Client Assignment Problems for Latency Minimization

Gruia Călinescu * Xiaolang Wang†

July 6, 2018

Abstract

Interactivity is a primary performance measure for distributed interactive applications (DIAs). In a network supporting a DIA, interactivity performance depends on both client-to-server network latencies and inter-server network latencies. An optimization problem, which we term FCSA, is to find an optimum way how clients are assigned to servers such that the largest latency on an interactivity path between two clients (client 1 to server 1, server 1 to server 2, then server 2 to client 2) is minimized. Previous work showed that it is NP-hard to approximate this problem with a ratio better than 4/3 and gave a 3-approximation algorithm. In this paper, we give a (3/2)-approximation algorithm for FCSA, and show that it is NP-hard to obtain a better ratio. We also give a (3/2)-approximation algorithm when server capacity constraints are considered.

1 Introduction

Interactivity is a primary performance measure for distributed applications (DIAs) that enable participants at different locations to interact with each other in real time. The interactivity performance depends on not only client-to-server network latencies but also inter-server network latencies [11]. Zhang

*Gruia Călinescu is with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA; calinescu@iit.edu.

†Xiaolang Wang is with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA; xwang122@hawk.iit.edu.

and Tang [11] modeled the network supporting a DIA by a graph $G(V, E)$, in which V was the set of nodes and E was the set of links between the nodes. A length $d(u, v)$ of link $(u, v) \in E$ represented the network latency between nodes u and v.

A network-latency optimization problem, which we term FCSA, is described as following. The input consists of a finite set V and two subsets of V: $C = \{c_1, c_2, \ldots, c_m\}$ ("clients") and S ("servers"), and a semimetric d on V [1]. A feasible solution consists of a function ("assignment") $f : C \to S$. The objective is to minimize

$$\max_{x,y \in C} d(y, f(y)) + d(x, f(x)) + d(f(x), f(y)).$$

[11] uses a more complete name for FCSA: the Client Assignment Problems for Continuous Distributed Interactive Applications. Another optimization problem with the consideration of capacity constraints on servers, which we term FCSA-SC is a generalization of FCSA. The only difference is that the input of FCSA-SC also contains another function ("capacity") $cap : S \to \mathbb{Z}^+$, which represents the maximum number of clients that can be assigned to a server.

[11] showed that it is NP-hard to approximate the FCSA problem with a ratio better than 4/3, and gave a 3-approximation algorithm. It also gave a 5-approximation algorithm for FCSA-SC. In this paper, we give a (3/2)-approximation algorithm for FCSA, and show that it is NP-hard to obtain a better ratio. Moreover, we give a (3/2)-approximation algorithm for FCSA-SC.

FCSA is a "bottleneck"-type problem (see [6]), such as the classic Metric k-Center problem, covered in [10]. In the Metric k-Center problem, the input is a metric undirected graph and a number k, and one needs to select k vertices (the "centers") to minimize the maximum, over vertices, of the distance from each vertex to the closest center. Hochbaum and Shmoys [5] give a 2-approximation for k-Center, matching the NP-Hardness result of Hsu and Nemhauser [7]. Our results can be seen as a counterpart of these papers. In particular, for FCSA-SC we use the technique of parametric pruning [10].

Further work on the Metric k-Center includes the capacitated version [8, 3], which has constant factor algorithms and a lower bound of 3 on the

[1] A function $d : V \times V \to \mathbb{R}$ is a *semimetric* on V iff for every $i_1, i_2, i_3 \in V$, $d(i_1, i_1) = 0$, $d(i_1, i_2) \geq 0$, $d(i_1, i_2) = d(i_2, i_1)$, and $d(i_1, i_2) + d(i_2, i_3) \geq d(i_1, i_3)$. If, in addition, $d(i_1, i_2) = 0$ implies $i_1 = i_2$, then d is a *metric*.

best possible approximation. [9, 1, 4] obtain approximation algorithms for Metric k-Center in directed metric graphs, matching (up to a constant) the hardness results in [2].

2 The algorithm for FCSA

For every client c_i and server s_x, compute $L(i,x) = \max_{c_j \in C,\, j \neq i} d(c_i, s_x) + d(s_x, c_j)$. Then let $x_i = \arg\min_x L(i,x)$, breaking ties arbitrarily.

