Message optimal fully decentralised evaluation of associative and commutative functions

S.-Y. Yuan

Abstract: Decentralised protocols can be characterised by successive rounds of message interchanges. In this article, we show that at least \( kN(N^{k-1}) - 1 \) messages are required for fully decentralised evaluating functions that are both associative and commutative if \( k \) rounds of message interchanges are used in an \( N \)-node system. We then present a family of fully decentralised algorithms that requires, at most, a total of \( kN(N^{k-1}) - 1 \) messages to be sent with \( k \) rounds of message interchanges. Therefore, the family of algorithms is optimal with respect to the total number of messages exchanged among the processing nodes. The problems which can be modelled as an evaluation of associative and commutative functions include extrema findings and distributed transaction commitments.

1 Introduction

Distributed systems consist of a collection of processing nodes connected via a communication network such that multiple processing nodes can interact to address the processing requirements of the users. The main use of such systems is in handling distributed applications/ algorithms which require that parts of the processing be carried out at different nodes and that the nodes communicate with each other. In this paper, we assume that the communication network provides a point-to-point message based communication facility in which error-free, in-sequence delivery of messages is assured. We also assume that nodes do not share any global memory or any central clock.

In a distributed algorithm designed to achieve a global objective, each processing node has to carry out its computations and actions based on the information available to it. To achieve the global objective therefore, an algorithm has to be designed to coordinate the processes in such a way that the proper, consistent information is available to each processing node at the proper time.

An algorithm using a centralised control mechanism contains a single process, called the coordinator, which coordinates the actions of others. The two-phase commit protocol [1] is a good example of such an algorithm. In this algorithm, the coordinator sends a transaction request to all other nodes and waits for their replies in the first phase. After receiving all replies, the coordinator sends a final decision to the others in the second phase.

Decentralised algorithms do not use a single coordinator. All participants are considered to be coordinators, and they all execute an identical program. Since there is no central controller in a decentralised algorithm, the information required for a node to accomplish the global objective cannot be obtained by exchanging messages with the controller only. So how to coordinate the message flow such that every node only exchanges messages with a few other nodes to obtain all required information becomes a major issue for designing decentralised algorithms.

Although the centralised algorithms usually require fewer messages than their decentralised counterparts, the use of a single central controller for interprocess coordinations may lead to some severe problems. The central coordinator becomes the critical point of the whole system. If it fails, the whole system has to stop operation until a new coordinator is designated. Further, the failure of the coordinator may result in the loss of some critical information. To prevent this situation from happening, backup controllers are introduced to keep track of the information of the primary controller.

Another problem of the centralised algorithms is that the controller may become a performance bottleneck for a large distributed system because usually the coordinator has to exchange information with all others.

In this article, we consider the case where the global objective is to evaluate a function or predicate that is both associative and commutative. Each node possesses one of the arguments of the function initially. The computation is required to be decentralised and the result of the computation to be known by all processing nodes. Such a problem includes many decentralised applications such as extrema finding [2–7], coordination of distributed checkpoints [8], and maintenance of transaction atomicity [9–11].

One simple way to carry out this computation is by requiring each process to send its information to every other process. The relevant computation can then be performed by each node. This method requires \( N(N - 1) \) messages, where \( N \) is the number of nodes in the system, to be sent at one round of message exchange. Lakshman uses a communication structure based on the finite projective plane [12] to construct decentralised protocols with \( O(N^2) \) messages and two rounds of message exchanges.

2 Computation model and problem domain

Distributed computations can be classified into two categories: centralised and decentralised. A centralised computation contains a special node called coordinator.
In general, a fully decentralised computation can be defined in terms of rounds. In each round, nodes exchange messages and then carry out some local computations. The local computation at each node in round \(i\) is based on the information available at that node up to and including the communication phase of round \(i\). A node can proceed with its local computation of \(i\)th round whenever its communication phase of round \(i\) completes.

The problem we are considering here is to evaluate an associative and commutative function by a fully decentralised computation. A function \(F: D \times D \rightarrow D\), where \(D\) is a set of objects, is said to be associative and commutative if, and only if,

- \(a\) for all \(x, y, z \in D\), \(F(F(x, y), z) = F(x, F(y, z))\)
- \(b\) for all \(x, y, z \in D\), \(F(x, y) = F(y, x)\).

