Distributed Computing on Core-Periphery Networks: Axiom-Based Design*

Chen Avin[1,**], Michael Borokhovich[1], Zvi Lotker[1], and David Peleg[2]

[1] Ben-Gurion University of the Negev, Israel
{avin,borokhom,zvilo}@cse.bgu.ac.il
[2] The Weizmann Institute, Israel
david.peleg@weizmann.ac.il

Abstract. Inspired by social networks and complex systems, we propose a *core-periphery* network architecture that supports fast computation for many distributed algorithms and is robust and efficient in number of links. Rather than providing a concrete network model, we take an axiom-based design approach. We provide three intuitive (and independent) algorithmic axioms and prove that any network that satisfies all axioms enjoys an efficient algorithm for a range of tasks (e.g., MST, sparse matrix multiplication, etc.). We also show the *minimality* of our axiom set: for networks that satisfy any subset of the axioms, the same efficiency cannot be guaranteed for *any* deterministic algorithm.

1 Introduction

A fundamental goal in distributed computing is designing a network architecture that allows fast running times for various distributed algorithms, but at the same time is cost-efficient in terms of minimizing the number of communication links between machines and the amount of memory used by each machine.

For illustration, let's consider three basic networks topologies: a star, a clique and a constant degree expander. The *star graph* has only a linear number of links and can compute every computable function after only one round of communication. But clearly, such an architecture has two major disadvantages: the memory requirements of the central node do not scale, and the network is not robust. The *complete graph*, on the other hand, is very robust and can support extremely fast performance for tasks such as information dissemination, distributed sorting and minimum spanning tree, to name a few [1,2,3]. Also, in a complete graph the amount of memory used by a single processor is minimal. But obviously, the main drawback of that architecture is the high number of links it uses. *Constant degree expanders* are a family of graphs that support efficient computation for many tasks. They also have linear number of links and can effectively balance the workload between many machines. But the diameter of these graphs is lower bounded by $\Omega(\log n)$ which implies similar lower bound for most of the interesting tasks one can consider.

* Supported in part by the Israel Science Foundation (grant 1549/13).
** Part of this work was done while the author was visiting ICERM, Brown university.

J. Esparza et al. (Eds.): ICALP 2014, Part II, LNCS 8573, pp. 399–410, 2014.

A natural question is therefore whether there are other candidate topologies with guaranteed good performance. We are interested in the best compromise solution: a network on which distributed algorithms have small running times, memory requirements at each node are limited, the architecture is robust to link and node failures, and the total number of links is minimized (preferably linear).

To try to answer this question we adopt in this paper an *axiomatic* approach to the design of efficient networks. In contrast to the direct approach to network design, which is based on providing a *concrete* type of networks (by deterministic or random construction) and showing its efficiency, the axiomatic approach attempts to abstract away the algorithmic requirements that are imposed on the concrete model. This allows one to isolate and identify the basic requirements that a network needs for a certain type of tasks. Moreover, while usually the performance of distributed algorithms is dictated by specific *structural* properties of a network (e.g., diameter, conductance, degree, etc.), the axioms proposed in this work are expressed in terms of desired *algorithmic* properties that the network should have.

The axioms[1] proposed in the current work are motivated and inspired by the *core-periphery* structure exhibited by many social networks and complex systems. A core-periphery network is a network structured of two distinct groups of nodes, namely, a large, sparse, and weakly connected group identified as the *periphery*, which is loosely organized around a small, cohesive and densely connected group identified as the *core*. Such a dichotomic structure appears in many areas of our life, and has been observed in many social organizations including modern social networks [4]. It can also be found in urban and even global systems (e.g., in global economy, the wealthiest countries constitute the core which is highly connected by trade and transportation routes) [5,6,7]. There are also peer-to-peer networks that use a similar hierarchical structure, e.g., FastTrack [8] and Skype [9], in which the supernodes can be viewed as the core and the regular users as the periphery.

The main technical contribution of this paper is proposing a minimal set of simple core-periphery-oriented axioms and demonstrating that networks satisfying these axioms achieve efficient running time for various distributed computing tasks while being able to maintain linear number of edges and limited memory use. We identify three basic, abstract and conceptually simple (parameterized) properties, that turn out to be highly relevant to the effective interplay between core and periphery. For each of these three properties, we propose a corresponding axiom, which in our belief captures some intuitive aspect of the desired behavior expected of a network based on a core-periphery structure. Let us briefly describe our three properties, along with their "real life" interpretation, technical formulation, and associated axioms.

The three axioms are: (i) clique-like structure of the core, (ii) fast convergecast from periphery to the core and (iii) balanced boundary between the core and

[1] One may ask whether the properties we define qualify as "axioms". Our answer is that the axiomatic lens helps us focus attention on the fundamental issues of minimality, independence and necessity of our properties.

