2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS) | 978-1-6654-8106-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IPDPS53621.2022.00082

Topological Modeling and Parallelization of Multidimensional Data on Microelectrode Arrays

Olamide Timothy Tawose, Bin Li, Lei Yang, Feng Yan, and Dongfang Zhao
Department of Computer Science and Engineering
University of Nevada, Reno, NV 89557, United States of America

***Abstract*—Microelectrode arrays (MEAs) are physical devices widely used in various science and engineering fields. One common computational challenge when applying a high-density MEA (i.e., a larger number of wires, more accurate locations of abnormal cells) is how to efficiently compute those resistance values provided the nonlinearity of the system of equations with the unknown resistance values per the Kirchhoff law. This paper proposes an algebraic-topological model for MEAs such that we can identify the intrinsic parallelism that cannot be identified by conventional approaches. We implement a system prototype called Parma based on the proposed topological methodology. Experimental results show that Parma outperforms the state-of-the-practice in time, scalability and memory usage: the computation time is two orders of magnitude faster on up to 1,024 cores with almost linear scalability and the memory is much better utilized with proportionally less warm-up time with respect to the number of concurrent threads.**

***Index Terms*—multidimensional data, parallel processing, scientific computing, applied topology**

I. Introduction

Microelectrode arrays (MEAs) are physical devices widely used in various science and engineering fields. For example in the pandemic of COVID-19, electrode arrays were involved in both vaccine development [1] and fast testing methods [2, 3]. More conventionally: (i) in biomedical engineering [4], an MEA can be applied to a patient's wound surface and report the anomalies of the skin; (ii) in biological sciences [5], an MEA can be placed on a cell medium to electronically detect the potential cancer regions; and (iii) in electronic engineering [6, 7], similar techniques are applied for the trade-offs between currents and signals in the very-large-scale integration (VLSI) design of CPU chips. These applications utilize multidimensional arrays as the default format for storing and managing large volumes of measurement data.

One of the most notable limitations of applying MEAs lies in its scalabilty: an MEA cannot be efficiently parametrized due to the complicated, nonlinear equations. Formally, the parameterization of an MEA aims to quantify the physical resistance of the devices given four inputs: (i) the topology of the MEA, (ii) the context where the MEA is placed, e.g., cell medium, patient skin; (iii) a provided voltage, e.g., 5 volts, and (iv) measured current values in the MEA. While in the real-world applications we can easily control the voltage, accurately measure the currents, and discreetly choose the object/context, the main challenge lies in the *arbitrary topology* exhibited by MEAs: the complexity of an MEA topology can lead to considerable computational time that is considered impractical for applications. For instance, it takes hours to parameterize a two-dimensional 64×64 MEA [8]. Specifically, the electrical resistance values in the complex circuit cannot be efficiently computed due to a large number of circuits at a very fine granularity and the nonlinearity of the system of equations with the unknown resistance values per the Kirchhoff law [9]. Kirchhoff law is one of the most fundamental laws governing the physical characteristics of electrode arrays. In practice, the law is applied repeatedly to every entity in the electronic device and more importantly, the equations are correlated and thus hard to be parallelized. To make it worse, if the internal resistors are unknown, the system of equations becomes nonlinear, making the problem prohibitively expensive to solve analytically. Conventional computational approaches include Landweber method [10], linear back projection [11], and Tikhonov regularization methods [12], all of which exhibit an ill-posed computational problem [13, 14]: the solution is largely dependent on the input and results in an unacceptable variance, which hinders its adoption in practice.

From a computational point of view, researchers have recently started to seek non-analytic paradigms such as machine learning to *estimate* the solution, e.g., training a convolutional neural network to approximate the unknown resistor distribution in an electrode array [9]. This approach is demonstrated as an effective means to "learn" the nonlinear function between inputs and outputs: the error rate is as low as 0.49%. In [15], the authors demonstrated a 20×20 microelectrode array device manufactured in a wet lab, and showed that a graph-theoretical conversion from the original MEA data allowed them to efficiently utilize storage space for the expensive computation. Later, Wang et al. [8] presented a forward labeling technique for effectively training an artificial neural network (ANN) to predict the unknown variables in the 64 × 64 MEA, which was more than two orders of magnitude larger than the one shown in [9]. While the ANN can be efficiently trained, how to collect the training data, i.e., parameterizing the MEAs, at such scales pose unprecedented challenges in terms of computation cost. The problem is further exacerbated by the fact that the intrinsic parallelism, if any, does not appear observational.

While aforementioned work focuses on adopting machine learning for an estimated parametrization of MEAs, this paper tackles the MEA-parametrization problem from an orthogonal viewpoint enlightened by algebraic topology, which allows us to *algebraically* parametrize MEAs whose structure and intrinsic parallelism are hard to identify otherwise. Firstly, the seemingly complex, interconnected circuit flows among MEA nodes can be abstracted and simplified as a series of *abstract complex*, a well-studied object in algebraic topology that we will detail shortly. Secondly, we can apply homological analysis of the abstract complex and extract the independent high-dimensional circles for parallelization. We show that the algebraic objects represented by the MEA data are well defined and further, form the highly-desired topological invariant, namely *homology groups*, under rigorous group-theoretical analysis. The algebraic invariant, such as Betti numbers, allows us to employ a fine-grained parallelization technique for applying Kirchhoff's laws concurrently, each of which works by itself on a k-dimensional cycle.

To demonstrate the effectiveness of the proposed approach, we implement a new paradigm, namely Parma. Preliminary results show that the proposed approach outperforms the state-of-the-practice in various metrics: (i) the computation time is three orders of magnitude faster; (ii) the I/O time is proportionally reduced with multithreading; (iii) the memory is better utilized with proportionally less warm-up time with respect to the number of concurrent processes/threads.

In summary, this paper makes the following contributions:

- We take an algebraic-topological approach to model the parametrization of MEAs that involves computationally-intensive Kirchhoff laws; (§III)
- We propose a new parallelization paradigm, which identifies the high-dimensional "holes" that can be computed in parallel; (§IV)
- We implement a prototype system called Parma that is extensively evaluated on various test beds at large scales of up to 1,024 cores. (§V)

II. Background and Problem Formulation

A. Kirchhoff Laws and Maxwell Cyclomatic Numbers

In Kirchhoff's 1847 paper, he proved that the currents of a direct circuit could be uniquely determined by two sets of linear equations given fixed source voltage and wire resistance. The two systems of linear equations are also called the first and second Kirchhoff's laws. The first law ($L1$) states that overall flow at a specific vertex is zero, and the second law ($L2$) states that the overall voltage change along a loop of edges stays the same. If we model a circuit as a graph $G(V, E)$, then there are $|V|$ equations of $L1$, and there are $|E|$ unknown currents. It can be shown that the $|V|$ equations of $L1$ are not independent, while any $|V| - 1$ equations are indeed independent. Consequently, we need to have $|E| - |V| + 1$ or more equations from $L2$ to find the $|E|$ unknown currents. It can be further shown that these $|E| - |V| + 1$ equations from $L2$ are all independent of $|V|$ equations in $L1$, indicating that both $L1$ and $L2$ equations can collectively determine the current values. While Kirchhoff proved this for the physical case where resistances are positive real numbers, a more general case can be proven using algebraic topology, i.e., the introduction of *cochain* and *coboundary*, see [16] for more details.