Claim 1 *For all clients c_i, we have $OPT \geq L(i, x_i)$, where OPT is the objective of the optimal solution.*

Proof: Let the optimal solution assigns client c_j to server s_{j^*} (for all j). Then, for all clients $c_j \neq c_i$, we have $OPT \geq d(c_i, s_{i^*}) + d(s_{i^*}, s_{j^*}) + d(c_j, s_{j^*}) \geq d(c_i, s_{i^*}) + d(s_{i^*}, c_j)$ (we used the triangle inequality). As this holds for all $c_j \neq c_i$, we deduce $OPT \geq L(i, i^*)$. From the definition of x_i, we have $L(i, i^*) \geq L(i, x_i)$ and the claim follows. ∎

Let $L = \max_{c_i \in C} L(i, x_i)$. Then:

Corollary 2 *$OPT \geq L$.*

Our algorithm returns the best of the following $m + 1$ solutions:

- Assign all clients to server s_{x_1}.
- Assign all clients to server s_{x_2}.
- ...
- Assign all clients to server s_{x_m}.
- For all i, assign client c_i to server s_{x_i}.

It is straightforward to implement this algorithm in polynomial time. Assume that our input is a distance matrix of size $(m+n)^2$, where m and n are the number of clients and servers respectively. For each server s_j, we can find two clients with longest and second longest distances to it in $O(m)$ steps, we call them y_j and z_j respectively (if more than one clients have the same longest distance to the server, arbitrarily choose two of them as y_j and z_j).

Then, we can find x_i for a client c_i by finding a server s_j with minimum $d(c_i, s_j) + d(s_j, y_j)$ as x_i; specially, if c_i is the y_j for server s_j, we use value $d(c_i, s_j) + d(s_j, z_j)$ instead of $d(c_i, s_j) + d(s_j, y_j)$ for this server. Thus in $O(m \cdot n)$ steps, can we find x_i for all clients. There are $n+1$ solutions. Each of the first n solutions takes only constant time, since an assignment of all clients to server s_j returns $d(s_j, y_j) + d(s_j, z_j)$. The last solution takes $O(m^2)$ steps: we need to calculate the objective for each pair of clients then return the maximum among them. Therefore, in conclusion, our algorithm has a time complexity $O(m \cdot (n+m))$.

Theorem 3 *The algorithm above is a* $(3/2)$*-approximation for FCSA.*

Proof: In the first case, there exists a client c_i such that $d(c_i, s_{x_i}) \geq L/4$. Then we look at the solution that assigns all the clients to server s_{x_i}. First, we notice that for any client $c_j \neq c_i$, we have

$$d(c_i, s_{x_i}) + d(s_{x_i}, s_{x_i}) + d(c_j, s_{x_i}) \leq L.$$

Second, we notice that for all $c_j \neq c_i$, we have

$$d(c_j, s_{x_i}) \leq L - d(c_i, s_{x_i}) \leq (3/4)L.$$

Therefore, for all $c_j \neq c_k$ and $c_i \notin \{c_j, c_k\}$, we have:

$$d(c_j, s_{x_i}) + d(s_{x_i}, s_{x_i}) + d(c_k, s_{x_i}) \leq 2(3/4)L = (3/2)L,$$

which combined with Corollary 2 finishes the proof in the first case.

In the second and the last case, we have that for all clients c_i, $d(c_i, s_{x_i}) \leq L/4$. We look at the last solution, where each client c_i is assigned to its s_{x_i}. Consider now two distinct clients c_j and c_k, we have

$$\begin{aligned}
d(c_j, s_{x_j}) + d(s_{x_j}, s_{x_k}) + d(c_k, s_{x_k}) \leq\ & d(c_j, s_{x_j}) + d(s_{x_j}, c_k) + d(c_k, s_{x_k}) + d(c_k, s_{x_k}) \\
\leq\ & L(j, x_j) + 2d(c_k, s_{x_k}) \\
\leq\ & L + 2d(c_k, s_{x_k}) \\
\leq\ & L + 2(L/4) \\
=\ & (3/2)L.
\end{aligned}$$

The first inequality comes from the triangle inequality, the second inequality comes from the definition of $L(j, x_j)$, the third inequality comes from the

definition of L, and the fourth inequality comes from the fact we are in the second case. Combined with Corollary 2, this finishes the proof in the second and final case. ■

