Spatial Decomposition of Structured Grids for Nuclear Reactor Simulations

Andrew P. Fitzgerald[a,*], Brendan Kochunas[a], Shane Stimpson[b] and Thomas Downar[a]

[a] *University of Michigan, Department of Nuclear Engineering and Radiological Sciences, 2200 Bonisteel Blvd. Floor 4, Ann Arbor, MI 48109*
[b] *Oak Ridge National Laboratory, Reactor and Nuclear Systems Division, 1 Bethel Valley Rd, Oak Ridge, TN 37830*

ARTICLE INFO

Keywords:
spatial decomposition
graphs
nuclear reactor
neutronics
transport

ABSTRACT

Spatial decomposition methods based on graph partitioning are developed and implemented in the high fidelity neutron transport code MPACT. These graph-based spatial decomposition methods are more general than previous decomposition methods and typically provide better load balance and reduced runtimes due to their improved parallel efficiency. Correlations are drawn between simulation runtime and the balance of the partition for 2D simulations. Comparisons are made using partition balance metrics for different decomposition schemes in 2D and 3D simulations. For typical ranges of subdomains, graph-based partitioning methods offer significant reductions to runtimes. However, for highly decomposed problems, these graph-based methods may decrease convergence rates, thus reducing parallel efficiency compared to older methods.

1. Introduction

Simulation is a necessary tool for nuclear reactor design and analysis. There are many different simulation methods for predicting the distribution of neutrons, and therefore power, in reactor systems; until recently, the method of choice has been neutron diffusion. Neutron diffusion codes can perform whole-core calculations, on a typical workstation, in relatively short runtimes. More recently there has been a shift toward higher fidelity methods such as discrete ordinates (S_N), and the method of characteristics (MOC) [1]. High fidelity methods allow for more detailed analysis through finer resolution and the use of fewer approximations, but they require significantly greater computational resources.

High fidelity whole-core simulations often require more memory than is available on a typical workstation, so larger distributed computing clusters are generally used. Therefore, it is often necessary to perform spatial domain decomposition. The simulation domain is divided into spatial subdomains, with each subdomain being handled by separate processes. In general, these processes need to communicate with their nearest neighbors. Each process only stores data relevant to its subdomain, making whole-core simulations possible. Additionally, because each subdomain can be processed in parallel, the overall runtime of the simulation is expected to decrease with increasing numbers of subdomains.

The MPACT code is a neutron transport code based on the method of characteristics [1] and is developed jointly by the University of Michigan and Oak Ridge National Laboratory [2]. MPACT has capabilities for domain decomposition over two separate domains: angle and space. In MPACT, each discrete angle has a calculable amount of work, and the decomposition is trivial; in general, the same cannot be said of the spatial domain. The focus of this work is the decomposition of the spatial domain in MPACT.

Spatial decomposition of a reactor can be abstracted to a graph partitioning problem [3], which has been well studied in computer science [4] and applied to other fields such as computational fluid dynamics [5]. In general, the graph partitioning problem is NP-complete, meaning that a partitioning cannot easily be verified as optimal; therefore, graph partitioning relies on approximate heuristic methods. Many different methods have been developed for graph partitioning, several of which are discussed in section 3.

The remainder of this article is structured as follows. In section 2, a description of spatial decomposition in MPACT is given. Section 3 introduces relevant graph theory concepts, and the methods used for spatial decomposition in this work. Section 4 describes the applications of these graph theory methods in MPACT. Section 5 compares methods for 2D and 3D reactor simulations. Finally, section 6 lists the conclusions that are drawn from this work.

