Replica Placement Algorithms for Mobile Transaction Systems

Manghui Tu, Student Member, IEEE, Peng Li, Liangliang Xiao, I-Ling Yen, Member, IEEE, and Farokh B. Bastani, Member, IEEE

Abstract—In distributed mobile systems, communication cost and disconnections are major concerns. In this paper, we address replica placement issues to achieve improved performance for systems supporting mobile transactions. We focus on handling correlated data objects and disconnections. Frequently, requests and/or transactions issued by mobile clients may access multiple data objects and should be considered together in terms of replica allocation. We discuss the replication cost model for correlated data objects and show that the problem of finding an optimal solution is NP. We further adjust the replication cost model for disconnections. A heuristic “expansion-shrinking” algorithm is developed to efficiently make replica placement decisions. The algorithm obtains near optimal solutions for the correlated data model and yields significant performance gains when disconnection is considered. Experimental studies show that the heuristic expansion-shrinking algorithm significantly outperforms the general frequency-based replication schemes.

Index Terms—Mobile transaction systems, cost, correlated data, replica placement, disconnection.

1 INTRODUCTION

Many modern distributed applications support mobile clients and it is common for the mobile clients to access backend databases and/or shared data files [21], [4], [17], [18]. For example, traders may issue business transactions via mobile devices, salesmen may access the inventory data and client information, and travelers may access sites for hotel reservation or make travel plan changes.

Due to the limited bandwidth and frequent disconnections in mobile environments, many efforts have been made to improve the performance and reliability for mobile transaction processing. Most of the research works in this direction focus on transaction processing protocols. In [14], a new lock management scheme is presented which allows a read unlock to be executed at any replica site. Multiple read unlock messages are sent at the first write lock request to reduce the communication cost. In [5], a Kangaroo mobile transaction model is proposed to address the efficient processing of transactions issued by mobile clients roaming around different coverage areas. It allows each subtransaction to be issued to a different base station and the system automatically forwards the transaction information to the current site. In [17], a time-out-based commit protocol, TCOT, has been proposed. TCOT allows the coordinator to commit a transaction if, within a timeout period, no abort messages are received from the cohorts executing the subtransactions.

The research works discussed above focus on mobile transaction processing protocols executed at the stationary nodes. In [6], the precommit concept is used to allow disconnected mobile clients to execute and precommit transactions. Precommitted transactions can be committed by the primary site after the mobile client gets connected again. Mazumdar et al. [22] improve the precommit model to achieve easier reconciliation at reconnection time. In [11], mobile clients can have the sign-off and checked-out modes. A disconnected mobile client can have read/write accesses on checked-out data and read-only accesses to other replicated objects during the sign-off period. Correspondingly, the other nodes have read-only accesses to objects that are checked-out by some mobile clients, but full accesses to objects that are not checked-out. By doing so, the reconciliation process can be easy and efficient.

All mobile side transaction processing protocols assume data replication on mobile nodes, but most of them do not consider how the data objects are replicated [6], [22], [24], [19]. Actually, replica placement in a mobile environment is an important issue and good placement strategies can result in significant performance gains [12], [25], [8]. In a mobile environment, full replication is generally not feasible [10]. Replica placement algorithms that are specific for mobile environment have been investigated. Huang et al. [12] propose a sliding window algorithm for replica allocation at the mobile nodes. The algorithm is based on the simple access frequency scheme, which determines whether to allocate or deallocate a data object according to the numbers of read and write accesses. Hara [8] proposed three methods for allocating replicas on ad hoc networks for data objects shared by multiple mobile hosts. The major goal is to allocate data objects such that they are distributed evenly in the system without having many replicas cluttered in a neighborhood. The paper assumes that there are no update accesses and all hosts know the read access probability of each independent data object. Hara [9] extends the earlier work and considers update accesses,
and they replicate the data objects with higher access frequencies and larger update time periods.

All existing works addressing data replication issues, including the mobile [12], [25], [8] and Internet environments [1], [27], [16], have not considered correlation among data objects. Data objects are considered correlated if they are accessed by the same transactions. For example, for a read transaction accessing multiple data objects, replicating all of these data objects at a mobile node can reduce the communication overhead between the mobile node and other nodes. However, if only a part of the data set to be accessed is replicated at a mobile node, then the replication will not bring much benefit because the read transaction still needs to be forwarded to retrieve the remaining data objects. The same applies to an update transaction accessing multiple data objects. Even if only a part of the data set to be updated is replicated, a message is needed to propagate the update request. Thus, to achieve better replica allocation, it is essential to consider the access correlation among data objects.

In this paper, we address the placement issues for partial replication of correlated data objects. Our goal is to improve the mobile nodes data accessing performance, especially to reduce the communication cost due to data accesses. We propose a data placement algorithm to dynamically replicate correlated data objects based on historical access patterns. We first show that the problem of finding optimal solutions based on the data correlation model is NP. An optimal replica placement algorithm based on the data correlation model is introduced. To reduce the time complexity, we further develop a heuristic “expansion-shrinking” algorithm to obtain near optimal replica placement solutions.

Besides communication cost, disconnection is also a common issue in mobile transaction processing. As discussed above, disconnection is the major concern in mobile transaction processing protocols [15], [6], [5], [21], [17]. Also, in [8], [9], frequently accessed data objects, subject to space constraints, are placed on mobile nodes to achieve high availability during disconnection time. Actually, disconnection also has major impact on the communication cost for transaction processing. For example, to commit an update transaction issued by a mobile node, the updates should be propagated to a primary site if the mobile node is connected. Delayed propagation of update transactions in many mobile transaction processing protocols [6], [10] may actually result in reduced communication cost. Existing mobile replica placement algorithms [12], [25] do not consider such impact.

In this paper, we also develop a heuristic algorithm to take the disconnection effect into account. Essentially, the communication cost for propagating multiple update requests together instead of propagating them independently (one at a time) is factored in the replication cost model to achieve more accurate replica placement decisions. Unlike data access patterns, which generally have a high degree of locality [3], [20], grouping update propagation messages due to disconnection is a random process. Thus, the scheme we consider attempts to improve the accuracy in placement decisions, but does not consider optimality.

The remainder of this paper is organized as follows: Section 2 describes the system model and problem specification. Section 3 defines the replication cost model based on the concept of data correlation, but without considering disconnection effect. Section 4 discusses the required modification to the cost model for addressing the effect of mobile disconnections. Section 5 introduces the optimal replica placement algorithm considering only the correlated data model. In Section 6, an “expansion-shrinking” algorithm that can be used for both cost models (with or without the disconnection effect) is introduced. To allow performance comparisons of the heuristic algorithm with existing, most commonly used frequency-based partial replication algorithms, we introduce a general form of frequency-based algorithm in Section 7. Section 8 discusses the experimental studies for comparing various algorithms and presents the results. Section 9 states the conclusion of the paper.

2 SYSTEM MODEL AND PROBLEM SPECIFICATION

Consider a mobile system consisting of a stationary base node $P_{BC}$ and a set of mobile nodes. Node $P_{BC}$ hosts a database which can be a partial replica of a remote database, and mobile clients issue transactions to access the database. The database may be partially replicated at mobile nodes. Let $D_{BC} = \{d_1, d_2, \ldots, d_N\}$ denote the set of $N$ data objects that are available on $P_{BC}$. Assume that all mobile nodes are independent with each other, i.e., they directly interact with $P_{BC}$. Thus, without loss of generality, we can consider only one mobile node $P_{MC}$. Let $D_{MC}$ denote the set of data objects replicated on a mobile node $P_{MC}$, $D_{MC} \subseteq D_{BC}$.

When considering replica placement, we need to consider data set read or updated by the transaction. The update consistency protocol for the system we consider is the primary lazy update protocol, and the read protocol is to read from the closest replica. Let $t$ denote a transaction; it can be a read transaction or an update transaction. Let $D(t)$ denote the set of data objects read or updated by $t$, $D(t) \subseteq D_{BC}$, and $D(t)$ is defined as a transaction-data set of $t$. For a read transaction $t$, if $D(t) \subseteq D_{MC}$, then $t$ is served at $P_{MC}$ locally; otherwise, $t$ is forwarded to $P_{BC}$. For an update transaction $t$, it needs to be forwarded to $P_{BC}$ first. If $D(t) \cap D_{MC} \neq \emptyset$, $t$ needs to be propagated to $P_{MC}$, which is the overhead due to the replication of any subset of $D(t)$ at $P_{MC}$. When a transaction contains both read and update operations, we can simply consider it as an update transaction and the effect of the read operations can be ignored. Since the transaction needs to be forwarded to $P_{BC}$ for execution anyway, the read operations can also be executed at $P_{BC}$ without incurring additional cost.

Without loss of generality, we consider the traffic on the link between $P_{MC}$ and $P_{BC}$, which is measured by the number of messages between $P_{MC}$ and $P_{BC}$. We assume that for any transaction, if communication between $P_{BC}$ and $P_{MC}$ is required, then only one message is needed, assuming that, in general, the data sets accessed by the transactions are not of large size. A read transaction issued by a mobile node other than $P_{MC}$ can either be served locally or at $P_{BC}$, so it does not affect the
communication cost between $P_{BC}$ and $P_{MC}$. Thus, these read transactions can be ignored in the analysis. Thus, we only need to consider read transactions issued by $P_{MC}$. An update transaction needs to be propagated to $P_{MC}$ if $D(t) \cap D_{MC} \neq \emptyset$. So, in the analysis for $P_{MC}$, we need to consider all update transactions $t$ where $D(t) \cap D_{MC} \neq \emptyset$, but there is no need to consider update transactions issued by $P_{MC}$ itself. For $P_{BC}$, we need to consider all update transactions but those update transactions issued by $P_{MC}$.

During the period when $P_{MC}$ is disconnected, update transactions at $P_{BC}$ cannot be propagated to $P_{MC}$. We assume that $P_{BC}$ will propagate those update transactions in one message the next time $P_{MC}$ is connected. Note that here we only consider the disconnection effect on data consistency maintenance, and do not consider the issues of data accesses during disconnection period (which have been addressed by some existing works [14], [6], [5], [21], [17]). We assume that when some data objects to be accessed by a read transaction issued at $P_{MC}$ are not available on $P_{MC}$, then $P_{MC}$ will forward the request to $P_{BC}$. Also, $P_{MC}$ always connects to forward update transactions issued locally to $P_{BC}$.

To minimize the traffic on the link between $P_{MC}$ and $P_{BC}$, we need to determine the set of data objects to be placed on $P_{MC}$. Since client access patterns show some locality, we can determine the best $D_{MC}$ based on the historical access patterns. Consider a time period $T$. $P_{BC}$ maintains a transaction log $L_{BC}$, and $P_{MC}$ maintains a transaction log $L_{MC}$. When $P_{BC}$ or $P_{MC}$ receives a transaction $t$, a transaction record is made and appended to $L_{BC}$ or $L_{MC}$, respectively. Let $Q(S)$ be the set of read transactions issued by $P_{MC}$ in the time duration $T$, accessing data set $S$, i.e.,

$$Q(S) = \{t | D(t) = S \land S \subseteq D_{BC} \land D(t) \neq \emptyset, \}$$

and $t$ is a read transaction issued by $P_{MC}$. Also, let $W(S)$ be the set of update transactions issued within the time period $T$, accessing all the data objects in $S$, i.e.,

$$(S) = \{t | D(t) = S \land S \subseteq D_{BC} \land D(t) \neq \emptyset, \}$$

and $t$ is an update transaction. $|Q(S)|$ and $|W(S)|$ are, hence, the number of read and update transactions accessing data set $S$, respectively. They will be used to compute the tradeoffs for replicating $S$.