Table 1. Summary of algorithms for core-periphery networks

Task	Running time on $\mathcal{CP}$ networks	Lower bounds	
		All Axioms	Any 2 Axioms
MST *	$O(\log^2 n)$	$\Omega(1)$	$\tilde{\Omega}(\sqrt[4]{n})$
Matrix transposition	$O(k)$	$\Omega(k)$	$\Omega(n)$
Vector by matrix multiplication	$O(k)$	$\Omega(k/\log n)$	$\Omega(n/\log n)$
Matrix multiplication	$O(k^2)$	$\Omega(k^2)$	$\Omega(n/\log n)$
Find my rank	$O(1)$	$\Omega(1)$	$\Omega(n)$
Find median	$O(1)$	$\Omega(1)$	$\Omega(\log n)$
Find mode	$O(1)$	$\Omega(1)$	$\Omega(n/\log n)$
Find number of distinct values	$O(1)$	$\Omega(1)$	$\Omega(n/\log n)$
Top r ranked by areas	$O(r)$	$\Omega(r)$	$\Omega(r\sqrt{n})$

k - maximum number of nonzero entries in a row or column. * - randomized algorithm

periphery. The first property deals with the flow of information within the core. It is guided by the key observation that to be influential, the core must be able to accomplish fast information dissemination internally among its members. The corresponding Axiom $\mathcal{A}_E$ postulates that the core must be a $\Theta(1)$-clique emulator (to be defined formally later). Note that this requirement is stronger than just requiring the core to possess a dense interconnection subgraph, since the latter permits the existence of "bottlenecks", whereas the requirement of the axiom disallows such bottlenecks.

The second property focuses on the flow of information from the periphery to the core and measures its efficiency. The core-periphery structure of the network is said to be a γ-*convergecaster* if this data collection operation can be performed in time γ. The corresponding Axiom $\mathcal{A}_C$ postulates that information can flow from the periphery nodes to the core efficiently (i.e., in constant time). Note that one implication of this requirement is that the presence of periphery nodes that are far away from the core, or bottleneck edges that bridge between many periphery nodes and the core, is forbidden.

The third and last property concerns the "boundary" between the core and the periphery and claim that core nodes are "*effective ambassadors*". Ambassadors serve as bidirectional channels through which information flows into the core and influence flows from the core to the periphery. However, to be effective as an ambassador, the core node must maintain a balance between its interactions with the "external" periphery and its interactions with the other core members, serving as its natural "support"; a core node which is significantly more connected to the periphery than to the core becomes ineffective as a channel of influence. In distributed computing terms, a core node that has many connections to the periphery has to be able to distribute all the information it collected from them to other core nodes. The corresponding Axiom $\mathcal{A}_B$ states that the core must have a $\Theta(1)$-balanced boundary (to be defined formally later).

To support and justify our selection of axioms, we examine their usefulness for effective distributed computations on core-periphery networks. We consider a collection of different types of tasks, and show that they can be efficiently solved on core-periphery networks, by providing a distributed algorithm for each task

and bounding its running time. Moreover, for each task we argue the necessity of all three axioms, by showing that if at least one of the axioms is not satisfied by the network under consideration, then the same efficiency can not be guaranteed by *any* algorithm for the given task.

Table 1 provides an overview of the main tasks we studied along with the upper and lower bounds on the running time when the network satisfies our axioms and a worst case lower bound on the time required when at least one of the axioms is not satisfied. For each task we provide an algorithm and prove formally its running time and the necessity of the axioms. As it turns out, some of the necessity proofs make use of an interesting connection to known communication complexity results.

The most technically challenging part of the paper is the distributed construction of a *minimum-weight spanning tree* (MST), a significant task in both the distributed systems world (cf.[10,11,12]) and the social networks world [13,14,15]. Thus, the main algorithmic result of the current paper is proving that MST can be computed efficiently (in $O(\log^2 n)$ rounds) on core-periphery networks. To position this result in context we recall that for the complete graph $G = K_n$, an MST can be constructed distributedly in $O(\log \log n)$ time [1]. For the wider class of graphs of diameter at most 2, this task can still be performed in time $O(\log n)$. In contrast, taking the next step, and considering graphs of diameter 3, drastically changes the picture, as there are examples of such graphs for which any distributed MST construction requires $\Omega\left(\sqrt[4]{n}\right)$ time [16].

The rest of the paper is organized as follows. Section 2 formally describes core-periphery networks, the axioms and their basic structural implications. Section 3 provides an overview on the MST algorithm and Section 4 an overview on the rest of the task we study. Due to lack of space we defer many of the technical details and proofs to the report [17].