2. Spatial Decomposition in MPACT

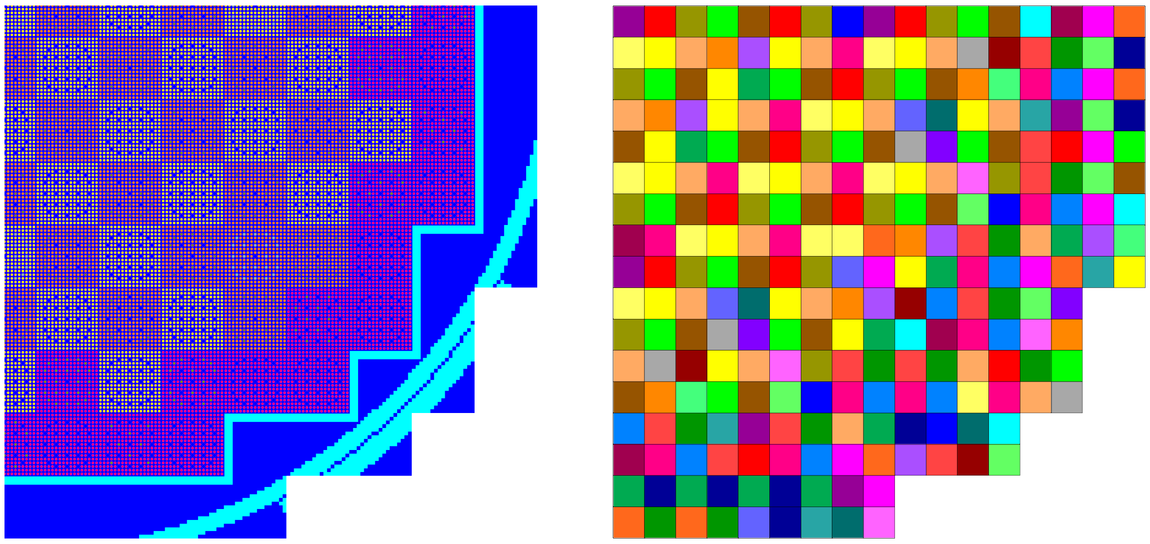
MPACT is a neutron transport code, based on the MOC. It was originally developed for direct whole-core simulation of light water reactors (LWRs). In the MOC, an approximate transport equation is solved analytically along characteristic rays that traverse the problem. By using many of these characteristic rays, an accurate solution is obtained; however, storing the data of these characteristic rays can use a considerable amount of memory.

In MPACT, the modular ray-tracing technique [6] is used to reduce the memory used for storing characteristic ray information. Modular ray-tracing involves dividing the reactor system into *ray-tracing modules*, which are small geometries that are often repeated in the reactor. Characteristic rays are constructed in each ray-tracing module such that each ray directly links to a ray in an adjacent module. In this method,

*Corresponding author

apfitzge@umich.edu (A.P. Fitzgerald); bkochuna@umich.edu (B. Kochunas); stimpsonsg@ornl.gov (S. Stimpson); downar@umich.edu (T. Downar)

ORCID(s): 0000-0002-4066-0949 (A.P. Fitzgerald); 0000-0001-7109-9368 (B. Kochunas); 0000-0002-0120-9928 (S. Stimpson); 0000-0002-5810-6727 (T. Downar)



(a) fine mesh (b) ray-tracing module mesh

Figure 1: Example quarter core core configuration and corresponding ray-tracing modular mesh in MPACT.

characteristic ray information is only stored for each *unique* ray-tracing module.

Ray-tracing modules are the smallest unit for spatial decomposition in MPACT [7]. These ray-tracing modules are typically an axial slice of a quarter of a full fuel assembly, as shown in figure 1 The core consists of a structured grid of these modules in which each module has the same dimensions but may have different numbers of computational cells. Therefore, in MPACT, the spatial decomposition is a structured grid partitioning problem.

In general, it is possible to use the computational cells as the smallest unit in the decomposition. However, this causes the decomposition problem to become an unstructured mesh partitioning problem. This is not done in MPACT because communication would become significantly more complicated. Additionally, there would be more re-entrant rays which would have negative impacts on the rate of convergence.