In this paper, the communication cost introduced by replica allocation and deallocation, message losses, and node failures are not considered.

## 3 Replication Cost Model Based on Data Correlations

From the system model, we can see that $P_{BC}$ does not know the read transactions served locally at $P_{MC}$ (read data objects already replicated at $P_{MC}$) and $P_{MC}$ does not know the update transactions that are not propagated to $P_{MC}$ (updated data objects not replicated at $P_{MC}$). Thus, neither of them alone can complete the allocation decision computation (unless they exchange logs, which incurs high communication cost). Thus, the replication algorithm based on data correlation should be decomposed into replica allocation (executed at $P_{BC}$) and deallocation processes (executed at $P_{MC}$). Correspondingly, two cost models are needed for the data correlation model, including the replica allocation cost model and deallocation cost model. Consider a set of data objects $\delta$, $\delta \subseteq D_{BC}$. If $\delta$ is replicated to $P_{MC}$, a read transaction $t$, $D(t) \subseteq \delta$, can be processed locally and one message can be saved. We consider this as one unit of benefit for replicating $\delta$ on $P_{MC}$. However, due to the replication of $\delta$ on $P_{MC}$, an update transaction $t$, $D(t) \cap \delta \neq \emptyset$, needs to be propagated to $P_{MC}$. We consider this as one unit of cost introduced by replicating $\delta$ on $P_{MC}$.

We define $update_{a}(\delta)$ as the additional cost if $\delta$ is replicated to $P_{MC}$. Essentially, $update_{a}(\delta)$ is the number of update transactions accessing some data objects in $\delta$, issued by node other than $P_{MC}$. It can be expressed as:

$$update_{a}(\delta) = \sum |W(S)|, \text{ where } (S \cap \delta \neq \emptyset) \land (S \cap D_{MC} = \emptyset).$$

We define $read_{a}(\delta)$ as the additional benefit if $\delta$ is replicated to $P_{MC}$. Essentially, the $read_{a}(\delta)$ is the number of read transactions issued by $P_{MC}$, accessing some data objects in $\delta$. It can be expressed as:

$$read_{a}(\delta) = \sum |Q(S)|, \text{ where } (S \subseteq \delta \cup D_{MC}) \land (S \cap \delta \neq \emptyset).$$

Now, we define $cost_{a}(\delta)$ as the access cost for replicating $\delta$ at $P_{MC}$. It is the difference between update cost and read benefit for replicating $\delta$ at $P_{MC}$. It can be expressed as:

$$cost_{a}(\delta) = update_{a}(\delta) - read_{a}(\delta).$$

The partial replication algorithm should allocate a data set $S$ on $P_{MC}$ if $cost_{a}(S) = \min cost_{a}(\delta)$, for all $\delta, \delta \subseteq D_{BC}$, and $cost_{a}(S) < 0$.

Now, we describe the cost model for the partial replica deallocation. A set of data objects, $\delta \subseteq D_{MC}$, may need to be deallocated from $P_{MC}$ if replicating $\delta$ no longer brings benefit in terms of communication cost. Let $update_{d}(\delta)$ denote the deallocation benefit for deallocating $\delta$. Essentially, the deallocation benefit of $\delta$ is the number of update transactions from $P_{BC}$, accessing a subset of data objects in $\delta$.

$$update_{d}(\delta) = \sum |W(S)|, \text{ where } (S \subseteq \delta) \land (S \subseteq D_{MC}) \land (S \neq \emptyset).$$

Let $read_{d}(\delta)$ denote the deallocation cost. Essentially, the deallocation cost of $\delta$ is the number of all read transactions issued by $P_{MC}$, accessing a subset of $D_{MC}$ and some data objects in $\delta$.

$$read_{d}(\delta) = \sum |Q(S)|, \text{ where } (S \cap \delta \neq \emptyset) \land (S \subseteq D_{MC}) \land (S \subseteq D_{MC}).$$

The replica deallocation access cost $cost_{d}(\delta)$, is the difference between read cost and update benefit for deallocating $\delta$ from $P_{MC}$.

$$cost_{d}(\delta) = read_{d}(\delta) - update_{d}(\delta).$$
The partial replication scheme should deallocate δ from 
P_{MC} when \text{cost}_{d}(\delta) is minimal among all subsets of D_{MC} and \text{cost}_{d}(\delta) \leq 0.

The goal is to obtain the optimal δ, such that the system incurs a minimum cost (minimal \text{cost}_{d}(\delta) for replica allocation or minimal \text{cost}_{d}(\delta) for replica deallocation) among all possible data sets. Next, we will show some basic properties of optimal replica allocation. We first prove that finding such an optimal data set δ is NP-hard. Since the proofs for the complexity of optimal replica allocation and deallocation algorithms are similar, we will only discuss the NP-hard proof for replica allocation algorithm.

**Theorem 1.** The problem of finding an optimal set δ such that \text{cost}_{d}(\delta) is minimal is NP-hard.

**Proof.** As defined earlier, \(D_{BC} = \{d_1, d_2, \ldots, d_N\}\)

Let \(\theta_i = \begin{cases} 1, & \text{if } d_i \text{ is on } P_{BC} \text{ only} \\ 0, & \text{if } d_i \text{ is replicated at } P_{MC} \text{ and } \overline{\theta_i} = 1 - \theta_i \end{cases} \)

\(\text{cost}_{d}(\delta) = \text{update}_{d}(\delta) - \text{read}_{d}(\delta) = \Sigma [W(S)], \text{ where} \)

\((S \cap \delta \neq \emptyset) \land (S \cap D_{MC} = \emptyset) = \Sigma [Q(S)], \text{ where} \)

\((S \subseteq \delta \cup D_{MC}) \land (S \cap \delta \neq \emptyset). \)

For a specific S in \(|W(S)|, S \subseteq D_{BC}\), then

\(S = \{d_{i_1}, d_{i_2}, \ldots, d_{i_j}\}, \)

where \(1 \leq j \leq N, \text{ and } 1 \leq i_1, i_2, \ldots, i_j \leq N. \) Let \(a_{i_1, i_2, \ldots, i_j}\) denote the number of update transactions accessing \(S\), then

\(|W(S)| = a_{i_1, i_2, \ldots, i_j} (1 - \theta_{i_1}) (1 - \theta_{i_2}) \ldots (1 - \theta_{i_j})

= \prod_{i_1, i_2, \ldots, i_j} a_{i_1, i_2, \ldots, i_j} \theta_{i_1} \theta_{i_2} \ldots \theta_{i_j}, \)

Similarly, for a specific S in \(|Q(S)|, S \subseteq D_{BC}\), then

\(S = \{d_{i_1}, d_{i_2}, \ldots, d_{i_j}\}, \)

where \(1 \leq j \leq N \) and \(1 \leq i_1, i_2, \ldots, i_j \leq N. \) Let \(a_{i_1, i_2, \ldots, i_j}\) denote the number of read transactions accessing \(S\), then

\(|Q(S)| = a_{i_1, i_2, \ldots, i_j} (1 - \theta_{i_1}) (1 - \theta_{i_2}) \ldots (1 - \theta_{i_j})

= \prod_{i_1, i_2, \ldots, i_j} a_{i_1, i_2, \ldots, i_j} \theta_{i_1} \theta_{i_2} \ldots \theta_{i_j}, \)

Thus, \text{cost}_{d}(\delta) = \sum_{i_1, i_2, \ldots, i_j} a_{i_1, i_2, \ldots, i_j} \theta_{i_1} \theta_{i_2} \ldots \theta_{i_j}, \) where \(1 \leq j \leq N, \) and \(1 \leq i_j \leq N. \) To obtain a solution for a specific problem, \(a_{i_1, i_2, \ldots, i_j}\) is known as constant, then we will ignore the constant part. Let

\(\text{cost}_{d'}(\delta) = \sum_{i_1, i_2, \ldots, i_j} \theta_{i_1} \theta_{i_2} \ldots \theta_{i_j}, \)

which is equal to \(\sum_{i_1, i_2, \ldots, i_j} \theta_{i_1} \wedge \theta_{i_2} \wedge \ldots \wedge \theta_{i_j}. \) If we can obtain \(\sum_{i_1, i_2, \ldots, i_j} \theta_{i_1} \wedge \theta_{i_2} \wedge \ldots \wedge \theta_{i_j}, \) we should know \(\wedge_{i_1, i_2, \ldots, i_j} (\theta_{i_1} \lor \theta_{i_2} \lor \ldots \lor \theta_{i_j}). \) We know that \(\wedge_{i_1, i_2, \ldots, i_j} (\theta_{i_1} \lor \theta_{i_2} \lor \ldots \lor \theta_{i_j}) \) is NP-complete and, hence, to find a solution for it is NP-hard. Thus, it follows that to find an optimal set δ such that \text{cost}_{d}(\delta) is minimum is NP-hard. \(\square\)

To show other properties, we first define some useful concepts. Let \(U\) denote the set of \(N\) transactions recorded at \(P_{BC}(P_{MC})\) during the time period \(T, \) i.e., \(U = \{t_1, t_2, \ldots, t_N\}. \)

We define a transaction-data union of \(P_{BC}(P_{MC})\) as the union of data sets of any subset of transactions in \(U. \)

More specifically, a transaction-data union is \(\bigcup_{t \subseteq U} D(t), \) where \(\emptyset \subseteq U \subseteq \emptyset. \) A data set \(S'\) is a nontransaction-data union of \(P_{BC}(P_{MC})\) if there exists a data item \(d\) in \(S'\) such that \(d\) does not belong to any transaction-data set that is a subset of \(S',\) i.e., \(\exists d \in S', \text{ s.t. } \exists t \subseteq U \land d \in D(t) \land D(t) \subseteq S'. \)

For example, suppose there are only two read transactions \(t_1\) and \(t_2\) recorded at \(P_{BC}\) in a certain time period \(T,\) with \(D(t_1) = \{A, B\}\) and \(D(t_2) = \{C, D\}. \)

To obtain \(\text{cost}_{d}(\delta) \leq \text{cost}_{d}(\delta').\)

**Theorem 2.** For a given nontransaction-data union \(\delta' \cup \delta'', \) where \(\delta' \subseteq D_{MC}, \delta' \cap D_{MC} = \emptyset, \text{ and } \text{cost}_{d}(\delta') < 0, \) there exists at least one transaction-data union \(\delta \cup \delta''\) such that \text{cost}_{d}(\delta) \leq \text{cost}_{d}(\delta').\)

**Proof.** Consider creating \(\delta\) by deleting all \(d_i, \) where \(d_i \in \delta' \cup \delta'', \text{ s.t. } \forall t \subseteq U. \)

\((D(t) = S) \land d_i \in S \land S \subseteq (\delta' \cup \delta''). \)

Thus, \(\delta\) is a transaction-data union, and

\(\delta \subseteq \delta' \cup \delta'' \land \delta \cap \delta' \neq \emptyset. \)

Note that \text{cost}_{d}(\delta') < 0 indicates \text{read}_{d}(\delta') > 0. \) Therefore, there must exist a read transaction \(t, \)

\((D(t) = S) \land S \subseteq (\delta' \cup \delta'') \land S \cap \delta' \neq \emptyset. \)

Since for a read transaction \(t,\) only when all data objects included in \(D(t)\) are replicated, \(t\) can be served locally. This means that replicating any \(d_i \in \delta' - \delta\) will not bring communication savings. So, when no \(d_i \in \delta' - \delta\) is involved in any update transactions,

\(\text{cost}_{d}(\delta) = \text{cost}_{d}(\delta'); \)

otherwise, \(\text{cost}_{d}(\delta) < \text{cost}_{d}(\delta'). \)

With Theorem 2, the partial replication algorithm based on the data correlation model only needs to consider transaction data unions for replication decisions. Similar to Theorem 2, for a given access pattern, there always exists an optimal replica deallocation data set that is a transaction-data union of \(P_{MC} \) (due to the similarity, we will not go into details for replica deallocation).
4 REPLICATION COST MODEL CONSIDERING DISCONNECTIONS

In a mobile environment, mobile nodes may disconnect from the system planned or unplanned [11]. The disconnection and reconnection protocols are performed in the network management level, which is independent of the applications [13]. We assume that the applications can always have the timely knowledge of the connection status of the mobile nodes. However, disconnection can impact the access behavior and, subsequently, the replica placement decisions. During the period when \( P_{MC} \) is disconnected, update transactions at \( P_{BC} \) cannot be propagated to \( P_{MC} \). When \( P_{MC} \) is reconnected, \( P_{BC} \) will send \( P_{MC} \) all the update transactions that have not been processed by \( P_{MC} \). The log \( L_{MC} \) at \( P_{MC} \) and \( L_{BC} \) at \( P_{BC} \) record the data sets of the update transactions and are being used by the replica placement algorithm to estimate the communication cost. Since all the update transactions can be sent in one message (or at the cost similar to one message when the number of update transactions to be propagated to \( P_{MC} \) is relatively small), the communication cost estimation is no longer valid. Thus, the cost model needs to be adjusted to take such effects into consideration.

