Analysis of Minimum Interaction Time for Continuous Distributed Interactive Computing

Lu Zhang, *Member, IEEE,* Xueyan Tang, *Member, IEEE,* and Bingsheng He, *Member, IEEE*

Abstract—Distributed interactive computing allows participants at different locations to interact with each other in real time. In this paper, we study the interaction times of continuous Distributed Interactive Applications (DIAs) in which the application states change due to not only user-initiated operations but also time passing. Given the clients and servers of a continuous DIA, its interaction time is directly affected by how the clients are assigned to the servers as well as the simulation time settings of the servers. We formulate the Minimum Interaction Time (MIT) problem as a combinatorial problem of these two tuning knobs and prove that it is NP-hard. We then approximate the problem by fixing the client assignment or the simulation time offsets among the servers. When the client assignment is fixed, we show that finding the minimum achievable interaction time can be reduced to a weighted bipartite matching problem. We further show that this approach establishes a tight approximation factor of 3 to the MIT problem if each client is assigned to its nearest server. When the simulation time offsets among the servers are fixed, we show that finding the minimum achievable interaction time is still NP-hard. This approach can approximate the MIT problem by a factor within 2 if the simulation times of all servers are synchronized. A mix of the above two approaches better approximates the MIT problem within a factor of 5/3. We further conduct experimental evaluation of these approaches with three real Internet latency datasets.

Index Terms—distributed interactive computing, interaction time, consistency, approximation algorithm.

1 Introduction

Recent years have witnessed rapid development of distributed interactive computing in many areas such as interactive digital media and entertainment [21], distributed interactive simulation [2], and collaborative computer-aided design and engineering [1]. In large-scale distributed interactive computing, the application state (such as the virtual worlds in multiplayer online games) is typically maintained across a group of geographically distributed servers [21]. Each participant, known as a client, is assigned to one server and connects to it for sending user-initiated operations. When the application state changes, state updates are delivered to the clients by their assigned servers to reflect the changes. In this way, Distributed Interactive Applications (DIAs) enable participants at different locations to interact with each other in real time.

Interactivity is of crucial importance to DIAs for supporting graceful interactions among participants. The interactivity performance can be characterized by the duration from the time when a client issues an operation to the time when the effect of the operation is presented to others [14]. This duration is known as the *interaction time* between clients. Since the clients interact with one another through their assigned servers, the interaction time between any pair of clients must include not only the network latencies between the clients and their assigned servers, but also the network latency between their assigned servers. These latencies are directly affected by how the clients are assigned to the servers. In addition to network latencies, the interaction time is also influenced by the need for consistency maintenance in DIAs. Consistency means that shared common views of the application state must be created among all clients and it is a fundamental requirement for supporting meaningful interactions [7].

In this paper, we study the interaction times of DIAs. We focus on continuous DIAs in which the application state changes due to not only user-initiated operations but also time passing. Examples of continuous DIAs include distributed virtual environments [22], distributed interactive simulations [2], and multiplayer online games [9]. In continuous DIAs, the progress of the application state is normally measured along a synthetic time scale known as the *simulation time* (for example, the time elapsed in the virtual game world). To ensure consistency among the application states at the servers, each user operation must be executed by all servers at the same simulation time [17]. As a result, maintaining consistency in continuous DIAs often entails artificial synchronization delays in the interactions among clients [3], [6], [12], [16], [17]. The amount of synchronization delays is dependent on the simulation time settings of clients and servers.

We formulate the Minimum Interaction Time (MIT) problem for continuous DIAs as a combinatorial optimization problem that includes two sets of variables: the client assignment and the simulation time offsets among servers. We show that the MIT problem is NP-hard. Two approaches are then considered to approximate the MIT problem: by fixing the client assignment and by fixing the simulation time offsets among servers. When the client assignment is fixed, we show that finding the minimum achievable interaction time can be reduced to a weighted bipartite

- *Lu Zhang is with the Computer Science and Computer Engineering Department, University of Arkansas, Fayetteville, AR, 72701. Email: lz006@uark.edu.*
- *Xueyan Tang is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, 639798. E-mail: asxytang@ntu.edu.sg.*
- *Bingsheng He is with the School of Computing, National University of Singapore, Singapore 119077. E-mail: he.bingsheng@gmail.com.*

matching problem. We further show that for networks with the triangle inequality, this approach establishes a tight approximation factor of 3 to the MIT problem if each client is assigned to its nearest server. When the simulation time offsets among servers are fixed, we show that finding the minimum achievable interaction time is still NP-hard. For networks with the triangle inequality, this approach can approximate the MIT problem within a factor of 2 if the simulation times of all servers are synchronized. A mix of the above two approaches better approximates the MIT problem within a factor of 5/3. These approaches are also experimentally evaluated using three real Internet latency datasets.

Most previous studies on client assignment for DIAs have focused on reducing client-to-server latencies only [24], [25]. Some recent work has investigated client assignment by taking inter-server latencies into account as well for optimizing interactivity [29]. However, no synchronization delay for consistency maintenance was considered. Therefore, it was only applicable to discrete DIAs that change their application states in response to user operations only (such as collaborative text editors). In addition, some client assignment heuristics have also been studied for optimizing the maximum length of interaction paths among all clients in DIAs [30]. Different from the above work, this paper conducts in-depth theoretical analysis of consistency-constrained simulation time settings and achievable interactivity for continuous DIAs. Where to place servers in the network is another relevant issue that has been extensively studied for various Internet applications [5], [16], [33]. Server placement is often related to several graph theoretic problems including the facility location problem and the k-center and k-median problems [5], [20]. Our work complements server placement in that we consider how to optimize the client assignment and the simulation time settings given a set of servers placed. To the best of our knowledge, there has been no study on the interplay of these two tuning knobs for improving the interactivity of continuous DIAs. Some preliminary results of our work were presented as a brief announcement [31].

The rest of this paper is organized as follows. Section 2 formulates the Minimum Interaction Time (MIT) problem for continuous DIAs. Sections 3 and 4 study the MIT problem and present the hardness results. Section 5 analyzes several approaches to approximate the MIT problem. Section 6 describes the experimental evaluation. Finally, Section 7 concludes the paper.

2 Problem Formulation

A DIA can be modeled by a network consisting of a set of nodes V. A distance $d(u, v)$ is associated with each pair of nodes $(u, v) \in V \times V$, representing the network latency of the routing path between nodes u and v. Denote by $S \subseteq V$ the set of servers and $C \subseteq V$ the set of clients in the network. Each client is assigned to one server for sending user operations and receiving state updates.

Each server and client has an associated simulation time to characterize its view of the application state. To provide realistic real-time interaction experiences, the simulation times of all the servers and clients should advance at the same rate as that of the wall-clock time. When a client issues an operation, the effect of the operation is presented to other clients through the following process. First, the client sends the operation to its assigned server. Then, the server forwards the operation to all the other servers. On receiving the operation, each server executes the operation, possibly after some synchronization delay, to compute the new state of the application. Finally, each server delivers the resultant state update to all the clients assigned to it. Since clients inherit the application state from their assigned servers, to allow all clients to always see identical states at the same simulation time, the application states at all the servers must be consistent at any simulation time. This in turn requires each user operation to be executed by all servers at the same simulation time, since the state of a continuous DIA changes due to both user operations and time passing.

In general, the simulation times of servers and clients do not have to be synchronized. The offsets between simulation times at servers and clients can actually be exploited to improve interaction times. To facilitate the presentation of our analysis, we shall denote the clients by $c_1, c_2, \cdots, c_{|C|}$, where $|C|$ is the number of clients. For each client c_i, we denote by $s_A(c_i) \in S$ the assigned server of c_i in a client assignment A. We also denote by $\delta_{c_i} \in \mathbb{R}$ the offset of each client c_i's simulation time relative to a reference clock (the larger the offset, the further ahead the simulation time). Similarly, for each server $s \in S$, we denote by $\delta_s \in \mathbb{R}$ the offset of s's simulation time relative to the reference clock.

Consider an operation issued at simulation time t by a client c_i. Suppose that all servers execute the operation at simulation time $t + \pi_i$ where $\pi_i \geq 0$. Due to state inheritance, all clients would also see the operation taking effect when their respective simulation times reach $t + \pi_i$. When the simulation time of a client c_j reaches $t + \pi_i$, the simulation time at c_i is $t + \pi_i + \delta_{c_i} - \delta_{c_j}$. Thus, the interaction time for c_j to see the effect of c_i's operation is $t + \pi_i + \delta_{c_i} - \delta_{c_j} - t = \pi_i + \delta_{c_i} - \delta_{c_j}$. As a result, the average interaction time between all pairs of clients[1] is given by

$$\frac{1}{|C|^2} \sum_{i=1}^{|C|} \sum_{j=1}^{|C|} \{\pi_i + \delta_{c_i} - \delta_{c_j}\} = \frac{1}{|C|} \sum_{i=1}^{|C|} \pi_i,$$

which indicates how long it takes on average for the effect of an operation to be presented to a client.

To make operation execution and state updates feasible, the π_i for each client c_i must satisfy the following constraints:

(i) When client c_i issues an operation at its simulation time t, all servers are able to receive the operation before their respective simulation times reach $t + \pi_i$.
(ii) All clients are able to receive the resultant state update of executing the operation before their respective simulation times reach $t + \pi_i$.

Note that each client c_i's operation is first sent to its assigned server $s_A(c_i)$, and then delivered to all the other servers. Thus, based on constraint (i), for each client $c_i \in C$

1. We also include here the interaction time from a client to itself, which represents the time for the client to see the effect of its own operation.

and each server $s \in S$, we have

$$t + d(c_i, s_A(c_i)) + d(s_A(c_i), s) + \delta_s - \delta_{c_i} \leq t + \pi_i,$$

which can be rewritten as

$$\forall c_i \in C, s \in S,\ d(c_i, s_A(c_i)) + d(s_A(c_i), s) + \delta_s - \delta_{c_i} \leq \pi_i. \quad (1)$$

The state update, on the other hand, is delivered to each client after its assigned server executes the operation. Thus, based on constraint (ii), for each client $c_i \in C$, we have

$$t + \pi_i + d(s_A(c_i), c_i) + \delta_{c_i} - \delta_{s_A(c_i)} \leq t + \pi_i,$$

which can be rewritten as

$$\forall c_i \in C, \quad d(s_A(c_i), c_i) + \delta_{c_i} - \delta_{s_A(c_i)} \leq 0. \quad (2)$$

Combining (1) and (2), we have

$$\forall c_i \in C, s \in S,\ 2 \cdot d(c_i, s_A(c_i)) + d(s_A(c_i), s) + \delta_s - \delta_{s_A(c_i)} \leq \pi_i,$$

which are equivalent to

$$\forall c_i \in C,\ 2 \cdot d(c_i, s_A(c_i)) + \max_{s \in S} \left\{ d(s_A(c_i), s) + \delta_s \right\} - \delta_{s_A(c_i)} \leq \pi_i.$$

Therefore,

$$\begin{aligned} \sum_{i=1}^{|C|} \pi_i \geq\ & 2 \cdot \sum_{i=1}^{|C|} d(c_i, s_A(c_i)) \\ & + \sum_{i=1}^{|C|} \max_{s \in S} \left\{ d(s_A(c_i), s) + \delta_s \right\} - \sum_{i=1}^{|C|} \delta_{s_A(c_i)}. \end{aligned} \quad (3)$$

