Message optimal fully decentralised evaluation of associative and commutative functions

S.-Y. Yuan

Indexing terms: Associative and commutative functions, Distributed algorithms, Decentralised computations, k-dimensional array, Message complexity

Abstract: Decentralised protocols can be characterised by successive rounds of message interchanges. In this article, we show that at least $kN(\lfloor N^{1/k} \rfloor - 1)$ 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 $kN(\lceil N^{1/k} \rceil - 1)$ 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 exchange 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 $O[N\sqrt{(N)}]$ 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

Paper 1165E (C2, C3), received 27th August 1993

The author is with the Department of Computer & Information Science, National Chiao Tung University, Hsinchu, Taiwan 30050, Republic of China

which coordinates the interactions among all processing nodes. A decentralised computation has no outstanding nodes. All processing nodes are considered to be coordinators. The coordination responsibility is shared by all nodes.

In this study, we are only interested in decentralised computations. In particular, we are concentrating on the class of decentralised computations in which all processing nodes execute an identical programme, send out the same number of messages, and receiver the same number of messages. In other words, the coordination responsibility is equally distributed to all processing nodes. We say that this kind of decentralised computation is fully decentralised.

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

Round $(i-1)$	— — — — — — — — — — — —	
	Compute	Computation phase
	Exchange information	Communication phase
Round i	— — — — — — — — — — — —	
	Compute	Computation phase
	Exchange information	Communication phase
Round $(i+1)$	— — — — — — — — — — — —	

available at that node up to and including the communication phase of round i. A node can proceed with its local computation of ith 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 $x, y, z \in D$, $F[F(x, y), z] = F[x, F(y, z)]$ and
(*b*) for all $x, y \in D$, $F(x, y) = F(y, x)$.

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, $0 \leqslant X \leqslant N-1$ 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 Ψ such that $F(V_0, V_1, \ldots, V_{N-1}) = \Psi_{i=0}^{N-1} V_i$. An operator Ψ is said to be associative and commutative over a set of object D if, and only if

(*a*) for all $x, y, z \in D$, $x\Psi(y\Psi z) = (x\Psi y)\Psi z$ and
(*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 $\prod_{i=1}^{k}(p_i+1) \geqslant N$, the $\sum_{i=1}^{k} p_i$ is at least $k(\lceil N^{1/k}\rceil - 1)$.

Proof: Since $\lceil N^{1/k}\rceil^k \geqslant N \geqslant \lfloor N^{1/k}\rfloor^k$ and $\forall i$, $0 \leqslant i \leqslant k$, $\lceil N^{1/k}\rceil^i \lfloor N^{1/k}\rfloor^{k-i} \geqslant \lceil N^{1/k}\rceil^{i-1}\lfloor N^{1/k}\rfloor^{k-i+1}$, there exists an m, $0 \leqslant m \leqslant k$ such that $\lceil N^{1/k}\rceil^m \lfloor N^{1/k}\rfloor^{k-m} \geqslant N > \lceil N^{1/k}\rceil^{m-1}\lfloor N^{1/k}\rfloor^{k-m+1}$. We claim that the minimal value of the $\sum_{i=1}^{k} p_i$ can be obtained by assigning mp_is as $(\lceil N^{1/k}\rceil - 1)$ and the other $(k-m)p_i$s as $(\lfloor N^{1/k}\rfloor - 1)$. Therefore, the minimum of the $\sum_{i=1}^{k} p_i$ is $m(\lceil N^{1/k}\rceil - 1) + (k-m)(\lfloor N^{1/k}\rfloor - 1)$.

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

From Q, we can construct a new set of k positive integers $Q' = \{q'_i \mid i = 1, 2, \ldots, k\}$ where $\forall l \neq i, j$, $q'_l = q_l$, $q'_i = q_i - 1$, and $q'_j = q_j + 1$.