Here, we assume that the transactions executed during a disconnection period are random. Thus, we consider the impact of the disconnection frequency on the message cost and also apply a weight \((< 1)\) to the cost function to model the effect of disconnections. Since the access behavior and, subsequently, the replica placement algorithm determines the optimal transaction-data union \( S \) of \( P_{MC} \), where \( S = D_{MC} \) is \( \arg\min_{\delta \in \delta} (\text{cost}_a(S)) \), where \( \delta \) is a subset of data set \( D_{MC} = D_{MC} \). \( P_{BC} \) then allocates \( S = D_{MC} \) to \( P_{MC} \). Next, \( P_{MC} \) executes the replica deallocation algorithm and determines a transaction-data union \( S \) of \( P_{MC} \), where \( S = \arg\min_{\delta} (\text{cost}_d(S)) \). \( P_{MC} \) then deallocates \( S \) if \( \text{cost}(S) < 0 \).

5 OPTIMAL PARTIAL REPLICA ALGORITHM BASED ON DATA CORRELATIONS

In the section, we discuss the optimal partial replication (OPR) algorithm under the replication cost model considering only data correlations. In this algorithm, \( P_{BC} \) first computes \( \left| Q(S) \right| \) and \( \left| W(S) \right| \) for each \( S \) accessed by transactions recorded in \( L_{BC} \). Then, the replica allocation algorithm determines the optimal transaction-data union \( S \) of \( P_{BC} \), where \( S = D_{MC} \) is \( \arg\min_{\delta \in \delta} (\text{cost}_a(S)) \), where \( \delta \) is a subset of data set \( D_{MC} = D_{MC} \). \( P_{BC} \) then allocates \( S = D_{MC} \) to \( P_{MC} \). Next, \( P_{MC} \) executes the replica deallocation algorithm and determines a transaction-data union \( S \) of \( P_{MC} \), where \( S = \arg\min_{\delta} (\text{cost}_d(S)) \). \( P_{MC} \) then deallocates \( S \) if \( \text{cost}(S) < 0 \).

5.1 OPR Algorithm

The OPR algorithm consists of optimal partial replica allocation (OPRA) executed at \( P_{BC} \) and optimal partial replica deallocation (OPRD) executed at \( P_{MC} \). The algorithm is developed based on the depth-first branch and bound approach. Fig. 1 shows the OPR algorithm. In this algorithm, \( \text{TranList} \) is a list of all transactions recorded in \( L_{BC} \). Note that there may be multiple transactions of the same type accessing the same data set. For such transactions, we only insert one of them into \( \text{TranList} \).

Note that, in the OPR algorithm, we need to compute the lower bound cost for each new transaction-data union \( S_{\text{TranListIndex}} \cup S \). One simple approach to compute such a lower bound cost is as follows: The update cost is still computed by \( \text{update}_a(S_{\text{TranListIndex}} \cup S) \). The read cost is computed by \( \text{read}_a(S_{\text{supper}}) \), where

\[
S_{\text{supper}} = S \cup \bigcup_{i=\text{TranListIndex}}^{\text{TranListLength}-1} S_i,
\]

where \( S_i \) is the transaction-data set of the transaction \( \text{TranList}[i] \), for \( \text{TranListIndex} \leq i \leq n \). This assumes that all transaction-data sets left are included in the transaction-data union so as to make the value of the read cost highest.

The OPR algorithm explores the search tree recursively in a depth-first manner. It essentially checks all possible transaction-data unions exhaustively. Given \( n \) transactions, the number of all possible transaction-data unions is \( \sum_{i=1}^{n} \binom{n}{i} = 2^n - 1 \). Therefore, the computation complexity of OPRA is \( O(2^n) \).

Since \( P_{BC} \) does not have knowledge of read transactions served locally at \( P_{MC} \) (which is only logged in \( L_{MC} \)), \( P_{BC} \) alone cannot make optimal allocation decisions. Here, \( P_{BC} \) assumes that all data in \( D_{MC} \) should remain at \( P_{MC} \) (i.e., they yield more benefit than cost). Based on this assumption,
Let $D_{org}^{MC}$ denote the data set on $PMC$ before $P_{BC}$ executes OPRA and $D_{new}^{MC} = S_{opt} \cup D_{org}^{MC}$. Let $D_{opt}^{MC}$ denote the minimum set of data that should be replicated on $PMC$ to minimize the communication cost between $PMC$ and $P_{BC}$. We show in Theorem 3 that $D_{new}^{MC}$ is a superset of $D_{opt}^{MC}$. Note that with the knowledge of both $L_{MC}$ and $L_{BC}$ (PMC sends $L_{MC}$ to $P_{BC}$ or $P_{BC}$ sends $L_{BC}$ to $PMC$), then $P_{BC}$ or $PMC$ alone can compute $D_{opt}^{MC}$ by executing OPRA once.

**Theorem 3.** $D_{new}^{MC} := S_{opt} \cup D_{opt}^{MC}$, obtained by executing OPRA, is a super set of $D_{opt}^{MC}$.

**Proof.** Please refer to the Appendix.

The algorithm OPRD is developed in a similar way as OPRA. OPRD is executed at $PMC$, and it deallocates a data set to minimize the communication cost along the link between $PMC$ and $P_{BC}$. Since there may be multiple data sets that can be deallocated while yielding the same minimal cost, the set with maximal size is chosen by OPRD. (For example, a data object that is never accessed by any transaction can be kept or removed without impact on cost.) The OPRA is shown in Fig. 2.

The algorithm OPRD is developed in a similar way as OPRA. OPRD is executed at $PMC$, and it deallocates a data set to minimize the communication cost along the link between $PMC$ and $P_{BC}$. Since there may be multiple data sets that can be deallocated while yielding the same minimal cost, the set with maximal size is chosen by OPRD. (For example, a data object that is never accessed by any transaction can be kept or removed without impact on cost.) The OPRA is shown in Fig. 2.

**Theorem 4.** $D_{opt}^{MC} = D_{new}^{MC} = S_{opt}^{d}$

**Proof.** Please refer to the Appendix.

**6 HEURISTIC PARTIAL REPLICATION ALGORITHM**

We develop a heuristic replication algorithm to obtain near optimal solutions. Similar to the OPRA algorithm, the heuristic algorithm is decomposed into heuristic replica allocation and deallocation algorithms. They can be used for both the original data correlation cost model and the modified cost model considering disconnections. However, the algorithms are developed mainly based on the data correlation concept such that the near-optimal properties can be obtained. The disconnection model simply adjusts the cost model of the algorithm by the weights (as defined in Section 4) to reflect the random effect of update transaction grouping. The near-optimality properties and the corresponding proofs only apply to the original data correlation cost model.

In Section 6.1, we discuss a log reduction scheme to reduce the size of the log as well as reduce the time in
processing the log. Note that the reduced log can be used for the OPR algorithm as well. In Sections 6.2 and 6.3, we discuss the details of the heuristic replica allocation and deallocation algorithms, respectively. Note that when discussing the properties of the algorithms in these two sections, we only consider the data correlation model without considering disconnections.

6.1 Log Reduction

Note that for a read only transaction $t$, the data objects in $D(t) \cap D_{MC}$ would not affect the read benefit computation. Thus, we can only consider to record $S$, $S = D(t) - D_{MC}$ instead of $D(t)$. By doing so, if two transactions $t_1$ and $t_2$ have different $D(t_1)$ and $D(t_2)$, but the same $D(t_1) - D_{MC}$ and $D(t_2) - D_{MC}$, then they still can be combined into one record. This can reduce the number of records in the log and, hence, improves the log processing time for both optimal and heuristic algorithms. Let $L_{BC}$ denote the log obtained using the new recording scheme, specifically, $L_{BC}$ contains records for distinct data sets

$$S \subseteq D_{MC}(D_{MC} = D_{BC} - D_{MC}),$$

and its corresponding $|Q(S)|$ and $|W(S)|$. $|Q(S)|$ now is the number of read transactions $t$ where $D(t) - D_{MC} = S$. A recorded update only transaction $t$ affects the communication cost computation only if $D(t) \neq \phi$ and $D(t) \subseteq D_{MC}$, and we can ignore all other update transactions. A similar log reduction concept can be applied to $L_{MC}$. For update transactions recorded in $L_{MC}$, only the data set $S = D(t) \cap D_{MC}$ will affect the cost computation and thus need to be recorded. So, $|W(S)|$ now is the number of update transactions $t$ accessing $S = D(t) \cap D_{MC}$. For recorded read transactions, only those read transactions $t$ with $D(t) \subseteq D_{MC}$ will affect the cost computation and thus need to be recorded. So, $|Q(S)|$ now is the number of read transactions $t$ with $D(t) = S$ and $D(t) \subseteq D_{MC}$. In the following two sections, we use $L_{BC}$ and $L_{MC}$ for the heuristic algorithms.

6.2 Heuristic Replica Allocation Algorithm

We develop a Set-Expansion heuristic algorithm to determine a data set to be allocated to $P_{MC}$. Let $S_{heu}$ denote a data set. At each step, the Set-Expansion algorithm tries to find a minimal set of data objects $S$ recorded in $L_{BC}$ such that $cost_{\alpha}(S_{heu} \cup S) < cost_{\alpha}(S_{heu})$, and then expands the set $S_{heu}$ to $S_{heu} \cup S$. Note that $S$ is a data set recorded in $L_{BC}$ and $|S - S_{heu}|$ is minimal among all data sets remained in $L_{BC}$. If there is more than one data set with the same size, then one of them is chosen arbitrarily. After testing $S$, the record with data set $S$ is deleted from $L_{BC}$. The Set-Expansion algorithm obtains a subset of the set computed by OPR. The Set-Expansion heuristic algorithm is shown in Fig. 3.