It is easy to see that the right side of the above inequality is a function of the client assignment A and the simulation time offsets of the servers which we shall denote by $\Delta = \{\delta_s \mid s \in S\}$. To facilitate presentation, we define

$$\begin{aligned} D(A, \Delta) =\ & 2 \cdot \sum_{i=1}^{|C|} d(c_i, s_A(c_i)) \\ & + \sum_{i=1}^{|C|} \max_{s \in S} \left\{ d(s_A(c_i), s) + \delta_s \right\} - \sum_{i=1}^{|C|} \delta_{s_A(c_i)}. \end{aligned} \quad (4)$$

It follows from (3) that the average interaction time $\frac{1}{|C|} \sum_{i=1}^{|C|} \pi_i$ is no less than $\frac{1}{|C|} D(A, \Delta)$. Given A and Δ, the equality of (3) can be satisfied by setting the simulation time offset δ_{c_i} of each client c_i at $\delta_{c_i} = \delta_{s_A(c_i)} - d(c_i, s_A(c_i))$ to fulfill constraint (2) and setting $\pi_i = 2 \cdot d(c_i, s_A(c_i)) + \max_{s \in S} \left\{ d(s_A(c_i), s) + \delta_s \right\} - \delta_{s_A(c_i)}$ to fulfill constraint (1). Therefore, the lowest achievable average interaction time is exactly $\frac{1}{|C|} D(A, \Delta)$. Since the factor $\frac{1}{|C|}$ is fixed given the set of clients, for simplicity, we shall omit it in studying interactivity optimization and define the Minimum Interaction Time (MIT) problem as follows:

Definition 1. Given a set of servers S and a set of clients C in a network, and the distance $d(u, v)$ between each pair of nodes $u, v \in C \cup S$, the MIT problem is to find a client assignment A and the simulation time offsets of servers Δ that minimize the average interaction time, i.e., to find

$$\min_{A, \Delta} D(A, \Delta).$$

To keep the above problem definition general, we do not impose any constraints such as load balancing on the client assignment. This is in sync with the current trend towards cloud-based application hosting. In this case, each server s in our model can be viewed as a data center available to host the DIA. Due to elastic computing resources offered by clouds, there is practically no limit on the number of clients that can connect to a data center. On the other hand, if the service provider hosts the application on its own server infrastructure, the servers may have capacity constraints. We shall discuss how to deal with server capacity limitations in Section 5.5.

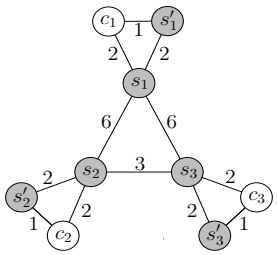


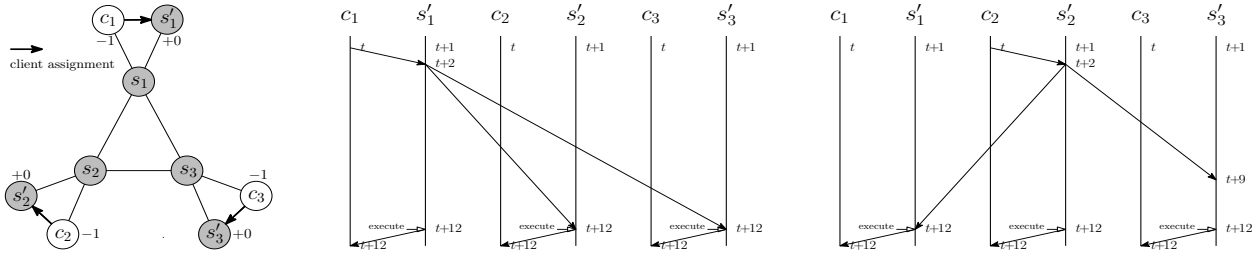
Fig. 1. An example network in a DIA.

We present an example to illustrate how A and Δ affect the average interaction time. In the network shown in Figure 1, there are three clients c_1, c_2, c_3 and six servers $s_1, s_2, s_3, s'_1, s'_2, s'_3$. A natural configuration is to assign each client c_i to server s'_i (i.e., the nearest server), and synchronize the simulation times of the assigned servers of all the clients, as shown in Figure 2a, where each client and server is marked with its simulation time offset.[2] Note that the simulation time of each client must lag behind the simulation time of its server due to the network latency of delivering state updates. Suppose that client c_1 issues an operation at simulation time t. As shown in Figure 2b, the operation first reaches server s'_1 at simulation time $t+2$, and is then delivered to the other two servers at simulation time $t + 12$. Thus, the operation can be executed by all the three servers at the same simulation time $t+12$ at the earliest, and finally, all the clients receive and present the resultant state updates at simulation time $t + 12$. Therefore, the interaction time from client c_1 to all the clients is 12. Figure 2c shows that the interaction time from client c_2 (or c_3) to all the clients is also 12. Consequently, $D(A, \Delta) = 36$ under this natural configuration.

We can improve the interactivity by tuning the simulation time offsets Δ. Figure 3a shows a simulation time setting that reduces $D(A, \Delta)$ to 33. Alternatively, we can also improve the interactivity by tuning the client assignment A. Figure 3b shows a client assignment that leads to $D(A, \Delta) = 30$. The optimal solution to the MIT problem is to tune both A and Δ together as shown in Figure 3c, which gives the best achievable interaction time of $D(A, \Delta) = 27$.

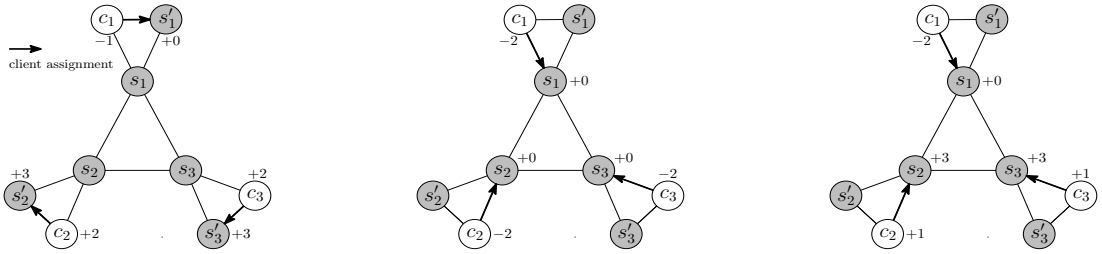
As seen from Definition 1, the MIT problem is a combinatorial optimization problem with two sets of variables: the client assignment A and the simulation time offsets Δ of servers. When the client assignment is fixed, finding the sim-

2. The simulation times of the servers not assigned any client are not really restricted by the consistency constraint. They can be set to lag behind the reference clock by an arbitrarily large amount and do not affect the interaction time between clients. Thus, they are not marked in the figure.



(a) Natural client assignment and simulation time setting (b) Interaction time from client c_1 (c) Interaction time from client c_2 (or c_3)

Fig. 2. A natural configuration.



(a) Tuning simulation times (b) Tuning client assignment (c) Optimal configuration

Fig. 3. Improved configurations.

ulation time offsets of servers that minimize the interaction time, i.e., finding $\min_{\Delta} D(A, \Delta)$, is a linear optimization problem. Next, we study the MIT problem by fixing the client assignment first.

3 MIT Problem with Fixed Client Assignment

To find $\min_{\Delta} D(A, \Delta)$, we first consider a more generalized min-max problem $\mathcal{P}$.

Definition 2. Given a $n \times n$ matrix $\mathbf{Q} = \left(q_{i,j}\right)_{n \times n}$, problem $\mathcal{P}$ is to find the values of σ_i $(i = 1, 2, \cdots, n)$ that minimize

$$F(\sigma_1, \sigma_2, \cdots, \sigma_n) = \sum_{i=1}^{n} \max_j \{q_{i,j} + \sigma_j\} - \sum_{i=1}^{n} \sigma_i.$$

In the following, we show that the above problem $\mathcal{P}$ can be reduced to a weighted bipartite matching problem.

Given an undirected bipartite graph with vertex partition (X, Y), a *matching* is a subset of edges M such that for all vertices $v \in X \cup Y$, at most one edge of M is incident on v. A vertex v is said to be matched by matching M if some edge in M is incident on v. Otherwise, v is unmatched. We refer to a matching in which every vertex is matched as a *perfect matching* [4]. Perfect matchings only exist in bipartite graphs with equal size partitions $|X| = |Y|$. Hall's Theorem [27] gives the condition for a perfect matching to exist in a bipartite graph.

Theorem 1 (Hall's Theorem). A bipartite graph with vertex partition (X, Y) has a matching of size $|X|$ if and only if every subset of vertices $S \subseteq X$ is connected to at least $|S|$ vertices in Y.

Suppose that each edge in a bipartite graph is associated with a non-negative weight. A *maximum-weight matching* is a matching of maximum total edge weight. Given a $n \times n$ matrix $\mathbf{Q} = \left(q_{i,j}\right)_{n \times n}$, we can construct a (X, Y)-bipartite graph where $|X| = |Y| = n$ and each edge in the graph is mapped onto an element located at the intersection of the corresponding row and column of the matrix. Since the constructed graph is a complete bipartite graph with two equal size partitions, a maximum-weight matching in the graph must be a perfect matching. Next, we show that the minimum value of F in problem $\mathcal{P}$ (Definition 2) is given by the weight of a maximum-weight matching in the (X, Y)-bipartite graph constructed above.

Definition 3. Given a matrix $\mathbf{Q}$, define $\mathbb{M}(\mathbf{Q})$ as the weight of a maximum-weight matching in the bipartite graph constructed from $\mathbf{Q}$.

Proposition 1. The minimum value of $F(\sigma_1, \sigma_2, \cdots, \sigma_n)$ in problem $\mathcal{P}$ is equal to $\mathbb{M}(\mathbf{Q})$.

Proof: Let σ_i^* $(i = 1, 2, \cdots, n)$ be the optimal solution to problem $\mathcal{P}$. Then, the minimum F value is given by $F(\sigma_1^*, \sigma_2^*, \cdots, \sigma_n^*) = \sum_{i=1}^{n} \max_j \{q_{i,j} + \sigma_j^*\} - \sum_{i=1}^{n} \sigma_i^*$. Define a new matrix $\mathbf{Q}^* = \left(q_{i,j}^*\right)_{n \times n}$ where $q_{i,j}^* = q_{i,j} + \sigma_j^* - \sigma_i^*$. In each row i of $\mathbf{Q}^*$, we mark the largest element(s) in the row, i.e., all the elements that are equal to $\max_j \{q_{i,j} + \sigma_j^* - \sigma_i^*\}$.