$$\begin{aligned}
\prod_{l=1}^{k}(q'_l+1) - \prod_{l=1}^{k}(q_l+1) &= \prod_{l=1,\, l\neq i,\, l\neq j}^{k}(q_l+1)(q'_i q'_j - q_i q_j) \\
&\geqslant (q'_i q'_j - q_i q_j) \\
&= (q_i - 1)(q_j + 1) - q_i q_j \\
&= q_i - q_j - 1 \\
&\geqslant 2 - 1 > 0 \qquad (1)
\end{aligned}$$

$$\Rightarrow \quad \prod_{l=1}^{k}(q'_l + 1) \geqslant N \qquad (2)$$

$$\begin{aligned}
\sum_{i=1}^{k} q'_i - \sum_{i=1}^{k} q_i &= q'_i + q'_j - q_i - q_j \\
&= q_i - 1 + q_j + 1 - q_i - q_j = 0 \qquad (3)
\end{aligned}$$

$$\Rightarrow \quad \sum_{i=1}^{k} q'_i = \sum_{i=1}^{k} q_i \qquad (4)$$

Therefore, the new set Q' not only satisfies $\prod_{i=1}^{k}(q'_i + 1) \geqslant N$ 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 $P = \{p_i \mid i = 1, 2, \ldots, k\}$, where $\forall p_i, p_j \in P$, $|p_i - p_j| \leqslant 1$, satisfying $\prod_{i=1}^{k}(p_i+1) \geqslant N$ with $\sum_{i=1}^{k} p_i$ being minimum.

We now show that the set P actually contains m elements of $(\lceil N^{1/k}\rceil - 1)$ and $k-m$ elements of $(\lfloor N^{1/k}\rfloor - 1)$. Since, for all sets, Q satisfies $\prod_{i=1}^{k}(q_i+1) \geqslant N$ with $\sum_{i=1}^{k} q_i$ being minimum, we can always find a corresponding set Q' satisfying $\prod_{i=1}^{k}(q'_i+1) \geqslant N$ 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 $\sum_{i=1}^{k} q_i'$ 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 $\sum_{i=1}^{k} q_i'$ is greater than $\sum_{i=1}^{k} p_i$. This contradicts the assumption that $\sum_{i=1}^{k} q_i'$ is minimum.

Without loss generality, we assume that $\forall 1 \leqslant i < j \leqslant k$, $p_i \geqslant p_j$ and $q_i' \geqslant q_j'$ where $p_i, p_j \in P$ and $g_i', g_j' \in Q'$. 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 $\forall 1 \leqslant i \leqslant l-1$, $q_i' = p_i$ and $q_l' < p_l$. There are two cases to be considered.

Case 1: $l \leqslant m$: Since $p_l = \lceil N^{1/k} \rceil - 1$, $q_l' \leqslant p_l - 1 = \lceil N^{1/k} \rceil - 2 \leqslant \lfloor N^{1/k} \rfloor - 1$. Thus $\forall l \leqslant i \leqslant k$, $q_i' \leqslant \lfloor N^{1/k} \rfloor - 1$. Therefore,

$$\begin{aligned}\prod_{i=1}^{k}(q_i'+1) &= \prod_{i=1}^{l-1}(p_i+1)\prod_{i=l}^{k}(q_i'+1)\\ &\leqslant \lceil N^{1/k}\rceil^{l-1}\lfloor N^{1/k}\rfloor^{k-l+1}\\ &\leqslant \lceil N^{1/k}\rceil^{m-1}\lfloor N^{1/k}\rfloor^{k-m+1}\\ &< N \end{aligned} \tag{5}$$

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

Case 2: $l > m$: Since $p_l = [N^{1/k}] - 1$, $q_l' \leqslant p_l - 1 = \lfloor N^{1/k} \rfloor - 2$. Thus, $\forall l \leqslant i \leqslant k$, $q_i' \leqslant [N^{1/k}] - 2$.

$$\begin{aligned}\prod_{i=1}^{k}(q_i'+1) &= \prod_{i=1}^{m}(p_i+1)\prod_{i=m+1}^{l-1}(p_i+1)\prod_{i=1}^{k}(q_i'+1)\\ &\leqslant \lceil N^{1/k}\rceil^{m}\lfloor N^{1/k}\rfloor^{l-m-1}(\lfloor N^{1/k}\rfloor - 1)^{k-l+1}\\ &< \lceil N^{1/k}\rceil^{m-1}\lfloor N^{1/k}\rfloor^{k-m-1}\lceil N^{1/k}\rceil(\lfloor N^{1/k}\rfloor - 1)\\ &\leqslant \lceil N^{1/k}\rceil^{m-1}\lfloor N^{1/k}\rfloor^{k-m-1}\\ &\quad\times(\lfloor N^{1/k}\rfloor + 1)(\lfloor N^{1/k}\rfloor - 1)\\ &< \lceil N^{1/k}\rceil^{m-1}\lfloor N^{1/k}\rfloor^{k-m-1}\lfloor N^{1/k}\rfloor^{2}\\ &= \lceil N^{1/k}\rceil^{m-1}\lfloor N^{1/k}\rfloor^{k-m+1}\\ &< N \end{aligned} \tag{6}$$

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

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

$$m(\lceil N^{1/k}\rceil - 1) + (k-m)(\lfloor N^{1/k}\rfloor - 1) \geqslant k(\lfloor N^{1/k}\rfloor - 1) \tag{7}$$

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

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 $kN(\lfloor N^{1/k} \rfloor - 1)$ messages passed among the N nodes.