Figure 1 gives a tight example of (3/2)-approximation using our algorithm. There are three clients (c_1, c_2 and c_3) and six servers ($s_{1,1}$, $s_{1,2}$, $s_{2,1}$, $s_{2,2}$, $s_{3,1}$ and $s_{3,2}$). The distance between two vertices a and b without an edge (a, b) is the length of the shortest path between them. For example, $\mathrm{d}(s_{1,1}, s_{1,2}) = 2 + 2\epsilon$, where ϵ is any value greater than 0. Assigning all clients to one of the servers among s_{x_1} ($s_{1,2}$), s_{x_2} ($s_{2,1}$) and s_{x_3} ($s_{3,1}$) has a smallest objective 6 (assigning all clients to $s_{1,2}$). Assigning each client c_i to its s_{x_i} has an objective $6 + \epsilon$. Our algorithm returns the best result among four solutions above, which is 6. The optimal solution is to assign c_1 to $s_{1,1}$, c_2 to $s_{2,1}$ and c_3 to $s_{3,2}$ with an objective $4 + 2\epsilon$. By making ϵ be arbitrarily small, we obtain that the approximation ratio of our algorithm is exactly 3/2.

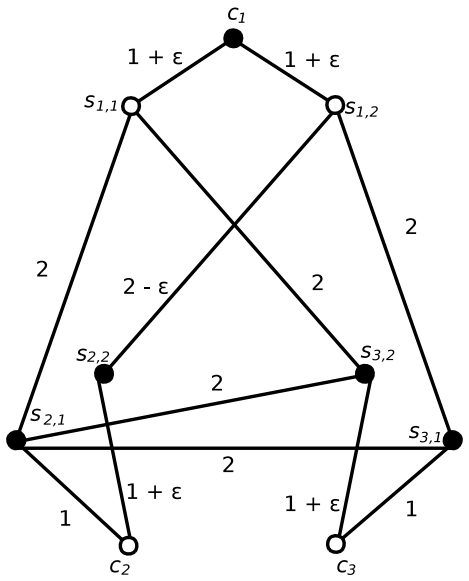

Figure 1: Example of tight 3/2-approximation for FCSA

3 Inapproximability result

Consider the following reduction from 3SAT to FCSA (part of the construction is the textbook reduction from 3SAT to Clique). Let Φ be a 3SAT Boolean formula. For every clause C_j with literal $x_{j_1}, x_{j_2}, x_{j_3}$, produce a client c_j at distance 1 to servers $s_{j,1}$, $s_{j,2}$, and $s_{j,3}$. For two servers $s_{j,i}$ and $s_{j',i'}$, make the distance between them 2 if $j \neq j'$ and the two literals x_{j_i} and $x_{j'_{i'}}$ are not complementary of each other. Use shortest paths to determine the distance between other pairs of vertices. Call the constructed FCSA instance $f(\Phi)$. Figure 2 is an example of an FCSA instance constructed from a 3SAT instance.

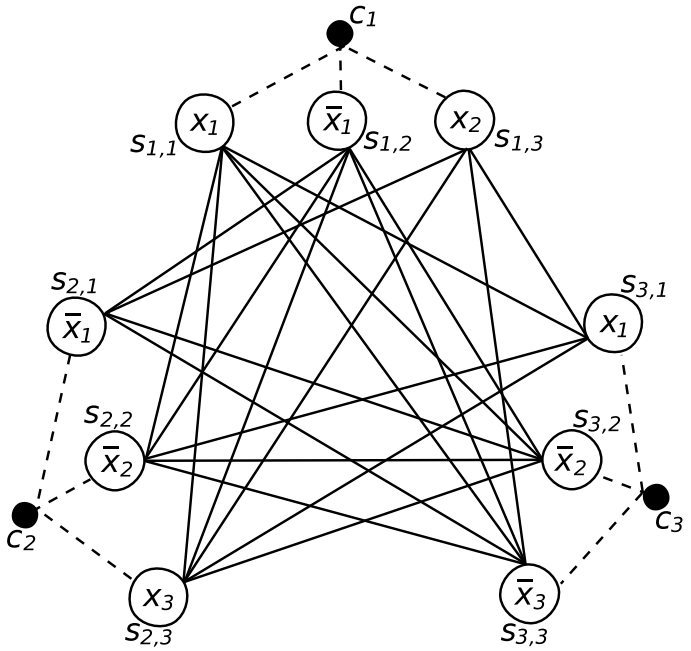

Figure 2: Example of the FCSA instance constructed from the 3SAT $\Phi = (x_1 \vee \bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3)$. A solid edge has length 2 and a dashed edge has length 1. The distance between two vertices without an edge between them is the length of the shortest path from one to the other.