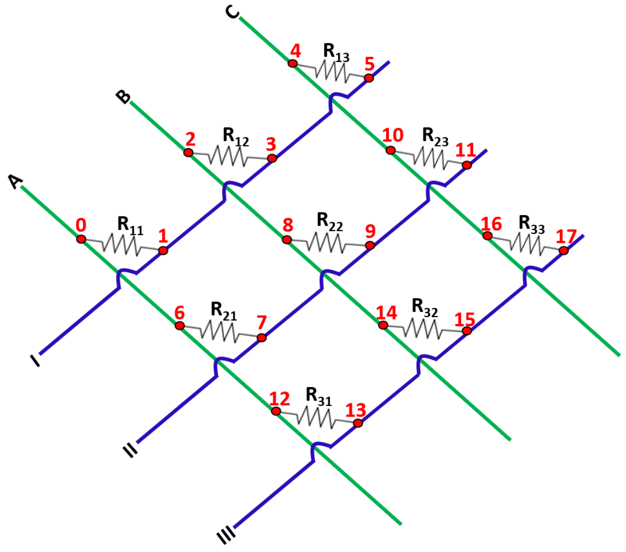


Figure 1. **Abstract architecture of a three-dimensional** 3×3 **electrode array, in a physical device.** Three horizontal wires (*A*, *B*, and *C*) and three vertical wires (*I*, *II* and *III*) are interconnected through the 18 joints $\{0, \cdots, 17\}$ and nine resistors (R_{ij}, $1 \leq i, j \leq 3$).

The number of independent loops represented by $|E| - |V| + 1$ is historically called the *cyclomatic number* by Maxwell in the context of the circuit and is an important topological property in graph theory. It should be noted, however, in many engineering applications, Kirchhoff's laws are applied indirectly: the currents are sometimes easy to measure, and it is the resistance that is of interest and yet unknown, making the systems of equations nonlinear.

B. Electrode Array and Graph Abstraction

Electrode Arrays are widely used in biomedical engineering, electrical engineering, and mechanical engineering. A typical $n \times n$ dimensional electrode array consists of a set of horizontal and vertical wires, joined through point-wise resistors. The n's scale or size highly depends on the application under consideration. For example, a continuous-flow device [5] used for the geometric screening of core/shell hydrogel microcapsules consists of 15 electrode pairs (i.e., n = 15), whereas a device designed for 2D electrical imaging surveys can consist of more than 20 electrode pairs [17]. Overall, a $n \times n$ array comprises $2n^2$ joints/junctions and n^2 resistors.

An example of such physical system is shown in Figure 1 with a size of n = 3. It consists of three horizontal wires (*A*, *B*, and *C*) and three vertical wires (*I*, *II* and *III*) connected with nine resistors resulting into a total of 18 joints $\{0, \cdots, 17\}$.

In general, a $n \times n$ electrode array can be abstracted into a two-dimensional graph where each vertex represents a resistor, as shown in Figure 2. In practice, the physical device of an electrode array is usually in a square shape, although the following discussion can be trivially extended to arbitrary shapes $m \times n$ ($m \neq n$).

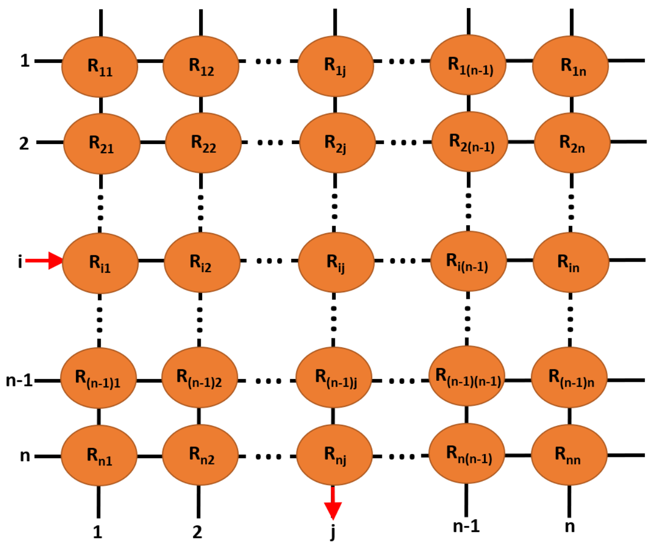


Figure 2. Abstraction of a general electrode array in two-dimensional space.

C. *Anomaly Detection through Electrode Arrays*

To put it in the real-world context, the electrode device is used for (almost) real-time anomaly detection such as the wound surface of an injured athlete. A common workload of such a $n \times n$ electrode array system is to find out the unknown resistances R_{ij}'s ($1 \leq i, j \leq n$) given the pair-wise measured resistance Z_{ij}'s between the end-points of n horizontal and n vertical wires. These R_{ij}'s are usually equal and negligible in values from the device. When the tested medium exhibits anomaly areas (e.g., cancer cells), the local resistance (i.e., R) will significantly increase. Therefore, we have n^2 unknowns (R's) and also n^2 measured values (Z's), namely $Z_{A,I}$, $Z_{B,I}$, $Z_{C,I}$, $Z_{A,II}$, $Z_{B,II}$, $Z_{C,II}$, $Z_{A,III}$, $Z_{B,III}$, and $Z_{C,III}$ in the example shown in Figure 1.

The challenge here lies in that the measured resistance between two endpoints is a nonlinear function of all the n^2 unknowns through many possible paths; To see this, take $Z_{B,III}$ for example, the most straightforward circuit is through R_{32} (between endpoints 14 and 15). And yet, there are other circuits flowing through, one possible path being:

$$B \rightarrow 8 \xrightarrow{R_{22}} 9 \rightarrow 7 \xrightarrow{R_{21}} 6 \rightarrow 12 \xrightarrow{R_{33}} 13 \rightarrow III$$

More examples from other pairs of endpoints include (cf. Figure 1):

$$C \rightarrow 16 \xrightarrow{R_{33}} 17 \rightarrow 15 \xrightarrow{R_{32}} 14 \rightarrow 2 \xrightarrow{R_{12}} 3 \rightarrow I$$
$$A \rightarrow 12 \xrightarrow{R_{31}} 13 \rightarrow III$$
$$C \rightarrow 10 \xrightarrow{R_{23}} 11 \rightarrow 9 \xrightarrow{R_{22}} 8 \rightarrow 14 \xrightarrow{R_{32}} 15 \rightarrow III$$
$$B \rightarrow 2 \xrightarrow{R_{12}} 3 \rightarrow 5 \xrightarrow{R_{13}} 4 \rightarrow 10 \xrightarrow{R_{23}} 11 \rightarrow II$$
$$A \rightarrow 0 \xrightarrow{R_{11}} 1 \rightarrow 3 \xrightarrow{R_{12}} 2 \rightarrow 8 \xrightarrow{R_{22}} 9 \rightarrow II$$

For a $n \times n$ array, there are overall $n^{(n+1)}$ possible paths. To see this, we can start with a specific pair of endpoints. Whenever the circuit flows from one joint to the next step, there are n possible choices. In total, there are only $(n-1)$ steps between the source and the destination. Therefore, there are $n^{(n-1)}$ possibilities between any pair of endpoints. Note that there are a total n^2 pairs of endpoints. Consequently, the total number of paths is $n^{(n-1)} \cdot n^2 = n^{(n+1)}$. To save all of these possible paths, the required space is even larger than the n exponential because each path has to store all the joint numbers as well. In [15], authors reported that the data growth is so fast that the path-based approach is unfeasible on mainstream computer hardware and systems when $n > 6$.

If we assume the paths can be stored efficiently (which is true only for small n's), then the question is how to find those paths efficiently. This is a classical problem in graph theory, which is solvable using either depth-first or breadth-first recursive traversal algorithms. Since the number of possible paths is exponential, any algorithm for finding them must be at least exponential, which is indeed the case of both depth-first and breadth-first recursive traversal algorithms. To see this, again, in the case of breadth-first recursion, there are n neighbors to the current position, and each of the n neighbors might lead to a depth of $(n-1)$, resulting in the total of an exponential number of recursion calls.

After finding out and storing the paths, the next step is to solve the equations built upon the paths satisfying the constraints, i.e., the total incoming circuit flow is equal to the total outgoing circuit flow according to the Kirchhoff law. In essence, all the paths are considered as the parallel circuit flows between two endpoints and can be aggregated through this form:

$$Z_{ij}^{-1} = \sum_{k=1}^{n^{(n-1)}} P_k^{-1}(R)$$

where $P_k(R)$ indicates the summation of resistors along the k-th path between the i-th horizontal wire and the j-th vertical wire in a $n \times n$ array. Therefore, the goal is to solve a system of n^2 nonlinear equations, each of which comprises an exponential number of terms, and each term exhibits a summation of selected unknowns as the divisor. This equation-solving procedure itself is also compute-intensive, requiring iterative method to find roots of the unknown resistors. The state-of-the-art is to leverage deep learning to estimate the unknowns, e.g., conventional neural networks [9]. Once the R values are known (or, estimated), the anomaly can be simply detected.