MPACT has had two spatial decomposition methods in the past: manual decomposition, and assembly-based decomposition. A user may manually enter a decomposition [7], but it is time consuming to construct a balanced decomposition and will likely still be suboptimal to some degree. An automated method exists that recursively bisects the core using Morton-ordering [8] and the reactor assembly geometries. While this method is automated, it often yields very imbalanced domains, and also restricts the number of subdomains that can be used.

Previous work has shown that spatial decomposition of reactors can be abstracted to a graph partitioning problem [3]. The use of graph partitioning methods in MPACT is expected to solve the issues encountered in each of the two approaches described above. These methods can be used to decompose into an arbitrary number of domains with high quality results, without user input.

Existing graph partitioning libraries such as METIS [9] partition graphs very efficiently and have very high quality results. To use all given processors, MPACT requires that each spatial subdomain contains at least one module, i.e. no partition can be empty. However, in some cases, particularly when the number of partitions is high, METIS may generate empty partitions. This means METIS cannot be used to decompose the core into an arbitrary number of subdomains without modifying the resulting partitions. For this reason, MPACT does not rely on third-party libraries for graph partitioning in the spatial decomposition process.

3. Applied Graph Theory

The spatial decomposition of a reactor core can be abstracted to the partitioning of a graph. Specifically, this would be a weighted graph, $G(V, E)$, which is comprised of a set of vertices, V, and a set of edges, E, that connect pairs of vertices. In general, these vertices and edges may have weights; a vertex v_i will have weight w_i, and an edge e_i between vertices v_i and v_j will have weight c_{ij}. In MPACT, a vertex represents a ray-tracing module, and the edges represent communication between adjacent modules in the MOC. The graphs are undirected because communication between ray-tracing modules is two-way.

Previous work [3] applied unweighted graph partitioning techniques to the reactor spatial decomposition problem; the work presented here applies generalizations and improvements to the methods used for graphs with weighted vertices and edges. A vertex's weight indicates the amount of computational work that is needed; as one might expect, this is highly correlated with the number of computational cells. This is shown in section 5. In general, the edges may also be weighted to account for different amounts of data transfer. This is discussed in more detail in section 4.

The goal is for each partition to have equal weight, with minimal weight of edges cut by partition boundaries. This is equivalent to each subdomain having the same amount of computational work with minimized communication between processes. If each process has roughly the same amount of work to perform, then less time will be spent waiting for other processes, thus improving parallel efficiency. Also, with less communication, less time will be spent passing data between processes, so the parallel overhead will be reduced.

In this work, methods were separated into two distinct categories: partitioning methods and partition refinement (improvement) methods. Partitioning methods give a near-balanced partitioning for a given graph. Refinement methods attempt to reduce communication between existing partitions in a graph. As applied in MPACT, these refinement methods typically did not significantly reduce communication. These methods and results are presented in appendix A.

3.1. Graph Partitioning Methods

In this work, recursive partition methods were considered due to their capability to partition into arbitrary numbers of domains. Each of these recursive partitioning methods sorts the graph, using different methods, and then divides or “cuts” the graph into two subgraphs with approximately equal vertex weights. Once a graph’s vertices are sorted into a list, V_s, the graph can be bisected using algorithm 1.

Multi-level partitioning methods are widely used in other fields such as networking, where graphs can become very large; however, in MPACT, the number of ray-tracing modules is on the order of a few hundred to several thousand, which directly correlates to the size of the graph. Additionally, for MPACT, the decomposition problem is static, so the computation time for partitioning is expected to be negligible as it can simply be performed one time at the outset. Due to the small graph size, multi-level methods were not considered as part of this work.