We show that for each $x = 1, 2, \cdots, n$, the marked elements in any x rows of $\mathbf{Q}^*$ must be distributed in at least x columns of $\mathbf{Q}^*$. Assume on the contrary that the marked elements in a set of x rows with indexes $I = \{i_1, i_2, \cdots, i_x\}$ are distributed in a set of y columns with indexes $J = \{j_1, j_2, \cdots, j_y\}$ where $y < x$. Now, suppose that we reduce $\sigma_{j_1}^*, \sigma_{j_2}^*, \cdots, \sigma_{j_y}^*$ by a small amount

$\varepsilon > 0$. Then, for the rows with indexes in $I \cap J$, the elements in columns $j_1, j_2, \cdots, j_y$ (which include all the marked elements in these rows) remain unchanged whereas the elements in the other columns increase by ε. For the rows with indexes in $I - J$, the elements in columns $j_1, j_2, \cdots, j_y$ (which include all the marked elements in these rows) reduce by ε while the elements in the other columns do not change. Therefore, for all the rows with indexes in I, the marked elements remain to be the largest of their respective rows as long as ε is small enough (for example, setting ε to be less than the minimum difference between two elements of different values in $\mathbf{Q}^*$). Thus, by using such small ε, we construct a new solution:

$$\sigma'_i = \begin{cases} \sigma^*_i - \varepsilon & \text{if } i \in J, \\ \sigma^*_i & \text{if } i \notin J. \end{cases}$$

The F value for the new solution is given by

$$\begin{aligned} &F(\sigma'_1, \sigma'_2, \cdots, \sigma'_n) = \sum_{i=1}^{n} \max_j \{q_{i,j} + \sigma'_j\} - \sum_{i=1}^{n} \sigma'_i \\ &= \sum_{i=1}^{n} \max_j \{q_{i,j} + \sigma'_j\} - \sum_{i=1}^{n} \sigma^*_i + |J| \cdot \varepsilon \\ &= \sum_{i \notin I} \max_j \{q_{i,j} + \sigma'_j - \sigma^*_i\} + \sum_{i \in I} \max_j \{q_{i,j} + \sigma'_j - \sigma^*_i\} + |J| \cdot \varepsilon. \end{aligned}$$

Since $\sigma'_j \leq \sigma^*_j$ for any j, it is obvious that $\sum_{i \notin I} \max_j \{q_{i,j} + \sigma'_j - \sigma^*_i\} \leq \sum_{i \notin I} \max_j \{q_{i,j} + \sigma^*_j - \sigma^*_i\}$. As the largest elements in each row of I are still the marked elements, which are distributed in columns J, we have

$$\begin{aligned} &\sum_{i \in I} \max_j \{q_{i,j} + \sigma'_j - \sigma^*_i\} = \sum_{i \in I} \max_{j \in J} \{q_{i,j} + \sigma'_j - \sigma^*_i\} \\ &= \sum_{i \in I} \max_{j \in J} \{q_{i,j} + \sigma^*_j - \varepsilon - \sigma^*_i\} = \sum_{i \in I} \Big(\max_{j \in J} \{q_{i,j} + \sigma^*_j - \sigma^*_i\} - \varepsilon \Big) \\ &= \sum_{i \in I} \max_j \{q_{i,j} + \sigma^*_j - \sigma^*_i\} - |I| \cdot \varepsilon. \end{aligned}$$

Therefore, it follows that

$$\begin{aligned} &F(\sigma'_1, \sigma'_2, \cdots, \sigma'_n) \\ &\leq \sum_{i \notin I} \max_j \{q_{i,j} + \sigma^*_j - \sigma^*_i\} + \sum_{i \in I} \max_j \{q_{i,j} + \sigma^*_j - \sigma^*_i\} \\ &\quad + |J| \cdot \varepsilon - |I| \cdot \varepsilon \\ &= \sum_{i=1}^{n} \max_j \{q_{i,j} + \sigma^*_j\} - \sum_{i=1}^{n} \sigma^*_i + |J| \cdot \varepsilon - |I| \cdot \varepsilon \\ &< \sum_{i=1}^{n} \max_j \{q_{i,j} + \sigma^*_j\} - \sum_{i=1}^{n} \sigma^*_i = F(\sigma^*_1, \sigma^*_2, \cdots, \sigma^*_n), \end{aligned}$$

which contradicts to that $F(\sigma^*_1, \sigma^*_2, \cdots, \sigma^*_n)$ is minimum. Thus, for each $x = 1, 2, \cdots, n$, the marked elements in any x rows of $\mathbf{Q}^*$ must be distributed in at least x columns.

Now consider the bipartite graph constructed from matrix $\mathbf{Q}^*$. According to *Hall's Theorem*, we can find a perfect matching in the bipartite graph consisting of only the edges corresponding to the marked elements of $\mathbf{Q}^*$. Suppose the edge weights in this perfect matching are given by $q^*_{i,\phi(i)}$ $(i = 1, 2, \cdots, n)$ where $\phi(1), \phi(2), \cdots, \phi(n)$ is a permutation of $1, 2, \cdots, n$. Since $q^*_{i,\phi(i)}$ is a marked element, $q^*_{i,\phi(i)} = \max_j \{q_{i,j} + \sigma^*_j - \sigma^*_i\}$. Thus, $F(\sigma^*_1, \sigma^*_2, \cdots, \sigma^*_n) = \sum_{i=1}^{n} q^*_{i,\phi(i)}$. By Definition 3, $\sum_{i=1}^{n} q^*_{i,\phi(i)} \leq \mathbb{M}(\mathbf{Q}^*)$. Therefore, $F(\sigma^*_1, \sigma^*_2, \cdots, \sigma^*_n) \leq \mathbb{M}(\mathbf{Q}^*)$. On the other hand, for any perfect matching $q^*_{i,\psi(i)}$ $(i = 1, 2, \cdots, n)$, we have

$$\begin{aligned} &\sum_{i=1}^{n} q^*_{i,\psi(i)} \leq \sum_{i=1}^{n} \max_j q^*_{i,j} = \sum_{i=1}^{n} \max_j \{q_{i,j} + \sigma^*_j\} - \sum_{i=1}^{n} \sigma^*_i \\ &= F(\sigma^*_1, \sigma^*_2, \cdots, \sigma^*_n). \end{aligned}$$

Since the bipartite graph constructed from $\mathbf{Q}^*$ is a complete bipartite graph with two equal size partitions, a maximum-weight matching in the graph must be a perfect matching. Thus, it follows that $F(\sigma^*_1, \sigma^*_2, \cdots, \sigma^*_n) \geq \mathbb{M}(\mathbf{Q}^*)$. Hence, $F(\sigma^*_1, \sigma^*_2, \cdots, \sigma^*_n) = \mathbb{M}(\mathbf{Q}^*)$.

Note that for any perfect matching $q^*_{i,\psi(i)}$,

$$\begin{aligned} &\sum_{i=1}^{n} q^*_{i,\psi(i)} = \sum_{i=1}^{n} \{q_{i,\psi(i)} + \sigma^*_{\psi(i)} - \sigma^*_i\} \\ &= \sum_{i=1}^{n} q_{i,\psi(i)} + \sum_{i=1}^{n} \sigma^*_{\psi(i)} - \sum_{i=1}^{n} \sigma^*_i = \sum_{i=1}^{n} q_{i,\psi(i)}. \end{aligned}$$

Therefore, $F(\sigma^*_1, \sigma^*_2, \cdots, \sigma^*_n) = \mathbb{M}(\mathbf{Q}^*) = \mathbb{M}(\mathbf{Q})$.

Hence, the proposition is proven. □

The following analysis establishes the relationship between $\min_\Delta D(A, \Delta)$ and the minimum F value.

Proposition 2. In problem $\mathcal{P}$, if columns j_1 and j_2 of $\mathbf{Q}$ are identical, then any optimal solution $\sigma^*_1, \sigma^*_2, \cdots, \sigma^*_n$ must satisfy $\sigma^*_{j_1} = \sigma^*_{j_2}$.

Proof: Since columns j_1 and j_2 are identical, $q_{i,j_1} = q_{i,j_2}$ for each row i. Suppose that $\sigma^*_{j_1} \neq \sigma^*_{j_2}$ in the optimal solution. Without loss of generality, we assume $\sigma^*_{j_1} < \sigma^*_{j_2}$. Then, $\max_{j \in \{j_1, j_2\}} \{q_{i,j} + \sigma^*_j\} = q_{i,j_2} + \sigma^*_{j_2}$ for each $1 \leq i \leq n$. Based on the optimal solution, we construct a new solution $\sigma'_j, \sigma'_2, \cdots, \sigma'_n$ by letting $\sigma'_{j_1} = \sigma^*_{j_2}$ and keeping the other variables unchanged, i.e., $\forall j \neq j_1, \sigma'_j = \sigma^*_j$. Then, the F value for the new solution is

$$\begin{aligned} &F(\sigma'_1, \sigma'_2, \cdots, \sigma'_n) = \sum_{i=1}^{n} \max_j \{q_{i,j} + \sigma'_j\} - \sum_{i=1}^{n} \sigma'_i \\ &= \sum_{i=1}^{n} \max \Big\{ \max_{j \neq j_1, j_2} \{q_{i,j} + \sigma'_j\}, \max_{j \in \{j_1, j_2\}} \{q_{i,j} + \sigma'_j\} \Big\} - \sigma'_{j_1} - \sum_{i \neq j_1} \sigma'_i \\ &= \sum_{i=1}^{n} \max \Big\{ \max_{j \neq j_1, j_2} \{q_{i,j} + \sigma^*_j\}, q_{i,j_2} + \sigma^*_{j_2} \Big\} - \sigma^*_{j_2} - \sum_{i \neq j_1} \sigma^*_i \\ &= \sum_{i=1}^{n} \max \Big\{ \max_{j \neq j_1, j_2} \{q_{i,j} + \sigma^*_j\}, \max_{j \in \{j_1, j_2\}} \{q_{i,j} + \sigma^*_j\} \Big\} \\ &\quad - \sigma^*_{j_2} + \sigma^*_{j_1} - \sum_{i=1}^{n} \sigma^*_i \\ &< \sum_{i=1}^{n} \max_j \{q_{i,j} + \sigma^*_j\} - \sum_{i=1}^{n} \sigma^*_i = F(\sigma^*_1, \sigma^*_2, \cdots, \sigma^*_n). \end{aligned}$$

This contradicts to the optimality of $\sigma^*_1, \sigma^*_2, \cdots, \sigma^*_n$.

Hence, the proposition is proven. □

Corollary 1. Let $\mathbf{Q}_A$ be a $|C| \times |C|$ matrix of $d(s_A(c_i), s_A(c_j))$ $(i, j = 1, 2, \cdots, |C|)$. Then,

$$\min_\Delta D(A, \Delta) = 2 \cdot \sum_{i=1}^{|C|} d(c_i, s_A(c_i)) + \mathbb{M}(\mathbf{Q}_A). \quad (5)$$

Proof: According to Proposition 2, to solve problem $\mathcal{P}$ for matrix $\mathbf{Q}_A$, we only need to consider the solutions that satisfy $\sigma_i = \sigma_j$ for any pair of clients c_i, c_j where $s_A(c_i) = s_A(c_j)$. In this case, for each server s, all the σ_is that satisfy $s_A(c_i) = s$ must have the same value. Thus, we can replace them by a common variable δ_s and obtain

$$\begin{aligned} & F(\sigma_1, \sigma_2, \cdots, \sigma_{|C|}) \\ = & \sum_{i=1}^{|C|} \max_j \{d(s_A(c_i), s_A(c_j)) + \sigma_j\} - \sum_{i=1}^{|C|} \sigma_i \\ = & \sum_{i=1}^{|C|} \max_{s \in S_A} \left\{d(s_A(c_i), s) + \delta_s\right\} - \sum_{i=1}^{|C|} \delta_{s_A(c_i)}, \end{aligned} \quad (6)$$

where S_A is the set of servers that are assigned clients under the client assignment A. Note that the simulation times of the servers that are not assigned clients are not really restricted by constraint (2). Their simulation times can be set to lag behind those of the servers in S_A by an arbitrarily large amount to fulfill constraint (1), i.e., setting $\delta_s \ll 0$ for each $s \in S - S_A$. Then, for any client c_i, $\max_{s \in S} \left\{d(s_A(c_i), s) + \delta_s\right\} = \max_{s \in S_A} \left\{d(s_A(c_i), s) + \delta_s\right\}$. Therefore, $F(\sigma_1, \sigma_2, \cdots, \sigma_{|C|})$ can be rewritten as

$$\begin{aligned} & F(\sigma_1, \sigma_2, \cdots, \sigma_{|C|}) \\ & = \sum_{i=1}^{|C|} \max_{s \in S} \left\{d(s_A(c_i), s) + \delta_s\right\} - \sum_{i=1}^{|C|} \delta_{s_A(c_i)}. \end{aligned}$$

This implies that $F(\sigma_1, \sigma_2, \cdots, \sigma_n)$ has exactly the same format as the second and third terms of $D(A, \Delta)$ in (4). Note that given a client assignment A, the first term of $D(A, \Delta)$ in (4) is fixed. Therefore,

$$\begin{aligned} \min_{\Delta} D(A, \Delta) = & 2 \cdot \sum_{i=1}^{|C|} d(c_i, s_A(c_i)) \\ & + \min_{\sigma_1, \sigma_2, \cdots, \sigma_{|C|}} \left\{F(\sigma_1, \sigma_2, \cdots, \sigma_{|C|})\right\} \\ = & 2 \cdot \sum_{i=1}^{|C|} d(c_i, s_A(c_i)) + \mathbb{M}(\mathbf{Q}_A). \end{aligned}$$

Hence, the corollary is proven. □

3.1 Hungarian Algorithm

Given a client assignment A, we construct a bipartite graph G with vertex partitions $X = \{x_1, x_2, \cdots, x_{|C|}\}$ and $Y = \{y_1, y_2, \cdots, y_{|C|}\}$, and assign weight $w(x_i, y_j) = d(s_A(c_i), s_A(c_j))$ to each edge (x_i, y_j). Then, $\mathbb{M}(\mathbf{Q}_A)$ can be calculated in polynomial time using the Hungarian algorithm for weighted bipartite matching [15]. The pseudo code of the Hungarian algorithm is presented in Algorithm 1, where the equality subgraph G_e is defined as the spanning subgraph of G that includes all vertices of G and those edges (x_i, y_j) whose weights satisfy $w(x_i, y_j) = l(x_i) + l(y_j)$.