Let $L$ denote the number of records in $L_{BC}$. At each step, the Set-Expansion algorithm needs to determine a recorded data set $S$ such that $|S - S_{heu}|$ is minimal among all data sets remained in $L_{BC}$. This has to go through all records remaining in $L_{BC}$ and also needs to scan the data set of each record. In the worst case, there will be $L$ records remaining in $L_{BC}$ and the size of data set $S$ or $S_{heu}$ is $N$ (note that $N$ is the size of $D_{BC}$), thus the running time for this step is $O(LN)$. Once we find the data set $S$ such that $|S - S_{heu}|$ is minimal, we need to calculate $cost_{\alpha}(S_{heu} \cup S)$ or $cost_{\alpha}(S)$, which also requires a running time of $O(LN)$. So, each step has time complexity $O(LN)$. The Set-Expansion algorithm repeats this step until $L_{BC}$ is empty, which takes a total of $L$ steps. Thus, the time complexity of the Set-Expansion algorithm is $O(L^2N)$.

The Set-Expansion heuristic algorithm cannot always compute optimal data set to be allocated to $P_{MC}$. For example, consider six read transactions accessing $\{A, B\}$, seven read transactions accessing $\{B, C\}$, and 10 update transactions accessing $\{B\}$, $D_{MC}$ is assumed to be $\phi$, and $S_{heu}$ is initially empty. The Set-Expansion algorithm works as follows: In the first step, the algorithm chooses data set $\{B\}$ for testing because $|\{B\}|$ is minimal. Since $cost_{\alpha}(\{B\}) = 10 > 0$, $B$ will not be added to $S_{heu}$. Next, the data sets $\{A, B\}$ and $\{B, C\}$ are tested, and $cost_{\alpha}(\{A, B\}) = 4 > 0$ and $cost_{\alpha}(\{B, C\}) = 3 > 0$. So, neither $\{A, B\}$ nor $\{B, C\}$ should be added to $S_{heu}$. So, the Set-Expansion algorithm will not replicate any data objects in $\{A, B, C\}$. However, data set $\{A, B, C\}$ should be replicated at $P_{MC}$ because $cost_{\alpha}(\{A, B, C\}) = -3 < 0$.

To compensate for this problem, we also consider a Set-Shrinking heuristic algorithm. Set-Shrinking algorithm assumes that all data objects in the set $D_{BC} - D_{MC}$ should be allocated to $P_{MC}$. Initially, $S_{heu}$ is set to be $\phi$. At each step,
the algorithm tries to find a minimal data set \( S \) recorded in \( L'_{BC} \), such that

\[
\text{cost}_a(D_{BC} - D_{MC} - S - S_{\text{heu}}) < \text{cost}_a(D_{BC} - D_{MC} - S_{\text{heu}}),
\]

and adds \( S \) to \( S_{\text{heu}} \). After testing all the data set recorded in \( L'_{BC} \), we obtain \( S_{\text{heu}} \) and shrink the allocation data set from \( D_{BC} - D_{MC} \) to \( D_{BC} - D_{MC} - S_{\text{heu}} \). The Set-Shrinking algorithm is similar to the Set-Expansion algorithm and the pseudocode is shown in Fig. 4. The Set-Shrinking algorithm also has time complexity \( O(L^2N) \). It does not always obtain optimal solutions either. Actually, the Set-Shrinking algorithm computes a superset of the optimal data set computed by OPRA.

The Set-Expansion and Set-Shrinking algorithms can be combined to obtain an improved allocation algorithm, the Expansion-Shrinking algorithm. In this algorithm, two data sets are obtained by executing the Set-Expansion and Set-Shrinking algorithms, and the one yielding lower cost is chosen \( P_{MC} \).

Next, we discuss the properties of the Set-Expansion algorithm, the Set-Shrinking algorithm, and the Expansion-Shrinking algorithm. In Theorem 5, we show that the Set-Expansion algorithm yields a subset of the optimal data set computed by OPRA. In Theorem 6, we show that the Set-Shrinking algorithm yields a superset of the optimal data set computed by OPRA. In Lemma 1, we show that every data set \( \delta \) added to \( S_{\text{heu}} \), then \( \text{cost}_a(S_{\text{heu}}) < \text{cost}_a(S_{\text{heu}} - \delta) \). We show in Lemma 2 that any subset of \( S_{\text{heu}} \) will introduce positive reduction on communication cost. Note that all the proofs are based on the data correlation cost model, without considering the adjusted weight in disconnection cost model.

**Lemma 1.** Let \( \delta \) denote the minimal-sized data set chosen by the Set-Expansion algorithm for testing. If \( \delta \) is added to \( S_{\text{heu}} \), then \( \text{cost}_a(S_{\text{heu}}) < \text{cost}_a(S_{\text{heu}} - \delta) \).

**Proof.** Please refer to the Appendix.

**Lemma 2.** Let \( \delta_k \) be an arbitrary set of data object, such that \( \delta_k \subseteq S_{\text{heu}} \wedge \delta_k \neq \phi \). Then, \( \text{cost}_a(S_{\text{heu}}) < \text{cost}_a(S_{\text{heu}} - \delta_k) \), where \( S_{\text{heu}} \) is the data set obtained from the Set-Expansion algorithm.

**Proof.** Please refer to the Appendix.

**Theorem 5.** Let \( S^a_{\text{opt}} \) denote the optimal set obtained by the Optimal Partial Replica Allocation (OPRA) algorithm, then \( S_{\text{heu}} \subseteq S^a_{\text{opt}} \).

**Proof.** Assume \( S^a_{\text{opt}} \cap S_{\text{heu}} = \phi \). If \( S_{\text{heu}} = \phi \), then \( S_{\text{heu}} \subseteq S^a_{\text{opt}} \) follows. Otherwise, we know that \( S^a_{\text{opt}} \neq \phi \) and \( \text{cost}_a(S_{\text{heu}}) < 0 \). According to the proof in Lemma 1, we know that \( \text{cost}_a(S^a_{\text{opt}} \cup S_{\text{heu}}) < \text{cost}_a(S^a_{\text{opt}}) \), thus \( S^a_{\text{opt}} \cap S_{\text{heu}} \neq \phi \).

Suppose \( S^a_{\text{opt}} \cap S_{\text{heu}} = S_n, S_n \neq \phi \), let

\[
S_{\text{heu}} - S_n = S_1, S_1 \neq S_2.
\]

We show that \( \text{cost}_a(S^a_{\text{opt}} \cup S_1) < \text{cost}_a(S^a_{\text{opt}}) \). Let \( u(X, S') = |W(S)| \), where

\[
(S \cap (D_{MC} \cup S') = \phi \wedge (S \cap X \neq \phi).
\]

Let \( r(X, S') = |Q(S)| \), where

\[
(S \subseteq (X \cup D_{MC} \cup S') \wedge (S \cap X \neq \phi).
\]

\[
\text{cost}_a(S^a_{\text{opt}} \cup S_1) - \text{cost}_a(S^a_{\text{opt}}) = \text{cost}_a(S_1, S^a_{\text{opt}} \cup D_{MC}) = u(S_1, S^a_{\text{opt}}) - r(S_1, S^a_{\text{opt}}).
\]

(7)

Since \( S_n \subseteq S^a_{\text{opt}} \), we have (7) which is

\[
\leq u(S_1, S_n) - r(S_1, S_n) = u(S_1, S_{\text{heu}} - S_1) - r(S_1, S_{\text{heu}} - S_1).
\]

Following the proof of Lemma 2, we can easily get

\[
u(S_1, S_{\text{heu}} - S_1) - r(S_1, S_{\text{heu}} - S_1) < 0.\]

So, \( \text{cost}_a(S_{\text{heu}} \cup S^n_{\text{opt}}) < \text{cost}_a(S^n_{\text{opt}}) \) follows.

Suppose \( S^n_{\text{opt}} \subseteq S_{\text{heu}} \) then according to Lemma 2, such \( S^n_{\text{opt}} \) is impossible because

\[
\text{cost}_a(S_{\text{heu}}) < \text{cost}_a(S^n_{\text{opt}}).
\]

\( \square \)
Theorem 6. Let $S_{opt}^a$ denote the optimal set obtained by the Optimal Partial Replica Allocation (OPRA) algorithm. Let $S_{heu}^a$ denote the data set computed by the Set-Shrinking algorithm. Then $S_{heu}^a \supseteq S_{opt}^a$.

Proof. It can be shown in a way similar to Theorem 5. \hfill \Box

6.3 Heuristic Replica Deallocation Algorithm

Replica allocation and replica deallocation are similar to each other except for the cost models. We can develop replica heuristic deallocation algorithm in a similar way as heuristic replica allocation algorithm. To get a subset of the optimal set of data objects to be deallocated from $P_{MC}$, we can develop an algorithm similar to the Set-Expansion algorithm. At each step, the deallocation algorithm tries to find a minimal set of data objects $S$ recorded in $L_{MC}'$ such that $\text{cost}_d(S_{heu} \cup S) < \text{cost}_d(S_{heu})$, and then expands the set $S_{heu}$ to $S_{heu} \cup S$. Note that $S$ is a data set recorded in $L_{MC}'$ and $|S - S_{heu}|$ is minimal among all data sets remaining in $L_{MC}'$. After testing $S$, the record with data set $S$ is deleted from $L_{MC}$. The deallocation process repeats this until $L_{MC}'$ is empty, and $S_{heu}$ is the data set to be deallocated from $P_{MC}$.

To get a superset of the optimal set of data objects to be deallocated from $P_{MC}$, we can develop an algorithm similar to the Set-Shrinking algorithm. The deallocation algorithm assumes that all data objects in $D_{MC}$ should be deallocated from $P_{MC}$. Initially, $S_{heu}$ is set to $\emptyset$. Each step, the algorithm tries to find a minimal set of data objects $S$ recorded in $L_{MC}'$ such that $\text{cost}_d(D_{MC} - S - S_{heu}) < \text{cost}_d(D_{MC} - S_{heu})$,

and adds $S$ to $S_{heu}$. After testing all the data sets recorded in $L_{BC}$, we obtain $S_{heu}$ and shrink the deallocation data set from $D_{MC}$ to $D_{MC} - S_{heu}$. Since the deallocation algorithms are similar to the replica allocation algorithms, we will not give the detailed pseudocode here.

7 Frequency-Based Partial Replication

In this section, we introduce a general form of the existing frequency-based partial replication scheme. The frequency-based partial replication scheme determines whether a data object should be allocated to $P_{MC}$ or deallocated from $P_{MC}$ by analyzing its access frequency [12], [25]. The allocation cost models will be discussed based on the system model we defined in Section 2. Note that different partial replication algorithms may consider different data granularities.

We define $\text{update}_a^P(d_i)$ as the additional cost if $d_i$ is replicated to $P_{MC}$. Essentially, $\text{update}_a^P(d_i)$ is the number of update transactions accessing $d_i$ issued by node other than $P_{MC}$.

$\text{update}_a^P(d_i) = \sum |W(S)|$, where $(d_i \in S)$ and $S \subseteq D_{BC}$.

We define $\text{read}_a^P(d_i)$ as the additional benefit if $d_i$ is replicated to $P_{MC}$. Essentially, the $\text{read}_a^P(d_i)$ is the number of read transactions accessing $d_i$ issued by node $P_{MC}$.

$\text{read}_a^P(d_i) = \sum |Q(S)|$, where $d_i \in S$ and $S \subseteq D_{BC}$.