2 Axiomatic Design for Core-Periphery Networks

Preliminaries. Let $G(V, E)$ denote our (simple, undirected) network, where V is the set of nodes, $|V| = n$, and E is the set of edges, $|E| = m$. The network can be thought of as representing a distributed system. We assume the synchronous CONGEST model (cf. [12]), where communication proceeds in *rounds* and in each round each node can send a message of at most $O(\log n)$ bits to each of its neighbors. Initially each node has a unique ID of $O(\log n)$ bits.

For a node v, let $N(v)$ denote its set of neighbors and $d(v) = |N(v)|$ its degree. For a set $S \subset V$ and a node $v \in S$, let $N_{\text{in}}(v, S) = N(v) \cap S$ denote its set of neighbors within S and denote the number of neighbors of v in the set S by $d_{\text{in}}(v, S) = |N_{\text{in}}(v, S)|$. Analogously, let $N_{\text{out}}(v, S) = N(v) \cap V \setminus S$ denote v's set of neighbors outside the set S and let $d_{\text{out}}(v) = |N_{\text{out}}(v, S)|$. For two subsets $S, T \subseteq V$, let $\partial(S, T)$ be the *edge boundary* (or cut) of S and T, namely the set of edges with exactly one endpoint in S and one in T and $|\partial(S, T)| = \sum_{v \in S} |N_{\text{out}}(v, S) \cap T|$. Let $\partial(S)$ denote the special case where $T = V \setminus S$.

Core-Periphery Networks. Given a network $G(V, E)$, a $\langle \mathcal{C}, \mathcal{P} \rangle$-partition is a partition of the nodes of V into two sets, the *core* $\mathcal{C}$ and the *periphery* $\mathcal{P}$. Denote the sizes of the core and the periphery by $n_{\mathcal{C}}$ and $n_{\mathcal{P}}$ respectively. To represent the partition along with the network itself, we denote the *partitioned network* by $G(V, E, \mathcal{C}, \mathcal{P})$.

Intuitively, the core $\mathcal{C}$ consists of a relatively small group of strong and highly connected machines designed to act as central servers, whereas the periphery $\mathcal{P}$ consists of the remaining nodes, typically acting as clients. The periphery machines are expected to be weaker and less well connected than the core machines, and they may perform much of their communication via the dense interconnection network of the core. In particular, a central component in many of our algorithms for various coordination and computational tasks is based on assigning each node v a *representative* core node $r(v)$, essentially a neighbor acting as a "channel" between v and the core. The representative chosen for each periphery node is fixed.

For a partitioned network to be effective, the $\langle \mathcal{C}, \mathcal{P} \rangle$ partition must possess certain desirable properties. In particular, a partitioned network $G(V, E, \mathcal{C}, \mathcal{P})$ is called a *core-periphery network*, or *$\mathcal{CP}$-network* for short, if the $\langle \mathcal{C}, \mathcal{P} \rangle$-partition satisfies three properties, defined formally later on in the form of three axioms.

Core-periphery Properties and Axioms. We first define certain key parameterized properties of node groups in networks that are of particular relevance to the relationships between core and periphery in our partitioned network architectures. We then state our axioms, which capture the expected behavior of those properties in core-periphery networks, and demonstrate their independence and necessity. Our three basic properties are:

α-Balanced Boundary. *A subset of nodes S is said to have an* α-balanced boundary *iff* $\frac{d_{\text{out}}(v,S)}{d_{\text{in}}(v,S)+1} = O(\alpha)$ *for every node $v \in S$.*

β-Clique Emulation. *The task of* clique emulation *on an n-node graph G involves delivering a distinct message $M_{v,w}$ from v to w for every pair of nodes v, w in $V(G)$. An n-node graph G is a* β-clique-emulator *if it is possible to perform clique emulation on G within β rounds (in the CONGEST model).*

γ-convergecast. *For $S, T \subseteq V$, the task of* $\langle S, T \rangle$-convergecast *on a graph G involves delivering $|S|$ distinct messages M_v, originated at the nodes $v \in S$, to some nodes in T (i.e., each message must reach at least one node in T). The sets $S, T \subset V$ form a* γ-convergecaster *if it is possible to perform $\langle S, T \rangle$-convergecast on G in γ rounds (in the CONGEST model).*

Consider a partitioned network $G(V, E, \mathcal{C}, \mathcal{P})$. We propose the following set of *axioms* concerning the core $\mathcal{C}$ and periphery $\mathcal{P}$.

$\mathcal{A}_B$. **Core Boundary.** The core $\mathcal{C}$ has a $\Theta(1)$-balanced boundary.