III. Algebraic-Topological Modeling of MEAs

This section will first briefly review the basics of algebraic topology and show the natural correspondence between MEAs and the topological objects such as simplex and simplicial complex. We will then demonstrate that this correspondence is mathematically sound, based on which of those topological objects can form more sophisticated ones that exhibit strong and otherwise unnoticeable algebraic invariant, including but not limited to homology groups and Betti numbers. As a result, the proposed modeling and analysis lead to a new parallelization paradigm that will be discussed in §IV.

A. Topology Basics

Mathematically speaking, a *topology* of a set S is a collection of subsets of S, denoted $\mathcal{T}$, satisfying certain properties[1] that distinguish a topology from the set of *hyperedges* in a *hypergraph* [18]. One example topology of S is then the power set of S, $\mathcal{P}(S)$, which consists of all the possible $2^{|S|}$ subsets of S. This is also called the *discrete topology* of S. The tuple $(S, \mathcal{T})$ is called the *topological space* of S. Each of the subsets U from $\mathcal{T}$ is called an *open set*, and the complement set $S \setminus U$ is a *closed set* by definition. A function g from space X to Y is called *continuous* if $\forall v$ is an open set in Y, then $g^{-1}(v)$ is an open set in X. The composition of two continuous functions is also continuous. If both g and g^{-1} are continuous and bijective (one-on-one mapping), we call g a *homeomorphism*. Because a homeomorphism is defined purely on open and closed sets, two topological spaces are considered equivalent if such a homeomorphism exists. Usually, we expect to migrate a complex problem in one topological space to another such that the problem can be solved more efficiently or more intuitively. The aforementioned concepts and techniques are also referred to as *point-set topology*.

In addition to point-set topology, there is another branch of *algebraic-topological* methods that study *homotopy* groups and *homology* groups. Informally, these groups break the complex down into the smaller pieces and map the geometrical objects into algebraic objects, such as *groups*. Some of the hardest problems were shown to be elegantly solvable through algebraic topology [19]. Remarkably, a unique subbranch of topology, namely *combinatorial topology*, specifically studied the topological properties of distributed computing models [20, 21].

The building blocks we are interested in for topological modeling of MEAs are called *simplices* (the plural form of simplex). In this work, by *simplex* σ we mean an *abstract simplex*, defined as a set S of vertices. Any subset of σ is also a simplex, and is called a *face* of σ. The dimension of σ is defined as the number of vertices minus 1:

$$\mathtt{dim}\,\sigma = |\sigma| - 1.$$

Geometrically, a simplex σ consists of all the possible points, edges, triangles, tetrahedrons, and higher-dimensional objects that can be composed of the vertices in S. From a combinatorial perspective, a collection of σ's can be thought of as an object representing more sophisticated relationships among the vertices in S, which is called an *abstract simplicial complex*, denoted K. The dimension of a complex is defined as the highest dimension from any simplex in the complex:

$$\mathtt{dim}\,K = \mathtt{max}(\mathtt{dim}\,\sigma), \forall \sigma \in K.$$

It is "simplicial" in the sense that any $\sigma_1 \cap \sigma_2 \in \sigma_1, \sigma_2$, meaning that the simplices (including the empty set $\emptyset$) shared

[1]Namely, both $\emptyset$ and S are in $\mathcal{T}$, a finite number of intersections of elements in $\mathcal{T}$ is in $\mathcal{T}$, and any union of elements in $\mathcal{T}$ is in $\mathcal{T}$.

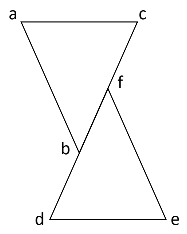


Figure 3. A polyhedron of two simplices (triangles $\{a, b, c\}$ and $\{d, e, f\}$) that is *not* a simplicial complex. The overlap of two triangles is segment $\{b, f\}$, which is not an element of the set of 1-simplices $\{\{a, b\}, \{b, c\}, \{a, c\}, \{d, e\}, \{d, f\}, \{e, f\}\}$.

by σ_1 and σ_2 must also be valid simplices of both σ_1 and σ_2. This requirement might sound self-evident, but actually might be violated in practice: Figure 3 shows that the shared line segment $\{b, f\}$ is not an element of

$$\{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}.$$

B. Modeling through Homology Groups

It follows that an MEA can be represented by an abstract simplicial complex, or *complex* if no ambiguity arises. Before we go on the discussion on modeling MEAs with complexes, we need to verify that the MEA can be indeed abstracted as a complex, per the above definition.

Proposition 1. *Every microelectrode array is an abstract simplicial complex with the set of vertices represented by the joints between wires.*

Proof. We will prove this for the two-dimensional case; higher-dimensional cases follow similarly.

First, we show that the dimension of a 2-dimensional MEA is one. Let P denote the polyhedron of the MEA object. We will show that the dimensions of all simplices are (i) larger than or equal to one and (ii) smaller than or equal to one, both of which will collectively prove our claim. For (i), suppose, for contradiction, that $\mathtt{dim}\,P < 1$, that is, $\mathtt{dim}\,P = 0$. However, a 0-dimensional complex has only vertices without any edges, indicating an MEA with joints without wires, which is impossible, thus a contradiction. For (ii), suppose, again for contradiction, that $\mathtt{dim}\,P > 1$. We will use induction to show that $\mathtt{dim}\,P$ cannot be any numbers larger than 1. We first check $\mathtt{dim}\,P \neq k$, $k = 2$. This is easy to verify since if $\mathtt{dim}\,P = 2$, there must be at least one triangle in P, whose dimension is 2. However, in 2-dimensional MEAs, there are only vertical and horizontal wires, and forming triangles is not possible. Now we start checking $k + 1$. Recall that by the definition of simplex, any subset of a simplex (i.e., a face) is again a simplex. It follows that if γ is a $(k + 1)$-dimensional simplex, then its subset, say a k-dimensional simplex σ must be a simplex. But we just show that a simplex cannot have dimension k, starting $k = 2$, leading to a contradiction.

Second, we show that any shared portion between two simplices in an MEA is a face of both simplices. Because the dimension of a 2-dimensional MEA is 1, as shown above, we only need to verify that the shared simplex is either a common vertex or a shared edge. It is trivial to check the shared edge, however, because that would indicate that the two wire-segments overlay each other. Therefore, we only need to show that the only intersection between any two edges is their joint (or nothing if they are parallel, which will be covered at the end of this proof). This is indeed the case because otherwise, the two segments would touch each other on two endpoints, making them identical. Lastly, to complete the proof, if two simplices do not share any simplex (e.g., they are parallel wires along the vertical or horizontal axes), then their intersection is $\emptyset$ and belongs to both simplices. $\square$

Having shown that an MEA is a complex allows us to explore strong properties that have been extensively studied in algebraic topology. Recall that Kirchhoff's laws say that the voltage changes over the "loop" of a circuit. This property can be accurately and efficiently characterized by the topological invariant called *homology groups*. We do not have space to elaborate either *homology* or *groups*, introductory texts on these topics can be found in [22, 23]. In the following, we will give a very brief overview of the concepts when they are absolutely necessary for our discussion. A *group* is a set G along with a binary operation $\star$ between two elements in G such that $\star$ is closed and associative in G, and there is a special identity element $e \in G$ such that any element has a counterpart to which its multiplication equals e:

$$\forall g, h, k \in G, g \star h \in G, (g \star h) \star k = g \star (h \star k), g \star e = g,$$
$$\exists g^{-1} \in G \texttt{ such that } g \star g^{-1} = e.$$