Proof: Assume that, for all i, $1 \leqslant i \leqslant k$, in the ith 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 kth round, there are at most $\prod_{i=1}^{k} (p_i + 1)$ nodes having the knowledge about the initial value of node 1. Since only k rounds of message interchanges are used, $\prod_{i=1}^{k} (p_i + 1)$ 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, $1 \leqslant i \leqslant k$, the total number of messages passed among the N nodes is $N \sum_{i=1}^{k} p_i$. Because $\prod_{i=1}^{k} (p_i + 1) \geqslant N$ and for any k positive integers $p_1, p_2, \ldots, p_k$, the $\sum_{i=1}^{k} p_i$ is at least $k(\lfloor N^{1/k} \rfloor - 1)$ (from Lemma 3.1), the total number of messages $N \sum_{i=1}^{k} p_i$ is at least $kN(\lfloor N^{1/k} \rfloor - 1)$.

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 $\sum_{i=1}^{k} p_i$ becoming $m(\lceil N^{1/k} \rceil - 1) + (k - m)(\lfloor N^{1/k} \rfloor - 1)$ under the constraint of $\prod_{i=1}^{k} (p_i + 1) \geqslant N$, where $0 \leqslant m \leqslant k$ and $[N^{1/k}]^m [N^{1/k}]^{k-m} \geqslant N > [N^{1/k}]^{m-1} [N^{1/k}]^{k-m+1}$. Therefore, if every node sends out p_i messages at the ith round of message interchange and after the ith round, there are $\prod_{j=1}^{i} (p_j + 1)$ nodes having the knowledge of the initial value of any node, 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 $\prod_{i=1}^{k} (p_i + 1) \geqslant N$ and $\sum_{i=1}^{k} p_i$ is the minimum. We then add $M - N$ dummy nodes into the system, where $M = \prod_{i=1}^{k} (p_i + 1)$. We consider the M-node system as a k-dimensional array of $[0 \cdots p_1, 0 \cdots p_2, \ldots, 0 \cdots p_k]$ such that every node X, $0 \leqslant X \leqslant M - 1$ can be addressed as a k-tuple $(x_1, x_2, \ldots, x_k)$, where $\forall i$, $1 \leqslant i \leqslant k$, x_i is integer between 0 and p_i and $X = \sum_{i=1}^{k} (x_i \prod_{j=i+1}^{k} (p_j + 1))$. Since there are mp_is equal to $([N^{1/k}] - 1)$ and $(k - m)p_i$s equal to $(\lfloor N^{1/k} \rfloor - 1)$, where $0 \leqslant m \leqslant k$ and $\lceil N^{1/k} \rceil^m \lfloor N^{1/k} \rfloor^{k-m} \geqslant N > \lceil N^{1/k} \rceil^{m-1} \lfloor N^{1/k} \rfloor^{k-m+1}$ (from the proof of Lemma 3.1), $M - N < \lceil N^{1/k} \rceil^{m-1} . \lfloor N^{1/k} \rfloor^{k-m+1} < N$, 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, $1 \leqslant i \leqslant k$: 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 $0 \leqslant y_i \leqslant p_i$ 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, $1 \leqslant i \leqslant k$, each node exchange messages with all nodes that differ from itself only in the ith 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 $\lceil N^{1/k}\rceil^m \lfloor N^{1/k}\rfloor^{k-m} \geqslant N > \lceil N^{1/k}\rceil^{m-1} \lfloor N^{1/k}\rfloor^{k-m+1}$. Let $p_1 = p_2 = \cdots = p_m = (\lceil N^{1/k}\rceil - 1)$, $p_{m+1} = p_{m+2} = \cdots = p_k = (\lfloor N^{1/k}\rfloor - 1)$, and $M = \prod_{i=1}^{k} (p_i + 1)$. Add $M - N$ dummy nodes to the system and address every node X, $0 \leqslant X \leqslant M - 1$ by a k-tuple $(x_1, x_2, \ldots, x_k)$, where $\forall i$, $1 \leqslant i \leqslant k$, $0 \leqslant x_i \leqslant p_i$ and $X = \sum_{i=1}^{k} (x_i \prod_{j=i+1}^{k} (p_j + 1))$. Initially, each real node $0 \leqslant X \leqslant N - 1$ 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.