Claim 4 *If the formula Φ is satisfiable, then $f(\Phi)$ has a solution of objective 4.*

Proof: In a satisfying truth assignment g of Φ, each clause C_j has (at least) one literal $x_{j_{j'}}$ that is $True$. Assign each client c_j to server $s_{j,j'}$; then we have

$$d(c_j, s_{j,j'}) + d(s_{j,j'}, s_{k,k'}) + d(c_k, s_{k,k'}) = 1 + 2 + 1 = 4.$$

Here, $d(s_{j,j'}, s_{k,k'}) = 2$ comes from the fact that $x_{j_{j'}}$ and $x_{k_{k'}}$ are not literals complementary of each other, as both are set to $True$ in g. ■

Claim 5 *If $f(\Phi)$ has a solution of objective less than 6, then Φ is satisfiable.*

As an illustration, for the example in Figure 2, there is an assignment with objective 4 in $f(\Phi)$: assigning c_1 to $s_{1,2}$, c_2 to $s_{2,2}$, and c_3 to $s_{3,2}$. Using this assignment for $f(\Phi)$, we can construct a satisfying assignment for Φ as following: assign $\bar{x}_1$ to $True$ for the first clause, assign $\bar{x}_2$ to $True$ for the second clause, assign $\bar{x}_2$ to $True$ for the third clause, then assign the unassigned x_3 to $True$. A satisfying assignment for Φ is $x_1 = False, x_2 = False, x_3 = True$.

Proof: Any two-clause formula is satisfiable. We assume from now on that Φ has at least three clauses, and as a result $f(\Phi)$ has at least three clients.

Based on our construction and on the fact that $f(\Phi)$ has a finite objective, for any two servers $s_{j,j'}$ and $s_{k,k'}$, the value of $d(s_{j,j'}, s_{k,k'})$ can only be 2 or 4; $d(s_{j,j'}, s_{k,k'}) = 4$ only when $x_{j_{j'}}$ is the negation of $x_{k_{k'}}$ and they appear in different clauses. For any client c_j and any server $s_{k,k'}$, their distance is at least 1. So, for any path between two clients involving two servers, say $s_{j,j'}$ and $s_{k,k'}$, with $d(s_{j,j'}, s_{k,k'}) = 4$, we have

$$d(c_a, s_{j,j'}) + d(s_{j,j'}, s_{k,k'}) + d(c_b, s_{k,k'}) \geq 1 + 4 + 1 = 6.$$

Thus, if $f(\Phi)$ has a solution of objective less than 6, we have an assignment without assigning two clients to two servers at relative distance 4.

Another observation is that, there does not exist a client-server pair at distance 2: c_j is at distance 1 to only three servers $s_{j,1}$, $s_{j,2}$ and $s_{j,3}$; c_j is at distance at least 3 to server $s_{k,k'}$ $(k \neq j)$ because $d(s_{j,j'}, s_{k,k'})$ is at least 2.

Thus, any assignment including assigning a client c_j to a server not among $s_{j,1}$, $s_{j,2}$ or $s_{j,3}$, say $s_{k,k'}$ for $k \neq j$, will have an objective at least

$$d(c_j, s_{k,k'}) + d(s_{k,k'}, s_{b,b'}) + d(c_a, s_{b,b'}) \geq d(c_j, s_{k,k'}) + d(s_{k,k'}, c_a) \geq 3 + 3 = 6,$$

where c_a is a client other than c_j and c_k, and $s_{b,b'}$ is the server assigned to c_a (the first inequality comes from the triangle inequality). Therefore,

in conclusion, if $f(\Phi)$ has a solution of objective less than 6, we have an assignment that only uses a set of servers at relative distances 2 and only assigns a client c_j to one of the $s_{j,j'}$.