$\mathcal{A}_E$. **Clique Emulation.** The core $\mathcal{C}$ is a $\Theta(1)$-clique emulator.

$\mathcal{A}_C$. **Periphery-Core Convergecast**. The periphery $\mathcal{P}$ and the core $\mathcal{C}$ form a $\Theta(1)$-convergecaster.

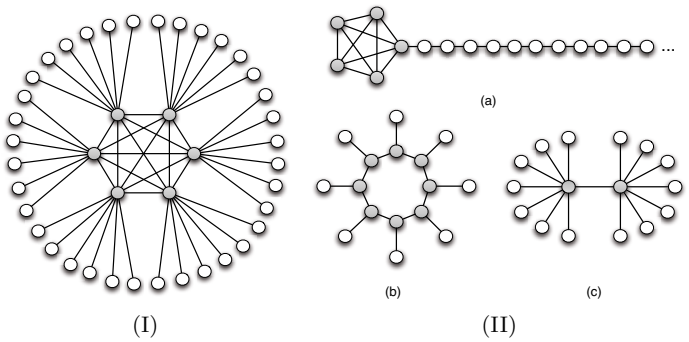

Fig. 1. (I) An example for a 36-node $\mathcal{CP}$-network that satisfies all three axioms. The 6 core nodes (in gray) are connected in clique. In this example every core node is also an ambassadors with equal number of edges to the core and outside from the core. The core and periphery form a convergecaster since the periphery can send all its information to the core in one round. (II) Networks used in proofs: (a) The lollipop partitioned network L_{25}. (b) The sun partitioned network S_{16}. (c) The dumbbell partitioned network D_{16} .

Let us briefly explain the axioms. Axiom $\mathcal{A}_B$ talks about the boundary between the core and periphery. Think of core nodes with a high out-degree (i.e., with many links to the periphery) as "*ambassadors*" of the core to the periphery. Axiom $\mathcal{A}_B$ states that while not all nodes in the core must serve as ambassadors, if a node is indeed an ambassador, then it must also have many links *within* the core. Axiom $\mathcal{A}_E$ talks about the flow of information within the core, and postulates that the core must be dense, and in a sense behave almost like a complete graph: "everyone must know everyone else". The clique-emulation requirement is actually stronger than just being a dense subgraph, since the latter permits the existence of bottlenecks nodes, which a clique-emulator must avoid. Axiom $\mathcal{A}_C$ also concerns the boundary between the core and periphery, but in addition it refers also to the structure of the periphery. It postulates that information can flow efficiently from the periphery to the core. For example, it forbids the presence of periphery nodes that are far away from the core, or bottleneck edges that bridge between many periphery nodes and the core. Fig. 1 (I) provides an example for $\mathcal{CP}$-network satisfying the three axioms.

We next show that the axioms are independent. Later, we prove the necessity of the axioms for the efficient performance of a variety of computational tasks.

Theorem 1. *Axioms $\mathcal{A}_B$, $\mathcal{A}_E$, $\mathcal{A}_C$ are independent, namely, assuming any two of them does not imply the third.*

We prove this theorem by considering three examples of partitioned networks, described next, each of which satisfies two of the axiomes but not the third (hence they are not $\mathcal{CP}$-networks), implying independence.

The lollipop partitioned network L_n: (Fig. 1 (II)(a) The lollipop graph consists of a $\sqrt{n}$-node clique and an $n-\sqrt{n}$-node line attached to some node of the clique.

Set the core $\mathcal{C}$ to be the clique and the periphery $\mathcal{P}$ to be the line. Observe that Axioms $\mathcal{A}_E$ and $\mathcal{A}_B$ hold but $\mathcal{A}_C$ is not satisfied since the long line will require linear time for periphery to core convergcast.

The sun partitioned network S_n: (Fig. 1 (II)(b)) The sun graph consists of an $n/2$-node cycle with an additional leaf node attached to each cycle node. Set the core $\mathcal{C}$ to be the cycle and the periphery $\mathcal{P}$ to contain the $n/2$ leaves. Axioms $\mathcal{A}_C$ and $\mathcal{A}_B$ hold but Axiom $\mathcal{A}_E$ does not, since the distance between diametrically opposing nodes in the cycle is $n/4$, preventing fast clique emulation.

The dumbbell partitioned network D_n: (Fig. 1 (II)(c)) The dumbbell graph is composed of two stars, each consisting of a center node connected to $n/2 - 1$ leaves, whose centers are connected by an edge. Set the core $\mathcal{C}$ to be the two centers, and the periphery $\mathcal{P}$ to contain all the leaves. Then Axioms $\mathcal{A}_E$ and $\mathcal{A}_C$ hold while Axiom $\mathcal{A}_B$ does not.