Let \(F\) be an associative and commutative function, and the \(N\) values \(V_0, V_1, \ldots, V_{N-1}\) be distributed across the \(N\) nodes in a distributed system, such that each node \(X\), \(0 < X < N - 1\) contains the value \(V_X\). Our problem is to fully decentralise evaluation of the function \(F\) over the \(N\) values and make the result known to the \(N\) nodes. Since \(F\) is an associative and commutative function, there exists a corresponding associative and commutative operator \(\Psi\) such that \(F(V_0, V_1, \ldots, V_{N-1}) = \Psi V_0 \Psi V_1 \Psi \cdots \Psi V_{N-1} \Psi\). An operator \(\Psi\) is said to be associative and commutative over a set of object \(D\), and only if

- \(a\) for all \(x, y, z \in D\), \(\Psi(x \Psi y) = (x \Psi y) \Psi z\)
- \(b\) for all \(x, y \in D\), \(x \Psi y = y \Psi x\).

Some of the well-known associative and commutative operators are addition, multiplication, logic-or, logic-and, set-union, and set-intersection.

### 3 Message complexity for fully decentralised evaluating associative and commutative functions

The following theorems describe the lower bound of message complexity for fully decentralised evaluation of associative and commutative functions in an \(N\)-node system.

---

**Lemma 3.1:** Given positive integers \(N\) and \(k\), for any \(k\) positive integers \(p_1, p_2, \ldots, p_k\) with \(\sum_{i=1}^{k} (p_i + 1) = N\), the \(\sum_{i=1}^{k} p_i\) is at least \(k\lceil N/\log_2 N \rceil - 1\).

**Proof:** Since \(N^{1/\log_2 N} \geq N^{1/\log_2 N} \geq 1\), \(\sum_{i=1}^{k} \lceil N^{1/\log_2 N} \rceil \geq \sum_{i=1}^{k} \lceil N/\log_2 N \rceil - 1\), and there exists an \(m\), \(0 < m < k\) such that \(\sum_{i=m+1}^{k} \lceil N^{1/\log_2 N} \rceil \geq \sum_{i=m+1}^{k} \lceil N/\log_2 N \rceil - 1\). We claim that the minimal value of the \(\sum_{i=1}^{k} p_i\) can be obtained by assigning \(m\) messages as \(\lceil N^{1/\log_2 N} \rceil - 1\) and the other \((k-m)p_i\) as \(\lceil N/\log_2 N \rceil - 1\). Therefore, the minimum of the \(\sum_{i=1}^{k} p_i\) is \(m\lceil N^{1/\log_2 N} \rceil - 1 + (k-m)\lceil N/\log_2 N \rceil - 1\).

To show the above claim, we first prove that there exists a set of \(k\) positive integers \(P = \{p_i\}_{i=1}^{k}\), where \(\forall p_i, p_j \in P, |p_i - p_j| \leq 1\), satisfies \(\prod_{i=1}^{k} (p_i + 1) = N\) with \(\sum_{i=1}^{k} p_i\) being minimum. Assume that \(Q = \{q_i\}_{i=1}^{k}\) satisfies \(\prod_{i=1}^{k} (q_i + 1) \geq N\) with \(\sum_{i=1}^{k} q_i\) being minimum and there exists \(q_i, q_j \in Q\) with \(q_i - q_j \geq 2\). Without loss generality, we let \(q_i \geq q_j + 1\).

From \(Q\), we can construct a new set of \(k\) positive integers \(Q' = \{q_i'\}_{i=1}^{k}\), where \(\forall \neq i, j, q_i' = q_i, q_j' = q_i - 1, q_j = q_j + 1\).

\[
\prod_{i=1}^{k} (q_i' + 1) = \prod_{i=1}^{k} (q_i + 1) - \prod_{i=1}^{k} (q_i + 1) + q_i = \prod_{i=1}^{k} (q_i + 1) - q_i q_j
\]

\[
\geq (q_i' q_j' - q_i q_j)
\]

\[
= (q_i - 1)(q_j + 1) - q_i q_j
\]

\[
= q_i - q_j - 1
\]

\[
\geq 1
\]

Therefore, the new set \(Q'\) not only satisfies \(\prod_{i=1}^{k} (q_i' + 1) \geq N\) but also maintains the property of \(\sum_{i=1}^{k} q_i\), being minimum. By applying the same procedure repeatedly, eventually we can construct a set of \(k\) positive integers \(P = \{p_i\}_{i=1}^{k}\), where \(\forall p_i, p_j \in P, |p_i - p_j| \leq 1\), satisfying \(\prod_{i=1}^{k} (p_i + 1) = N\) with \(\sum_{i=1}^{k} p_i\) being minimum.