The complexity of the original Hungarian algorithm is $O(|C|^3)$. In our case, for each vertex in X, there are only $|S|$ different weights for all the edges incident on it, where $|S|$ is the number of servers. Specifically, if $s_A(c_{j_1}) = s_A(c_{j_2})$, then $w(x_i, y_{j_1}) = w(x_i, y_{j_2})$. So, vertices y_{j_1} and y_{j_2} should be either both in T or both in $Y - T$, which implies that $l(y_{j_1}) = l(y_{j_2})$ holds throughout algorithm execution. Thus, for each $s_k \in S$, we can define $z_k = \{y_j \mid s_A(c_j) = s_k\}$. By maintaining the $slack[\cdot]$ values for each z_k instead of for each y_j, the complexity of the Hungarian algorithm for our special case can be improved to $O(|C|^2|S|)$.

Algorithm 1: The Hungarian Algorithm

```
for each node x_i ∈ X do
    l(x_i) ← max_{y_j ∈ Y} {w(x_i, y_j)};
for each node y_j ∈ Y do
    l(y_j) ← 0;
initialize an empty matching;
construct equality subgraph G_e;
for each exposed node u ∈ X do
    R ← {u}, T ← ∅;
    for each node y_j ∈ Y − T do
        slack[y_j] = min_{x_i ∈ R} {l(x_i) + l(y_j) − w(x_i, y_j)};
    while T ≠ Y do
        find N(R): the set of vertices adjacent to R in
        equality subgraph G_e;
        while T ⊂ N(R) do
            for each node v ∈ N(R) − T do
                if v is exposed then
                    find the augmenting path from u to v
                    and update the matching;
                    break;
                else
                    R ← R ∪ {the matched node of v};
                    T ← T ∪ {v};
                    for each node y_j ∈ Y − T do
                        update slack[y_j];
            update N(R): the set of vertices adjacent to R
            in equality subgraph G_e;
        ε ← min_{y_j ∈ Y−T} slack[y_j];
        for each node x_i ∈ R do
            l(x_i) ← l(x_i) − ε;
        for each node y_j ∈ T do
            l(y_j) ← l(y_j) + ε;
        update equality graph G_e;
```

4 Hardness of MIT Problem

Based on the analysis in the previous section, we now analyze the hardness of the MIT problem. According to (5), the minimum interaction time can be computed by

$$\min_{A, \Delta} D(A, \Delta) = \min_{A} \left\{2 \cdot \sum_{i=1}^{|C|} d(c_i, s_A(c_i)) + \mathbb{M}(\mathbf{Q}_A)\right\}.$$

Lemma 1. Given a $n \times n$ matrix $\mathbf{Q}$, if all diagonal elements of $\mathbf{Q}$ are 0, one non-diagonal element is $1 + x$ ($x \geq 0$), and all the other non-diagonal elements are 1, then $\mathbb{M}(\mathbf{Q}) = n + x$.

Lemma 2. Given a $n \times n$ matrix $\mathbf{Q}$, if there exist y ($y \leq n$) elements that are located in y different rows and y different columns of $\mathbf{Q}$ and have a sum of x, then $\mathbb{M}(\mathbf{Q}) \geq x$.

Proof: The above two lemmas are self-evident. □

Theorem 2. The MIT problem is NP-hard.

Proof: We show the NP-hardness of the MIT problem by a polynomial reduction from the minimum set cover problem which is known to be NP-hard [11]. The decision version of the minimum set cover problem is defined as follows: given a finite set P and a collection $\mathbb{U}$ of its subsets, and a positive integer $k \leq |\mathbb{U}|$, find out whether $\mathbb{U}$ contains a set cover for P of size at most k, i.e., whether there exists a subcollection $\mathbb{U}' \subseteq \mathbb{U}$ with $|\mathbb{U}'| \leq k$ such that $\bigcup_{U \in \mathbb{U}'} U = P$.

Let R be an instance of the minimum set cover problem. Suppose that set P contains n elements $p_1, p_2, \cdots, p_n$ and collection $\mathbb{U}$ contains m subsets $U_1, U_2, \cdots, U_m$. We first construct a network with n clients $c_1, c_2, \cdots, c_n$ and k groups of servers. Each client c_i corresponds to one element p_i in set P. Each server group consists of m subgroups, each of which corresponds to one subset in $\mathbb{U}$. The number of servers in a subgroup is equal to the cardinality of its corresponding subset, and each server in the subgroup corresponds to one element in the subset. In the constructed network, we say that a server and a client are *associated* if they correspond to the same element in P. Clearly, each server has only one client associated with it. The distance between a server and a client is 1 if they are associated and is 2 otherwise. The inter-server distance is 1 if the two servers are in different groups or in the same subgroup, and is 2 if they are in different subgroups of the same group. An instance T of the MIT problem in its decision version is then defined on this network. In the following, we show that, $\mathbb{U}$ contains a set cover $\mathbb{U}'$ of size at most k for instance R if and only if there exist a client assignment A and simulation time offsets Δ satisfying $D(A, \Delta) \leq 3n$ for instance T of the MIT problem.

Sufficiency: Suppose there exists a set cover $\mathbb{U}' = \{U_{j_1}, U_{j_2}, \cdots, U_{j_l}\}$ of size not exceeding k, i.e., $l \leq k$. Then, we construct a client assignment A that assigns all clients to the servers in the following subgroups: the j_1-th subgroup of server group 1, the j_2-th subgroup of server group 2, $\ldots$, the j_l-th subgroup of server group l. Since $\mathbb{U}'$ is a set cover, each client would have at least one server associated with it in the above subgroups. By assigning each client to a server associated with it, the distance between any client and its assigned server is 1. Since all the aforementioned subgroups belong to different server groups, the distance between any two servers in these subgroups is 1. Thus, in the matrix $\mathbf{Q}_A = \Big(d(s_A(c_i), s_A(c_j))\Big)_{n \times n}$, all non-diagonal elements must be 1 and all diagonal elements are 0, which, according to Lemma 1, leads to $\mathbb{M}(\mathbf{Q}_A) = n$. Let Δ^* be the optimal simulation time offsets of the servers under the constructed client assignment A. It follows from Corollary 1 that

$$D(A, \Delta^*) = 2 \cdot \sum_{i=1}^{n} 1 + \mathbb{M}(\mathbf{Q}_A) = 3n.$$

Necessity: Suppose that there exist a client assignment A and simulation time offsets Δ_A satisfying $D(A, \Delta_A) \leq 3n$. Then, we have $\min_{\Delta} D(A, \Delta) \leq D(A, \Delta_A) \leq 3n$. We first show that in the client assignment A, each client must be assigned to a server associated with it. Assume on the contrary that there are a set X of clients that are not assigned to servers associated with them. As a result, the distances from these clients to their assigned servers are 2. The remaining $n - |X|$ clients are assigned to servers associated with them. Since each server is associated with only one client, these $n - |X|$ clients must be assigned to $n - |X|$ different servers. Therefore, the distances between their assigned servers are at least 1. Denote these $n - |X|$ clients by $c_{r_1}, c_{r_2}, \cdots, c_{r_{n-|X|}}$. Consider the following $n - |X|$ elements in the matrix $\mathbf{Q}_A$: $d(s_A(c_{r_1}), s_A(c_{r_2}))$, $d(s_A(c_{r_2}), s_A(c_{r_3}))$, $\cdots$, $d(s_A(c_{r_{n-|X|-1}}), s_A(c_{r_{n-|X|}}))$, $d(s_A(c_{r_{n-|X|}}), s_A(c_{r_1}))$. These elements are located in $n - |X|$ different rows and $n - |X|$ different columns of $\mathbf{Q}_A$, and their values are at least 1. According to Lemma 2, $\mathbb{M}(\mathbf{Q}_A) \geq n - |X|$. Thus, it follows from Corollary 1 that

$$\begin{aligned}
\min_{\Delta} D(A, \Delta) &= 2 \cdot \sum_{c \notin X} d(c, s_A(c)) + 2 \cdot \sum_{c \in X} d(c, s_A(c)) + \mathbb{M}(\mathbf{Q}_A) \\
&\geq 2 \cdot \sum_{c \notin X} 1 + 2 \cdot \sum_{c \in X} 2 + n - |X| \\
&= 2 \cdot \big(n - |X|\big) + 4 \cdot |X| + n - |X| = 3n + |X| \geq 3n + 1,
\end{aligned}$$

which contradicts to that $\min_{\Delta} D(A, \Delta) \leq 3n$.

Next, we show that in the client assignment A, at most one subgroup in each server group is assigned clients. Otherwise, if two subgroups of the same server group are both assigned clients, at least one non-diagonal element in the matrix $\mathbf{Q}_A$ would be 2. Since all diagonal elements in $\mathbf{Q}_A$ are 0 and all non-diagonal elements are no less than 1, according to Lemma 1, $\mathbb{M}(\mathbf{Q}_A) \geq n + 1$. Note that the distance from each client to its assigned server is at least 1. Therefore,

$$\begin{aligned}
\min_{\Delta} D(A, \Delta) &= 2 \cdot \sum_{i=1}^{n} d(c_i, s_A(c_i)) + \mathbb{M}(\mathbf{Q}_A) \\
&\geq 2 \cdot n \cdot 1 + n + 1 = 3n + 1,
\end{aligned}$$

which again contradicts to that $\min_{\Delta} D(A, \Delta) \leq 3n$.

We now construct a set cover $\mathbb{U}'$ based on the client assignment A by picking a subset from $\mathbb{U}$ if and only if at least one of its corresponding server subgroups is assigned clients in A. Since at most one subgroup in each server group is assigned clients, the number of subgroups that are assigned clients must not exceed the number of server groups, i.e., k. Thus, $|\mathbb{U}'| \leq k$. Moreover, for each element p_i in set P, since client c_i is assigned to a server associated with it, the subset corresponding to its server's subgroup must cover p_i and is picked by $\mathbb{U}'$. It follows that $\mathbb{U}'$ must cover all the elements in P. Therefore, $\mathbb{U}'$ is a set cover of size at most k.