Algorithm 1 The algorithm used to determine how to cut a graph into two subgraphs based on a sorted vertex list V_s, and that the graph will be recursively partitioned into N groups.

```
1: procedure GRAPH CUT($G(V,E), V_s, N$)
2:     $N_1 \leftarrow \lfloor N/2 \rfloor$        ▷ Desired number of recursive partitions for first subgraph
3:     $W_1 \leftarrow \frac{N_1}{N} \sum_{v_i \in V} w_i$        ▷ Ideal weight of first subgraph
4:     Let $V_1$ be a set of vertices such that:
           • $V_1 \subset V$
           • The vertices $V_1$ are taken in order from $V_s$
           • $W_1 - \sum_{v_i \in V_1} w_i$ is minimized
5:     Let $V_2$ be the subset $V \setminus V_1$
6:     Optionally call a refinement method
7:     Create a graph $G_1$ from $V_1$
8:     Create a graph $G_2$ from $V_2$
9: end procedure
```

3.1.1. *Recursive Spectral Bisection*

The recursive spectral bisection (RSB) method, originally developed by Pothen et al.[10], has been highly successful and widely used in graph partitioning [11, 12]. This method relies entirely on the connectivity of the graph and not on its geometry. The RSB method has been improved to allow to allow for partitioning of *weighted* graphs into any number of domains [13].

The RSB method makes use of the Laplacian matrix of a graph; specifically the second-smallest eigenvalue of this matrix, referred to by Fiedler as the *algebraic connectivity* [14]. The eigenvector associated with this eigenvalue has also been known as the *Fiedler vector*. For weighted graphs, the weighted Laplacian matrix is used in lieu of the Laplacian matrix; matrix elements are given by

$$L_{ij} = \begin{cases} d_i, & i = j, \\ c_{ij}, & i \neq j, \\ 0, & \text{else}, \end{cases} \tag{1}$$

where d_i is the sum of edge weights from vertex v_i, and c_{ij} is the weight of the edge between vertices v_i and v_j. The Fiedler vector is found from this weighted Laplacian matrix and can be used to sort the vertices of the graph in a one-dimensional list V_s. This list of vertices is then divided into two sets, based on weight and total number of partitions needed (see algorithm 1). The recursive spectral bisection algorithm is listed in algorithm 2.

Algorithm 2 The recursive spectral bisection (RSB) algorithm.

```
1: procedure RSB($G(V,E)$)
2:     Let $L$ be the weighted Laplacian of $G(V,E)$
3:     Compute eigenvectors of $L$
4:     Use the Fiedler vector to sort $V \rightarrow V_s$
5:                                ▷ If tie, use larger eigenvectors
6:     Cut graph into $G_1(V_1,E_1), G_2(V_2,E_2)$: Algorithm 1
7:     RSB($G_1(V_1,E_1)$)
8:     RSB($G_2(V_2,E_2)$)
9: end procedure
```

3.1.2. *Recursive Inertial Bisection*

Another class of recursive partitioning methods are coordinate or geometric methods. There are many different geometric partitioning methods in existence; in this study, the recursive inertial bisection (RIB) method [4, 15] was investigated. This method uses only the geometry of the graph to construct a bisector and does not consider the connectivity (edges) in any way.

The RIB method determines a bisector which cuts the graph into two approximately equally sized subdomains. This is easily generalized for weighted graphs. The bisector should have approximately equal amounts of weight on each side. The RIB makes no assumption of the orientation of the graph in space, unlike some other coordinate partitioning methods. The principle axes of the graph are equivalent to the eigenvectors of the inertial matrix given by

$$\boldsymbol{I} \equiv \sum_{i=1}^{n} w_i \left(\boldsymbol{x}_i - \overline{\boldsymbol{x}}\right)^T \left(\boldsymbol{x}_i - \overline{\boldsymbol{x}}\right), \tag{2}$$

where n is the number of vertices, $\boldsymbol{x}_i$ is a row-vector containing coordinates of vertex v_i, and $\overline{\boldsymbol{x}}$ is the mean coordinate vector given by

$$\overline{\boldsymbol{x}} \equiv \frac{\sum_{i=1}^{n} w_i \boldsymbol{x_i}}{\sum_{i=1}^{n} w_i}. \tag{3}$$

An approximate bisector is given as passing through the weighted centroid with normal vector given as one of the eigenvectors of $\boldsymbol{I}$.