We now show that the set \(P\) actually contains \(m\) elements of \(\lceil N/\log_2 N \rceil - 1\) and \(k - m\) elements of \(\lceil N^{1/\log_2 N} \rceil - 1\). Since, for all sets, \(Q\) satisfies \(\prod_{i=1}^{k} (q_i + 1) \geq N\) with \(\sum_{i=1}^{k} q_i\), being minimum, we can always find a corresponding set \(Q'\) satisfying \(\prod_{i=1}^{k} (q_i' + 1) \geq N\) with \(\sum_{i=1}^{k} q_i\), being minimum also such that all elements in \(Q'\) differ by,
at most, one. If the set $Q$ is not $P$, because the $\sum_{i=1}^{m} q_i$ is minimum, the maximal element in $Q$ must be less than, or equal to, the maximal element in $P$. Otherwise, since no two elements in $Q$ differ more than 1, and no two elements in $P$ differ more than 1, the minimal element in $Q$ must be greater than the minimal element in $P$ also. Thus $\sum_{i=1}^{m} q_i$ is greater than $\sum_{i=1}^{m} p_i$. This contradicts the assumption that $\sum_{i=1}^{m} q_i$ is minimum.

Without loss generality, we assume that $1 \leq i < j \leq K$, $p_j \geq p_i$ and $q_i > q_j$ where $p_i, p_j \in P$ and $q_i, q_j \in Q$. Since the maximal element in $Q$ is no greater than the maximal element in $P$, there exists at least one element in $Q$ being less than its counterpart in $P$. Let $l$ be the smallest integer between 1 and $k$ such that $\forall l < i < j \leq K, q_i = p_l$ and $q_j < p_j$. There are two cases to be considered.

Case 1: $i \leq j$: Since $p_i = [N^{1/k}] - 1$, $q_i \leq p_i - 1 = [N^{1/k}] - 2 \leq [N^{1/k}] - 1$. Thus $\forall 1 \leq i \leq k, q_i \leq [N^{1/k}] - 1$. Therefore,

$$\prod_{i=1}^{m} (q_i + 1) \leq \prod_{i=1}^{m} (p_i + 1)\prod_{i=1}^{m} (q_i + 1)$$

\[= [N^{1/k}]^{m-1} [N^{1/k}]^{j-i+1} \leq [N^{1/k}]^{m-1} [N^{1/k}]^{j-i+1} \leq < N \quad (5)\]

This is contradictory to the assumption that $\prod_{i=1}^{m} (q_i + 1) \geq N$.

Case 2: $i > j$: Since $p_j = [N^{1/k}] - 1$, $q_j \leq p_j - 1 = [N^{1/k}] - 2$. Thus, $\forall l \leq i \leq k, q_i \leq [N^{1/k}] - 2$.

$$\prod_{i=1}^{m} (q_i + 1) = \prod_{i=1}^{m} (p_i + 1)\prod_{i=1}^{m} (q_i + 1)$$

\[\leq [N^{1/k}]^{m-1} [N^{1/k}]^{j-i+1} [N^{1/k}]^{j-i+1} \leq [N^{1/k}]^{m-1} [N^{1/k}]^{j-i+1} \leq < N \quad (6)\]

This is contradictory to the assumption that $\prod_{i=1}^{m} (q_i + 1) \geq N$.

Therefore, for any set $Q$ satisfying $\prod_{i=1}^{m} (q_i + 1) \geq N$ with $\sum_{i=1}^{m} q_i$ being minimum, we can always find a corresponding set $P$ which satisfies $\prod_{i=1}^{m} (p_i + 1) \geq N$ with $\sum_{i=1}^{m} p_i$ being minimum and contains exactly $m$ elements of $[N^{1/k}] - 1$ and $k - m$ elements of $[N^{1/k}] - 1$. Therefore, the minimum of the $\sum_{i=1}^{m} p_i$ is $m([N^{1/k}] - 1) + (k - m)([N^{1/k}] - 1)$. Since

$$m([N^{1/k}] - 1) + (k - m)([N^{1/k}] - 1) \geq k([N^{1/k}] - 1) \quad (7)$$

the $\sum_{i=1}^{m} p_i$ is at least $k([N^{1/k}] - 1)$. □

**Theorem 3.1:** Any algorithm for evaluating associative and commutative functions fully decentralised in an $N$-node system with $k$ rounds of message exchanges requires at least $\kappa N^{1/k} - 1$ messages passed among the $N$ nodes.