Structural Implications of the Axioms. The axioms imply a number of simple properties of the network structure.

Theorem 2. *If $G(V, E, \mathcal{C}, \mathcal{P})$ is a core-periphery network (i.e., it satisfies Axioms $\mathcal{A}_B$, $\mathcal{A}_E$ and $\mathcal{A}_C$), then the following properties hold:*

1. *The size of the core satisfies $\Omega(\sqrt{n}) \leq n_c \leq O(\sqrt{m})$.*
2. *Every node v in the core satisfies $d_{\text{out}}(v, \mathcal{C}) = O(n_c)$ and $d_{\text{in}}(v, \mathcal{C}) = \Omega(n_c)$.*
3. *The number of outgoing edges from the core is $|\partial(\mathcal{C})| = \Theta(n_c^2)$.*
4. *The core is dense, i.e., the number of edges in it is $\sum_{v \in \mathcal{C}} d_{\text{in}}(v, \mathcal{C}) = \Theta(n_c^2)$.*

Proof. Axiom $\mathcal{A}_E$ necessitates that the inner degree of each node v is $d_{\text{in}}(v, \mathcal{C}) = \Omega(n_c)$ (or else it would not be possible to complete clique emulation in constant time), implying the second part of claim 2. It follows that the number of edges in the core is $\sum_{v \in \mathcal{C}} d_{\text{in}}(v, \mathcal{C}) = \Theta(n_c^2)$, hence it is dense; claim 4 follows. Since also $\sum_{v \in \mathcal{C}} d_{\text{in}}(v, \mathcal{C}) \leq 2m$, we must have the upper bound of claim 1, that is, $n_c = O(\sqrt{m})$. Axiom $\mathcal{A}_B$ yields that for every v, $d_{\text{out}}(v, \mathcal{C}) = O(n_c)$, so the first part of claim 2 follows. Note that $|\partial(\mathcal{C})| = \sum_{v \in \mathcal{C}} d_{\text{out}}(v, \mathcal{C}) = O(n_c^2)$, so the upper bound of claim 3 follows. To give a lower bound on n_c, note that by Axiom $\mathcal{A}_C$ we have $|\partial(\mathcal{C})| = \Omega(n - n_c)$ (otherwise the information from the $n - n_c$ nodes of $\mathcal{P}$ could not flow in $O(1)$ time to $\mathcal{C}$), so $n_c = \Omega(\sqrt{n})$ and the lower bounds of claims 1 and 3 follow. □

An interesting case for efficient networks is where the number of edges is linear in the number of nodes. In this case we have the following corollary.

Corollary 1. *In a core-periphery network where $m = O(n)$, the following properties hold:*

1. *The size of the core satisfies $n_c = \Theta(\sqrt{n})$*
2. *The number of outgoing edges from the core is $|\partial(\mathcal{C})| = \Theta(n)$.*
3. *The number of edges in the core is $\sum_{v \in \mathcal{C}} d_{\text{in}}(v, \mathcal{C}) = \Theta(n)$.*

Now we show a key property relating our axioms to the network diameter.

Claim 1. *If the partitioned network $G(V, E, \mathcal{C}, \mathcal{P})$ satisfies Axioms $\mathcal{A}_E$ and $\mathcal{A}_C$ then its diameter is $\Theta(1)$.*

The following claim shows that the above conditions are necessary.

Claim 2. *For $X \in \{E, C\}$, there exists a family of n-node partitioned networks $G_X(V, E, \mathcal{C}, \mathcal{P})$ of diameter $\Omega(n)$ that satisfy all axioms except $\mathcal{A}_X$.*

3 MST on a Core-Periphery Network

In this section we present a time-efficient randomized distributed algorithm for computing a *minimum-weight spanning tree* (MST) on a core-periphery network. In particular, we consider an n-node core periphery network $G(V, E, \mathcal{C}, \mathcal{P})$, namely, a partitioned network satisfying all three axioms, and show that an MST can be distributedly computed on such a network in $O(\log^2 n)$ rounds with high probability. Upon termination, each node knows which of its edges belong to the MST. Formally, we state the following theorem.

Theorem 3. *On a $\mathcal{CP}$-network $G(V, E, \mathcal{C}, \mathcal{P})$, Algorithm $\mathcal{CP}$-MST constructs an MST in $O(\log^2 n)$ rounds with high probability.*

We also show that Axioms $\mathcal{A}_B$, $\mathcal{A}_E$, and $\mathcal{A}_C$ are indeed necessary for our distributed MST algorithm to be efficient.

