is totally monotone in columns, according to Lemma 1, in time $O(B)$ we obtain minimum entries of each column of matrix A_i by applying algorithm from [Aggarwal et al., 1987].

Theorem

The multiple allocation CFLP on a line graph can be solved in $O(mB)$ time.

Proof.

An improved exact algorithm for the multiple allocation CFLP on a line works as follows.

1. We reduce the multiple allocation CFLP to the multiple allocation CFLP with unit demands as in [Mirchandani et al., 1996].
2. We compute the sums from Remark 1 in $O(m+B)$ time and the values l_i, r_i for $1 \leq i \leq m$, so that we could further calculate any element $w_i(k, j)$ in $O(1)$ time.
3. For each $i = 1, \dots, m$ we compute the i-th row of table S defined by (6) as follows:
4. In a $B \times B$-matrix A_i defined as (7), which is totally monotone in columns, according to Lemma 1, in time $O(B)$ we obtain minimum entries of each column of matrix A_i by applying algorithm from [Aggarwal et al., 1987].
5. Having all the minimum entries of each column of A_i been calculated, we can compute all elements of the i-th row of S in additional $O(B)$ time.

Theorem

The multiple allocation CFLP on a line graph can be solved in $O(mB)$ time.

Proof.
An improved exact algorithm for the multiple allocation CFLP on a line works as follows.

1. We reduce the multiple allocation CFLP to the multiple allocation CFLP with unit demands as in [Mirchandani et al., 1996].
2. We compute the sums from Remark 1 in $O(m+B)$ time and the values l_i, r_i for $1 \leq i \leq m$, so that we could further calculate any element $w_i(k,j)$ in $O(1)$ time.
3. For each $i = 1,\ldots,m$ we compute the i-th row of table S defined by (6) as follows:
4. In a $B \times B$-matrix A_i defined as (7), which is totally monotone in columns, according to Lemma 1, in time $O(B)$ we obtain minimum entries of each column of matrix A_i by applying algorithm from [Aggarwal et al., 1987].
5. Having all the minimum entries of each column of A_i been calculated, we can compute all elements of the i-th row of S in additional $O(B)$ time.
6. Finally, since S has m rows the total time complexity of the algorithm is $O(mB)$.

Conclusion and final remarks

For the multiple allocation CFLP on a line

- **In [Mirchandani et al., 1996]:** $O(mB\min\{a_{max}, B\})$ time algorithm
- **Our first modification:** $O(mB\log(\min\{a_{max}, B\}))$ time algorithm.
- **Our second modification:** $O(mB)$ time algorithm.

Conclusion and final remarks

For the multiple allocation CFLP on a line

- **In [Mirchandani et al., 1996]:** $O(mB\min\{a_{max},B\})$ time algorithm
- **Our first modification:** $O(mB\log(\min\{a_{max},B\}))$ time algorithm.
- **Our second modification:** $O(mB)$ time algorithm.

Remark

The second modification contains the algorithm from [Aggarwal et al., 1987], which has a large constant factor in the big O. Therefore, despite of the obvious advantage in the theoretical evaluation of the running-time, in practice for small values of B the second modification may work slower than the first one.

Conclusion and final remarks

For the multiple allocation CFLP on a line

- **In [Mirchandani et al., 1996]:** $O(mB\min\{a_{max}, B\})$ time algorithm
- **Our first modification:** $O(mB\log(\min\{a_{max}, B\}))$ time algorithm.
- **Our second modification:** $O(mB)$ time algorithm.

Remark

The second modification contains the algorithm from [Aggarwal et al., 1987], which has a large constant factor in the big O. Therefore, despite of the obvious advantage in the theoretical evaluation of the running-time, in practice for small values of B the second modification may work slower than the first one.

Open questions:

- Is there an $O(B+m)$ time algorithm for multiple allocation CFLP on a line?
- Is there an efficient pseudopolinomial-time algorithm for the **single allocation** CFLP on a line graph?