Using this assignment of $f(\Phi)$, a satisfying assignment for Φ can be constructed in polynomial time: for each client c_j and its assigned server $s_{j,j'}$, assign $x_{j_{j'}}$ to $True$; then if neither x_i nor $\bar{x}_i$ is assigned with a Boolean value, assign x_i to $True$. Here are the reasons why this is a satisfying assignment. We assigned at least one literal to $True$ in each clause, because clause C_j is represented by a client c_j in $f(\Phi)$, and we assigned one of the literals in clause C_j represented by c_j's assigned server to $True$. We never assigned any literal and its negation both to $True$, because in $f(\Phi)$ we never assigned two clients to two servers at relative distance 4. ■

Thus an algorithm with approximation ratio smaller than 3/2 can determine if Φ is satisfiable or not: if the algorithm returns a solution of objective less than 6, we know from the previous claim that Φ is satisfiable; on the other hand, if the algorithm returns a solution of objective at least 6, we know that the optimal solution has objective strictly greater than 4, and it follows from Claim 4 that Φ is not satisfiable. We obtain that

Theorem 6 *Unless $P = NP$, no polynomial-time algorithm for FCSA can have approximation ratio smaller than* 3/2.

4 The algorithm for FCSA-SC

Here, we only consider the case that all servers have a capacity 1, because

Claim 7 *For each FCSA-SC instance, there is an equivalent FCSA-SC instance in which $\forall s_i \in S,\ cap(s_i) = 1$.*

Proof: Given an FCSA-SC instance Λ, we can construct another FCSA-SC instance $\hat{\Lambda}$ as following. For each $s_i \in S(\Lambda)$, we construct in $\hat{\Lambda}$ $cap(i)$ servers with capacity 1: $s_{i_1}, s_{i_2} ... s_{i_{cap(s_i)}}$. We construct the same set of client in $\hat{\Lambda}$ as in Λ: $C(\hat{\Lambda}) = C(\Lambda)$. And we define the length function $\hat{d}$ in $\hat{\Lambda}$ as following:

$$\begin{cases} \hat{d}(s_{i_j}, s_{i_k}) = 0, & j \neq k; \\ \hat{d}(s_{i_p}, s_{j_q}) = d(s_i, s_j), & i \neq j, s_i, s_j \in S(\Lambda); \\ \hat{d}(s_{i_j}, c) = d(s_i, c), & s_i \in S(\Lambda), c \in C(\Lambda) = C(\hat{\Lambda}); \\ \hat{d}(c_i, c_j) = d(c_i, c_j), & i \neq j, c_i, c_j \in C(\Lambda) = C(\hat{\Lambda}). \end{cases}$$

If $\hat{\Lambda}$ has feasible solution $\hat{f}$, the following is a feasible solution for Λ with the same objective: for each client c_i, if $\hat{f}(c_i) = s_{j_k}$, then $f(c_i) = s_j$. Because k is at most $cap(j)$, the number of clients assigned to c_i in Λ will not exceed the capacity of s_j. And similarly, if Λ has feasible solution f, we can construct a feasible solution $\hat{f}$ for $\hat{\Lambda}$ with the same objective as following: for each client c_i, if $f(c_i) = s_j$, assign c_i to one of the s_{j_k} in $\hat{\Lambda}$. ■

It is obvious that no feasible solution exists if there are more clients than servers under the assumption that each server has a capacity 1. Thus, we assume $|C| \leq |S|$ in order to guarantee a feasible solution. We denote OPT as the optimum objective for an FCSA-SC instance, and we denote P as a constant which represents a guessed value of the optimum objective. The value of P will be adjusted through the process of the algorithm, and the way to adjust it will be talked about later in this section.

For a value P, for every pair of client c_i and server s_j, and for every other client c_k, we find a set $\Psi^P_{i,j,k}$ of servers with

$$\Psi^P_{i,j,k} = \{s \mid d(c_i, s_j)+d(s, s_j)+d(s, c_k) \leq P \wedge d(c_i, s_j) \geq (1/4)P, s \in S \backslash \{s_j\}, c_k \neq c_i\}.$$

Then, for a value P and for each client c_i, we find a set Θ^P_i of servers with

$$\Theta^P_i = \{s \mid d(c_i, s) + \max_{j \neq i} d(s, c_j) \leq P \wedge d(c_i, s) \leq (1/4)P, s \in S, c_j \in C\}.$$

We build the following $m \cdot n$ bipartite graphs:

- $G^P_{1,1}(L^P_{1,1}, R^P_{1,1}, E^P_{1,1})$, $L^P_{1,1} = C \setminus \{c_1\}$, $R^P_{1,1} = S \setminus \{s_1\}$, $E^P_{1,1} = \{(c_k, s) \mid s \in \Psi^P_{1,1,k}\}$
- $G^P_{1,2}(L^P_{1,2}, R^P_{1,2}, E^P_{1,2})$, $L^P_{1,2} = C \setminus \{c_1\}$, $R^P_{1,2} = S \setminus \{s_2\}$, $E^P_{1,2} = \{(c_k, s) \mid s \in \Psi^P_{1,2,k}\}$
- ...
- $G^P_{2,1}(L^P_{2,1}, R^P_{2,1}, E^P_{2,1})$, $L^P_{2,1} = C \setminus \{c_2\}$, $R^P_{2,1} = S \setminus \{s_1\}$, $E^P_{2,1} = \{(c_k, s) \mid s \in \Psi^P_{2,1,k}\}$
- $G^P_{2,2}(L^P_{2,2}, R^P_{2,2}, E^P_{2,2})$, $L^P_{2,2} = C \setminus \{c_2\}$, $R^P_{2,2} = S \setminus \{s_2\}$, $E^P_{2,2} = \{(c_k, s) \mid s \in \Psi^P_{2,2,k}\}$
- ...

- $G^P_{m,n}(L^P_{m,n}, R^P_{m,n}, E^P_{m,n})$, $L^P_{m,n} = C \setminus \{c_m\}$, $R^P_{m,n} = S \setminus \{s_n\}$, $E^P_{m,n} = \{(c_k, s) \mid s \in \Psi^P_{m,n,k}\}$

Then we build another bipartite graph G^P_0 as following:

$$G^P_0(L^P_0, R^P_0, E^P_0), L^P_0 = C, R^P_0 = S, E^P_0 = \{(c_k, s) \mid s \in \Theta^P_k\}.$$

We check the existence of at least one complete matching in any of the $m \cdot n+1$ graphs, where a complete matching is defined as following.

Definition 1 *A matching h in a bipartite graph $G(L, R, E)$ is a* complete matching *if $|h| = min(|L|, |R|)$.*

Note that all our graphs have $L^P_* \leq R^P_*$, as we assumed the number of servers is at least the number of clients.

Till here, we have the first half of our algorithm which is called *matching-based assignment*. From here on in the second half of our algorithm, we will use matching-based assignment to adjust the value of P and to find a feasible solution that is a (3/2)-approximation of FCSA-SC problem.

For every two clients c_i, c_j and every two servers s_x, s_y, we calculate

$$p_{i,x,y,j} = d(c_i, s_x) + d(s_x, s_y) + d(s_y, c_j).$$

Then we sort all $m^2 \cdot n^2$ results in non-decreasing order, and we use P_z to re-denote each $p_{i,x,y,j}$ where z is its ordering number, $1 \leq z \leq m^2 \cdot n^2$. We adjust the value of P as following:

1. First, let $P = P_1$, if at least one complete matching exists, then we finish the adjustment with $P = P_1$.

2. Otherwise, we keep two values of P: P_i, P_j where $j > i$, such that no complete matching exists if $P = P_i$ and at least one complete matching exists if $P = P_j$. (the reason why this holds for the initial value of j given below is proven later while proving Claim 10). Let $i = 1$ and $j = m^2 \cdot n^2$ as initial values and use binary search to shrink $j - i$ by half: if a complete matching exists when $P = P_{\lfloor (i+j)/2 \rfloor}$ then let $j = \lfloor (i + j)/2 \rfloor$; otherwise, let $i = \lfloor (i + j)/2 \rfloor$.

3. Repeat this shrinking procedure until we find consecutive values P_i, P_j where $i = j-1$, such that no complete matching exists if $P = P_i = P_{j-1}$ and at least one complete matching exists if $P = P_j$. We finish the adjustment with $P = P_j$.

With the adjusted value of P, there exists at least one complete matching among $m \cdot n + 1$ graphs. Our algorithm returns one of the following results:

- if a complete matching is found in one of the first $m \cdot n$ graphs, $G^P_{i,j}$, then assign c_i to s_j and assign every other client to its matched server in $G^P_{i,j}$.
- if a complete matching is found in G^P_0, assign every client to its matched server in G^P_0.