Theorem 4. *For each $X \in \{B, E, C\}$ there exists a family $\mathcal{F}_X = \{G_X(V, E, \mathcal{C}, \mathcal{P})(n)\}$ of partitioned networks that do not satisfy Axiom $\mathcal{A}_X$ but satisfy the other two axioms, and the time complexity of* any *distributed MST algorithm on $\mathcal{F}_X$ (as a function of the network size n) is $\Omega(n^{\alpha_X})$ for some constant $\alpha_X > 0$.*

The formal proof of Theorem 4 can be found in [17], but the idea of the proof is as following. For each case of Theorem 4 we show a graph in which, for a certain weight assignment, there exist two nodes s and r such that in order to decide which of the edges incident to r belong to the MST, it is required to know the weights of all the edges incident to s. Thus, at least $deg(s)$ (i.e., degree of s) messages have to be delivered from s to r in order to complete the MST task, which implies a lower bound on any distributed MST algorithm.

Now let us give a high level description of the algorithm. Our $\mathcal{CP}$-MST algorithm is based on Boruvka's MST algorithm [18], and runs in $O(\log n)$ phases, each consisting of several steps. The algorithm proceeds by maintaining a forest of tree *fragments* (initially singletons), and merging fragments until the forest converges to a single tree. Throughout the execution, each node has two *officials*, namely, core nodes that represent it. In particular, recall that each node v is assigned a *representative* core neighbor $r(v)$ passing information between v and the core. In addition, v is also managed by the *leader* $l(i)$ of its current fragment i. An important distinction between these two roles is that the representative of each node is fixed, while its fragment leader may change in each phase (as its fragment grows). At the beginning of each phase, every node knows the IDs of

its fragment and its leader. Then, every node finds its minimum weight outgoing edge, i.e., the edge with the second endpoint belonging to the other fragment and having the minimum weight. This information is delivered to the core by the means of the representative nodes, which receive the information, aggregate it (as much as possible) and forward it to the leaders of the appropriate fragments. The leaders decide on the fragment merging and inform all the nodes about new fragments IDs.

The correctness of the algorithm follows from emulating Boruvka's algorithm and the correctness of the fragments merging procedure, described in the technical report [17]. The main challenges in obtaining the proof were in bounding the running time, which required careful analysis and observations. There are two major sources of problems that can cause delays in the algorithm. The first involves sending information between officials (representatives to leaders and vice versa). Note that there are only $O(\sqrt{m})$ officials, but they may need to send information about m edges, which can lead to congestion. For example, if more than $\alpha \cdot \sqrt{m}$ messages need to be sent to an officials of degree $\sqrt{m}$, then this will take at least α rounds. We use randomization of leaders and the property of clique emulation to avoid this situation and make sure that officials do not have to send or receive more than $O(\sqrt{m} \log m)$ messages in a phase. The second source for delays is the fragments merging procedure. This further splits into two types of problems. The first is that a chain of fragments that need to be merged could be long, and in the basic distributed Boruvka's algorithm will take long time (up to n) to resolve. This problem is overcome by using a modified pointer jumping technique similar to [16]. The second problem is that the number of fragments that need to be merged could be large, resulting in a large number of *merging* messages that contain, for example, the new fragment ID. This problem is overcome by using randomization and by reducing the number of messages needed for resolving a merge. Full description of the algorithm along with the proofs of correctness and running time can be found in [17].

4 Additional Algorithms in Core-Periphery Networks

In addition to MST, we have considered a number of other distributed problems of different types, and developed algorithms for these problems that can be efficiently executed on core-periphery networks. In particular, we dealt with the following set of tasks related to matrix operations. (M1) Sparse matrix transposition. (M2) Multiplication of a sparse matrix by a vector. (M3) Multiplication of two sparse matrices.

We then considered problems related to calculating aggregate functions of initial values initially stored one at each node in V. In particular, we developed efficient algorithms for the following problems. (A1) Finding the rank of each value, assuming the values are ordered. (As output, each node should know the rank of the element it stores.) (A2) Finding the median of the values. (A3) Finding the (statistical) mode, namely, the most frequent value. (A4) Finding the number of distinct values stored in the network. Each of these problems

requires $\Omega(Diam)$ rounds on general networks, whereas on a $\mathcal{CP}$-network it can be performed in $O(1)$ rounds.

An additional interesting task is defined in a setting where the initial values are split into disjoint groups, and requires finding the r largest values of each group. This task can be used, for example, for finding the most popular headlines in each area of news. Here, there is an $O(r)$ round solution on a $\mathcal{CP}$-network, whereas in general networks the diameter is still a lower bound.

In all of these problems, we also establish the necessity of all 3 axioms, by showing that there are network families satisfying 2 of the 3 axioms for which the general lower bound holds. Due to space limitation, we discuss in this section only one of these problems, namely, multiplication of a vector by a sparse matrix. Our results for the other problems can be found in [17].