References:

- [Aggarwal et al., 1987] A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, R. Wilber, "Geometric applications of a matrix searching algorithm," Algorthmica, 2, 1987, pp. 195–208.
- [Mirchandani et al., 1996] P. Mirchandani, R. Kohli, A. Tamir, "Capacitated location problem on a line," Transportation Science, 30(1), 1996, pp. 75–80.

Thanks for your attention!

Lemma

For each $1 \leq i \leq m$, the $B \times B$-matrix A_i defined by (7) is totally monotone in columns.

Proof. The proof is by contradiction. But first, we need to show that the function $w_i(k, j)$ is concave for each i, that is, for each $i : 1 \leq i \leq m$ and every $1 \leq k_0 < k_1 \leq j_0 < j_1 \leq B$:

$$w_i(k_0, j_0) + w_i(k_1, j_1) \leq w_i(k_0, j_1) + w_i(k_1, j_0). \quad (8)$$

It's proved by definition.

Lemma

For each $1 \le i \le m$, the $B \times B$-matrix A_i defined by (7) is totally monotone in columns.

Proof. The proof is by contradiction. But first, we need to show that the function $w_i(k, j)$ is concave for each i, that is, for each $i : 1 \le i \le m$ and every $1 \le k_0 < k_1 \le j_0 < j_1 \le B$:

$$w_i(k_0, j_0) + w_i(k_1, j_1) \le w_i(k_0, j_1) + w_i(k_1, j_0). \tag{8}$$

It's proved by definition.

Suppose that the matrix A_i defined by (7) is not totally monotone. Therefore, there exist indices $k_0 < k_1$ and $j_0 < j_1$, such that

$$A_i(k_0, j_0) > A_i(k_1, j_0) \text{ and } A_i(k_0, j_1) < A_i(k_1, j_1). \tag{9}$$

- Suppose that the four elements of matrix A_i in (9) are the white elements of A_i. Since element $A_i(k_1, j_0)$ is white, we have $k_1 \leq j_0$, and, therefore, $k_0 < k_1 \leq j_0 < j_1$. Summing the inequalities from (9) and using the definition of $A_i(k, j)$ from (7), we get:

$$S(i-1, k_0) + S(i-1, k_1) + w_i(k_0, j_0) + w_i(k_1, j_1) >$$

$$S(i-1, k_0) + S(i-1, k_1) + w_i(k_0, j_1) + w_i(k_1, j_0),$$

which contradicts the concave property (8) of $w_i(k, j)$.

- Suppose that the four elements of matrix A_i in (9) are the white elements of A_i. Since element $A_i(k_1, j_0)$ is white, we have $k_1 \leq j_0$, and, therefore, $k_0 < k_1 \leq j_0 < j_1$. Summing the inequalities from (9) and using the definition of $A_i(k, j)$ from (7), we get:

$$S(i-1, k_0) + S(i-1, k_1) + w_i(k_0, j_0) + w_i(k_1, j_1) >$$

$$S(i-1, k_0) + S(i-1, k_1) + w_i(k_0, j_1) + w_i(k_1, j_0),$$

which contradicts the concave property (8) of $w_i(k, j)$.

- Suppose that among the four elements of matrix A_i in (9), there exists a gray element.
    - If element $A_i(k_1, j_0)$ is gray, then we get a straightaway contradiction with the first inequality in (9).
    - If element $A_i(k_0, j_1)$ is gray, then we obtain the same for second inequality in (9).
    - If element $A_i(k_0, j_0)$ is gray, then according to the definition of A_i and the choice of indices $k_0 < k_1$ and $j_0 < j_1$, either $A_i(k_1, j_0) = \infty$, or $A_i(k_0, j_1) = \infty$, and we get the same type of contradiction with (9).
    - If element $A_i(k_1, j_1)$ is gray, then again either $A_i(k_1, j_0) = \infty$, or $A_i(k_0, j_1) = \infty$, and we obtain the same contradiction with (9).

Therefore, matrix A_i is totally monotone in columns.

□