Now, think of the set C consists of all the possible subsets of line segments in an MEA, and let us define the binary operation between *any* pair of subsets as modulo-2 inclusion, meaning that any duplicate simplices will cancel out. So, two 1-dimensional simplices (i.e., edges), say $\sigma_1 = \{a, b\} \in G$ and $\sigma_2 = \{b, c\} \in G$, can be calculated as

$$\sigma_1 \star \sigma_2 = \{a, c\}.$$

This group is called the *complex chain* group in the literature of algebraic topology, usually denoted C. Obviously, some elements of C are cycles and others are not; for example in Figure 1, a sequence $0 \to R_{11} \to 1 \to 3 \to R_{12} \to 2 \to 8 \to R_{22} \to 9 \to 7 \to R_{21} \to 6 \to 0$ is a cycle. We are interested in this subset of cycles, denoted D, because they are closely related to the Kirchhoff laws. Obviously, if we apply the defined modulo-2 operation along the cycle, the eventual result would be empty (i.e., the identity element in C); in fact, there is another name to summarize the series of modulo-2 operations above, *boundary*[2], denoted δ. It is easy to verify that the boundary δ can map the set of $k+1$-dimensional simplices into k-dimensional simplices, which results in the following sequence:

$$\cdots \xrightarrow{\delta} C^k \xrightarrow{\delta} C^{k-1} \xrightarrow{\delta} \cdots C^1 \xrightarrow{\delta} C^0,$$

where C^k denotes a k-dimensional complex chain, or a k-chain group. The result, or *image*, of the δ operation, is a subset of simplices, called the *boundary group*, denoted B^k, and is called k-boundary group at dimension k. In group theory, the *preimage* of the empty image, $\delta^{-1}(e)$, is called the *kernel* of the map δ; therefore, D^k is the kernel of δ whose result is $e \in C^{k-1}$. D^k is called the k-cycle group.

[2]There is a more formal definition of the boundary operation in algebraic topology; we do not mention it as it has no direct implication to our discussion.

We can then define $H^k = D^k / B^k$, the quotient group at each dimension, which also compose a series of groups, also called the *homology groups*. The order, or cardinality, i.e., the number of elements, of these quotient groups can be calculated as:

$$|H^k| = |D^k| / |B^k|,$$

according to the Lagrange Law (in group theory). Because the chains of groups in simplicial complexes are defined under the modulo-2 operation, the number of involved simplices is $\log |H^k|$, which is defined as the *rank* of a group, or the *Betti number* for H^k, denoted β_k, which can be efficiently calculated as:

$$\begin{aligned}\beta_k &= \texttt{rank}(H^k) = \log |H^k| = \log \left(|D^k| / |B^k| \right) \\ &= \log |D^k| - \log |B^k| = \texttt{rank}(D^k) - \texttt{rank}(B^k).\end{aligned}$$

Betti number implies the number of k-dimensional "basic" hole embedded in the topology of the MEA data; by "basic", we mean that the hole is not a composition of other holes. In our MEA applications, the Betti number implies the parallelism for applying Kirchhoff's laws concurrently.

IV. Parallel Processing of Multidimensional Electrode Arrays using Algebraic Invariant

This section presents the potential parallelism enabled by the algebraic invariant we developed in §III. For completeness, we will first briefly review the baseline approach that is built upon the vertex-correlation in graph theory [15]. Then, we describe how to apply work-stealing to improve the parallelism. Finally, we show that the parallelism exhibited by algebraic invariant can be naturally leveraged by popular paradigms such as multithreading and multiprocessing across nodes.

A. Categorizing Vertex-oriented Constraints

Due to the existence of redundant or several possible sub-paths between distinct pair of end points, we propose a new approach that is not tightly correlated to the $n^{(n+1)}$ paths between end points. Instead, we concentrate just on the n joints and attempt to translate a set of paths into a set of joints while preserving all topological features. To put it another way, our aim is to reduce the number of constraints from $O(n^n)$ to $O(n^c)$[3] without losing any information, i.e., lossless conversion. Such conversion is only achievable if we can somehow express the partially redundant paths as a single joint. The key idea is inspired by the observation that many distinct end-to-end paths take the same sub-paths for a lot of times. Hence, we try to reconstruct a different/equivalent graph topology for the $n \times n$.

Figure 4 shows a concrete example of converting all feasible paths (i.e., 9 paths) between two end points C and I in Figure 1

[3]We use c to denote a constant number.

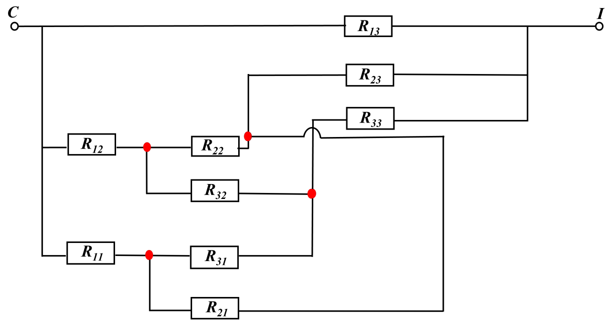


Figure 4. **A corresponding topology between C and I as Figure 1.** There also exists nine paths between C and I, which are semantically equivalent to Figure 1 but with possible loops across sub-paths.

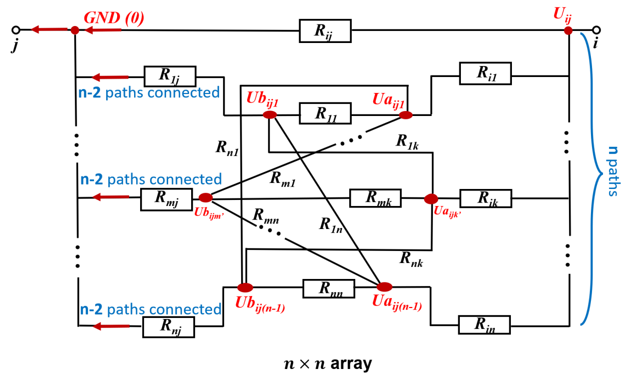


Figure 5. **A corresponding topology between the i-th horizontal axis and the j-th vertical axis in a $n \times n$ Array, in a 2D space.** There are total $(n-1)$ Ua's and $(n-1)$ Ub's as two sets of distinct voltage values for a specific pair of ij end points, assuming the original voltage can be measured between i and j. The conditions of subscripts are: $k \in \{1, \cdots, j-1, j+1, \cdots, n\}$; $m \in \{1, \cdots, i-1, i+1, \cdots, n\}$; $k' = k$ if $k \leq j$, $k' = (k-1)$ otherwise; and $m' = m$ if $m \leq i$, $m' = (m-1)$ otherwise.

where n = 3. We identify the following nine paths from C to I to verify that the transformed topology and the original array are equal.

(i) $C \to R_{13} \to I$
(ii) $C \to R_{13} \to R_{23} \to R_{21} \to R_{31} \to R_{33} \to I$
(iii) $C \to R_{13} \to R_{33} \to R_{32} \to R_{22} \to R_{23} \to I$
(iv) $C \to R_{12} \to R_{22} \to R_{23} \to I$
(v) $C \to R_{12} \to R_{32} \to R_{33} \to I$
(vi) $C \to R_{12} \to R_{22} \to R_{21} \to R_{31} \to R_{33} \to I$
(vii) $C \to R_{11} \to R_{31} \to R_{33} \to I$
(viii) $C \to R_{11} \to R_{21} \to R_{23} \to I$
(ix) $C \to R_{11} \to R_{31} \to R_{32} \to R_{22} \to R_{23} \to I$

For an arbitrary pair of endpoints between the i-th horizontal wire and the j-th vertical wire as shown in Figure 2, the equivalent topology can be expressed as Figure 5. In essence, the most straightforward path between i and j goes only through R_{ij}, shown as the top path (or the top main route) in the figure. Then, there are $(n-1)$ main routes starting with R_{ik} where $k \in \{1, \cdots, j-1, j+1, \cdots, n\}$, corresponding to the first set of voltage values called $Ua_{ijk'}$ where $k' = k$ if $k \leq j$ and $k' = (k-1)$ otherwise. Similarly, toward the end of each main route, there are $(n-1)$ R_{mj}'s where $m \in \{1, \cdots, i-1, i+1, \cdots, n\}$ and $(n-1)$ $Ub_{ijm'}$'s where $m' = m$ if $m \leq i$ and $m' = (m-1)$ otherwise. We do not assign a variable to the end-to-end voltage between i and j (i.e., U_{ij}) because it can be easily measured in practice. Both Ua and Ub have three subscripts, with i and j indicating the two endpoints and the third subscript indicating the top-down ordering of those voltage values from 1 to $(n-1)$. As we will see soon, this equivalent topology would yield a polynomial number of equations by enforcing the constraints on those $2(n-1)$ voltage points Ua's and Ub's, as opposed to an exponential number of equations as discussed before.