Now, we define $\text{cost}_a^P(d_i)$ as the access cost for replicating $d_i$ at $P_{MC}$. It is the difference between update cost and read benefit for replicating $d_i$ at $P_{MC}$. Also, similar to Section 4, we need to adjust the update cost to reflect the effect of update transaction grouping during disconnection period. A weight $w (w < 1)$ is applied to the update cost and the cost function is given as follows:

$$\text{cost}_a^P(d_i) = w \text{update}_a^P(d_i) - \text{read}_a^P(d_i).$$

Note that $w$ can be set to 1 when not considering disconnections and adjusted to an appropriate value as discussed in Section 4 when disconnection model is used. The frequency-based partial replication scheme computes $\text{cost}_a^P(d_i)$ for each $d_i \in D_{BC}$, and replicates $d_i$ to $P_{MC}$ when $\text{cost}_a^P(d_i) < 0$.

8 Experimental Studies

We conduct experimental studies to compare the replication algorithms based on the correlated data as well as the disconnection cost models. The expansion-shrinking (heuristic) algorithm is compared with the commonly used frequency-based partial replication (frequency) scheme to study its performance and effectiveness. We also compare the heuristic algorithm with the optimal partial replication (OPR) algorithm to see how close the heuristic solutions are to the optimal ones. Due to the exponential time complexity in OPR algorithm, we can only measure its performance with a small number of data objects (small size of $D_{BC}$).

Thus, we conduct separate experiments. We first compare the heuristic and frequency-based algorithms with a reasonable data set size. Then, we compare the heuristic and OPR algorithms with a very small data set size.

We also conduct experimental studies to evaluate the algorithms under the consideration of disconnections. To simulate the disconnection effect, we generate transactions and group multiple update transactions into one message for propagation. For both the heuristic algorithm and frequency-based algorithm, a weight is added to adjust the update cost (as discussed in Section 4). In this experiment, we set the average number of update transactions in a group (to be propagated in one message) to 2.0. The weight is set to be the same as the average number of update transactions in one group. We evaluate the effect of the weighted approach by comparing it with the original algorithms (by setting weight = 1.0).

The metric we consider is the message saving ratio. Let $M_B$ denote the number of messages required when a specific replication algorithm is used and $M_{NR}$ denote the number of messages required when there is no replication at $P_{MC}$. The message saving ratio is $(M_{NR} - M_B)/M_{NR}$, and we present it in percentage. Note that the total number of messages in the mobile system with no replication is actually the number of read access transactions issued by the mobile node. To compare the heuristic and the OPR algorithms, we compute the cost ratio, which is the ratio of the number of messages saved by the heuristic algorithm compared to the number of messages saved by the OPR.

8.1 Transaction Generation

We assume that the data objects accessed by most of the transactions follow some patterns. Also, due to access locality, we assume that the patterns will be stable for some
time periods. In our experimental study, we first generate a series of transactions and analyze the data objects they access. The set of data objects accessed by a transaction is defined as a data group. We generate transactions until $M$ distinct data groups are identified. Note that the data groups may have overlapping data objects but no two data groups are exactly the same.

To generate a transaction, the number of data objects to be accessed by the transaction, denoted as $\mu$, is first determined. $\mu$ is bounded by $T_S$ and follows a Zipf distribution. Specifically, the probability of a transaction with data set size $\mu$ is proportional to $1/\mu^Z_S$, where $Z_S$ is the skew parameter of the Zipf distribution [26]. Next, the $\mu$ data objects are determined. Let $N_S$ denote the total number of data objects we consider in the experimental study. All data objects are ranked. The probability that a data object is accessed by a transaction is proportional to $rZ_D$, where $Z_D$ is the skew parameter. Finally, whether a transaction is read only or update is determined by the ratio, $R/W$, in a uniform distribution, where $R$ is the total number of read transactions issued by $P_{MC}$ and $W$ is the total number of update transactions issued by nodes other than $P_{MC}$.

From the first batch of transactions generated, we obtain $M$ data groups and use them as the basis to formulate an access distribution. The $M$ data groups are divided into two clusters, including the read cluster with $M_1$ data groups for the read transactions and the write cluster with $M_2$ data groups for the update transactions, where $M_1 + M_2 = M$ and $M_1/M_2 = R/W$. For each cluster, 0.5$M_1$ (or 0.5$M_2$) virtual data groups are added and the pregenerated $M_1$ (or $M_2$) data groups together with the virtual data groups are ranked.

For the experimental study, we assume that in a time period $T$, $T_N$ transactions are issued. After the first $T_N$ transactions, an allocation decision is made and data objects are replicated and allocated or deallocated accordingly. Subsequently, 5,000 additional transactions are randomly generated based on the same access pattern. Performance data for the 5,000 transactions are collected. Note that for performance data collection, the first $T_N$ transactions are not considered. For each transaction, its type (read of write) is first determined based on the $R/W$ ratio. The probability distribution of the data groups the transaction accesses follows a Zipf distribution with the skew parameter $Z_f$. If the data group to be accessed by a transaction falls in the virtual data groups, then a new transaction is generated following the original transaction generation method.

To avoid having biased data access patterns, we repeat the data access pattern and experimental data collection procedures 100 times. The final data is the average of the 100 trials.

8.2 Total Number of Data Objects on Node $P_{BC}$

First, we compare the heuristic algorithm with the frequency-based algorithm and measure the impact of $N_S$ (the number of data objects on $P_{BC}$) on their performance. The parameters in this experiment are set as follows: $Z_D = 0.2$, $Z_S = 0.5$, $Z_f = 0.5$, $R/W = 1$, $T_N = 5,000$, and $T_S = 5$. $M$ and $N_S$ varies from 100 to 1,000 ($M$ is adjusted to make sure that most data objects will be accessed). Fig. 5 shows the message saving results for the two algorithms. The performance of the two algorithms remains stable when $N_S$ changes. The heuristic algorithm achieves a much higher message saving ratio than the frequency-based algorithm.

Now, we compare the heuristic algorithm with OPR under various $N_S$ values. The parameters in this experiment are set as follows: $Z_D = 0.2$, $Z_S = 0.5$, $Z_f = 0.5$, $R/W = 1.5$, $T_N = 200$, and $T_S = 3$. $N_S$ is changing from 5 to 12 ($M$ changes from 10 to 24). Fig. 6 shows that the cost ratio is around 85 percent, indicating that the heuristic algorithm achieves very good performance. Also, as can be seen, $N_S$ has little impact on the relative performance of these two algorithms.

Finally, we simulate the disconnection effect and study the performance of weighted algorithms under different $N_S$ values. The parameters in this experiment are set as follows: $Z_D = 0.2$, $Z_S = 0.5$, $Z_f = 0.5$, $R/W = 0.8$, $T_N = 5,000$, and $T_S = 5$. $M$ and $N_S$ values change from 100 to 1,000. The results in Fig. 7 show that the performance of the four algorithms varies little with the change of $M$ and $N_S$ values. The heuristic algorithm with weight = 2.0 saves about 1/3 of messages, more than double of message saving ratio of other algorithms.

8.3 Maximum Number of Data Objects in Transactions

The number of data objects in transactions ($T_S$) is expected to be a significant factor that impacts the relative performance of the algorithms. If $T_S$ is 1, then the heuristic...
Fig. 10. The disconnection effect of the four algorithms (variant $T_S$).

heuristic algorithm with weight = 2.0 has significantly better message savings than the other three algorithms.

### 8.4 Read Update Ratio

We compare the performance of the algorithms with different read/update ratio ($R/W$). First, we compare the heuristic algorithm with the frequency-based algorithm, and the parameters are set as follows: $Z_D = 0.5$, $Z_S = 0.5$, $Z_f = 0.5$, $R/W = 1$, $N_S = 5,000$, and $N_S = 200$, $M = 100$. $T_S$ changes from 1 to 10. The results are shown in Fig. 8. As $T_S$ increases, the two algorithms have degraded performance, but the difference of the message saving ratio of the two algorithms becomes larger. The heuristic algorithm always saves more messages than the frequency-based algorithm.

Next, we compare the heuristic algorithm with OPR under different $T_S$ values. The parameters in this experiment are set as follows: $Z_D = 0.2$, $Z_S = 0.5$, $Z_f = 0.5$, $R/W = 1.5$, $N_S = 10$, $T_N = 200$, and $M = 20$. $T_S$ changes from 1 to 6. We limit $T_S$ to 6 since when $T_S > 6$, the computation time for OPR becomes very high. The comparison results are shown in Fig. 9. The cost ratio decreases sharply from $T_S = 2$ to $T_S = 4$, which indicates that $T_S$ does affect the heuristic algorithm’s performance significantly. However, with $T_S > 4$, this effect becomes less significant.

Now, we simulate the disconnect effect and study the performance of weighted algorithms under different $T_S$ values. The parameters in this experiment are set as follows: $Z_D = 0.5$, $Z_S = 0.2$, $Z_f = 0.5$, $R/W = 0.8$, $N_S = 200$, $T_N = 5,000$, and $M = 100$. $T_S$ varies from 1 to 10. The results are shown in Fig. 10. It shows that $T_S$ has a great impact on the performance of the algorithms when considering disconnection. As $T_S$ value becomes higher, all these algorithms have degraded performances. However, performances degrade less with high $T_S$ value, except for the frequency-based algorithm with no weight. The
the results, we can conclude that $R/W$ is another important factor that affects the performance of the heuristic algorithm.

Finally, we simulate the disconnection impact and study the performance of weighted algorithms under different $R/W$ values. The parameters in this experiment are set as follows: $Z_D = 0.5$, $Z_S = 0.2$, $Z_f = 0.5$, $N_S = 200$, $T_S = 5$, $T_N = 5,000$, $M = 100$, and $R/W$ changes from 0.2 to 2. The results are shown in Fig. 13. Again, we can see that $R/W$ has a significant impact on the performance of the four algorithms when disconnection is considered. The performance of all the four weighted algorithms has improved performance when $R/W$ increases. The heuristic algorithm with weight $= 2.0$ is always the best.

8.5 Performance Impact by $Z_D$ and $Z_S$

We now measure the performance impact of $Z_D$, which is the skew parameter of the Zipf distribution of the access probability of data objects, and $Z_S$, which is the skew parameter of the Zipf distribution of the probability of data set size in transactions. Here, we will study the relative performance of the algorithms (with different weights) under the disconnection model.

Fig. 14 shows the performance of the algorithms under different $Z_D$ values. The parameters in this experiment are set as follows: $Z_S = 0.2$, $Z_f = 0.5$, $N_S = 200$, $T_S = 5$, $T_N = 5,000$, $R/W = 0.8$, and $M = 100$. $Z_D$ changes from 0.1 to 4. The heuristic algorithm with weight $= 2.0$ is always much better than the other three algorithms. The other three algorithms have degraded performance as $Z_D$ increases. However, as $Z_D$ becomes larger than 2, all these algorithms have stable performance.

Fig. 15 shows the performance impact on the algorithms with different $Z_S$ values. The parameters in this experiment are set as follows: $Z_D = 0.5$, $N_S = 200$, $Z_f = 0.5$, $T_S = 5$, $T_N = 5,000$, $R/W = 0.8$, and $M = 100$. $Z_S$ varies from 0.1 to 2. When $Z_S$ is less than 0.7, all four algorithms have stable performances. As $Z_S$ changes from 0.7 to 1, the performance of the four algorithms is greatly improved. With $Z_S$ increases after 1, all four algorithms have slightly improved performance. As $Z_S$ becomes larger than 0.7, it shows little performance differences among the four algorithms, mainly because that most transactions only access one data object.