Hence, the theorem is proven. □

5 Approximating MIT Problem

We now study approximations of the MIT problem by fixing the client assignment and/or the simulation time offsets. An intuitive and easy-to-implement strategy for client assignment is to assign each client to its nearest server, i.e., the server with the shortest distance (network latency) to it. This is known as the *nearest-server assignment* and is widely used in many applications [8], [16], [26]. On the other hand, a simple and straightforward setting of simulation times is to synchronize the simulation times of the servers. Denote such simulation time setting by Δ_0 and denote the nearest-server assignment by N.

If N and Δ_0 are employed together, the resultant interaction time $D(N, \Delta_0)$ can be arbitrarily worse than the minimum interaction time $\min_{A,\Delta} D(A, \Delta)$. Figure 5 gives an example network with n clients $c_1, c_2, \cdots, c_n$ and two servers s_1, s_2. In the nearest-server assignment, $c_1, c_2, \cdots, c_{n-1}$ are assigned to s_1, and c_n is assigned to s_2. It can be computed from (4) in Section 2 that $D(N, \Delta_0) = n + 2$. The optimal solution to the MIT problem can either assign each client to its nearest server but set $\delta_{s_1} = 1$ and $\delta_{s_2} = 0$, or assign all clients to server s_1. The minimum interaction time is $\min_{A,\Delta} D(A, \Delta) = 4$. Thus, the ratio between the two results can be made arbitrarily large as n goes towards infinity. In fact, it can be proved that $D(N, \Delta_0) \leq (|C|+1) \cdot \min_{A,\Delta} D(A, \Delta)$ as detailed in Section 5.1.

Interestingly, however, constant approximation factors can be achieved by *either* fixing the client assignment at N *or* fixing the simulation time offsets at Δ_0 as shall be shown in Sections 5.2 and 5.3.

5.1 Approximating MIT Problem by Fixing Client Assignment and Simulation Time Offsets

In this section, we show that for networks with the triangle inequality, the interaction time resulting from fixing the simulation time offsets at Δ_0 under the nearest-server assignment N has a tight approximation factor of $(|C| + 1)$ for the MIT problem, where $|C|$ is the number of clients.

Theorem 3.

$$D(N, \Delta_0) \leq (|C| + 1) \cdot \min_{A,\Delta} D(A, \Delta).$$

Proof: Let $n_i \in S$ be the nearest server of client c_i, and let $S_N = \{n_i | c_i \in C\}$ be the set of servers that are assigned clients in the nearest-server assignment N. Under Δ_0, the simulation times of the servers in S_N are synchronized. That is, for all servers n_i, δ_{n_i}'s have the same value. In this case, according to (6) in the proof to Corollary 1, we have

$$F(\sigma_1, \sigma_2, \cdots, \sigma_{|C|}) = \sum_{i=1}^{|C|} \max_{s \in S_N} \{d(n_i, s)\},$$

which is a constant independent of $\sigma_1, \sigma_2, \cdots, \sigma_{|C|}$. Thus, $D(N, \Delta_0)$ is given by

$$D(N, \Delta_0) = 2 \cdot \sum_{i=1}^{|C|} d(c_i, n_i) + \sum_{i=1}^{|C|} \max_{s \in S_N} \{d(n_i, s)\}. \quad (7)$$

Define $n_{m_i} = \arg\max_{s \in S_N} \{d(n_i, s)\}$. Then, the above expression can be rewritten as

$$D(N, \Delta_0) = 2 \cdot \sum_{i=1}^{|C|} d(c_i, n_i) + \sum_{i=1}^{|C|} d(n_i, n_{m_i}).$$

Let $o_i \in S$ be the assigned server of client c_i in the optimal solution to the MIT problem. By the triangle inequality and the fact that n_i is the nearest server of c_i, it follows that

$$\begin{aligned} D(N, \Delta_0) &\leq 2 \cdot \sum_{i=1}^{|C|} d(c_i, n_i) + \sum_{i=1}^{|C|} \Big(d(n_i, c_i) + d(c_i, o_i) \\ &\quad + d(o_i, o_{m_i}) + d(o_{m_i}, c_{m_i}) + d(c_{m_i}, n_{m_i}) \Big) \\ &\leq 4 \cdot \sum_{i=1}^{|C|} d(c_i, o_i) + 2 \cdot \sum_{i=1}^{|C|} d(c_{m_i}, o_{m_i}) + \sum_{i=1}^{|C|} d(o_i, o_{m_i}). \end{aligned} \quad (8)$$

We consider two different cases. In the first case, there exist two different clients c_j and c_k such that $m_j \neq m_k$. Then, $d(c_{m_j}, o_{m_j}) + d(c_{m_k}, o_{m_k}) \leq \sum_{i=1}^{|C|} d(c_i, o_i)$. In addition, it is straightforward that $d(c_{m_i}, o_{m_i}) \leq \sum_{i=1}^{|C|} d(c_i, o_i)$ for any other client c_i $(i \neq j, k)$. Thus, the second term of (8) satisfies

$$\begin{aligned} &2 \cdot \sum_{i=1}^{|C|} d(c_{m_i}, o_{m_i}) \\ &= 2 \cdot \Big(d(c_{m_j}, o_{m_j}) + d(c_{m_k}, o_{m_k}) + \sum_{i \neq j,k} d(c_{m_i}, o_{m_i}) \Big) \\ &\leq 2 \cdot \Big(\sum_{i=1}^{|C|} d(c_i, o_i) + (|C| - 2) \cdot \sum_{i=1}^{|C|} d(c_i, o_i) \Big) \\ &= (2|C| - 2) \cdot \sum_{i=1}^{|C|} d(c_i, o_i). \end{aligned}$$

For the third term of (8), we have

$$\sum_{i=1}^{|C|} d(o_i, o_{m_i}) \leq \sum_{i=1}^{|C|} \max_{j,k} \{d(o_j, o_k)\} = |C| \cdot \max_{j,k} \{d(o_j, o_k)\}.$$

Let $\mathbf{Q}_O$ be a $|C| \times |C|$ matrix of $d(o_i, o_j)$ $(i, j = 1, 2, \cdots, |C|)$. Since $\mathbf{Q}_O$ is symmetric, in the bipartite graph corresponding to $\mathbf{Q}_O$, we can always construct a perfect matching that contains $\max_{j<k}\{d(o_j, o_k)\}$ and $\max_{j>k}\{d(o_j, o_k)\}$. Note that we have $\max_{j<k}\{d(o_j, o_k)\} = \max_{j>k}\{d(o_j, o_k)\} = \max_{j,k}\{d(o_j, o_k)\}$. Therefore, we obtain $\mathbb{M}(\mathbf{Q}_O) \geq 2 \cdot \max_{j,k}\{d(o_j, o_k)\}$. It follows that

$$\sum_{i=1}^{|C|} d(o_i, o_{m_i}) \leq \frac{|C|}{2} \cdot \mathbb{M}(\mathbf{Q}_O).$$

As a result,

$$\begin{aligned} D(N, \Delta_0) &\leq (2|C| + 2) \cdot \sum_{i=1}^{|C|} d(c_i, o_i) + \frac{|C|}{2} \cdot \mathbb{M}(\mathbf{Q}_O) \\ &\leq (|C| + 1) \cdot \Big(2 \cdot \sum_{i=1}^{|C|} d(c_i, o_i) + \mathbb{M}(\mathbf{Q}_O) \Big) \\ &= (|C| + 1) \cdot \min_{A,\Delta} D(A, \Delta), \end{aligned}$$

where the last step follows from Corollary 1 in Section 3.

In the second case, m_i is the same for all clients c_i. Then, for client c_{m_i}, we have $\max_{s \in S_N} d(n_{m_i}, s) = d(n_{m_i}, n_{m_i}) = 0$. That means for any server $s \in S_N$, $d(n_{m_i}, s) = 0$. So, all clients have the same nearest server. Thus, it follows from

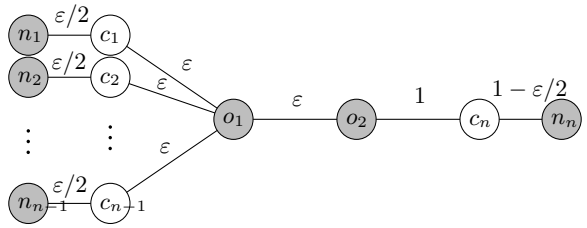


Fig. 4. The approximation factor ($|C|+1$) of $D(N,\Delta_0)$ is tight.

(7) that

$$D(N,\Delta_0) = 2\cdot\sum_{i=1}^{|C|} d(c_i,n_i) \le 2\cdot\sum_{i=1}^{|C|} d(c_i,o_i) \le \min_{A,\Delta} D(A,\Delta).$$

So, $D(N,\Delta_0)$ must be an optimal solution.

Therefore, summarizing the above two cases, the approximation factor is $(|C|+1)$. Hence, the theorem is proven. □

The above approximation factor is tight as shown by the example in Figure 4. In this example, the nearest-server assignment assigns each client c_i to server n_i ($i=1,2,\cdots,n$), which gives $D(N,\Delta_0) = 2\cdot((n-1)\cdot\frac{\varepsilon}{2}+1-\frac{\varepsilon}{2})+n\cdot(2+2\varepsilon) = 2n+2+(3n-2)\varepsilon$. On the other hand, the optimal solution is to assign all clients to server o_2 such that $\min_{A,\Delta} D(A,\Delta) = 2\cdot((n-1)\cdot 2\varepsilon+1) = 2+(4n-4)\varepsilon$. Thus, the ratio between the two results can be made arbitrarily close to $(n+1)$ as ε approaches 0.

5.2 Approximating MIT Problem by Fixing Client Assignment

In Section 3, we have shown that the minimum achievable interaction time under a fixed client assignment can be computed efficiently in polynomial time. For networks with the triangle inequality, it can be shown that the minimum achievable interaction time under the nearest-server assignment is within 3 times of the optimal solution to the MIT problem.

Theorem 4. $\min_{\Delta} D(N,\Delta) \le 3\cdot\min_{A,\Delta} D(A,\Delta)$.

Proof: Let $n_i \in S$ be the nearest server of client c_i. According to Corollary 1 in Section 3, we have

$$\min_{\Delta} D(N,\Delta) = 2\cdot\sum_{i=1}^{|C|} d(c_i,n_i) + \mathbb{M}(\mathbf{Q}_N),$$

where $\mathbf{Q}_N$ is a $|C|\times|C|$ matrix of $d(n_i,n_j)$ ($i,j = 1,2,\cdots,|C|$).

Suppose that the maximum-weight matching in the bipartite graph constructed from $\mathbf{Q}_N$ includes $d(n_i,n_{\phi(i)})$ ($i=1,2,\cdots,|C|$) where $\phi(1),\phi(2),\cdots,\phi(|C|)$ is a permutation of $1,2,\cdots,|C|$. It follows that

$$\min_{\Delta} D(N,\Delta) = 2\cdot\sum_{i=1}^{|C|} d(c_i,n_i) + \sum_{i=1}^{|C|} d(n_i,n_{\phi(i)}). \quad (9)$$

Let $o_i \in S$ be the assigned server of client c_i in the optimal solution to the MIT problem. Then,

$$\min_{A,\Delta} D(A,\Delta) = 2\cdot\sum_{i=1}^{|C|} d(c_i,o_i) + \mathbb{M}(\mathbf{Q}_O),$$

where $\mathbf{Q}_O$ is a $|C|\times|C|$ matrix of $d(o_i,o_j)$ ($i,j = 1,2,\cdots,|C|$).