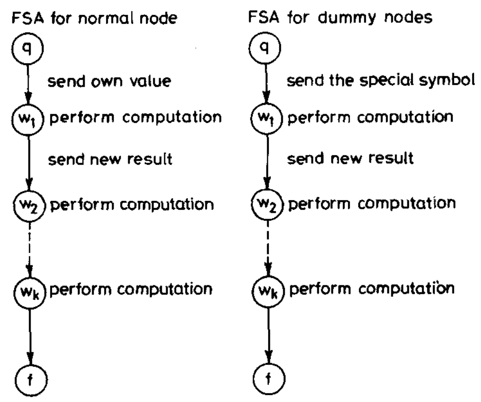

Fig. 1 *FSAs for computations of commutative functions*

State q for normal nodes: Send own value to all nodes addressed as $(y_1, x_2, \ldots, x_k)$, where $0 \leqslant y_1 \leqslant p_1$, and move to state w_1.

State q for dummy nodes: Send the special value '@' to all nodes addressed as $(y_1, x_2, \ldots, x_k)$, where $0 \leqslant y_1 \leqslant p_1$, and move to state w_1.

State w_i, $1 \leqslant i \leqslant (k - 1)$: Upon receiving values from all nodes addressed as $(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_k)$, where $0 \leqslant y_i \leqslant p_i$, perform the operation Ψ among received values (ignoring the special symbol '@') and the local value, send the result to all nodes addressed as $(x_1, \ldots, x_i, y_{i+1}, x_{i+2}, \ldots, x_k)$, where $0 \leqslant y_{i+1} \leqslant p_{i+1}$, 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 $(x_1, \ldots, x_{k-1}, y_k)$, where $0 \leqslant y_k \leqslant p_k$, perform the operation Ψ 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 $\Psi_{i=0}^{N-1} V^0(i)$, where $V^0(i) = V_i$ is the initial value of node i.

Proof: We show it by induction.

Inductive hypothesis: After the ith round of message exchanges, where $1 \leqslant i \leqslant k$, a node X addressed as $(x_1, x_2, \ldots, x_k)$ will have the value $V^i(X) = \Psi_{y_1=0}^{p_1} \Psi_{y_2=0}^{p_2} \cdots \Psi_{y_i=0}^{p_i} V^0(y_1, y_2, \ldots, y_i, x_{i+1}, \ldots, x_k)$.

Base case: After the first round of message exchange, the node X has received values from nodes $(y_1, x_2, \ldots, x_k)$, where $0 \leqslant y_1 \leqslant p_1$, which have the initial values $V^0(y_1, x_2, \ldots, x_k)$. Therefore, the node X will contain the value $V^1(X) = \Psi_{y_1=0}^{p_1} V^0(y_1, x_2, \ldots, x_k)$.

Inductive case: Assume for $i = l - 1$, where $l \leqslant k$, the inductive hypothesis is true. After the $(l - 1)$th round of message exchange, the node X would have the value of $V^{l-1}(X)$ which is $\Psi_{y_1=0}^{p_1} \Psi_{y_2=0}^{p_2} \cdots \Psi_{y_{l-1}=0}^{p_{l-1}} V^0(y_1, y_2, \ldots, y_{l-1}, x_l, \ldots, x_k)$. After the lth round of message exchanges, the node X would receive values from nodes $(x_1, \ldots, x_{l-1}, y_l, x_{l+1}, \ldots, x_k)$, where $0 \leqslant y_l \leqslant p_l$, which have values $V^{l-1}(x_1, \ldots, x_{l-1}, y_l, x_{l+1}, \ldots, x_k)$. Therefore,

$$V^l(X) = \Psi_{y_l=0}^{p_l} V^{l-1}(x_1, \ldots, x_{l-1}, y_l, x_{l+1}, \ldots, x_k) \tag{8}$$

$$= \Psi_{y_l=0}^{p_l} (\Psi_{y_1=0}^{p_1} \Psi_{y_2=0}^{p_2} \cdots \Psi_{y_{l-1}=0}^{p_{l-1}} V^0 \times (y_1, y_2, \ldots, y_{l-1}, y_l, x_{l+1}, \ldots, x_k))$$

$$= \Psi_{y_1=0}^{p_1} \Psi_{y_2=0}^{p_2} \cdots \Psi_{y_{l-1}=0}^{p_{l-1}} \Psi_{y_l=0}^{p_l} V^0 \times (y_1, y_2, \ldots, y_{l-1}, y_l, x_{l+1}, \ldots, x_k) \tag{9}$$