Given the equivalent topology, we are able to enforce the constraints on the joints (i, j, Ua's, and Ub's) instead of the paths. The saving is significant: for each pair of endpoints, there are $2n$ joints (1 at i, 1 at j, $(n-1)$ at Ua's, and $(n-1)$ at Ub's) and $n^{(n-1)}$ paths; or for the entire system, there are a polynomial number $2n \cdot n^2 = O(n^3)$ of joints and exponential number $n^{(n-1)} \cdot n^2 = O(n^n)$ of paths. The following of this section explains how we generate the equations on those $2n$ joints for a pair of endpoints i and j. The $2n$ equations (satisfying the circuit flow constraints by the Kirchhoff Law) for each pair of endpoints (i, j) are defined as follows:

$$\begin{cases} \frac{U_{ij}}{Z_{ij}} = \frac{U_{ij}}{R_{ij}} + \sum_k \frac{U_{ij}-U_{ijk'}}{R_{ik}}, & \texttt{\# One equation at } i \\ \frac{U_{ij}}{Z_{ij}} = \frac{U_{ij}}{R_{ij}} + \sum_m \frac{U_{ijm'}}{R_{mj}}, & \texttt{\# One equation at } j \\ \frac{U_{ij}-U_{ijk'}}{R_{ik}} = \sum_k \frac{U_{ijk'}-U_{ijm'}}{R_{mk}}, & \texttt{\# n-1 eq.'s for } Ua \\ \frac{U_{ijm'}}{R_{mj}} = \sum_m \frac{U_{ijk'}-U_{ijm'}}{R_{mk}}, & \texttt{\# n-1 eq.'s for } Ub \end{cases}$$

where (i) $k' = k$ if $k \leq j$ and $k' = (k-1)$ otherwise; and (ii) $m' = m$ if $m \leq i$ and $m' = (m-1)$ otherwise. For the entire array, there are $(n-1) \cdot n^2$ unknown Ua's, $(n-1) \cdot n^2$ unknown Ub's, and n^2 unknown R's; all U_{ij}'s and Z_{ij}'s are measured values. The total number of nonlinear equations for the entire $n \times n$ array is $2n^3$, with $(2n-1) \cdot n^2$ unknowns. Although a system of nonlinear equations does not guarantee unique or sensible roots (for instance, resistance cannot be non-positive values), specifying a practical and positive U_{ij} value usually precludes the problem, which is out of the scope of this paper.

Obviously, all the joints[4] can be categorized into four groups: (i) source points with 1-to-n flow constraints; (ii) destination points with n-to-1 flow constraints; (iii) intermediate points close to the source, with 1-to-n flow constraints; and (iv) intermediate points close to the destination, with n-to-1 flow constraints. Each of these four types is independent of the others, thanks to the resistors in-between. Therefore, the baseline implementation for parallelization is to assign a dedicated thread to each of the aforementioned four constraint types. We will refer to this parallelization simply as *parallel* in the following discussion and evaluation.

In *Parallel*, we are restricted from having more than four threads or processes to parallelize the entire set of equations. As we will see in §V, four threads will not saturate the

[4]Or, equivalently, vertices in the original array.

optimization room in this case. Another limitation of *Parallel* is that users will have to manually split the original system of nonlinear equations into sub-systems, which might represent a technical barrier for end-users without a deep programming background.

B. Parallelization on MEA Manifolds

From a geometric point of view, the circuit flows in an MEA can be thought of as in a *vector field* of the MEA *manifold*, if we consider the MEA device is sufficiently "dense" or "smooth" at a local region. By vector field, we mean a function from an n-dimensional point p to a vector v_p eminated from p; by manifold, we mean an arbitrary space where each sufficiently small region is isometric to a Euclidean space. Then, it is a well-known result from differential geometry that the circuit accumulation, i.e., calculus, can be efficiently computed with the local tangent spaces (along with associated metrics such as *normal* vectors) and drop the global (Euclidean) coordinates. This observation has a deeper implication than it seems: Because calculus can be applied with the local parameters, which are collectively called a *frame* in the literature of differential geometry, we can parallelize the computation at a finer granularity.

One advantage of adopting such a differential-geometric approach is the removal of some constraints on manufacturing MEAs. For example, our current 2-dimensional MEA device is an equidistant grid (see Figure 1) with orthogonal wires. With the introduction of frames, we can adopt the Jacobian matrix to covert any arbitrary MEA into a locally orthogonal frame for parallel computation on the directions of partial derivatives. That is, let $U_{i,j}$ denote the voltage value at a specific node, then elementary calculus on Euclidean space $\mathbb{R}^n$ tells us

$$\frac{\partial^2 U_{i,j}}{\partial x \partial y} = \frac{\partial^2 U_{i,j}}{\partial y \partial x},$$

where x and y represents the two orthogonal axes in $\mathbb{R}^2$, and in a manifold the change of $U_{i,j}$, denoted $D(U)$, can be calculated as

$$D(U) = \begin{bmatrix} \frac{\partial U_i}{\partial x} & \frac{\partial U_i}{\partial y} \\ \frac{\partial U_j}{\partial x} & \frac{\partial U_j}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} dx \\ dy \end{bmatrix},$$

which can then be plugged into the usual vector calculus and possibly calculate the voltage change along the wires by applying Stokes' theorem to the voltages:

$$\int_{xy-boundary} U = \iint_{xy-patch} D(U).$$

The above discussion shows that as long as the smoothness assumption holds, we can efficiently parametrize MEAs with local voltage values in parallel. In practice, although the spatial gap among MEA nodes is not negligible, we can repeat the measurement and consider the vector of repeated measurements as a more realistic manifold. The practicality of MEA manifolds depends on the nature of the MEA applications. That is, if the voltage change is *continuous*, meaning that there is no "abrupt" change exhibited in the application, then U is evidently differentible and integrable. In a microelectronic setup, it is usually assumed that the voltage change is continuous [24].

In §IV-A, we present a method taking a polynomial time (i.e., $O(n^c)$, c is a constant number, n is the number of endpoints in the MEA) to parametrize MEAs at joints rather than paths. We demonstrate that $c = 3$ for a two-dimensional MEA; the complexity can be trivially generalized into $O(n^{k+1})$ for an arbitrary k-dimensional MEA. With the topological parallelization introduced in this section, we can further improve the asymptotic time cost by paralleling the parameterization for the homology groups, or visually speaking, the "holes". In an k-dimensional equidistant MEA, that means we could further improve the parallelism by $(n-1)^k$-fold. Therefore, the overall complexity for parametrizing a k-dimensional MEA could be theoretically reduced to

$$\frac{O(n^{k+1})}{(n-1)^k} = \frac{O(n^{k+1})}{O(n^k)} = O(n).$$

That is, we would be able to achieve a method linear in time for MEA parametrization as long as the device is "smooth" enough, by which we mean the fact that the MEA has sufficiently dense endpoints being concurrently worked by a sufficiently large number of processes. This will be experimentally demonstrated in the evaluation, e.g., cf. Fig. 9 and Fig. 10.