**Proof:** Assume that, for all $i, 1 \leq i \leq k$, in the $i$th round of message exchange, each node sends out $p_i$ messages. Since in the fully decentralised computation, all nodes execute the same programme, send out the same number of messages, and receive the same number of messages in each round. Therefore, after the first round, there are at most $(p_1 + 1)$ nodes having the knowledge about the initial value of node 1. After the second round, there are at most $(p_1 + 1)(p_2 + 1)$ nodes having the knowledge about the initial value of node 1. Therefore, after the $k$th round, there are at most $\prod_{i=1}^{m} (p_i + 1)$ nodes having the knowledge about the initial value of node 1. Since only $k$ rounds of message interchanges are used, $\prod_{i=1}^{m} (p_i + 1)$ should be at least $N$. Otherwise, some nodes would have no knowledge about the initial value of node 1, so they will not be able to evaluate the function.

Since each node sends out $p_i$ messages in round $i, 1 \leq i \leq k$, the total number of messages passed among the $N$ nodes is $N \sum_{i=1}^{m} p_i$. Because $\prod_{i=1}^{m} (p_i + 1) \geq N$, and for any $k$ positive integers $p_1, p_2, \ldots, p_k, the \sum_{i=1}^{m} q_i$ is at least $k([N^{1/k}] - 1)$ (from Lemma 3.1), the total number of messages $N \sum_{i=1}^{m} p_i$ is at least $kN([N^{1/k}] - 1)$.

4 The KDAMS structure

We have shown that decentralised evaluation of associative and commutative functions require at least $kN([N^{1/k}] - 1)$ messages for $k$ rounds of message interchanges in an $N$-node systems. In this Section, we discuss a communication structure called KDAMS which can be used to develop protocols achieving the lower bound of message complexity.

In Lemma 3.1, we have shown that given positive integers $N$ and $k$, a set of $k$ positive integers $p_1, p_2, \ldots, p_k$ can be found to minimise the $\sum_{i=1}^{m} p_i$ becoming $m([N^{1/k}] - 1) + (k - m)([N^{1/k}] - 1)$ under the constraint of

$$\prod_{i=1}^{m} (p_i + 1) \geq N, \quad 0 \leq m \leq k \quad \text{and} \quad [N^{1/k}]^{m-1} [N^{1/k}]^{j-i+1} \geq N \geq [N^{1/k}]^{m-1} [N^{1/k}]^{j-i+1} \leq N \quad (8)$$

Therefore, when every node sends out $p_i$ messages at the $i$th round of message interchange and after the $i$th round, there are $\prod_{i=1}^{m} (p_i + 1)$ nodes having the knowledge of the initial value of node 1, the lower bound can be achieved.

For any given $N$ and $k$, from Lemma 3.1, we first find a set of $k$ positive integers $p_1, p_2, \ldots, p_k$ such that $\prod_{i=1}^{m} (p_i + 1) \geq N$ and $\sum_{i=1}^{m} p_i$ is the minimum. We then add $M - N$ dummy nodes into the system, where $M = \prod_{i=1}^{m} (p_i + 1)$. We consider the $M$-node system as a $k$-dimensional array of $[0 \ldots p_1, 0 \ldots p_2, \ldots, 0 \ldots p_k]$ such that every node $X, 0 \leq X \leq M - 1$ can be addressed as a $k$-tuple $(x_1, x_2, \ldots, x_k), where 1 \leq i \leq k, x_i$ is integer between 0 and $p_i$ and $X = \sum_{i=1}^{m} (x_i \prod_{j=i+1}^{m} (p_j + 1))$. Since there are $m$ nodes equal to $([N^{1/k}] - 1)$ and $(k - m)$ nodes equal to $([N^{1/k}] - 1)$, where $0 \leq m \leq k$ and $[N^{1/k}]^{m-1} [N^{1/k}]^{j-i+1} \geq N \geq [N^{1/k}]^{m-1} [N^{1/k}]^{j-i+1} \leq N \quad (9)$

Thus, the proof of Lemma 3.1, $M - N < [N^{1/k}]^{j-i+1}$, each real node only has to emulate at most one dummy node. The KDAMS communication structure is defined as follows.