So the inductive hypothesis is true for all i, $1 \leqslant i \leqslant k$. Thus, after k rounds of message interchanges, a node X should contain the value $V^k(X) = \Psi_{y_1=0}^{p_1} \Psi_{y_2=0}^{p_2} \cdots \Psi_{y_k=0}^{p_k} V^0(y_1, y_2, \ldots, y_k) = \Psi_{i=0}^{N-1} V_i = F(V_0, V_1, \ldots, V_{N-1})$ □

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 $\lfloor N^{1/k}\rfloor - 1$ or $\lfloor N^{1/k}\rfloor - 1$, the $\sum_{i=1}^{k} p_i$ should be less than or equal to $k(\lceil N^{1/k}\rceil - 1)$, so the total number of messages $\leqslant kN(\lceil N^{1/k}\rceil - 1)$, because the lower bound $kN(\lfloor N^{1/k}\rfloor - 1)$ and $kN(\lceil N^{1/k}\rceil - 1)$ have the same order of $\Theta(kNN^{1/k})$. 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 KDAMS structure, we note that $m = 1$ and $p_1 = 2$, $p_2 = 1$, and $p_3 = 1$. Therefore, each node X, $0 \leqslant X \leqslant 11$, can be addressed as a triple (x_1, x_2, x_3), where $0 \leqslant x_1 \leqslant 2$, $0 \leqslant x_2 \leqslant 1$, and $0 \leqslant x_3 \leqslant 1$. Assume that nodes have initial values as follows:

$V_{000}^0 = 10$	$V_{001}^0 = 31$	$V_{010}^0 = 43$	$V_{011}^0 = 50$
$V_{100}^0 = 1$	$V_{101}^0 = 39$	$V_{110}^0 = 17$	$V_{111}^0 = 3$
$V_{200}^0 = 28$	$V_{201}^0 = 9$	$V_{210}^0 = 20$	$V_{211}^0 = 11$

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 initial value to be its new local value. Thus, the local values after the first round are as follows:

$$V^1_{000} = V^0_{000} + V^0_{100} + V^0_{200} = 10 + 1 + 28 = 39$$
$$V^1_{100} = V^0_{000} + V^0_{100} + V^0_{200} = 10 + 1 + 28 = 39$$
$$V^1_{200} = V^0_{000} + V^0_{100} + V^0_{200} = 10 + 1 + 28 = 39$$
$$V^1_{001} = V^0_{001} + V^0_{101} + V^0_{201} = 31 + 39 + 9 = 79$$
$$V^1_{101} = V^0_{001} + V^0_{101} + V^0_{201} = 31 + 39 + 9 = 79$$
$$V^1_{201} = V^0_{001} + V^0_{101} + V^0_{201} = 31 + 39 + 9 = 79$$
$$V^1_{010} = V^0_{010} + V^0_{110} + V^0_{210} = 43 + 17 + 20 = 80$$
$$V^1_{110} = V^0_{010} + V^0_{110} + V^0_{210} = 43 + 17 + 20 = 80$$
$$V^1_{210} = V^0_{010} + V^0_{110} + V^0_{210} = 43 + 17 + 20 = 80$$
$$V^1_{011} = V^0_{011} + V^0_{111} + V^0_{211} = 50 + 3 + 11 = 64$$
$$V^1_{111} = V^0_{011} + V^0_{111} + V^0_{211} = 50 + 3 + 11 = 64$$
$$V^1_{211} = V^0_{011} + V^0_{111} + V^0_{211} = 50 + 3 + 11 = 64$$