C. System Optimization

1) Balanced Parallel: An improved parallelization can be achieved by balancing the workload through work-stealing. If we closely examine the four constraint categories, two of them comprise a lot more constraints: the number of sources and destination joints is n, while two intermediate types are $n^2 \cdot (n-1)$—roughly in the cubic order of the former. Therefore, in this optimization, we allow threads to continue working on other tasks instead of waiting idly. In theory, this approach could help reduce the end-to-end execution time if the overhead of switching threads is nicely controlled. We will refer to this implementation as *Balanced Parallel*.

It should be clear that, however, our implementation takes a deterministic approach to balance the workload rather than making the decision at runtime, which is stochastic. Determinacy, however, is a double-edged sword: it helps reduces the runtime overhead of switching threads, and yet might hurt the flexibility in practice, especially for large-scale applications. In later evaluations, we will see that *Balanced Parallel* achieves the highest performance at small scales and yet delivers suboptimal performance at larger scales. If we step back and look at the big picture, *Balanced Parallel* described here still falls into the category of coarse-grained parallelization.

2) Fine-grained Multiprocessing (PyMP-k).: Automatic multiprocessing (e.g., OpenMP) is designed for well-structured loops, which is, unfortunately, not the case in electrode arrays. First, the four constraint types cannot be programmatically expressed in the same loop. Second, the electrode data are highly skewed with two hefty tasks compared to others.

To leverage the OpenMP-like parallelization, we implement the Betti-number-aware multiprocessing approach by pushing the parallelization into each of the k-dimensional loops. That is, in addition to the parallelization between constraint types, we now enable the intra-type parallelism regardless of the constraint type. The downside is, however, for small n of lightweight constraints, the efficiency might be low due to the small workload (compared to the overhead). If the dominant workloads are at large scales, the performance gain might outweigh the low efficiency from lightweight workloads. As we will see this in the evaluation section, Parma incorporated with an OpenMP-like library, PyMP[5], taking the aforementioned approach delivers the highest performance at large scales up to 100×100 arrays. PyMP utilizes its work-sharing constructs to enable load balancing among processes. Constructs take an amount of work and distribute it over the specified number of processes in a parallel region.

The above approach can be extended to multiple nodes, e.g., being implemented with MPI. In general, the overhead across nodes (e.g., I/O cost of message passing) is higher than the parallelization within a physical node. Therefore, inter-node parallelization is preferable only when the workload share per process is significantly higher than the amortized overhead. We will quantify the workload impact to the performance of different scales (up to 1,024 processes) in the evaluation section.

V. Implementation and Evaluation

Our evaluation focuses on three metrics: the computation time (§V-C), the memory footprint (§V-D), the I/O cost (§V-E), and the scalability (§V-F). Three baseline systems are used when applicable: (i) *Single-thread*: the serialized implementation of MEA analysis as in the literature [15] , (ii) *Parallel*: the naive parallel processing based on vertex-correlation [8], and (iii) *Balanced Parallel*: a work-stealing approach based on *Parallel* that we discuss in this paper (§IV).

A. Implementation

We have implemented the proposed parallelization methods with Python v3.7.0, PyMP v0.4.2. Our whole framework is implemented in Python because the state-of-the-art system [15] upon which ours is built was implemented in Python. There are about 2,600 lines of Python code and other scripts (BASH, R, etc.) in our current implementation, which can be downloaded from the project online repository.

The current implementation comprises two main parts:

- MEA: This component converts the original exponential all-pair-path problems into polynomial ones.
- Parma: This component applies various optimizations to parallelize the formation of the system of nonlinear equations.

We have evaluated the system prototype on up to 100×100 arrays or end points. The electrode array hardware comprised 64×64 wires built in the wet lab of our collaborators from the Department of Biomedical Engineering. The environment can be conveniently set up using popular Python frameworks such as Anaconda.

[5]https://github.com/classner/pymp

B. Experimental Setup

Our test bed consists of an on-premises system comprised of a many-core server i.e., HP Z820 server and a high-performance computing (HPC) cluster:

1) The HP Z820 server has 32 Intel Xeon E5-2670 cores, 128 GB RAM, a 500 GB SSD, and a 2 TB HDD; and
2) The high-performance computing (HPC) cluster is comprised of 58 nodes interconnected with FDR InfiniBand. Each node is equipped with an Intel Core-i7 2.6 GHz 32-core CPU along with 296 GB 2400 MHz DDR4 memory. It has a remote 2.1 PB storage system managed by GPFS [25]. We use up to 32 nodes, or 1,024 cores, in the following experiments.

All test beds are installed with Ubuntu 16.04, Python 3.7.0, NumPy 1.15.4, SciPy 0.17.0, PyMP v0.4.2, mpi4py v2.0.0, and mpich2 v1.4.1.The performance results we have obtained and illustrated graphically are an average of multiple trials.

All of the experimental data (up to 100×100) are obtained from a microelectrode array device measuring (unknown) numbers of cells atop their media at a wet lab from the Department of Biomedical Engineering. The data are originally saved as Excel files and converted into text files before being fed to the Parma system prototype. The data at the wet lab are measured four times a day: 0 hour, 6 hour, 12 hour, and 24 hour, after the device setup is completed. The resistance values of cells range between 2,000 and 11,000 Kilohm, while the electrical voltage is 5 volts.

C. Computation Time

In Figure 6, we report the performance of three parallelization optimizations applied to Parma. The experiments were carried out on the on-premises system. `PyMP` delivers the highest performance at scales $n \geq 20$, despite of lower performance than `Balanced Parallel` at $n = 10$ where the parallelization overhead outweighs the speedup. Since `PyMP` seems to perform best at larger scales, which is not surprising as it offers fine-grained parallelism, the remainder of this subsection will further investigate the properties of `PyMP` in more detail unless otherwise noted.

In addition, we report the overall compute time at various levels of parallelism $k \in \{2, \cdots, 32\}$ in PyMP without the I/O time in Figure 7. The experiments were carried out on the HPC cluster. It can be observed that the improvement in performance or speedup becomes more significant at scales $n \geq 20$ for the various levels of parallelism $k \in \{2, \cdots, 32\}$ in PyMP despite of the inconsistent performance when $n = 10$.

D. Memory Footprint

We report the memory characterization at various scales, as reported in Figure 8. For all the scales of $n \in \{10..100\}$, the peak memory usage is about the same regardless of data parallelism. However, a higher parallelism on large scales ($n \geq$

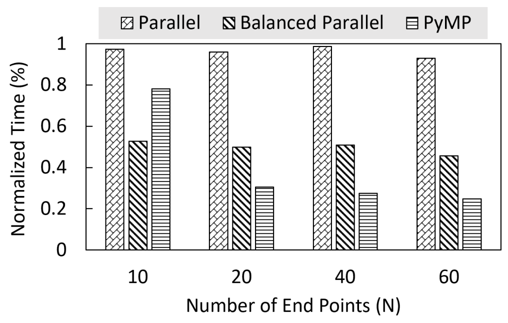


Figure 6. **Various approaches for parallel formulation of Kirchhoff law equations.** The proposed *joint constraints* enable various parallelization possibilities, namely *Parallel*, *Balanced Parallel*, and *PyMP*. *PyMP* delivers the highest performance at scales $n \geq 20$, despite of lower performance than *Balanced Parallel* at $n = 10$ where the parallelization overhead outweighs the speedup.

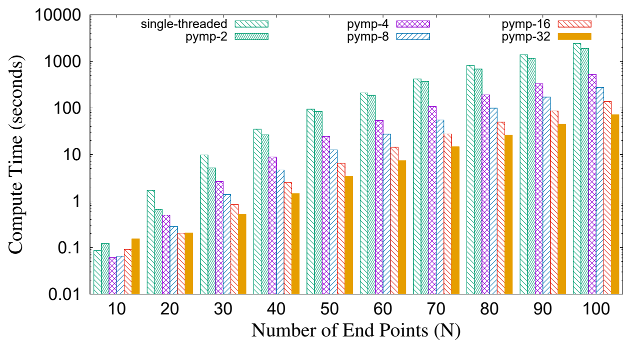


Figure 7. **Computation time of various parallelism in PyMP.** Applying fine-grained multiprocessing leads to a linear decrease in the overall compute time per workload at scales $n \geq 20$.