By the triangle inequality, we have

$$\begin{aligned} d(n_i,n_{\phi(i)}) \le\ & d(n_i,c_i)+d(c_i,o_i)+d(o_i,o_{\phi(i)}) \\ & + d(o_{\phi(i)},c_{\phi(i)})+d(c_{\phi(i)},n_{\phi(i)}). \end{aligned} \quad (10)$$

Since n_i and $n_{\phi(i)}$ are the nearest servers of clients c_i and $c_{\phi(i)}$ respectively, it follows from (9) and (10) that

$$\begin{aligned} \min_{\Delta} D(N,\Delta) &\le \sum_{i=1}^{|C|}\Big(3\cdot d(c_i,n_i)+d(c_i,o_i)+d(o_i,o_{\phi(i)}) \\ &\quad + d(o_{\phi(i)},c_{\phi(i)})+d(c_{\phi(i)},n_{\phi(i)})\Big) \\ &\le \sum_{i=1}^{|C|}\Big(4\cdot d(c_i,o_i)+d(o_i,o_{\phi(i)})+2\cdot d(o_{\phi(i)},c_{\phi(i)})\Big) \\ &= 4\cdot\sum_{i=1}^{|C|} d(c_i,o_i)+\sum_{i=1}^{|C|} d(o_i,o_{\phi(i)})+2\cdot\sum_{i=1}^{|C|} d(o_{\phi(i)},c_{\phi(i)}) \\ &= 6\cdot\sum_{i=1}^{|C|} d(c_i,o_i)+\sum_{i=1}^{|C|} d(o_i,o_{\phi(i)}). \end{aligned}$$

Since $d(n_i,n_{\phi(i)})$ ($i=1,2,\cdots,|C|$) is a perfect matching in the bipartite graph constructed from $\mathbf{Q}_N$, $d(o_i,o_{\phi(i)})$ ($i=1,2,\cdots,|C|$) must also be a perfect matching in the bipartite graph constructed from $\mathbf{Q}_O$. By Definition 3, we have

$$\sum_{i=1}^{|C|} d(o_i,o_{\phi(i)}) \le \mathbb{M}(\mathbf{Q}_O).$$

Therefore, it follows that

$$\begin{aligned} \min_{\Delta} D(N,\Delta) &\le 6\cdot\sum_{i=1}^{|C|} d(c_i,o_i)+\mathbb{M}(\mathbf{Q}_O) \quad (11)\\ &\le\ 3\cdot\Big(2\cdot\sum_{i=1}^{|C|} d(c_i,o_i)+\mathbb{M}(\mathbf{Q}_O)\Big) \le 3\cdot\min_{A,\Delta} D(A,\Delta). \end{aligned}$$

Hence, the theorem is proven. □

The above approximation factor of 3 is tight. Figure 6 gives an example network with two clients c_1,c_2 and three servers s,s_1,s_2. In the nearest-server assignment, c_1 and c_2 are assigned to s_1 and s_2 respectively. Under such client assignment, the minimum interaction time $\min_{\Delta} D(N,\Delta)$ is $12a-8\varepsilon$, which can be achieved as long as the relative offset between the simulation times of s_1 and s_2 is within $4a-2\varepsilon$ (i.e., the network latency between them). On the other hand, the optimal solution to the MIT problem is to assign both clients to server s, so that the interaction time is $4a$, i.e., $\min_{A,\Delta} D(A,\Delta) = 4a$. Thus, the ratio between the two results can be made arbitrarily close to 3 as ε approaches 0.

5.3 Approximating MIT Problem by Fixing Simulation Time Offsets

In this section, we approximate the MIT problem by fixing the simulation time offsets at Δ_0. Let S_A be the set of servers

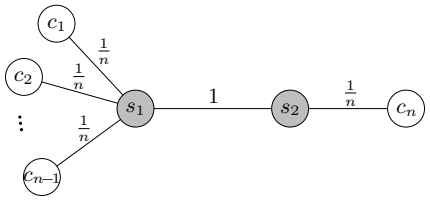


Fig. 5. The approximation factor of $D(N, \Delta_0)$ is not bounded.

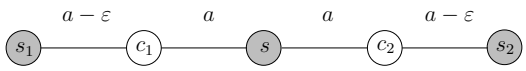


Fig. 6. The approximation factor 3 of $\min_\Delta D(N, \Delta)$ is tight.

that are assigned clients in a client assignment A. Similar to obtaining (7) in Section 5.1, $D(A, \Delta_0)$ is given by

$$D(A, \Delta_0) = 2 \cdot \sum_{i=1}^{|C|} d(c_i, s_A(c_i)) + \sum_{i=1}^{|C|} \max_{s \in S_A} \{d(s_A(c_i), s)\}, \tag{12}$$

In the following, we first show that finding $\min_A D(A, \Delta_0)$ is also an NP-hard problem.

Theorem 5. When the simulation times of the servers are synchronized, finding a client assignment that minimizes the interaction time, i.e., finding $\min_A D(A, \Delta_0)$, is NP-hard.

Proof: Similar to the proof to Theorem 2, we show the NP-hardness by a polynomial reduction from the minimum set cover problem. Given an instance of the minimum set cover problem, we construct the same network as in the proof to Theorem 2 and show that there exists a set cover of size at most k if and only if there exists a client assignment A satisfying $D(A, \Delta_0) \leq 3n$ on the constructed network.

Sufficiency: Suppose that there exists a set cover of size at most k. We construct a client assignment A in the same way as in the proof to Theorem 2. Under this client assignment, the distance between any client and its assigned server is 1, and the distance between any two servers that are assigned clients is also 1. Thus, we have

$$D(A, \Delta_0) = 2 \cdot \sum_{i=1}^{n} 1 + \sum_{i=1}^{n} 1 = 2n + n = 3n.$$

Necessity: In the proof to Theorem 2, we have shown that if there does not exist any set cover of size at most k, then $\forall A$ and Δ, $D(A, \Delta) > 3n$. The latter implies that $\forall A, D(A, \Delta_0) > 3n$. Thus, the necessity is proven. □

The following analysis shows that $\min_A D(A, \Delta_0)$ can approximate the MIT problem with a better factor than $\min_\Delta D(N, \Delta)$ for networks with the triangle inequality.

Lemma 3. For any $n \times n$ matrix $\mathbf{Q} = \left(q_{i,j}\right)_{n \times n}$, it holds that $\mathbb{M}(\mathbf{Q}) \geq \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} q_{i,j}$.

Proof: Consider the following n permutations of $1, 2, \cdots, n$:

$$\begin{aligned} \phi_1 = (1, 2, \cdots, n), \quad & \phi_2 = (2, 3, \cdots, 1), \\ \cdots\cdots, \quad & \phi_n = (n, 1, \cdots, n-1). \end{aligned}$$

For each permutation ϕ_j $(1 \leq j \leq n)$, $q_{i,\phi_j(i)}$ $(i = 1, 2, \cdots, n)$ is a perfect matching in the bipartite graph constructed from $\mathbf{Q}$. By Definition 3, we have $\mathbb{M}(\mathbf{Q}) \geq \sum_{i=1}^{n} q_{i,\phi_j(i)}$. By adding up the inequalities for all js, we have

$$n \cdot \mathbb{M}(\mathbf{Q}) \geq \sum_{j=1}^{n} \sum_{i=1}^{n} q_{i,\phi_j(i)} = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{i,j}.$$

Therefore,

$$\mathbb{M}(\mathbf{Q}) \geq \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} q_{i,j}.$$

Hence, the lemma is proven. □

Theorem 6. $\min_A D(A, \Delta_0) \leq 2 \cdot \min_{A,\Delta} D(A, \Delta)$.

Proof: Let $o_i \in S$ be the assigned server of client c_i in the optimal solution to the MIT problem. Then, we have

$$\min_{A,\Delta} D(A, \Delta) = 2 \cdot \sum_{i=1}^{|C|} d(c_i, o_i) + \mathbb{M}(\mathbf{Q}_O),$$

where $\mathbf{Q}_O$ is a $|C| \times |C|$ matrix of $d(o_i, o_j)$ $(i, j = 1, 2, \cdots, |C|)$.

For each server o_j $(j = 1, 2, \cdots, |C|)$, denote by A_{o_j} the client assignment that assigns all clients to o_j. Then, we have

$$D(A_{o_j}, \Delta_0) = 2 \cdot \sum_{i=1}^{|C|} d(c_i, o_j).$$

By adding up the above inequalities for all o_js, we have

$$\sum_{j=1}^{|C|} D(A_{o_j}, \Delta_0) = 2 \cdot \sum_{j=1}^{|C|} \sum_{i=1}^{|C|} d(c_i, o_j).$$

By the triangle inequality, we have

$$\begin{aligned} &\sum_{j=1}^{|C|} D(A_{o_j}, \Delta_0) \leq 2 \cdot \sum_{j=1}^{|C|} \sum_{i=1}^{|C|} \big(d(c_i, o_i) + d(o_i, o_j)\big) \\ &= 2|C| \cdot \sum_{i=1}^{|C|} d(c_i, o_i) + 2 \cdot \sum_{i=1}^{|C|} \sum_{j=1}^{|C|} d(o_i, o_j). \end{aligned}$$

According to Lemma 3,

$$\sum_{i=1}^{|C|} \sum_{j=1}^{|C|} d(o_i, o_j) \leq |C| \cdot \mathbb{M}(\mathbf{Q}_O).$$

Thus,

$$\begin{aligned} \sum_{j=1}^{|C|} D(A_{o_j}, \Delta_0) &\leq 2|C| \cdot \sum_{i=1}^{|C|} d(c_i, o_i) + 2|C| \cdot \mathbb{M}(\mathbf{Q}_O) \qquad (13) \\ &\leq 2|C| \cdot \Big(2 \cdot \sum_{i=1}^{|C|} d(c_i, o_i) + \mathbb{M}(\mathbf{Q}_O)\Big) = 2|C| \cdot \min_{A,\Delta} D(A, \Delta). \end{aligned}$$

Since for each server o_j $(j = 1, 2, \cdots, |C|)$,

$$\min_A D(A, \Delta_0) \leq D(A_{o_j}, \Delta_0),$$

it follows that

$$\min_A D(A, \Delta_0) \leq \frac{1}{|C|} \sum_{j=1}^{|C|} D(A_{o_j}, \Delta_0) \leq 2 \cdot \min_{A,\Delta} D(A, \Delta). \tag{14}$$

Hence, the theorem is proven. □

5.3.1 A Greedy Heuristic

The proof to Theorem 6 implies that the approximation factor 2 is in fact applicable to any client assignment algorithm that takes into consideration those of assigning all clients to each of the available servers. Thus, although finding the best client assignment under simulation time setting Δ_0 is NP-hard, it is straightforward to design efficient 2-approximation algorithms under Δ_0.

We present a heuristic called *Greedy Assignment* for assigning clients to servers to optimize $D(A, \Delta_0)$. The algorithm maintains a set of active servers S_{act}, and the clients are allowed to be assigned to the active servers only. Algorithm 2 presents the pseudo code of *Greedy Assignment*. The algorithm starts with $S_{act} = \emptyset$, and iteratively adds servers into the set to reduce the average interaction time. In each iteration, the algorithm considers each server s not currently in S_{act}, and computes a new client assignment assuming that s is added into S_{act}. Among all the servers s that can reduce the average interaction time if added to S_{act}, the algorithm chooses the one that produces the minimum average interaction time. This process continues until the average interaction time cannot be further reduced by adding any new server into S_{act}.