Other works [4, 15] have used the smallest eigenvalue's eigenvector as a normal vector to minimize the mean-square distance of vertices from the bisecting line or plane. However, in this work, the largest eigenvalue's eigenvector is used, so a smaller cut-size is typically given while still bisecting the graph into two subdomains of approximately equal weight. This is the case because in the MOC, communication scales with the surface area between adjacent ray-tracing modules. This may not be the case for other computational methods.

In general, a line or plane passing through the weighted centroid with the eigenvector normals will not cut the graph into two equally weighted subdomains. Instead, the vertices will be sorted according to their distance from the approximate bisectors, and then a cut will be made so that near equal amounts of weight are in each set using algorithm 1. This sorting and cutting based on weights is equivalent to shifting the bisector in the direction of the normal vector. An example is visualized in figure 2. The RIB algorithm is listed in algorithm 3.

Algorithm 3 The basic recursive inertial bisection (RIB) algorithm.

```
1:  procedure RIB($G(V, E)$)
2:      Compute the weighted centroid of the graph $\overline{x}$, given
        by equation (3)
3:      Shift coordinates relative to centroid:
        $x_i^c = x_i - \overline{x} \quad \forall i \in V$
4:      Compute inertial matrix $I$, given by equation (2)
5:      Compute eigenvectors of $I$. Largest eigenvalue's
        eigenvector $e_1$
6:      Compute distance from largest eigen-pair bisector:
        $d_i = x_i^c \cdot e_1$
7:      Sort $V \to V_s$ based on $d_i$.
8:              ▷ In ties use smaller eigenvalue's eigenvector
9:      Cut graph into $G_1(V_1, E_1), G_2(V_2, E_2)$: Algorithm 1
10:     RIB($G_1(V_1, E_1)$)
11:     RIB($G_2(V_2, E_2)$)
12: end procedure
```

3.1.3. *Recursive Expansion-Based Methods*

The recursive expansion-based bisection (REB) methods comprise the last class of partitioning methods examined in this work. These methods begin a bisection step by selecting a vertex as the starting point of a subdomain. This subdomain is then expanded until it is approximately half the size of the graph [16, 17, 4, 3]. In this work, the method outlined by Fitzgerald et al.[3] was slightly modified and generalized to

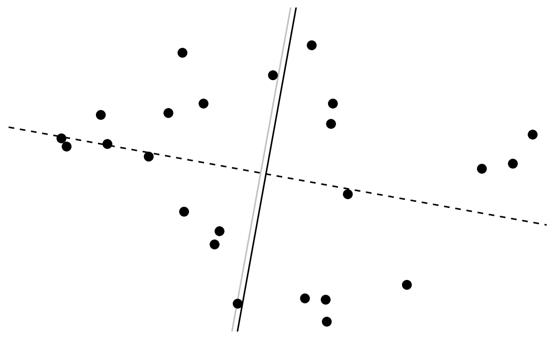


Figure 2: Example of an inertial bisection. Vertices are shown as black points, the bisectors of the largest eigen-pair is shown by the black solid, and the bisector of the smallest eigen-pair is shown by the black dashed line. The "shifted" bisector used in the partitioning is shown in grey. While the communication between vertices is not drawn, it is clear that the length (proportional to cut size) of the smallest eigen-pair bisector is larger than that of the largest eigen-pair bisector.

weighted graphs. For the remainder of this work, the acronym *REB* will be used to denote this specific expansion-based method rather than the entire class of methods.

This REB method considers both the geometry and connectivity of the graph. The method begins by choosing a starting vertex for the subdomain and then expands based on a set of prioritized rules. At each expansion step, the next vertex is chosen so that it is geometrically close to the vertices within the subdomain and to minimize edges between the subdomain and the remaining graph. However, this method makes the assumptions that the mesh is structured, and that every mesh element is the same shape and size. For the application in MPACT, this is always true.