40) implies a higher utilization of the memory: for instance, two threads ($k = 2$) on a 100×100 array ($n = 100$) incur a low memory footprint in about 60% of time while four threads ($k = 4$) incur the same memory footprint only in about 30% of time. At small scales ($n \leq 20$), little difference is observed. The memory usage is proportional to the rank of n and is under 20 GB for a 100 × 100 array.

This experiment shows that there is negligible memory overhead while improving temporal performance due to the spawning of new processes for any selected number of end-points or scales. For each selected scale, the peak memory usage for a different number of threads remains almost the same.

E. I/O Cost

We report the overall time taken to generate the set of equations and write them to a file in disk with Parma. The experiments were carried out on the HPC cluster. Figure 9 shows the results at up to $n = 100$. In comparison with results reported in Figure 7, the time taken to write the set of equations to disk exhibit noticeable differences at scales $n \geq 20$ for threads at various levels of parallelism. The results confirm our conjecture that spawning more threads is preferable for larger workloads such that the overhead can be amortized.

F. Scalability

We report the scalability of Parma in terms of spawning more processes as reported in Figure 10. Due to a maximum number of 32 physical cores on a single server, we have implemented the topological parallelism with MPI. We deploy the MPI implementation on up to 1,024 cores, and observe a linear strong scalability for practical workloads (e.g., 50×50 or larger MEAs). For smaller workloads (e.g., 10×10 and 20×20 MEAs), the inter-node parallelism is not effective and an intra-node parallelization (e.g., OpenMP) is recommended.

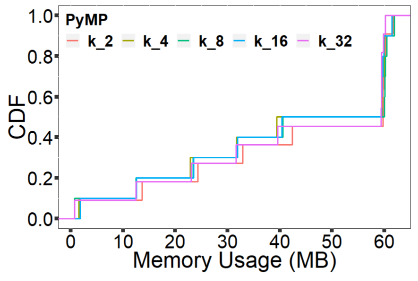
VI. Related Work

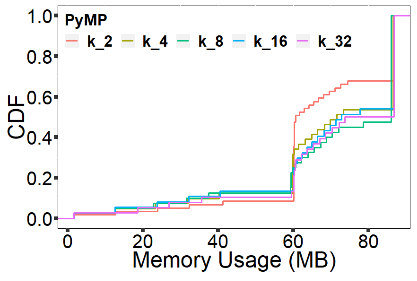
Loke et al. [17] proposed techniques for the fast computation of electrode arrays for two-dimensional (2D) resistivity surveys. An automatic graph-based method [26] was proposed for localizing distantly-spaced cochlear implant electrode arrays in clinical computed tomography with sub-voxel accuracy. In [27], a method based on the concept of Space-Amplitude Transform was proposed to transform time recordings from a 2D electrode array as a one-dimensional (1D) plus time signals in order to speed up and make simpler the data analysis. Kiele et al. presented the principles for a robust and precise alignment monitoring system, which allows the detection of linear and rotational displacements of two parallel electrode arrays [28]. In [29], finite element method (FEM) modeling was proposed for studying the impact of simultaneous impedance measurement of 100 electrodes of a Utah Electrode Array (UEA). Yassin et al. [30] proposed an energy-efficient spike data extraction solution for a high-density electrode array capable of reducing the data to be transferred by over 85%. Buccino et al. [31] proposed a semi-automatic approach involving an online implementation of the Independent Component Analysis (ICA) algorithm for real-time spike sorting of high-density Multi-Electrode Array data. Also, a method to automate spike sorting in electrical stimulation experiments using large multi-electrode arrays, where artifacts are a concern, was proposed in [32].

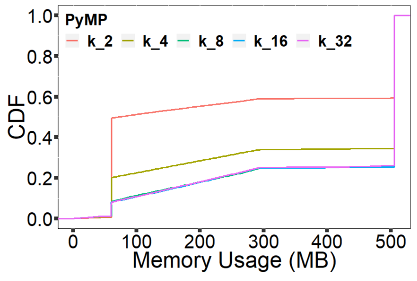
While aforementioned literature proposed approaches to alleviate existing challenges encountered with the utilization of electrode arrays in various scenarios, this paper instead, for the first time, focuses on a new approach to transform the original problem from spatial domain to temporal domain and enables unprecedented parallelization possibilities.

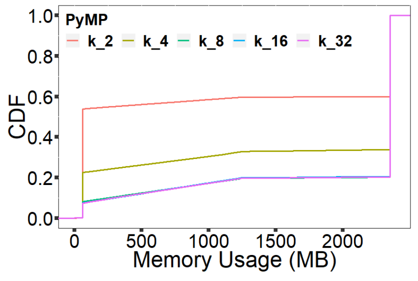
VII. Conclusion and Future Work

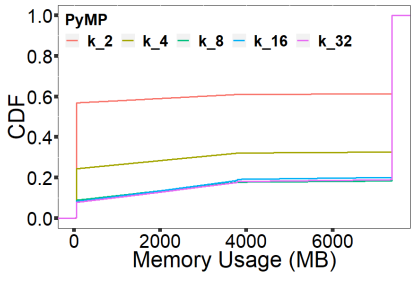
This paper addresses the long-existing computational challenge of multidimensional data in one of the most widely used engineering devices, namely microelectronic array (MEA). We propose a new algebraic model to abstract the entities in MEA; the new model then allows us to develop new methodology to parallelize the computation dictated by the Kirchhoff law. We implement a system prototype—namely Parma—with various optimizations backed by the proposed algebraic model and


(a) n=10


(b) n=20


(c) n=40


(d) n=60


(e) n=80

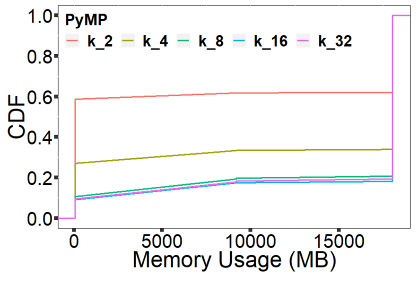

(f) n=100

Figure 8. **Cummulative Distribution Functions (CDFs) of Memory Usage.** For all the scales of $n \in \{10..100\}$, the peak memory usage is about the same regardless of data parallelism.

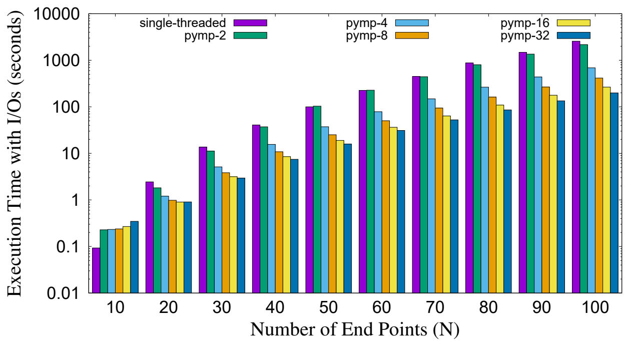


Figure 9. **The end-to-end time of various degrees of parallelism in PyMP, including disk I/Os.** Utilization of more threads $k \geq 2$ starting from a low rank of $n = 20$ makes significant effect to the overall I/O time.

parallelization, and evaluate its performance on up to 1,024 cores. Experimental results show that the proposed approach significantly outperforms the state-of-the-practice: the computation time is orders of magnitude faster; the I/O cost is proportionally reduced; and the memory is efficiently utilized.

Our future work along this line of research is threefold. Firstly, we will extend the proposed approach into a cluster of heterogeneous nodes. Secondly, we plan to develop a GPU version of Parma so that the massive number of GPU cores can be exploited. Finally, we are also planing to re-implement both the baseline system and the proposed parallelization techniques with low-level programming language like C or C++ in order to explore more opportunities for performance improvement.