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:

$$V^2_{000} = V^1_{000} + V^1_{010} = 39 + 80 = 119$$
$$V^2_{010} = V^1_{000} + V^1_{010} = 39 + 80 = 119$$
$$V^2_{001} = V^1_{001} + V^1_{011} = 79 + 64 = 143$$
$$V^2_{011} = V^1_{001} + V^1_{011} = 79 + 64 = 143$$
$$V^2_{100} = V^1_{100} + V^1_{110} = 39 + 80 = 119$$
$$V^2_{110} = V^1_{100} + V^1_{110} = 39 + 80 = 119$$
$$V^2_{101} = V^1_{101} + V^1_{111} = 79 + 64 = 143$$
$$V^2_{111} = V^1_{101} + V^1_{111} = 79 + 64 = 143$$
$$V^2_{200} = V^1_{200} + V^1_{210} = 39 + 80 = 119$$
$$V^2_{210} = V^1_{200} + V^1_{210} = 39 + 80 = 119$$
$$V^2_{201} = V^1_{201} + V^1_{211} = 79 + 64 = 143$$
$$V^2_{211} = V^1_{201} + V^1_{211} = 79 + 64 = 143$$

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 vaues are as follows:

$$V^3_{000} = V^2_{000} + V^2_{001} = 119 + 143 = 262$$
$$V^3_{001} = V^2_{000} + V^2_{001} = 119 + 143 = 262$$
$$V^3_{010} = V^2_{010} + V^2_{011} = 119 + 143 = 262$$
$$V^3_{011} = V^2_{010} + V^2_{011} = 119 + 143 = 262$$
$$V^3_{100} = V^2_{100} + V^2_{101} = 119 + 143 = 262$$
$$V^3_{101} = V^2_{100} + V^2_{101} = 119 + 143 = 262$$
$$V^3_{110} = V^2_{110} + V^2_{111} = 119 + 143 = 262$$
$$V^3_{111} = V^2_{110} + V^2_{111} = 119 + 143 = 262$$
$$V^3_{200} = V^2_{200} + V^2_{201} = 119 + 143 = 262$$
$$V^3_{201} = V^2_{200} + V^2_{201} = 119 + 143 = 262$$
$$V^3_{210} = V^2_{210} + V^2_{211} = 119 + 143 = 262$$
$$V^3_{211} = V^2_{210} + V^2_{211} = 119 + 143 = 262$$

7 Concluding remarks

We have shown that fully decentralised evaluating associative and commutative functions require at least $kN(\lfloor N^{1/k} \rfloor - 1)$ messages in an N-node system with k rounds of message exchanges. A family of fully decentralised algorithm for evaluation 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 $kN(\lceil N^{1/k} \rceil - 1)$ 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)
2 CHANG, E., and ROBERTS, R.: 'An improved algorithm for decentralised extrema finding in circular configurations of processors', *Commun. ACM*, 1979, **22**, pp. 281–283
3 HIRSCHBERG, D., and SINCLAIR, J.: 'Decentralised extrema-finding in circular configurations of processors', *Commun. ACM*, 1980, **23**, pp. 627–628
4 FRANKLIN, W.R.: 'On an improved algorithm for decentralised extrema-finding in circular configurations of processors', *Commun. ACM*, 1982, **25**, pp. 336–337
5 PETERSON, G.L.: 'An $o(n \log n)$ unidirectional algorithm for the circular extrema problem', *ACM Trans.*, 1982, **CS-4**, pp. 758–762
6 DOLEV, D., KLAWE, M., and RODEH, M.: 'An $o(n \log n)$ unidirectional distributed algorithm for extrema finding in a circle', *J. Algorithms*, 1982, **3**, pp. 245–260
7 FREDERICKSON, G.N., and LYNCH, N.: 'The impact of synchronous communication on the problem of electing a leader in a ring'. Proceedings of the 16th ACM symposium on theory of computing, Washington, DC, 1984, pp. 493–503 ACM
8 SON, S.H., and AGRAWALA, A.K.: 'A non-intrusive checkpointing scheme in distributed database system'. Proceedings of the 15th international symposium on fault-tolerant computing, 1985, pp. 99–104 IEEE
9 SKEEN, D.: 'Nonblocking commit protocols'. ACM SIGMOD international conference on management of data, 1981, pp. 133–142 ACM
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
11 SKEEN, D., and STONEBRAKER, M.: 'A formal model of crash recovery in a distributed system', *IEEE Trans.*, 1983, **SE-9**, pp. 219–228
12 LAKSHMAN, T.V., and AGRAWALA, A.K.: 'Efficient decentralised consensus protocols', *IEEE Trans.*, 1986, **SE-12**, (5), pp. 600–607