A few definitions are in place. Let A be a matrix in which each entry $A(i,j)$ can be represented by $O(\log n)$ bits (i.e., it fits in a single message in the CONGEST model). Denote by $A_{i,*}$ (respectively, $A_{*,i}$) the ith row (resp., column) of A. Denote the ith entry of a vector s by $s(i)$. We assume that the nodes in $\mathcal{C}$ have IDs $[1,\ldots,n_{\mathcal{C}}]$ and this is known to all of them. A square $n \times n$ matrix A with $O(k)$ nonzero entries in each row and each column is hereafter referred to as an $O(k)$-*sparse matrix*.

Let s be a vector of size n and A be a square $n \times n$ $O(k)$-sparse matrix. Initially, each node in V holds one entry of s (along with the index of the entry) and one row of A (along with the index of the row). The task is to distributively calculate vector $s' = sA$ and store its entries at the corresponding nodes in V, such that the node that initially stored $s(i)$ will store $s'(i)$. We start with a claim on the lower bound (the proof can be found in [17]).

Claim 3. *The* lower bound *for any algorithm for multiplication of a vector by a sparse matrix on any network is $\Omega(D)$, and on a $\mathcal{CP}$-network is $\Omega(k/\log n)$.*

Algorithm 1. The following algorithm solves the task in $O(k)$ rounds on a $\mathcal{CP}$-network $G(V,E,\mathcal{C},\mathcal{P})$.

1. Each $u \in V$ sends the entry of s it has (along with the index of the entry) to its representative $r(u) \in \mathcal{C}$ (recall that if $u \in \mathcal{C}$ then $r(u) = u$).
2. $\mathcal{C}$ nodes redistribute the s entries among them so that the node with ID i stores indices $[1 + (n/n_{\mathcal{C}})(i-1), \ldots, (n/n_{\mathcal{C}})i]$ (assume $n/n_{\mathcal{C}}$ is integer).
3. Each $u \in V$ sends the index of the row of A it has to $r(u) \in \mathcal{C}$.
4. Each representative requests the $s(i)$ entries corresponding to rows $A_{i,*}$ that it represents from the $\mathcal{C}$ node storing it.
5. Each representative gets the required elements of s and sends them to the nodes in $\mathcal{P}$ it represents.
6. Each $u \in V$ sends the products $\{A(i,j)s(i)\}_{j=1}^{n}$ to its representative.
7. Each representative sends each nonzero value $A(i,j)s(i)$ it has (up to $O(kn_{\mathcal{C}})$ values) to the representative responsible for $s(j)$, so it can calculate $s'(j)$.
8. Now, each node $u \in V$ that initially stored $s(i)$, requests $s'(i)$ from its representative. The representative gets the entry from the corresponding node in $\mathcal{C}$ and sends it back to u.

We state the following results regarding the running time of Algorithm 1.

Theorem 5. *On a $\mathcal{CP}$-network $G(V, E, \mathcal{C}, \mathcal{P})$, the multiplication of a $O(k)$-sparse matrix by a vector can be completed in $O(k)$ rounds w.h.p.*

Before we start with the proof, we present the following theorem from [2].

Theorem 6 ([2]). *Consider a fully connected system of n_c nodes. Each node is given up to M_s messages to send, and each node is the destination of at most M_r messages. There exists algorithm that delivers all the messages to their destinations in $O\left(\frac{M_s+M_r}{n_c}\right)$ rounds w.h.p.*

This theorem will be extensively used by our algorithms since it gives running time bound on messages delivery in a core that satisfies Axiom $\mathcal{A}_E$. The result of the theorem holds with high probability which implies that it exploits a randomized algorithm. Nevertheless, our algorithms can be considered deterministic in the sense that all the decisions they make are deterministic. The randomness of the information delivery algorithm of Theorem 6 does not affect our algorithms since the decisions when and what message will be sent along with the message source and destination, are deterministically controlled by our algorithms.

Proof of Theorem 5. Consider Algorithm 1 and the $\mathcal{CP}$-network $G(V, E, \mathcal{C}, \mathcal{P})$. At Step 1, due to $\mathcal{A}_B$ and $\mathcal{A}_C$, each representative will obtain $O(n_c)$ entries of s in $O(1)$ rounds. For Step 2, we use Theorem 6 with the parameters: $M_s = O(n_c)$ and $M_r = O(n/n_c)$, and thus such a redistribution will take $O((n_c+n/n_c)/n_c) = O(1)$ rounds. At Step 3, due to $\mathcal{A}_B$ and $\mathcal{A}_C$ each representative will obtain $O(n_c)$ row indices of A in $O(1)$ rounds.