8.6 Performance Impact by $Z_f$

We now measure the performance impact of $Z_f$. The parameters in this experiment are set as follows: $Z_D = 0.5$, $Z_S = 0.2$, $N_S = 200$, $T_S = 5$, $R/W = 0.8$, $T_N = 5,000$, and $M = 100$. Also, $Z_f$ changes from 0.1 to 1. Fig. 16 shows that $Z_f$ has little impact on the performance of the four algorithms. The heuristic algorithm is always better than the frequency-based algorithms. The algorithms with weight $= 2.0$ are always better than the algorithms with weight $= 1.0$.

8.7 Predefined Data Relationships

We further consider predefined data access relationships. Sometimes, access to one data object may trigger the accesses to other data objects. For example, a transaction accessing data object $A$ is always followed by a transaction accessing data objects $B$ and $C$. Based on this observation, we simulate sequences of accesses that are known a priori. First, 20 percent of the data objects in the data set are considered for the generation of the predefined data access sequences. Similar to

![Fig. 13. The disconnection effect of the four algorithms (variant $R/W$).](image)

![Fig. 14. The disconnection effect of the four algorithms (variant $Z_D$).](image)

![Fig. 15. The disconnection effect of the four algorithms (variant $Z_S$).](image)

![Fig. 16. The disconnection effect of the four algorithms (variant $Z_f$).](image)
the data group generation process, we generate 40 data group pairs (or 40 access sequences). Each data group pair consists of two nonoverlapping data groups.

These predefined access sequences are ranked and are accessed by transactions following a Zipf distribution with the skew parameter $Z_{pre} = 0.5$. During the transaction generation process, $\alpha$ percent of the $(T_N + 5,000)$ transactions access the predefined access sequences and the remaining ones are the original transactions which access the 100 data groups (as discussed in Section 8.1). Other parameters in this experiment are set as follows: $Z_D = 0.5$, $Z_S = 0.2$, $Z_I = 0.5$, $R/W = 0.8$, $N_S = 200$, $T_S = 5$, $T_N = 5,000$, and $M = 100$. The value of $\alpha$ (the predefined transaction ratio) changes from 0 to 100. The results are shown in Fig. 17. From the results, we can see that the performance of the four algorithms vary slightly with different predefined transaction ratio $\alpha$. As $\alpha$ becomes higher, the heuristic algorithm with weight 2.0 slightly improves its performance. The other three algorithms have the worst performance on message saving when 40 percent of transactions access predefined data groups. In all cases, the heuristic algorithm with weight 2.0 has much better performance than the other three algorithms.

9 CONCLUSION

In this paper, we proposed novel partial replication strategies for placement of correlated data objects with the consideration of disconnections of mobile clients. Our contributions include:

1. developing the cost models for accessing correlated data objects and for the effect of disconnection on communication cost,
2. proving that the optimal replica placement problem for correlated data object model is NP-
3. proposing the heuristic expansion-shrinking algorithm for the correlated data object model and show that it achieves near-optimal solutions, and
4. proposing the weighted heuristic algorithm to take the disconnection effect into account and to achieve better data object placement.

The experimental studies positively validate our proposed algorithms and show their significant performance gains.

We plan to consider correlated data object placement on the Internet with complex topology. Usually, mobile networks can connect to the Internet, and widely distributed systems deployed over the Internet can be extended to support mobile users. In such distributed systems with complex topology over the Internet, centralized replica placement algorithms could be too computation extensive, thus distributed algorithms may be preferred. We also plan to consider the replica placement problem of location dependent data objects [23], [7].

APPENDIX

Theorem 3. $D_{MC}^{new}$ (correlation data object model), obtained by OPRA, is a superset of $D_{MC}^{opt}$, i.e., $D_{MC}^{opt} \subseteq D_{MC}^{new}$.

Proof. Assume that $D_{MC}^{opt} \subseteq D_{MC}^{new}$ is not true. Let

$$D_{MC} = D_{MC}^{opt} - D_{MC}^{new}.$$ 

then $D_{MC}^{opt} \neq \emptyset$, and $D_{MC}^{opt} \subseteq (D_{MC}^{new} \cup D_{MC}^{opt})$. This means that the data set $D_{MC}^{opt}$ should also be replicated at $P_{MC}$, but OPRA fails to do so. Which means that

$$\text{cost}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{cost}_o(S_{opt}^{new}) \geq 0.$$ 

Since $D_{MC}^{opt}$ is the minimum optimal data set at $P_{MC}$, $D_{MC}^{opt}$ should not be deallocated. According to the benefit and cost trade-off computing for deallocation,

$$\text{cost}_d(D_{MC}^{opt}) = \text{read}_d(S_{opt}^{new}) - \text{update}_d(D_{MC}^{opt}) > 0,$$

with $D_{MC}^{opt}$ on $P_{MC}$. According to (5), $\text{read}_d(D_{MC}^{opt})$ is the total number of read transactions $t$, where $D(t) \cap D_{MC}^{opt} \neq \emptyset$ and $(t) \subseteq D_{MC}^{opt}$.

According to (4), $\text{update}_d(D_{MC}^{opt})$ is the total number of update transactions $t$ where $D(t) \subseteq D_{MC}^{opt}$.

Since $D_{MC}^{opt} \cap D_{MC}^{opt} = \emptyset$, then $D_{MC}^{opt} \cap D_{MC}^{opt} = \emptyset$. Thus, all of the read transactions $t$, where $D(t) \cap D_{MC}^{opt} \neq \emptyset$ should have been forwarded to $P_{MC}$ before receiving $S_{opt}^{new}$. Since $\text{cost}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{cost}_o(S_{opt}^{new}) = \text{update}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{update}_o(S_{opt}^{new}) - \text{read}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{read}_o(S_{opt}^{new}) < 0$, so

$$\text{cost}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{cost}_o(S_{opt}^{new}) = \text{update}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{update}_o(S_{opt}^{new}) - \text{read}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{read}_o(S_{opt}^{new}) < 0.$$ 

We know from definition that

$$\text{update}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{update}_o(S_{opt}^{new}) = \text{update}_d(D_{MC}^{opt}).$$ 

$$\text{read}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{read}_o(S_{opt}^{new})$$ is the total number of read transactions $t$, where $D(t) \cap D_{MC}^{opt} \neq \emptyset$ and

$$D(t) \subseteq (S_{opt} \cup D_{MC}^{new} \cup D_{MC}^{opt}),$$

which is $D(t) \subseteq (D_{MC}^{new} \cup D_{MC}^{opt})$. It can be seen that

$$D_{MC}^{opt} \subseteq (D_{MC}^{new} \cup D_{MC}^{opt}),$$

so we know that $\text{read}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{read}_o(S_{opt}^{new}) \geq \text{read}_d(D_{MC}^{opt})$ by their definitions. Therefore, we know that

$$\text{cost}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{cost}_o(S_{opt}^{new}) = \text{update}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{update}_o(S_{opt}^{new}) - \text{read}_o(S_{opt}^{new} \cup D_{MC}^{opt}) - \text{read}_o(S_{opt}^{new}) < 0,$$

which is a contradiction to the assumption. It follows that $D_{MC}^{opt} \subseteq D_{MC}^{new}$. $\Box$

Theorem 4. $D_{MC}^{opt} = D_{MC}^{new} - S_{opt}^{new}$. 

Fig. 17. The disconnection effect of the four algorithms with different predefined transaction ratio.
Proof. Assume that $D_{opt}^{\text{MC}} \notin D_{new}^{\text{MC}} - S_{opt}$. Consider four cases of relations between $D_{opt}^{\text{MC}}$, $D_{new}^{\text{MC}}$, and $S_{opt}^{d}$.

**Case 1.** $D_{opt}^{\text{MC}} \supset D_{new}^{\text{MC}} - S_{opt}^{d}$. According to Theorem 3, $D_{opt}^{\text{MC}} \subseteq D_{new}^{\text{MC}}$; let $S_{opt}^{d} = D_{opt}^{\text{MC}} - D_{opt}^{\text{MC}}$. According to the definition of $D_{opt}^{\text{MC}}$, $S_{opt}^{d}$ is the maximum set of data objects that could be deallocated from $D_{new}^{\text{MC}}$. Since $D_{opt}^{\text{MC}} \supset D_{new}^{\text{MC}} - S_{opt}^{d}$, so $S_{opt}^{d} \subseteq S_{opt}$. However, according to OPRD, $S_{opt}^{d}$ is the maximal set of data objects that should be deallocated from $D_{MC}$, so $S_{opt} = S_{opt}^{d}$, which is a contradiction to the assumption that $S_{opt}^{d} \subseteq S_{opt}$.

**Case 2.** $D_{opt}^{\text{MC}} \subset D_{new}^{\text{MC}} - S_{opt}^{d}$. The proof is similar to that in Case 1.

**Case 3.** $D_{opt}^{\text{MC}} - (D_{new}^{\text{MC}} - S_{opt}^{d}) \neq \phi$ and $D_{new}^{\text{MC}} - S_{opt} \notin D_{opt}^{\text{MC}}$.

Let $S_{opt}^{d} = D_{new}^{\text{MC}} - D_{opt}^{\text{MC}}$. We have $S_{opt}^{d} \neq S_{opt}^{d} \neq \phi$. Let $S_{opt}^{d} = S_{opt}^{d} \cap S_{opt}^{d}$, $d_{u} = S_{opt}^{d} - S_{opt}^{d}$, and

$$d_{u} = S_{opt}^{d} - S_{opt}^{d}.$$

We have $d_{u} \neq \phi$, $d_{u}^{d} \neq \phi$, and $d_{u} \cap d_{u}^{d} = \phi$.

We know that $\text{cost}_{d}(d_{u} \cup S_{opt}^{d}) - \text{cost}_{d}(S_{opt}^{d}) \leq 0$, so

$$\text{read}_{d}(d_{u} \cup S_{opt}^{d}) - \text{update}_{d}(d_{u} \cup S_{opt}^{d}) \leq \text{read}_{d}(S_{opt}^{d}) - \text{update}_{d}(S_{opt}^{d}),$$

which is

$$\text{read}_{d}(d_{u} \cup S_{opt}^{d}) - \text{read}_{d}(S_{opt}^{d}) \leq \text{update}_{d}(d_{u} \cup S_{opt}^{d}) - \text{update}_{d}(S_{opt}^{d}).$$

We also know that

$$\text{cost}_{d}(d_{u} \cup S_{opt}^{d} \cup d_{u}^{d}) - \text{cost}_{d}(S_{opt}^{d} \cup d_{u}^{d}) = \text{read}_{d}(d_{u} \cup S_{opt}^{d} \cup d_{u}^{d}) - \text{read}_{d}(S_{opt}^{d} \cup d_{u}^{d}) - (\text{update}_{d}(d_{u} \cup S_{opt}^{d} \cup d_{u}^{d}) - \text{update}_{d}(S_{opt}^{d} \cup d_{u}^{d})).$$

According to (4), (5), and (6)

$$\text{read}_{d}(d_{u} \cup S_{opt}^{d} \cup d_{u}^{d}) - \text{read}_{d}(S_{opt}^{d} \cup d_{u}^{d}) \leq \text{read}_{d}(d_{u} \cup S_{opt}^{d}) - \text{read}_{d}(S_{opt}^{d}),$$

and

$$\text{update}_{d}(d_{u} \cup S_{opt}^{d} \cup d_{u}^{d}) - \text{update}_{d}(S_{opt}^{d} \cup d_{u}^{d}) \geq \text{update}_{d}(d_{u} \cup S_{opt}^{d}) - \text{update}_{d}(S_{opt}^{d}).$$

It is obvious that

$$\text{cost}_{d}(d_{u} \cup S_{opt}^{d} \cup d_{u}^{d}) - \text{cost}_{d}(S_{opt}^{d} \cup d_{u}^{d}) \leq 0.$$

In other words,

$$\text{cost}_{d}(d_{u} \cup S_{opt}^{d} \cup d_{u}^{d}) \leq \text{cost}_{d}(S_{opt}^{d} \cup d_{u}^{d}).$$

This means that $S_{opt}^{d}$ is not the maximal set of data objects that should be deallocated from $D_{MC}$, which is a contradiction to the definition of $S_{opt}^{d}$.