This REB method uses the concept of a *sphere of influence* (SOI) around a vertex. The SOI includes directly neighboring vertices and vertices that neighbor more than one of the direct neighbors or that would if the direct neighbor were present in each structured position around the primary vertex. This is shown for 2D rectangular structured mesh in figure 3. For implementation simplicity, the sphere of influence is calculated using distance rather than connectivity.

The starting vertex in this REB method is chosen using a set of prioritized rules:

1. must be on graph boundary, i.e. at least one direct neighbor is not present,
2. must have the lowest summed weight of edges, and
3. must be located furthest from weighted centroid (given by equation (3)).

Vertices within the expanding subdomain are considered internal vertices, and the remaining vertices are considered to be external vertices. During expansion, the next vertex is determined using a set of prioritized rules:

1. must be neighboring at least one internal vertex,
2. must have the highest summed weight of edges with internal vertices,
3. must have the lowest summed weight of edges with external vertices,

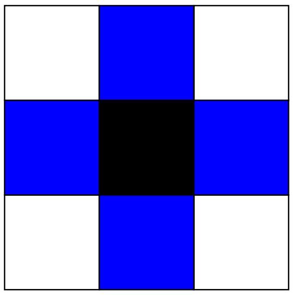


Figure 3: "Sphere of influence" example for 2D rectangular structured grid. The primary vertex is shown in black, direct neighbors are blue, and additional vertices in the sphere are white [3].

4. must have the largest number of internal SOI vertices,
5. must have the largest number of external SOI vertices, and
6. must have the smallest distance from reference vertex.

The reference vertex is in the expanding subdomain, which begins as the first vertex but changes during expansion; the reference vertex is the most recently added vertex with less external communication than the previously added vertex.

Algorithm 4 The chosen Recursive Expansion Bisection (REB) algorithm.

```
1: procedure REB($G(V, E)$)
2:     Compute weighted centroid of the graph
3:     Choose a starting vertex for the expanding domain:
       See rules in section 3.1.3
4:     Expand the domain from the starting vertex. Let $V_s$
       be the list of vertices in order of the expansion: See
       rules in section 3.1.3
5:     Cut graph into $G_1(V_1, E_1), G_2(V_2, E_2)$: Algorithm 1
6:     REB($G_1(V_1, E_1)$)
7:     REB($G_2(V_2, E_2)$)
8: end procedure
```

4. Applications for MPACT

As described in section 2, MPACT's spatial decomposition is performed on the ray-tracing module mesh. However, there are a couple restrictions on spatial subdomains in MPACT. Each spatial subdomain must be contiguous, and cannot wrap around other spatial subdomains. To account for these restrictions, adaptations are made to the graph partitioning process.

Due to restrictions in MPACT, at each recursive step, each subdomain in a bisection is made contiguous. If a partitioning method results in a noncontiguous subdomain, then each noncontiguous group of modules will be moved into the other subdomain except for the largest group. This fix is done at the expense of load-balance, but is necessary for these methods to be robust in MPACT. To ensure that no subdomain wraps around another, a fix is applied after the graph partitioning process. If a subdomain wraps around another, then the concave subdomain will be given the modules it wraps around.

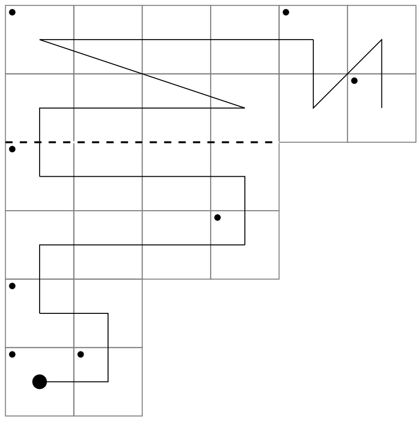


Figure 4: An example of the REB method expansion on a small graph. The grey lines show the square mesh cells. The expansion begins at the large black point in the center of a cell, and the black line from this shows the expansion's order. Each small black dot in the upper left of a cell indicates the reference vertices during expansion. The thick black dashed line shows the bisecting cut.