4.1 KDAMS structure

At round $i, 1 \leq i \leq k$: Every node $X(x_1, x_2, \ldots, x_k)$ exchanges information with nodes addressed as $(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_k)$, where $0 \leq y_i \leq p_i$ and $y_i \neq x_i$.

In other words, the KDAMS structure is to arrange nodes into a $k$-dimensional array and at round $i, 1 \leq i \leq k$, each node exchange messages with all nodes that differ from itself only in the $i$th dimension. Thus, for
a 2D array, each node communicates with nodes in the same column at the first round, and then with nodes in the same row at the second round.

5 The message optimal algorithm

The actions of the algorithm at each site are modelled by a finite state automation (FSA). The local state transitions of FSAs involve reading/writing messages to the network and transiting to another local state. For given $N$ and $k$, from Lemma 3.1, there exists an integer $m$ such that $[N^{1/k}]^{m}[N^{1/k}]^{-m} \geq N \geq [N^{1/k}]^{m}-[N^{1/k}]^{-m}$ + 1. Let $p_1 = p_2 = \ldots = p_m = ([N^{1/k}]^{m}-1), p_{m+1} = p_{m+2} = \ldots = p_k = ([N^{1/k}]^{m}-1) - 1$, and $M = \prod_{i=1}^{k} (p_i + 1)$. Add $M - N$ dummy nodes to the system and address every node $X$, $0 \leq X < M - 1$ by a k-tuple $(x_1, x_2, \ldots, x_k)$, where $0 \leq x_i < p_i$, $1 \leq i \leq k$, $0 \leq x_i \leq p_i$, and $X = \sum_{i=1}^{k} (x_i \prod_{j=1}^{i-1} (p_j + 1))$.

Initially, each real node $0 \leq X \leq N - 1$ has the value $v_X$ and all dummy nodes are assigned a special value which cannot affect the outcome of the computation (i.e. 0 for summation function and 1 for multiplication function). In the algorithm, we use the symbol @ to represent the special value.

The FSA of the algorithm for a node $X(x_1, x_2, \ldots, x_k)$ is shown in Fig. 1. The actions of each state for the node $X(x_1, x_2, \ldots, x_k)$ are described as follows.

**Fig. 1** FSAs for computations of commutative functions

State $q$ for normal nodes: Send own value to all nodes addressed as $(y_1, y_2, \ldots, y_k)$, where $0 \leq y_i \leq p_i$, and move to state $w_i$.

State $q$ for dummy nodes: Send the special symbol @ to all nodes addressed as $(y_1, y_2, \ldots, y_k)$, where $0 \leq y_i \leq p_i$, and move to state $w_i$.

State $w_i$, $1 \leq i \leq (k - 1)$: Upon receiving values from all nodes addressed as $(x_1, x_2, \ldots, y_{i-1}, y_i, y_{i+1}, \ldots, x_k)$, where $0 \leq y_i \leq p_i$, perform the operation $\psi$ among received values (ignoring the special symbol @) and the local value, send the result to all nodes addressed as $(x_1, \ldots, x_i, y_{i+1}, y_{i+2}, \ldots, x_k)$, where $0 \leq y_{i+1} \leq p_{i+1}$, replace the local value by the new result, and move to state $w_{i+1}$.

State $w_k$: Upon receiving values from all nodes addressed as $(x_1, \ldots, x_{k-1}, x_k)$, where $0 \leq x_k \leq p_k$, perform the operation $\psi$ among received values and the local value and move to state $f$.

State $f$: Final state.

The following theorem shows the correctness of the protocol.

**Theorem 5.1** If a node is in state $f$, it contains the value of $\psi_{i=0}^{N+1} v_i$, where $v_0(i)$ is the initial value of node $i$.