Till here, we finish our algorithm for FCSA-SC. And before the proof of our algorithm is a (3/2)-approximation of FCSA-SC, let us prove some claims first.

Claim 8 *If a complete matching exists in any of the $m \cdot n + 1$ graphs ($G^P_{1,1}, G^P_{1,2}...G^P_{n,m}, G^P_0$), then our matching-based algorithm obtains a feasible solution with objective at most $(3/2)P$.*

Proof: Assume that the solution given by our algorithm is called f'.

The first case is that, the solution is related to a complete matching in one of the first $m \cdot n$ graphs, say $G^P_{i,j}$. In graph $G^P_{i,j}$ we connect a client c_k to servers in the set $\Psi^P_{i,j,k}$, and based on the definition of set $\Psi^P_{i,j,k}$ we have:

$$\forall c_k \in L^P_{i,j},\ \forall s \in \Psi^P_{i,j,k},\ d(c_i, s_j) + d(s, s_j) + d(s, c_k) \leq P, \tag{1}$$

and

$$d(c_i, s_j) \geq (1/4)P,$$

which together with Inequality (1) implies

$$\forall c_k \in L^P_{i,j},\ \forall s \in \Psi^P_{i,j,k},\ d(s, s_j) + d(s, c_k) \leq (3/4)P. \tag{2}$$

For any two clients x and y other than c_i we have:

$$\begin{aligned}
& d(x, f'(x)) + d(y, f'(y)) + d(f'(x), f'(y)) \\
\leq\ & d(x, f'(x)) + d(y, f'(y)) + d(f'(x), s_j) + d(f'(y), s_j) \\
=\ & d(x, f'(x)) + d(f'(x), s_j) + d(y, f'(y)) + d(f'(y), s_j) \\
\leq\ & (3/4)P + (3/4)P \\
=\ & (3/2)P
\end{aligned}$$

The first inequality comes from triangle inequality, and the second inequality follows Inequality (2). Together with Inequality (1), we finish the proof of the first case.

The second and the last case is that, the solution is related to a complete matching in graph G_0^P. In graph G_0^P we connect a client c_i to servers in set Θ_i^P, and based on the definition of set Θ_i^P we have:

$$\forall s \in \Theta_i^P,\ \forall c_j \neq c_i,\ d(c_i, s) + d(s, c_j) \leq P, \tag{3}$$

and

$$\forall s \in \Theta_i^P,\ d(c_i, s) \leq (1/4)P. \tag{4}$$

For any two clients x and y we have:

$$\begin{aligned}
& d(x, f'(x)) + d(y, f'(y)) + d(f'(x), f'(y)) \\
\leq\ & d(x, f'(x)) + d(y, f'(y)) + d(f'(x), y) + d(y, f'(y)) \\
=\ & d(x, f'(x)) + d(f'(x), y) + 2 \cdot d(y, f'(y)) \\
\leq\ & d(x, f'(x)) + d(f'(x), y)) + 2 \cdot (1/4)P \\
\leq\ & P + (1/2)P \\
=\ & (3/2)P
\end{aligned}$$

The first inequality comes from triangle inequality, the second and the third inequalities follow the Inequality (4) and (3) respectively. Till here, we finish the proof of the second and the last case. ■

Claim 9 *If $P \geq OPT$, then at least one complete matching exists among the $m \cdot n + 1$ graphs.*

Proof: Remind that OPT is the objective of the optimal solution, f.

The first case is that, there exists a client c_i with $d(c_i, f(c_i)) \geq (1/4)P$. Let $s_j = f(c_i)$, and in this case a complete matching exists in graph $G_{i,j}^P$. Here is the proof.

For every client $c_k \neq c_i$ we have

$$d(c_i, s_j) + d(s_j, f(c_k)) + d(c_k, f(c_k)) \leq OPT.$$

Because $P \geq OPT$, we also have

$$d(c_i, s_j) + d(s_j, f(c_k)) + d(c_k, f(c_k)) \leq P. \tag{5}$$

Together with the fact that $d(c_i, s_j) \geq (1/4)P$, from Inequality (5) it is easy to see that:

$$\forall c_k \neq c_i,\ f(c_k) \in \Psi^P_{i,j,k},$$

which guarantees a complete matching in graph $G^P_{i,j}$, and we finish the proof of the first case.