Greedy Assignment adopts the following strategy, as described in Algorithm 3, to compute the client assignment given a set of active servers S_{act}. By assuming that each server in S_{act} is assigned at least one client, it follows from (12) that

$$D(A, \Delta_0) = \sum_{i=1}^{|C|} \Big(2 \cdot d(c_i, s_A(c_i)) + \max_{s \in S_{act}} \{ d(s_A(c_i), s) \} \Big).$$

Thus, the optimal assignment under this assumption is to assign each client c to the server in S_{act} that minimizes $2 \cdot d(c, s_A(c)) + \max_{s \in S_{act}} \{ d(s_A(c), s) \}$. Given S_{act}, the algorithm performs such assignment for all clients. Let $S_{act}\prime$ be the set of servers that are actually assigned clients in the aforementioned assignment. If $S_{act}\prime \subset S_{act}$, i.e., some servers in S_{act} are not assigned any client, the algorithm updates S_{act} with S'_{act} and repeats the above process. If $S_{act}\prime = S_{act}$, it stops and returns the current client assignment.

In the client assignment process (Algorithm 3), the complexity of each loop iteration (lines 3 - 9) is $O(|C||S_{act}|)$, and there are at most $O(|S_{act}|)$ iterations since each iteration reduces $|S_{act}|$ by at least 1 before the loop stops. Therefore, the total complexity of the *Greedy Assignment* algorithm is $O(\sum_{S_{act}}(|S| - |S_{act}|)|C||S_{act}|^2) = O(|C||S|^4)$.

Algorithm 2: The Greedy Assignment Algorithm

```
Output: D - The average interaction time
        A - The assignment
1  S_act = ∅;
2  D = ∞;
3  repeat
4      D* = D;
5      foreach s ∉ S_act do
6          S_act = S_act ∪ {s};
7          (D', A') = Assign(S_act);
8          if D' < D* then
9              D* = D';
10             A* = A';
11             s* = s;
12         S_act = S_act \ {s};
13     improve = D − D*;
14     if improve > 0 then
15         S_act = S_act ∪ {s*};
16         D = D*;
17         A = A*;
18 until improve ≤ 0;
```

Algorithm 3: Assign

```
Input  : S_act - The set of active servers
Output: D - The average interaction time
        A - The assignment
1  S'_act = S_act;
2  repeat
3      S_act = S'_act;
4      foreach s ∈ S_act do
5          m[s] = max_{s'∈S_act} {d(s, s')};
6      foreach c ∈ C do
7          s* = argmax_{s'∈S_act} {2 · d(c, s') + m[s']};
8          set s_A(c) = s*;
9      S'_act = {s | ∃c ∈ C, s_A(c) = s};
10 until S'_act == S_act;
11 D = Σ_c {2 · d(c, s_A(c)) + m[s_A(c)]};
12 return (D, A);
```

5.4 A Hybrid Approach

Inspired by a mixed heuristic for the server placement problem of DIAs [32], we further derive from the proofs to Theorems 4 and 6 that the better approximation between fixing client assignment and fixing simulation time offsets has an improved approximation factor of $5/3$.

Theorem 7. $\min\{ \min_{\Delta} D(N, \Delta), \min_{A} D(A, \Delta_0) \} \leq \frac{5}{3} \cdot \min_{A,\Delta} D(A, \Delta)$.

Proof: According to (11) in the proof to Theorem 4,

$$\min_{\Delta} D(N, \Delta) \leq 6 \cdot \sum_{i=1}^{|C|} d(c_i, o_i) + \mathbb{M}(\mathbf{Q}_O).$$

According to (13) and (14) in the proof to Theorem 6,

$$\begin{aligned} \min_{A} D(A, \Delta_0) &\leq \frac{1}{|C|} \sum_{j=1}^{|C|} D(A_{o_j}, \Delta_0) \\ &\leq 2 \cdot \sum_{i=1}^{|C|} d(c_i, o_i) + 2 \cdot \mathbb{M}(\mathbf{Q}_O). \end{aligned}$$

Therefore, the better approximation between $\min_{\Delta} D(N, \Delta)$ and $\min_{A} D(A, \Delta_0)$ satisfies

$$\begin{aligned} &\min\{ \min_{\Delta} D(N, \Delta), \min_{A} D(A, \Delta_0) \} \\ &\leq \frac{1}{3} \cdot \big(\min_{\Delta} D(N, \Delta) + 2 \cdot \min_{A} D(A, \Delta_0) \big) \\ &\leq \frac{1}{3} \cdot \Big(10 \cdot \sum_{i=1}^{|C|} d(c_i, o_i) + 5 \cdot \mathbb{M}(\mathbf{Q}_O) \Big) \end{aligned}$$

$$= \frac{5}{3} \cdot \Big(2 \cdot \sum_{i=1}^{|C|} d(c_i, o_i) + \mathbb{M}(\mathbf{Q}_O)\Big) = \frac{5}{3} \cdot \min_{A,\Delta} D(A, \Delta).$$

Hence, the theorem is proven. □

5.5 Dealing with Limited Server Capacities

So far, we have not assumed any limitation on the server capacity. If the capacity of each server is limited, the number of clients assigned to each server must be kept within its capacity to prevent the processing delay at the server from increasing significantly [18]. Our proposed algorithms can be easily adapted to ensure that the capacity of each server is not exceeded. In *Nearest-Server Assignment*, if the nearest server of a client is fully occupied, the client can try to connect to the remaining servers in increasing order of their distances to the client until a server not yet saturated is found. For *Greedy Assignment*, we can first rank all the servers in ascending order of their total distances to all the clients, and initialize the active server set S_{act} to include top $\lceil |C|/P \rceil$ servers in the ranking, where P is the capacity of a server. In the client assignment process (Algorithm 3), only the servers not yet saturated are considered when assigning the clients. With limited server capacities, the approximation factors analyzed earlier may not hold. We leave the approximability analysis of the "capacitated" MIT problem to the future work. In the next section, we experimentally evaluate the "capacitated" algorithms.

6 Experimental Evaluation

6.1 Experimental Setup

We evaluate the above approximation approaches using three real network latency datasets: the Meridian dataset, the PlanetLab-All-Pairs-Pings dataset, and the PlanetLab-Datacenter dataset. The Meridian dataset is collected in the Meridian project [19], which contains pairwise latency measurements between nodes in the Internet using the King technique [13]. In our simulation, the network is represented by a complete latency matrix among 1796 nodes. The PlanetLab-All-Pairs-Pings dataset is collected by J. Stribling [23], which includes periodic pairwise latency measurements between PlanetLab nodes performed every 15 minutes. In our simulation, we use the data collected on May 8, 2005 that contains 220 nodes and calculate the average pairwise latencies through the day. The PlanetLab-Datacenter dataset consists of two network latency datasets. The first dataset collected by Wu et al. [28] contains latency measurements between PlanetLab nodes and datacenters from Amazon EC2 and Microsoft Azure. The second dataset collected by Garcia-Dorado et al. [10] contains latency measurements between all pairs of datacenters from Amazon EC2 and Microsoft Azure. We integrate these two datasets and simulate a network including 253 PlanetLab nodes and 13 datacenters with 7 from Amazon EC2 and 6 from Microsoft Azure. Throughout our experiments, servers are placed at a certain number of selected nodes, and a client is assumed to be located at each of the remaining nodes. The experiments are carried out with different server numbers and capacities. All experiments are conducted with a PC workstation with 16GB RAM and Intel Core i7-4770 CPU.

Three types of server placement schemes are simulated in the experiments for the Meridian and PlanetLab-All-Pairs-Pings datasets: random, k-center, and k-median. For random placement, we perform 10 simulation runs using different sets of randomly selected nodes to place servers, and present the average performance results along with the 10th and 90th percentile results. The k-center and k-median problems are widely used to model server placement in the Internet [5]. Both problems are NP-hard [11]. In our experiments, we employ two greedy heuristics of the k-center and k-median problems [5], [20] to place servers. For the PlanetLab-Datacenter dataset, we straightforwardly place servers at all the datacenters.

We construct a theoretical lower bound of $\min_{A,\Delta} D(A, \Delta)$ to quantify the relative performance of different approaches. According to Corollary 1 and Lemma 3, we have

$$\min_{\Delta} D(A, \Delta) = 2 \cdot \sum_{i=1}^{|C|} d(c_i, s_A(c_i)) + \mathbb{M}(\mathbf{Q}_A)$$
$$\geq 2 \cdot \sum_{i=1}^{|C|} d(c_i, s_A(c_i)) + \frac{1}{|C|} \sum_{i=1}^{|C|} \sum_{j=1}^{|C|} d(s_A(c_i), s_A(c_j))$$
$$= \frac{1}{|C|} \sum_{i=1}^{|C|} \sum_{j=1}^{|C|} \Big(d(c_i, s_A(c_i)) + d(s_A(c_i), s_A(c_j)) + d(s_A(c_j), c_j)\Big).$$

For any client assignment A, we have

$$d(c_i, s_A(c_i)) + d(s_A(c_i), s_A(c_j)) + d(s_A(c_j), c_j)$$
$$\geq \min_{s_a, s_b \in S} \{d(c_i, s_a) + d(s_a, s_b) + d(s_b, c_j)\}.$$

Therefore, $\min_{A,\Delta} D(A, \Delta)$ must be bounded by

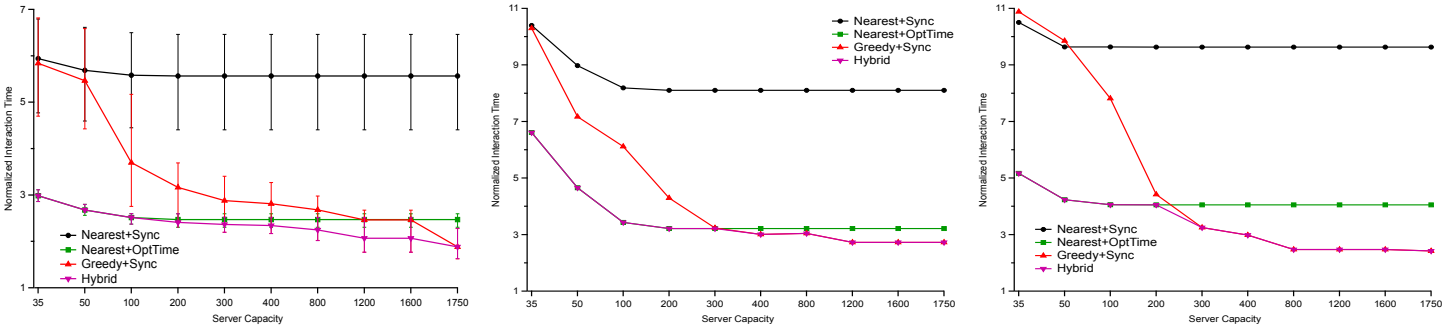
$$\frac{1}{|C|} \sum_{i=1}^{|C|} \sum_{j=1}^{|C|} \min_{s_a, s_b \in S} \{d(c_i, s_a) + d(s_a, s_b) + d(s_b, c_j)\}.$$

Note that this lower bound may not be achievable by any client assignment A and simulation time setting Δ, and is thus not tight. The interaction time $D(A, \Delta)$ produced by each approximation approach is normalized with respect to the above lower bound.

We evaluate four approaches: (1) assign each client to its nearest server and synchronize the simulation times of the assigned servers (denoted by Nearest+Sync, which is the baseline approach discussed at the beginning of Section 5); (2) assign each client to its nearest server and optimize the simulation time setting as discussed in Section 3 (denoted by Nearest+OptTime); (3) optimize the client assignment using *Greedy Assignment* as discussed in Section 5.3.1 and synchronize the simulation times of the assigned servers (denoted by Greedy+Sync); (4) assign the clients and/or optimize the simulation time setting using the hybrid approach as discussed in Section 5.4 (denoted by Hybrid).