**Proof:** We show it by induction.

**Inductive hypothesis:** After the $i$th round of message exchanges, where $1 \leq i \leq k$, a node $X$ addressed as $(x_1, x_2, \ldots, x_k)$ will have the value $V(X) = \psi_{i=0}^{x_1} \psi_{i=0}^{x_2} \ldots \psi_{i=0}^{x_k} \psi_{i=0}^{v_0}$, where $0 \leq y_k \leq p_k$, which have the initial values $v_0(y_1, y_2, \ldots, x_k)$. Therefore, the node $X$ will contain the value $V(X) = \psi_{i=0}^{x_1} \psi_{i=0}^{x_2} \psi_{i=0}^{x_3}$.\ldots\psi_{i=0}^{x_k}$.\ldots\psi_{i=0}^{v_0}$.

**Inductive case:** Assume for $i = i - 1$, where $i \leq k$, the inductive hypothesis is true. After the $(i - 1)$th round of message exchange, the node $X$ has received values from nodes $(y_1, x_2, \ldots, x_k)$, where $0 \leq y_1 \leq p_1$, which have the initial values $v_0(y_1, x_2, \ldots, x_k)$. Therefore, the node $X$ will contain the value $V(X) = \psi_{i=0}^{x_1} \psi_{i=0}^{x_2} \psi_{i=0}^{x_3}$.\ldots\psi_{i=0}^{x_k}$.\ldots\psi_{i=0}^{v_0}$.

Because every node sends $p_i$ messages in round $i$, the total number of messages passed among the $N$ nodes is $N \sum_{i=1}^{k} p_i$. Since every $p_i$ is either $[N^{1/k}] - 1$ or $[N^{1/k}] - 1 - 1$, and the $\sum_{i=1}^{k} p_i$ should be less than or equal to $k[N^{1/k}] - 1$, so the total number of messages $\leq k[N^{1/k}] - 1$ because the lower bound $k[N^{1/k}] - 1$ and $k[N^{1/k}] - 1$ have the same order of $\Theta(kN^{1/k})$. The algorithm is optimal with respect to the message complexity.

6 An example

Assume that we want to evaluate the summation function in a 12-node system with three rounds of message exchanges. From Lemma 3.1 and the KADMS structure, we note that $m = 1$ and $p_1 = 2, p_2 = 1, and p_3 = 1$. Therefore, each node $X$, $0 \leq X \leq 11$, can be addressed as a triple $(x_1, x_2, x_3)$, where $0 \leq x_1 \leq 2, 0 \leq x_2 \leq 1, and 0 \leq x_3 \leq 1$. Assume that nodes have initial values as follows:

- $v_0^0 = 10, v_0^1 = 31, v_0^2 = 43, v_0^3 = 50$
- $v_1^0 = 1, v_1^0 = 39, v_1^2 = 17, v_1^3 = 3$
- $v_2^0 = 28, v_2^0 = 9, v_2^0 = 20, v_2^0 = 11$
According to the KDAMS structure, in the first round, every node sends its current value to all nodes whose addresses differ itself only in the first dimension. After receiving all values, every node sums them and its current value to be its new local value. Thus, the local values after the first round are as follows:

\[
\begin{align*}
V_{00}^1 & = V_{00}^0 + V_{10}^0 + V_{20}^0 = 10 + 1 + 28 = 39 \\
V_{10}^1 & = V_{00}^0 + V_{10}^0 + V_{20}^0 = 10 + 1 + 28 = 39 \\
V_{20}^1 & = V_{00}^0 + V_{10}^0 + V_{20}^0 = 10 + 1 + 28 = 39 \\
V_{01}^1 & = V_{00}^0 + V_{11}^0 + V_{21}^0 = 31 + 39 + 9 = 79 \\
V_{11}^1 & = V_{00}^0 + V_{11}^0 + V_{21}^0 = 31 + 39 + 9 = 79 \\
V_{21}^1 & = V_{00}^0 + V_{11}^0 + V_{21}^0 = 31 + 39 + 9 = 79 \\
V_{10}^1 & = V_{01}^0 + V_{11}^0 + V_{21}^0 = 43 + 17 + 20 = 80 \\
V_{11}^1 & = V_{01}^0 + V_{11}^0 + V_{21}^0 = 43 + 17 + 20 = 80 \\
V_{21}^1 & = V_{01}^0 + V_{11}^0 + V_{21}^0 = 43 + 17 + 20 = 80 \\
V_{02}^1 & = V_{01}^0 + V_{11}^0 + V_{21}^0 = 50 + 3 + 11 = 64 \\
V_{12}^1 & = V_{01}^0 + V_{11}^0 + V_{21}^0 = 50 + 3 + 11 = 64 \\
V_{22}^1 & = V_{01}^0 + V_{11}^0 + V_{21}^0 = 50 + 3 + 11 = 64
\end{align*}
\]