The second and the last case is that, every client c_i has $d(c_i, f(c_i)) \leq (1/4)P$. Let $c_{k(i)} = \arg\max_{j \neq i} d(f(c_i), c_j)$, and in this case a complete matching exists in graph G^P_0. Here is the proof. We have

$$\begin{aligned} & d(c_i, f(c_i)) + d(f(c_i), c_{k(i)}) \\ \leq\ & d(c_i, f(c_i)) + d(f(c_i), f(c_{k(i)})) + d(c_{k(i)}, f(c_{k(i)})) \\ \leq\ & OPT \end{aligned}$$

The first inequality follows triangle inequality. Again, because $P \geq OPT$, we also have

$$d(c_i, f(c_i)) + d(c_k, f(c_i)) \leq P. \tag{6}$$

Together with the fact that $d(c_i, f(c_i)) \leq (1/4)P$, from Inequality (6) it is easy to see that:

$$\forall c_i,\ f(c_i) \in \Theta^P_i,$$

which guarantees a complete matching in graph G^P_0, and we finish the proof of the second and the last case. ■

Claim 10 *In the second half of our algorithm, after the adjustment for the value of P, $P \leq OPT$.*

Proof: It is easy to see OPT equals to one of the P_z, $1 \leq z \leq m^2 \cdot n^2$.

The first case is that, we finish the adjustment in the first step, which means we find at least one complete matching when $P = P_1$. Since P_1 is the smallest possible value of OPT, so in this case $P = P_1 \leq OPT$ and we finish the proof of the first case.

The second and the last case is that, we finish the adjustment in the third step. In this case, let us look into the adjustment procedure. In the second step we set the initial values $i = 1, j = m^2 \cdot n^2$. We can make sure that no complete matching exists when $P = P_1$ because we come to the second step already. Also, and we can make sure to find at least one complete matching when $P = P_{m^2 \cdot n^2}$ because of the fact that $P_{m^2 \cdot n^2} \geq OPT$ and Claim 9.

We finish the adjustment with $P = P_j$ in the third step, where at least one complete matching exists if $P = P_j$ but no complete matching exists if

$P = P_{j-1}$. From the contrapositive of Claim 9, we have $P_{j-1} < OPT$, which implies $P_j \leq OPT$. Till here, we finish the proof of the second and the last case. ■

Combining Claim 8 and Claim 10, it is easy to see our algorithm gives a solution with objective at most $(3/2)OPT$, which means

Theorem 11 *The algorithm we give is a (3/2)-approximation for FCSA-SC.*

Acknowledgments: Gruia Calinescu thanks Niranjana Sompura Ramakrishna Reddy, a student in his Combinatorial Optimization class, for bringing this problem to his attention.

References

[1] Aaron Archer. Two $O(\log^* k)$-approximation algorithms for the asymmetric k-center problem. In Karen Aardal and Bert Gerards, editors, *Integer Programming and Combinatorial Optimization*, volume 2081 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin Heidelberg, 2001.

[2] Julia Chuzhoy, Sudipto Guha, Eran Halperin, Sanjeev Khanna, Guy Kortsarz, Robert Krauthgamer, and Joseph (Seffi) Naor. Asymmetric k-center is $\log^* n$-hard to approximate. *J. ACM*, 52(4):538–551, July 2005.

[3] Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. LP rounding for k-centers with non-uniform hard capacities. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 273–282. IEEE Computer Society, 2012.

[4] Inge Li Gørtz and Anthony Wirth. Asymmetry in k-center variants. *Theoretical Computer Science*, 361(2–3):188 – 199, 2006.

[5] D.S. Hochbaum and D.B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10:180–184, 1985.

[6] D.S. Hochbaum and D.B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986.

[7] Wen-Lian Hsu and George L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209 – 215, 1979.

[8] Samir Khuller and Yoram J. Sussmann. The capacitated k-center problem. *SIAM J. Discrete Math.*, 13(3):403–418, 2000.

[9] Rina Panigrahy and Sundar Vishwanathan. An $O(\log^* n)$ approximation algorithm for the asymmetric p-center problem. *J. Algorithms*, 27(2):259–268, May 1998.

[10] V.V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[11] Lu Zhang and Xueyan Tang. The client assignment problem for continuous distributed interactive applications: Analysis, algorithms, and evaluation. *Parallel and Distributed Systems, IEEE Transactions on*, 25(3):785–795, March 2014.