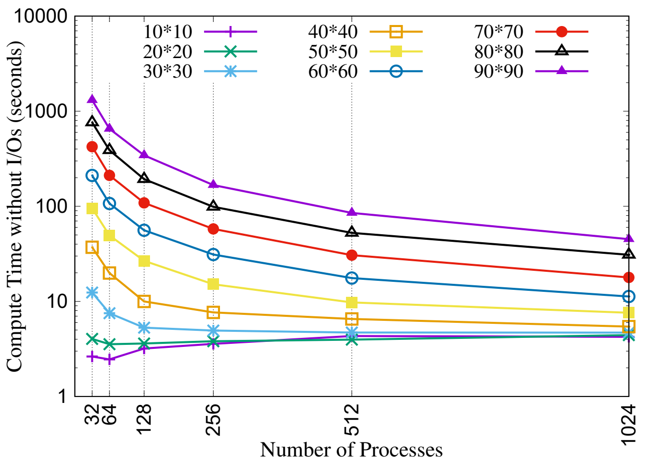


Figure 10. **Scalability of Parma across various number of processes and varying workloads.**

References

[1] E. Kim, G. Erdos, S. Huang, T. W. Kenniston, S. C. Balmert, C. D. Carey, V. S. Raj, M. W. Epperly, W. B. Klimstra, B. L. Haagmans, E. Korkmaz, L. D. Falo, and A. Gambotto, "Microneedle array delivered recombinant coronavirus vaccines: Immunogenicity and rapid translational development," *EBioMedicine*, vol. 55, p. 102743, 2020.

[2] V. Jain and K. Muralidhar, "Electrowetting-on-dielectric system for covid-19 testing," *Transactions of the Indian National Academy of Engineering*, May 2020.

[3] M. A. Ali, C. Hu, S. Jahan, B. Yuan, M. S. Saleh, E. Ju, S.-J. Gao, and R. P. Panat, "Sensing of covid-19 antibodies in seconds via aerosol jet printed three dimensional electrodes," *medRxiv*, 2020.

[4] S. Zips, L. Grob, P. Rinklin, K. Terkan, N. Y. Adly, L. J. K. Weiß, D. Mayer, and B. Wolfrum, "Fully printed u-needle electrode array from conductive polymer ink for bioelectronic applications," *ACS Applied*

Materials & Interfaces, vol. 11, no. 36, pp. 32 778–32 786, 2019.

[5] Y. Niu, L. Qi, F. Zhang, and Y. Zhao, "Geometric screening of core/shell hydrogel microcapsules using a tapered microchannel with interdigitated electrodes," *Biosensors and Bioelectronics*, vol. 112, pp. 162–169, 2018.

[6] H. Li, Z. Tian, J. Xu, R. K. V. Maeda, Z. Wang, and Z. Wang, "Chip-specific power delivery and consumption co-management for process-variation-aware manycore systems using reinforcement learning," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 5, pp. 1150–1163, 2020.

[7] W. Chang, C. Lin, S. Mu, L. Chen, C. Tsai, Y. Chiu, and M. C. . Chao, "Generating routing-driven power distribution networks with machine-learning technique," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 8, pp. 1237–1250, 2017.

[8] X. Wang, O. T. Tawose, F. Yan, and D. Zhao, "HDK: toward high-performance deep-learning-based kirchhoff analysis," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020, pp. 997–1004.

[9] C. Tan, S. Lv, F. Dong, and M. Takei, "Image reconstruction based on convolutional neural network for electrical resistance tomography," *IEEE Sensors Journal*, vol. 19, no. 1, pp. 196–204, 2019.

[10] J. Sun, W. Tian, H. Che, S. Sun, S. Gao, L. Xu, and W. Yang, "Proportional-integral controller modified landweber iterative method for image reconstruction in electrical capacitance tomography," *IEEE Sensors Journal*, 2019.

[11] R. Amirulah, S. Z. M. Muji, M. H. Jabbar, R. A. Rahim, and M. H. F. Rahiman, "Digitalization of linear back projection algorithm for fpga implementation," in *2016 IEEE Conference on Systems, Process and Control (ICSPC)*, 2016.

[12] M. Vauhkonen, D. Vadasz, P. A. Karjalainen, E. Somersalo, and J. P. Kaipio, "Tikhonov regularization and prior information in electrical impedance tomography," *IEEE Transactions on Medical Imaging*, vol. 17, no. 2, pp. 285–293, 1998.

[13] A. Bakushinsky and A. Goncharsky, *Ill-posed problems: theory and applications*. Springer Science and Business Media, 2012.

[14] M. M. Lavrentev, V. G. Romanov, and S. P. S., *Ill-posed problems of mathematical physics and analysis*. American Mathematical Society, 1986.

[15] Y. Niu, A. Al-Mamun, H. Lin, T. Li, Y. Zhao, and D. Zhao, "Toward scalable analysis of multidimensional scientific data: A case study of electrode arrays," in *IEEE International Conference on Big Data*, 2018.

[16] P. Giblin, *Graphs, Surfaces and Homology*. Cambridge University Press, 2010.

[17] M. H. Loke, P. B. Wilkinson, and J. E. Chambers, "Fast computation of optimized electrode arrays for 2d resistivity surveys," *Computers & Geosciences*, vol. 36, no. 11, Nov. 2010.

[18] J. Shun, "Practical parallel hypergraph algorithms," in *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 232–249.

[19] R. Guerraoui, M. Herlihy, and B. Pochon, "A topological treatment of early-deciding set-agreement," *Theor. Comput. Sci.*, vol. 410, no. 6-7, pp. 570–580, 2009.

[20] M. Herlihy and N. Shavit, "A simple constructive computability theorem for wait-free computation," in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC)*, 1994, pp. 243–252.

[21] ——, "The asynchronous computability theorem for t-resilient tasks," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*, 1993, pp. 111–120.

[22] A. Hatcher, *Algebraic topology*. Cambridge: Cambridge Univ. Press, 2000.

[23] D. S. Dummit and R. M. Foote, *Abstract algebra*. John Wiley, 1999.

[24] A. Hambley, *Electrical Engineering: Principles and Applications*. Pearson; 7th edition, 2017.

[25] F. Schmuck and R. Haskin, "Gpfs: A shared-disk file system for large computing clusters," in *Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST)*, 2002.

[26] Y. Zhao, S. Chakravorti, R. F. Labadie, B. M. Dawant, and J. H. Noble, "Automatic graph-based method for localization of cochlear implant electrode arrays in clinical ct with sub-voxel accuracy," *Medical image analysis*, vol. 52, Feb. 2019.

[27] F. Esposti, J. Lamanna, and M. G. Signorini, "A new approach to the spatio-temporal pattern identification in neuronal multi-electrode registrations," in *Proceedings of Neuroscience Today*, 2007, pp. 21–24.

[28] P. Kiele, A. Kohler, C. Pasluosta, and T. Stieglitz, "Robust and precise alignment monitoring of electrode arrays for capacitive energy supply and signal transmission," in *9th International IEEE/EMBS Conference on Neural Engineering (NER)*, 2019, pp. 686–689.

[29] E. della Valle and J. D. Weiland, "Simultaneous impedance measurements of the utah electrodes array: A finite element method analysis," in *9th International IEEE/EMBS Conference on Neural Engineering (NER)*, 2019, pp. 819–822.

[30] Y. H. Yassin, C. Francky, K. Fabian, J.-J. Sun, J. Couto, P. G. Kjeldsberg, and N. V. Helleputte, "Algorithm/architecture co-optimisation technique for automatic data reduction of wireless read-out in high-density electrode arrays," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 17, no. 3, Jun. 2018.

[31] A. P. Buccino, S.-H. Hsu, and G. Cauwenberghs, "Real-time spike sorting for multi-electrode arrays with online independent component analysis," in *IEEE Biomedical Circuits and Systems Conference*, 2018.

[32] G. E. Mena, L. E. Grosberg, S. Madugula, P. Hottowy, A. Litke, J. Cunningham, E. J. Chichilnisky, and L. Paninski, "Electrical stimulus artifact cancellation and neural spike detection on large multi-electrode arrays," *PLoS computational biology*, vol. 13, no. 11, Nov. 2017.