A group of ray-tracing modules in MPACT can be abstracted into a graph. Each vertex will have weight corresponding to the number of cells contained in the module. Edges can be drawn between directly neighboring ray-tracing modules. This represents communication in MPACT's MOC solver. Transport source iterations converge slowly, so MPACT relies on the coarse mesh finite difference (CMFD) [18] acceleration method. CMFD acceleration is performed by constructing a sparse linear system based on the finite differenced diffusion operator and then solving for the largest eigenpair of that linear system. In MPACT, solving the linear system is handled by a third-party library, PETSc [19].

For 2D simulations, the application of graph partitioning methods is clear: abstract the 2D mesh into a graph for partitioning. However, for 3D, there are additional concerns. MPACT's primary 3D transport method is the 2D-1D method, in which the MOC is used in the radial directions, and a lower-order solver couples axial planes [20]. For 2D-1D simulations, MPACT currently restricts spatial domains to be aligned in both the radial and axial directions; this is due to implementation, and is not a general requirement of the methods.

To comply with MPACT's restrictions on 3D spatial domains, the current approach is to axially average module weights (numbers of cells), perform a 2D graph partitioning on a single plane, and apply the resulting partitioning to all axial planes. This approach will restrict the number of spatial domains to be an integer multiple of the number of planes. This axially and radially aligned scheme is expected to work well in many cases since reactor cores do not typically vary significantly in the axial direction. However, for some designs, this may not be true, and planes near the top or bottom

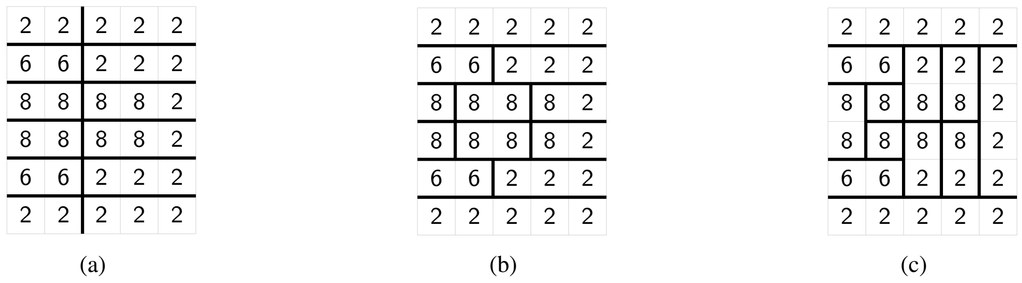


Figure 5: Sample decompositions for (a) axially aligned, (b) radially aligned, and (c) generalized decomposition strategies. Rows represent axial planes. Numbers are the vertex weights.

of the core have significantly fewer cells; in these cases, high load imbalance is to be expected.

It is possible to change MPACT's implementation to lift these alignment restrictions. If spatial domains were aligned in only the radial direction, there may be some benefit to load-balance. In this scheme, each axial plane can be assigned an appropriate number of processes, and a separate 2D decomposition can be performed for each plane. This also lifts restrictions on the number of domains; the number of domains must only be greater than or equal to the number of planes.

If all alignment restrictions were lifted on MPACT's spatial domains, then a direct partitioning of the 3D core can be performed by abstracting the entire core to a graph. This scheme provides the most freedom and would be expected to give the most balanced decompositions. In MPACT's 2D-1D solver, the amount of data communicated radially is significantly larger than that communicated axially, so it may be advantageous to assign the edges connecting the neighboring modules axially lower weights than those in radial directions. By doing so, the overall communication would be expected to be decreased.