In the second round, every node sends its current value to all nodes whose addresses differ itself only in the second dimension. After receiving all expected values, every node sums them and its current local value to be its new local value. The local values after second round are as follows:

\[
\begin{align*}
V_{00}^2 & = V_{00}^1 + V_{10}^1 + V_{20}^1 = 39 + 80 = 119 \\
V_{10}^2 & = V_{00}^1 + V_{10}^1 + V_{20}^1 = 39 + 80 = 119 \\
V_{20}^2 & = V_{00}^1 + V_{10}^1 + V_{20}^1 = 39 + 80 = 119 \\
V_{01}^2 & = V_{01}^1 + V_{11}^1 + V_{21}^1 = 79 + 64 + 143 = 286 \\
V_{11}^2 & = V_{01}^1 + V_{11}^1 + V_{21}^1 = 79 + 64 + 143 = 286 \\
V_{21}^2 & = V_{01}^1 + V_{11}^1 + V_{21}^1 = 79 + 64 + 143 = 286 \\
V_{10}^2 & = V_{02}^1 + V_{12}^1 + V_{22}^1 = 39 + 80 = 119 \\
V_{11}^2 & = V_{02}^1 + V_{12}^1 + V_{22}^1 = 39 + 80 = 119 \\
V_{12}^2 & = V_{02}^1 + V_{12}^1 + V_{22}^1 = 39 + 80 = 119 \\
V_{20}^2 & = V_{01}^1 + V_{11}^1 + V_{21}^1 = 79 + 64 + 143 = 286 \\
V_{21}^2 & = V_{01}^1 + V_{11}^1 + V_{21}^1 = 79 + 64 + 143 = 286 \\
V_{22}^2 & = V_{01}^1 + V_{11}^1 + V_{21}^1 = 79 + 64 + 143 = 286
\end{align*}
\]

In the third round, every node sends its current value to all nodes whose addresses differ itself only in the third dimension. After receiving all expected values, every node sums them and its current local value to be the final result. The final values are as follows:

\[
\begin{align*}
V_{00}^3 & = V_{00}^2 + V_{01}^2 = 119 + 143 = 262 \\
V_{10}^3 & = V_{00}^2 + V_{01}^2 = 119 + 143 = 262 \\
V_{20}^3 & = V_{00}^2 + V_{01}^2 = 119 + 143 = 262 \\
V_{01}^3 & = V_{00}^2 + V_{01}^2 = 119 + 143 = 262 \\
V_{11}^3 & = V_{00}^2 + V_{01}^2 = 119 + 143 = 262 \\
V_{21}^3 & = V_{00}^2 + V_{01}^2 = 119 + 143 = 262
\end{align*}
\]

7 Concluding remarks

We have shown that fully decentralised evaluating associative and commutative functions require at least \( kN([N^{1/k}] - 1) \) messages in an \( N \)-node system with \( k \) rounds of message exchanges. A family of fully decentralised algorithms for evaluating of associative and commutative function with optimal message complexity is developed by using the KDAMS communication structure. The family of algorithms are symmetric and require at most a total number of \( kN([N^{1/k}] - 1) \) messages for \( k \) rounds of message interchanges in an \( N \)-node system. This family of algorithms also permit a trade-off between the number of rounds of message exchanges and the total number of messages passed among the nodes. By associating proper operations, optimal decentralised algorithms for distributed transaction commitments, extrema finding, and computation of summation can be derived from the algorithm described in Section 5.

8 References

1 GRAY, J.N.: ‘Notes on database operating systems', in 'Operating systems: an advanced course' (Springer-Verlag, Berlin, 1979)
10 MOHAN, C., and LINDSAY, B.: ‘Efficient commit protocols for the tree of processes model of distributed transactions', Proceedings of the 2nd ACM SIGACT/SIGOPS symposium on principles of distributed computing, Montreal, Canada, 1983, pp. 76-80