**Case 4.** Otherwise. This can be proven in a similar way as Case 3.

Based on the four cases above, it follows that $D_{opt}^{\text{MC}} = D_{new}^{\text{MC}} - S_{opt}^{d}$.

**Lemma 1.** Let $\delta$ denote the minimal-sized data set chosen by the Set-Expansion algorithm for testing. If $\delta$ is added to $S_{heur}$, then $\text{cost}_{a}(S_{heur}) < \text{cost}_{a}(S_{heur} - \delta)$.

**Proof.** Let $S_{heur}^{d}$ denote the data set computed by the Set-Expansion algorithm when $\delta$ is tested. According to the Set-Expansion algorithm, we only need to show that if $\text{cost}_{a}(\delta) < 0$ and $\delta \cap S_{heur}^{d} = \phi$, then

$$\text{cost}_{a}(S_{heur}) < \text{cost}_{a}(S_{heur} - \delta).$$

We first show that

$$\text{read}_{a}(\delta) + \text{read}_{a}(S_{heur}^{d}) \leq \text{read}_{a}(S_{heur} \cup \delta).$$

According to (1), $\text{read}_{a}(\delta) + \text{read}_{a}(S_{heur}^{d}) = \Sigma\{Q(S)\}$, where $(S \subseteq (\delta \cup D_{MC})) \wedge (S \cap \delta \neq \phi) + \Sigma\{Q(S)\}$, where $(S \subseteq (S_{heur} \cup D_{MC})) \wedge (S \cap S_{heur}^{d} \neq \phi) = \Sigma\{Q(S)\}$, where $(S \subseteq (S_{heur}^{d} \cup \delta \cup D_{MC})) \wedge (S \cap S_{heur}^{d} \neq \phi) = \Sigma\{Q(S)\}$, where

$$(S \subseteq (S_{heur}^{d} \cup \delta \cup D_{MC})) \wedge (S \cap \delta \neq \phi) \wedge (S \cap S_{heur}^{d} \neq \phi) \leq \Sigma\{Q(S)\},$$

where

$$(S \subseteq (S_{heur}^{d} \cup \delta \cup D_{MC})) \wedge (S \cap S_{heur}^{d} \neq \phi) = \text{read}_{a}(S_{heur} \cup \delta).$$

Then, we show

$$\text{update}_{a}(\delta \cup S_{heur}^{d}) \leq \text{update}_{a}(S_{heur}^{d}) + \text{update}_{a}(\delta).$$

According to (2), $\text{update}_{a}(\delta \cup S_{heur}^{d}) = \Sigma\{W(S)\}$, where $(S \cap \delta \neq \phi) \wedge (S \cap D_{MC} = \phi) = \Sigma\{W(S)\}$, where $(S \cap \delta \neq \phi) \wedge (S \cap D_{MC} = \phi) = \Sigma\{W(S)\}$, where

$$(S \cap S_{heur}^{d} \neq \phi) \wedge (S \cap D_{MC} = \phi) = \Sigma\{W(S)\},$$

where

$$(S \cap \delta \neq \phi) \wedge (S \cap S_{heur}^{d} \neq \phi) \wedge (S \cap D_{MC} = \phi) \neq \Sigma\{W(S)\},$$

where

$$(S \cap \delta \neq \phi) \wedge (S \cap S_{heur}^{d} \neq \phi) \wedge (S \cap D_{MC} = \phi) \neq \Sigma\{W(S)\},$$

where

$$(S \subseteq (S_{heur}^{d} \cup \delta \cup D_{MC})) \wedge (S \cap \delta \neq \phi) \wedge (S \cap S_{heur}^{d} \neq \phi) \neq \Sigma\{Q(S)\},$$

where

$$(S \subseteq (S_{heur}^{d} \cup \delta \cup D_{MC})) \wedge (S \cap \delta \neq \phi) \wedge (S \cap D_{MC} = \phi) \neq \Sigma\{Q(S)\},$$

where

$$(S \subseteq (S_{heur}^{d} \cup \delta \cup D_{MC})) \wedge (S \cap \delta \neq \phi) \wedge (S \cap S_{heur}^{d} \neq \phi) \neq \Sigma\{Q(S)\}.$$

According to (3),

$$\text{cost}_{a}(S_{heur} \cup \delta) = \text{update}_{a}(S_{heur}^{d} \cup \delta) - \text{read}_{a}(S_{heur} \cup \delta) = \Sigma\{W(S)\},$$

where

$$(S \cap \delta \neq \phi) \wedge (S \cap S_{heur}^{d} \neq \phi) \wedge (S \cap D_{MC} = \phi) \neq \Sigma\{W(S)\},$$

where

$$(S \subseteq (S_{heur}^{d} \cup \delta \cup D_{MC})) \wedge (S \cap \delta \neq \phi) \wedge (S \cap S_{heur}^{d} \neq \phi) \neq \Sigma\{Q(S)\}.$$
Since \( \text{cost}_a(\delta) < 0 \), then we get
\[
\text{cost}_a(S'_{\text{heu}} \cup \delta) < \text{cost}_a(S'_{\text{heu}})
\]
\( \Box \)

**Lemma 2.** Let \( \delta_k \) be an arbitrary set of data, such that \( \delta_k \subseteq S_{\text{heu}} \land \delta_k \neq \phi \). Then, \( \text{cost}_a(S'_{\text{heu}}) < \text{cost}_a(S'_{\text{heu}} - \delta_k) \), where \( S_{\text{heu}} \) is the data set obtained by the Set-Expansion algorithm.

**Proof.** We first change the notation of \( \text{cost}_a(\delta) \) to \( \text{cost}_a(\delta, D_{\text{MC}}) \) to explicitly specify the data items that have allocated to the node \( P_{\text{MC}} \). Now, assume \( \delta_k \) was introduced into \( S_{\text{heu}} \) by the following transaction-data sets of transactions \( \{S_1, S_2, \ldots, S_m\} \) after log analysis. Let \( S'_{\text{heu}}(i) \) denote the current near optimal data set computed by the Set-Expansion algorithm just before \( S_i \) join in. We partition \( S_i \) into three parts: \( S_i = X_i \cup Y_i \cup Z_i \), where \( Z_i = S_i \cap S'_{\text{heu}}(i) \), \( X_i = (S_i - Z_i) \cap \delta_k \), \( Y_i = S_i - Z_i - X_i \).

We know that \( Y_i \cap \delta_k = \phi \), \( Z_i \subseteq S'_{\text{heu}}(i) \),

and \( (X_i \cup Y_i) \subseteq S'_{\text{heu}}(i + 1) \) when \( i \leq m \). From Lemma 1 and the Set-Expansion algorithm, each \( S_i \) joins in the \( S_{\text{heu}} \) with \( \text{cost}_a(S'_{\text{heu}}(i), D_{\text{MC}}) > \text{cost}_a(S'_{\text{heu}}(i) \cup S_i, D_{\text{MC}}) \).

From the cost function and the fact \( Y_i \subseteq S_i \), we can easily conclude that
\[
\text{cost}_a(S'_{\text{heu}}(i), D_{\text{MC}}) \leq \text{cost}_a(S'_{\text{heu}}(i) \cup Y_i, D_{\text{MC}}).
\]

Otherwise, \( Y_i \) already joins in \( S'_{\text{heu}}(i) \). Note that the Set-Expansion algorithm always tries to analyze a transaction-data set with smallest size, after comparison with \( S'_{\text{heu}} \) and \( D_{\text{MC}} \).

Let \( u(X_i, S'_{\text{heu}}(i)) = \Sigma|W(S)| \), where
\[
(S \cap (D_{\text{MC}} \cup S'_{\text{heu}}(i)) = \phi) \land (S \cap X_i = \phi).
\]

Let \( r(X_i, S'_{\text{heu}}(i)) = \Sigma|Q(S)| \), where
\[
(S \subseteq (X_i \cup D_{\text{MC}} \cup S'_{\text{heu}}(i))) \land (S \cap X_i = \phi).
\]

Let \( u'(X_i \cup Y_i, S'_{\text{heu}}(i)) = \Sigma|W(S)| \), where
\[
(S \cap (D_{\text{MC}} \cup S'_{\text{heu}}(i)) = \phi) \land (S \cap X_i = \phi) \land (S \cap Y_i = \phi).
\]

Let \( r'(X_i \cup Y_i, S'_{\text{heu}}(i)) = \Sigma|Q(S)| \), where
\[
(S \subseteq (X_i \cup Y_i \cup D_{\text{MC}} \cup S'_{\text{heu}}(i)))
\]
\[
\land (S \cap X_i = \phi) \land (S \cap Y_i = \phi).
\]

We know from the Set-Expansion algorithm
\[
\text{cost}_a(S'_{\text{heu}}(i), D_{\text{MC}}) > \text{cost}_a(S'_{\text{heu}}(i) \cup S_i, D_{\text{MC}}) \Rightarrow
\]
\[
\text{cost}_a(S'_{\text{heu}}(i) \cup S_i, D_{\text{MC}}) - \text{cost}_a(S'_{\text{heu}}(i), D_{\text{MC}}) =
\]
\[
\text{cost}_a((S'_{\text{heu}}(i) \cup S_i) - S'_{\text{heu}}(i), S'_{\text{heu}}(i) \cup D_{\text{MC}}) =
\]
\[
\text{cost}_a(X_i \cup Y_i, S'_{\text{heu}}(i) \cup D_{\text{MC}}) < 0.
\]

Since \( \text{cost}_a(S'_{\text{heu}}(i), D_{\text{MC}}) \leq \text{cost}_a(S'_{\text{heu}}(i) \cup Y_i, D_{\text{MC}}) \) and
\[
\text{cost}_a(Y_i, D_{\text{MC}} \cup S'_{\text{heu}}(i)) = \text{cost}_a(S'_{\text{heu}}(i) \cup Y_i, D_{\text{MC}})
\]
\[
- \text{cost}_a(S'_{\text{heu}}(i), D_{\text{MC}}),
\]
then
\[
\text{cost}_a(Y_i, D_{\text{MC}} \cup S'_{\text{heu}}(i)) = u(Y_i, S'_{\text{heu}}(i)) - r(Y_i, S'_{\text{heu}}(i)) \geq 0.
\]

We can find that
\[
r(X_i, S'_{\text{heu}}(i)) + r(Y_i, S'_{\text{heu}}(i)) + r'(X_i \oplus Y_i, S'_{\text{heu}}(i)) =
\]
\[
r(X_i \cup Y_i, S'_{\text{heu}}(i)),
\]

and
\[
u(X_i, S'_{\text{heu}}(i)) + u(Y_i, S'_{\text{heu}}(i)) - u'(X_i \oplus Y_i, S'_{\text{heu}}(i)) =
\]
\[
u(X_i \cup Y_i, S'_{\text{heu}}(i)).
\]