For Step 4, we again use Theorem 6 with the parameters: $M_s = O(n_c)$ (indices of rows each representative has), $M_r = O(n/n_c)$ (number of entries of s stored in each node in $\mathcal{C}$), and obtain the running time for this step: $O((n_c + n/n_c)/n_c) = O(1)$ rounds. At Step 5, each representative gets the required elements of s which takes running time is $O(1)$ due to Theorem 6, and then sends them to the nodes in $\mathcal{P}$ it represents which also takes $O(1)$ due to $\mathcal{A}_C$. Step 6 takes $O(k)$ rounds since A has up to k nonzero entries in each row. Step 7 again uses Theorem 6 with parameters $M_s = O(kn_c)$, $M_r = O(n/n_c)$, and thus the running time is $O(kn/n_c^2) = O(k)$.

At Step 8, a single message is sent by each node to its representative (takes $O(1)$ due to $\mathcal{A}_C$), then the requests are delivered to the appropriate nodes in $\mathcal{C}$ and the replies with the appropriate entries of s' are received back by the representatives. All this takes $O(1)$ rounds due to the Axiom $\mathcal{A}_E$ and Theorem 6. Then the entries of s' are delivered to the nodes that have requested them. Due to $\mathcal{A}_C$ this will also take $O(1)$ rounds. □

The following theorem shows the necessity of the axioms for achieving $O(k)$ running time. The proof of the theorem can be found in [17].

Theorem 7. *For each $X \in \{B, E, C\}$ there exist a family $\mathcal{F}_X = \{G_X(V, E, \mathcal{C}, \mathcal{P})(n)\}$ of partitioned networks that do not satisfy Axiom $\mathcal{A}_X$ but satisfy the other*

two axioms, and input matrices of size $n \times n$ and vectors of size n, for every n, such that the time complexity of any *algorithm for multiplying a vector by a matrix on the networks of $\mathcal{F}_X$ with the corresponding-size inputs is $\Omega(n/\log n)$.*

References

1. Lotker, Z., Patt-Shamir, B., Pavlov, E., Peleg, D.: Minimum-weight spanning tree construction in o(log log n) communication rounds. SIAM J. Computing 35(1), 120–131 (2005)
2. Lenzen, C., Wattenhofer, R.: Tight bounds for parallel randomized load balancing. In: STOC, pp. 11–20 (2011)
3. Lenzen, C.: Optimal deterministic routing and sorting on the congested clique. In: PODC, pp. 42–50 (2013)
4. Avin, C., Lotker, Z., Pignolet, Y.A., Turkel, I.: From caesar to twitter: An axiomatic approach to elites of social networks. CoRR abs/1111.3374 (2012)
5. Fujita, M., Krugman, P.R., Venables, A.J.: The spatial economy: Cities, regions, and international trade. MIT Press (2001)
6. Krugman, P.: Increasing Returns and Economic Geography. The Journal of Political Economy 99(3), 483–499 (1991)
7. Holme, P.: Core-periphery organization of complex networks. Physical Review E 72, 46111 (2005)
8. Liang, J., Kumar, R., Ross, K.W.: The fasttrack overlay: A measurement study. Computer Networks 50, 842 (2006)
9. Baset, S., Schulzrinne, H.: An analysis of the skype peer-to-peer internet telephony protocol. In: INFOCOM, pp. 1–11 (2006)
10. Attiya, H., Welch, J.: Distributed Computing: Fundamentals, Simulations and Advanced Topics. McGraw-Hill (1998)
11. Lynch, N.: Distributed Algorithms. Morgan Kaufmann (1995)
12. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM (2000)
13. Adamic, L.: The small world web. Research and Advanced Technology for Digital Libraries, 852–852 (1999)
14. Bonanno, G., Caldarelli, G., Lillo, F., Mantegna, R.: Topology of correlation-based minimal spanning trees in real and model markets. Phys. Rev. E 68 (2003)
15. Chen, C., Morris, S.: Visualizing evolving networks: Minimum spanning trees versus pathfinder networks. In: INFOVIS, pp. 67–74 (2003)
16. Lotker, Z., Patt-Shamir, B., Peleg, D.: Distributed MST for constant diameter graphs. Distributed Computing 18(6), 453–460 (2006)
17. Avin, C., Borokhovich, M., Lotker, Z., Peleg, D.: Distributed computing on core-periphery networks: Axiom-based design. CoRR abs/1404.6561 (2014)
18. Nesetril, J., Milkova, E., Nesetrilova, H.: Otakar boruvka on minimum spanning tree problem translation of both the 1926 papers, comments, history. Discrete Mathematics 233(1-3), 3–36 (2001)