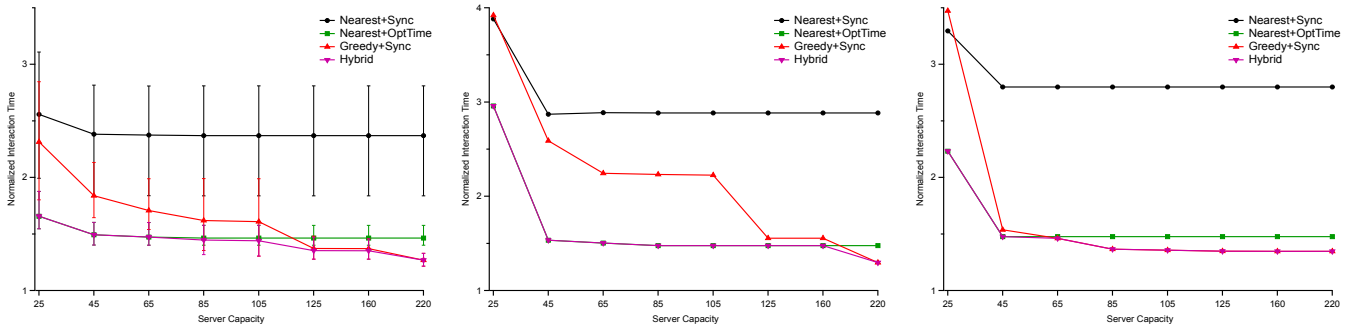
6.2 Experimental Results

Figure 7 shows the normalized interaction times as a function of the server capacity when there are 60 servers placed in the network represented by the Meridian dataset, where the server capacity refers to the maximum number of clients that can be assigned to each server. The error bars indicate



(a) Random server placement (b) k-center server placement (c) k-median server placement

Fig. 7. Results for Meridian dataset with 60 servers.



(a) Random server placement (b) k-center server placement (c) k-median server placement

Fig. 8. Results for PlanetLab-All-Pairs-Pings dataset with 10 servers.

the 10th and 90th percentile results for random server placement. Note that the theoretical lower bound does not assume any server capacity limitation and is thus independent of the server capacity. Therefore, all the results for each server placement are normalized with the same base. It can be seen that the interaction times resulting from all the algorithms usually increase with decreasing server capacity. This is because at lower server capacities, the algorithms are less likely to succeed in assigning clients to the most preferable servers. In general, Nearest+OptTime, Greedy+Sync and Hybrid significantly outperform the intuitive Nearest+Sync approach. This implies that simply assigning each client to its nearest server and synchronizing the simulation times may not be effective in minimizing the interaction time for continuous DIAs. Comparing the remaining three approaches, Hybrid produces the best performance. This indicates that it is beneficial to choose the approximation approach to the MIT problem in an adaptive manner. We remark that since Hybrid simply takes the better solution between Nearest+OptTime and Greedy+Sync, it adds very little execution time compared to these two algorithms.

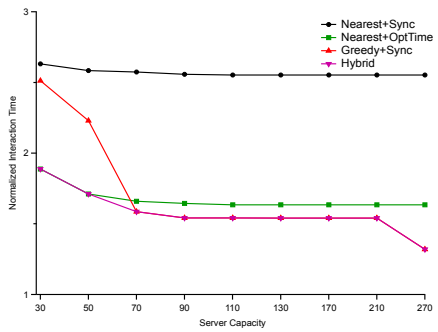


Fig. 9. Results for PlanetLab-Datacenter dataset .

Similar results are observed in the experiments with other numbers of servers and with the other two network latency datasets. Figure 8 shows the results for the PlanetLab-All-Pairs-Pings dataset when there are 10 servers placed in the network, and Figure 9 shows the results for the PlanetLab-Datacenter dataset.

7 Conclusions

In this paper, we have studied the MIT problem for minimizing the interaction times in continuous distributed interactive computing by optimizing the client assignment and the simulation time offsets among servers. We have shown that the MIT problem is NP-hard. To approximate the MIT problem, two approaches are presented: by fixing the client assignment and by fixing the simulation time offsets among servers. When the client assignment is fixed, we have shown that finding the minimum achievable interaction time can be reduced to a weighted bipartite matching problem. When the simulation time offsets among servers are fixed, we have shown that finding the minimum achievable interaction time is still NP-hard. The approximation factors of the above two approaches as well as a hybrid approach combining them have been analyzed. These approximation approaches

have been compared by experimental evaluation using real Internet latency data. For future work, we plan to extend the approximability analysis to the "capacitated" MIT problem.

ACKNOWLEDGMENTS

This work is supported by Singapore Ministry of Education Academic Research Fund Tier 2 under Grant MOE2013-T2-2-067, and Academic Research Fund Tier 1 under Grant 2014-T1-001-145.

REFERENCES

[1] A. Agustina, F. Liu, S. Xia, H. Shen, and C. Sun. CoMaya: incorporating advanced collaboration capabilities into 3D digital media design tools. In *Proceedings of ACM CSCW 2008*, pages 5–8, 2008.

[2] L. Ahmad, A. Boukerche, A. Al Hamidi, A. Shadid, and R. Pazzi. Web-based e-learning in 3D large scale distributed interactive simulations using HLA/RTI. In *Proceedings of IEEE IPDPS 2008*, pages 1–4, 2008.

[3] J. Brun, F. Safaei, and P. Boustead. Managing latency and fairness in networked games. *Communications of the ACM*, 49(11):46–51, 2006.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[5] E. Cronin, S. Jamin, C. Jin, A.R. Kurc, D. Raz, and Y. Shavitt. Constrained mirror placement on the Internet. *IEEE Journal on Selected Areas in Communications*, 20(7):1369–1382, 2002.

[6] E. Cronin, A.R. Kurc, B. Filstrup, and S. Jamin. An efficient synchronization mechanism for mirrored game architectures. *Multimedia Tools and Applications*, 23(1):7–30, 2004.

[7] D. Delaney, T. Ward, and S. McLoone. On consistency and network latency in distributed interactive applications: A survey-Part I. *Presence: Teleoperators & Virtual Environments*, 15(2):218–234, 2006.

[8] C. Ding, Y. Chen, T. Xu, and X. Fu. CloudGPS: A scalable and ISP-friendly server selection scheme in cloud computing environments. In *Proceedings of IEEE/ACM IWQoS 2012*, 2012.

[9] C. Diot and L. Gautier. A distributed architecture for multiplayer interactive applications on the Internet. *IEEE Network Magazine*, 13(4):6–15, 1999.

[10] Jose Garcia-Dorado and Sanjay Rao. Cost-aware multi data-center bulk transfers in the cloud from a customer-side perspective. *IEEE Transactions on Cloud Computing, PrePrints*, 2015.

[11] M.R. Garey and D.S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman and Company, Calif, 1979.

[12] L. Gautier, C. Diot, and J. Kurose. End-to-end transmission control mechanisms for multiparty interactive applications on the Internet. In *Proceedings of IEEE INFOCOM 1999*, pages 1470–1479, 1999.

[13] K.P. Gummadi, S. Saroiu, and S.D. Gribble. King: Estimating latency between arbitrary Internet end hosts. In *Proc. 2nd ACM SIGCOMM Workshop on Internet Measurment*, pages 5–18, 2002.

[14] C. Jay, M. Glencross, and R. Hubbold. Modeling the effects of delayed haptic and visual feedback in a collaborative virtual environment. *ACM Transactions on Computer-Human Interaction*, 14(2), 2007.

[15] E. Lawler. *Combinatorial optimization: networks and matroids*. Dover Publications, 2001.

[16] K.W. Lee, B.J. Ko, and S. Calo. Adaptive server selection for large scale interactive online games. *Computer Networks*, 49(1):84–102, 2005.

[17] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg. Local-lag and timewarp: Providing consistency for replicated continuous applications. *IEEE Transactions on Multimedia*, 6(1):47–57, 2004.

[18] P. Morillo, J.M. Orduna, M. Fernandez, and J. Duato. Improving the performance of distributed virtual environment systems. *IEEE Trans. Parallel Distrib. Syst.*, 16(7):637–649, 2005.

[19] Meridian Project. The meridian latency data set. [Online] Available: http://www.cs.cornell.edu/People/egs/meridian/, 2011.

[20] L. Qiu, V.N. Padmanabhan, and G.M. Voelker. On the placement of web server replicas. In *Proc. IEEE INFOCOM'01*, pages 1587–1596.

[21] F. Safaei, P. Boustead, C.D. Nguyen, J. Brun, and M. Dowlatshahi. Latency-driven distribution: infrastructure needs of participatory entertainment applications. *IEEE Communications Magazine*, 43(5):106–112, 2005.

[22] S. Singhal and M. Zyda. *Networked virtual environments: design and implementation*. Addison-Wesley Reading, MA, 1999.

[23] J. Stribling. Planetlab all-pairs-pings. [Online] Available: http://pdos.lcs.mit.edu/~strib/, 2011.

[24] D.N.B. Ta and S. Zhou. A two-phase approach to interactivity enhancement for large-scale distributed virtual environments. *Computer Networks*, 51(14):4131–4152, 2007.

[25] S.D. Webb and S. Soh. Adaptive client to mirrored-server assignment for massively multiplayer online games. In *Proceedings of MMCN 2008*, 2008.

[26] S.D. Webb, S. Soh, and W. Lau. Enhanced mirrored servers for network games. In *Proceedings of ACM SIGCOMM NetGames 2007*, pages 117–122, 2007.

[27] D.B. West. *Introduction to graph theory*. Prentice hall Englewood Cliffs, 2001.

[28] Zhe Wu and Harsha V Madhyastha. Understanding the latency benefits of multi-cloud webservice deployments. *ACM SIGCOMM Computer Communication Review*, 43(2):13–20, 2013.

[29] L. Zhang and X. Tang. Optimizing client assignment for enhancing interactivity in distribute interactive applications. *IEEE/ACM Transactions on Networking*, 20(6):1707–1720, 2012.

[30] L. Zhang and X. Tang. The client assignment problem for continuous distributed interactive applications: analysis, algorithms, and evaluation. *IEEE Transactions on Parallel and Distributed Systems*, 25(3):785–795, 2014.

[31] Lu Zhang, Xueyan Tang, and Bingsheng He. Brief announcement: on minimum interaction time for continuous distributed interactive computing. In *Proceedings of PODC 2013*, pages 122–124. ACM, 2013.

[32] H. Zheng. *Server Provisioning for Distributed Interactive Applications*. PhD Thesis, Nanyang Technological University, 2014.

[33] H. Zheng and X. Tang. Analysis of server provisioning for distributed interactive applications. *IEEE Transactions on Computers*, 64(10):2752–2766, 2015.

Lu Zhang Lu Zhang received the BEng degree in computer science and engineering from University of Science and Technology of China in 2008, and the PhD degree in computer science from Nanyang Technological University in 2013. He is currently a postdoctoral researcher in the Computer Science and Computer Engineering Department, University of Arkansas. His research interests include distributed computing, discrimination-aware data mining, and causal inference.

Xueyan Tang Xueyan Tang received the BEng degree in computer science and engineering from Shanghai Jiao Tong University in 1998, and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2003. He is currently an associate professor in the School of Computer Science and Engineering at Nanyang Technological University, Singapore. He has served as an associate editor of the IEEE Transactions on Parallel and Distributed Systems. His research interests include distributed systems, cloud computing, mobile and pervasive computing, and wireless sensor networks. He is a senior member of the IEEE.

Bingsheng He Bingsheng He received the bachelor degree in computer science from Shanghai Jiao Tong University (1999-2003), and the PhD degree in computer science in Hong Kong University of Science and Technology (2003-2008). He is an Associate Professor in School of Computing, National University of Singapore. His research interests are high performance computing, distributed and parallel systems, and database systems.