Then,
\[
\text{cost}_a(X_i \cup Y_i, S'_{\text{heu}}(i) \cup D_{\text{MC}}) - \text{cost}_a(Y_i, D_{\text{MC}} \cup S'_{\text{heu}}(i)) \leq
\]
\[
0 \Rightarrow u(X_i \cup Y_i, S'_{\text{heu}}(i)) - r(X_i \cup Y_i, S'_{\text{heu}}(i)) - (u(Y_i, S'_{\text{heu}}(i)) - r(Y_i, S'_{\text{heu}}(i))) =
\]
\[
u(X_i, S'_{\text{heu}}(i)) - u'(X_i \oplus Y_i, S'_{\text{heu}}(i)) + (r(X_i, S'_{\text{heu}}(i)) + r'(X_i \oplus Y_i, S'_{\text{heu}}(i))) \leq 0.
\]

From the definition of \( u(X_i, S'_{\text{heu}}(i)) \) and
\[
u(X_i \oplus Y_i, S'_{\text{heu}}(i)),
\]
we can get
\[
u(\delta_k, S_{\text{heu}} - \delta_k) = \Sigma|W(S)|,
\]
where
\[
(S \cap (D_{\text{MC}} \cup (S_{\text{heu}} - \delta_k)) = \phi) \land (S \cap \delta_k = \phi) =
\]
\[
u(X_1, S_{\text{heu}} - \delta_k) + \nu(X_2, S_{\text{heu}} - \delta_k) - \nu'(X_2 \oplus X_1, S_{\text{heu}} - \delta_k) +
\]
\[
u(X_3, S_{\text{heu}} - \delta_k) - \nu'(X_3 \oplus (X_1 \cup X_2), S_{\text{heu}} - \delta_k) + \ldots +
\]
\[
u(X_m, S_{\text{heu}} - \delta_k) - \nu'(X_m \oplus (X_1 \cup X_2 \cup \ldots \cup X_{m-1}), S_{\text{heu}} - \delta_k) =
\]
\[
u(X_1, S_{\text{heu}} - \delta_k) + \nu(X_2, S_{\text{heu}} - \delta_k) - \nu'(X_2 \oplus X_1, S_{\text{heu}} - \delta_k) +
\]
\[
u(X_3, S_{\text{heu}} - \delta_k) - (\delta_k - (X_1 \cup X_2 \cup X_3 \cup \ldots \cup X_{m-1}))).
\]

Since
\[
S'_{\text{heu}}(i) \cup Y_i \subseteq S_{\text{heu}} - \delta_k - (X_1 \cup X_2 \cup X_3 \cup \ldots \cup X_{i-1}),
\]
then we have
\[
(8 \leq \nu(X_1, S'_{\text{heu}}(1) \cup Y_1) + \nu(X_2, S'_{\text{heu}}(2) \cup Y_2) +
\]
\[
\nu(X_3, S'_{\text{heu}}(3) \cup Y_3) + \ldots + \nu(X_m, S'_{\text{heu}}(m) \cup Y_m) =
\]
\[
u(X_1, S_{\text{heu}}(1)) - \nu'(X_1 \oplus Y_1, S_{\text{heu}}(1)) + \nu(X_2, S_{\text{heu}}(2)) -
\]
\[
\nu'(X_2 \oplus Y_2, S_{\text{heu}}(2)) + \ldots + \nu(X_m, S_{\text{heu}}(m)) -
\]
\[
\nu'(X_m \oplus Y_m, S_{\text{heu}}(m)).
\]
Therefore, we have
\[
\begin{align*}
&u(\delta_k, S_{\text{heu}} - \delta_k) \leq u(X_1, S'_{\text{heu}}(1)) - u'(X_1 \oplus Y_1, S'_{\text{heu}}(1)) + u(X_2, S'_{\text{heu}}(2)) - u'(X_2 \oplus Y_2, S'_{\text{heu}}(2)) + \ldots + u(X_m, S'_{\text{heu}}(m)) - u'(X_m \oplus Y_m, S'_{\text{heu}}(m)).
\end{align*}
\]

(9)

From the definition of \(r(X_1, S'_{\text{heu}}(i))\) and
\[
\begin{align*}
r'(X_1 \oplus Y_1, S'_{\text{heu}}(i)),
\end{align*}
\]
we can get \(r(\delta_k, S_{\text{heu}} - \delta_k) = \Sigma |Q(S)|\), where
\[
(S \subseteq (D_{\text{MC}} \cup S_{\text{heu}})) \land (S \cap \delta_k \neq \emptyset) = r(X_1, S_{\text{heu}} - \delta_k) + r(X_2, S_{\text{heu}} - \delta_k) + r'(X_2 \oplus X_1, S_{\text{heu}} - \delta_k) + r(X_3, S_{\text{heu}} - \delta_k) + \ldots + r(X_m, S_{\text{heu}} - \delta_k) + r'(X_3 \oplus (X_1 \cup X_2, S_{\text{heu}} - \delta_k) + \ldots + r'(X_m \oplus (X_1 \cup X_2 \cup \ldots \cup X_{m-1}), S_{\text{heu}} - \delta_k) =
\]
\[
\begin{align*}
r(X_1, S_{\text{heu}} - \delta_k) + r(X_2, S_{\text{heu}} - (\delta_k - X_1)) + \ldots + r(X_m, S_{\text{heu}} - (\delta_k - (X_1 \cup X_2 \cup X_3 \cup \ldots \cup X_{m-1}))).
\end{align*}
\]
(10)

Since
\[
S_{\text{heu}}(i) \cup Y_i \subseteq S_{\text{heu}} - (\delta_k - (X_1 \cup X_2 \cup X_3 \cup \ldots \cup X_{i-1})),
\]
then we have
\[
(10) \geq r(X_1, S'_{\text{heu}}(1) \cup Y_1) + r(X_2, S'_{\text{heu}}(2) \cup Y_2) + \ldots + r'(X_3, S'_{\text{heu}}(3) \cup Y_3) + \ldots + r'(X_m, S'_{\text{heu}}(m) \cup Y_m) = r(X_1, S'_{\text{heu}}(1)) + r'(X_1 \oplus Y_1, S'_{\text{heu}}(1)) + r(X_2, S'_{\text{heu}}(2)) + \ldots + r'(X_m, S'_{\text{heu}}(m)) + r'(X_m \oplus Y_m, S'_{\text{heu}}(m)).
\]

Therefore, we have
\[
r(\delta_k, S_{\text{heu}} - \delta_k) \geq r(X_1, S'_{\text{heu}}(1)) + r'(X_1 \oplus Y_1, S'_{\text{heu}}(1)) + r(X_2, S'_{\text{heu}}(2)) + \ldots + r'(X_m, S'_{\text{heu}}(m)) + r'(X_m \oplus Y_m, S'_{\text{heu}}(m)).
\]
(11)

Since
\[
r(X_1, S'_{\text{heu}}(i)) + r'(X_1 \oplus Y_1, S'_{\text{heu}}(i)) \geq u(X_1, S'_{\text{heu}}(i)) - u'(X_1 \oplus Y_1, S'_{\text{heu}}(i))
\]
for each \(X_i\) and \(Y_i\). For some \(X_i\) and \(Y_i\),
\[
r(X_i, S'_{\text{heu}}(i)) + r'(X_i \oplus Y_i, S'_{\text{heu}}(i)) > u(X_i, S'_{\text{heu}}(i)) - u'(X_i \oplus Y_i, S'_{\text{heu}}(i))
\]
since \(X_i\) is not always an empty set with \(\delta_k \neq \emptyset\). Then, from (2) and (4), we can easily deduce that
\[
u(\delta_k, S_{\text{heu}} - \delta_k) < r(\delta_k, S_{\text{heu}} - \delta_k).
\]

We know that
\[
r(\delta_k, S_{\text{heu}} - \delta_k) + \text{read}_a(S_{\text{heu}} - \delta_k) = \text{read}_a(S_{\text{heu}})
\]
and
\[
u(\delta_k, S_{\text{heu}} - \delta_k) + \text{update}_a(S_{\text{heu}} - \delta_k) = \text{update}_a(S_{\text{heu}}).
\]
It follows that
\[
\text{cost}_a(S_{\text{heu}}, D_{\text{MC}}) < \text{cost}_a(S_{\text{heu}} - \delta_k, D_{\text{MC}}).
\]
\[\square\]

\[\text{REFERENCES}\]


Manghui Tu received the BS degree from Wuhan University, P.R. China, in 1996 and the MS degree in computer science from the University of Texas at Dallas in 2001. He is currently working toward the PhD degree in the Department of Computer Science at the University of Texas at Dallas. His research interests include distributed systems, grid computing, information security, mobile computing, and scientific computing. His PhD research work focuses on data management in secure and high-performance data grid. He is a student member of the IEEE.

Peng Li received the BSc and MS degrees in computer science from the Renmin University of China, and the PhD degree in computer science from the University of Texas at Dallas. He is the chief software architect of Didiom LLC. Before that, he was a visiting assistant professor of the Computer Science Department in Western Kentucky University. His research interests include database systems, database security, transaction processing, distributed computing, and E-commerce.

Liangliang Xiao received the BS degree in mathematics from Suzhou University in China and the PhD degree in mathematics from the Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences. He is a research associate in the Department of Computer Science at the University of Texas at Dallas. His research interests include algebra, number theory, and scientific databases.

I-Ling Yen received the BS degree from Tsing-Hua University, Taiwan, and the MS and PhD degrees in computer science from the University of Houston. She is currently an associate professor of computer science at the University of Texas at Dallas. Her research interests include fault-tolerant computing, security systems and algorithms, distributed systems, Internet technologies, E-commerce, and self-stabilizing systems. She has published more than 100 technical papers in these areas and received many research awards from US NSF, DOD, NASA, and several industry companies. She has served as a program committee member for many conferences and program chair/cochair for the IEEE Symposium on Application-Specific Software and System Engineering and Technology, the IEEE High Assurance Systems Engineering Symposium, the IEEE International Computer Software and Applications Conference, and the IEEE International Symposium on Autonomous Decentralized Systems. She is a member of the IEEE.

Farokh B. Bastani received the BTech degree in electrical engineering from the Indian Institute of Technology, Bombay, and the MS and PhD degrees in computer science from the University of California, Berkeley. He is currently a professor of computer science at the University of Texas at Dallas. His research interests are in the areas of relational programs, high-assurance hardware/software systems engineering, hardware/software reliability assessment, self-stabilizing systems, inherent fault tolerance, and high-performance modular parallel programs. Dr. Bastani was the editor-in-chief of the *IEEE Transactions on Knowledge and Data Engineering* and is on the editorial boards of the International Journal of Artificial Intelligence Tools, the Journal of Knowledge and Information Systems (KAIS), and the Springer-Verlag book series on Knowledge and Information Management (KAIM). He was the program cochair of the 1997 Symposium on Reliable Distributed Systems and the 1998 International Symposium on Software Reliability Engineering, the program vice-chair of the 1999 Symposium on Autonomous Decentralized Systems, and the program chair for the 1995 International Conference on Tools with Artificial Intelligence. He has been on the program committees of numerous conferences and workshops and on the steering committee of the IEEE Symposium on High-Assurance Systems Engineering, the IEEE Symposium on Application-Specific Software Engineering and Technology, and the IEEE Knowledge and Data Engineering Workshop. He was on the editorial board of the *IEEE Transactions on Software Engineering* and the High Integrity Systems Journal. He was a guest editor of the April 1992 special issue of the *IEEE Transactions on Knowledge and Data Engineering* devoted to self-organizing systems and the December 1985, January 1986, and November 1993 special issues of the *IEEE Transactions on Software Engineering* devoted to software reliability. He is a guest editor of a special issue of the *International Journal of Software Engineering and Knowledge Engineering* (JSEKE) devoted to embedded software systems. He is a member of ACM and the IEEE.

> For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).