Figure 5 shows a comparison of the three decomposition schemes. Looking at the maximum-to-minimum ratio (MMR), as an indicator of load-balance, the strategies have clear differences. The MMR for the axially aligned, radially aligned, and generalized strategies in this example are 4.00, 2.67, and 2.00, respectively. However, it is important to note that the largest domain in each case has the same weight (16), so while the different schemes give different balances, the overall runtimes are not expected to be different.

5. Results

5.1. 2D Results

Results were generated for the planar 2D version of VERA progression problem 5a [21]. This problem is a quarter core with reflector, barrel, and neutron pad regions surrounding several fuel assemblies, as shown in figure 6. In the model, there are 257 ray-tracing modules in total, which provides the upper bound for the number of domains. Each of the graph decomposition methods was applied to the geometry of this problem, and MPACT was run for each case without applying refinement methods. The assembly-based decomposition was run for all possible numbers of subdomains: 1, 4, 16, 73, and 257. Example decompositions for 73 subdomains are shown in figure 7.

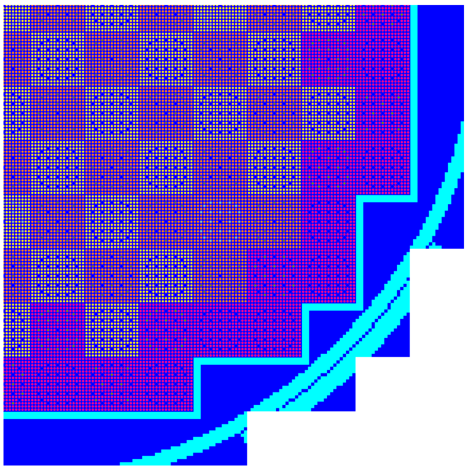


Figure 6: VERA progression problem 5a-2d core configuration.

5.1.1. Load-Balance

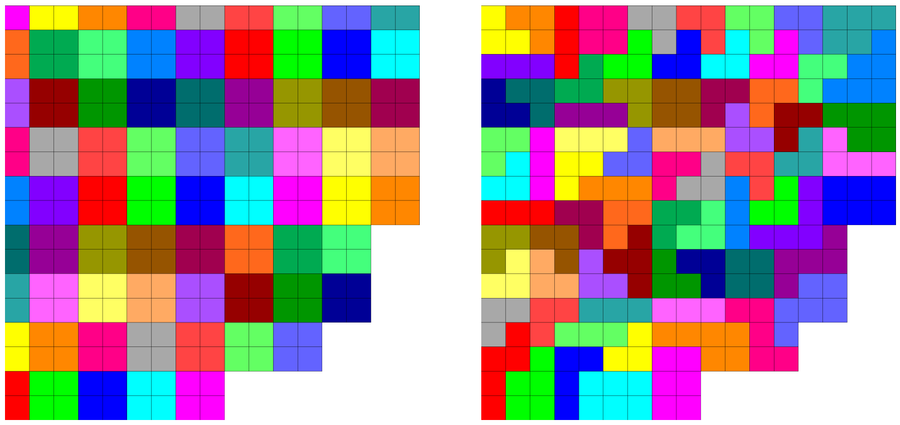
In MPACT, each spatial subdomain is simulated concurrently. After each iteration, parallel boundary conditions are communicated and updated in parallel. The time for solve routines is measured for each subdomain, as is the MOC communication time. However, the communication time includes time spent waiting for other subdomains to finish computation; that is, blocking communication.

This parallel iteration scheme means that the wall-time of each iteration is controlled by the subdomain with the longest run-time; this is expected to be the subdomain with the largest number of cells. However, the wall-time is not the only important consideration; it is important to consider how well the computational resources are used. A measure for how well computational resources are used is the parallel efficiency, as defined by

$$E \equiv \frac{T_s}{N \cdot T}, \tag{4}$$

where T_s is the time in serial, and T is the time in parallel with N processes.

If runtime is highly correlated with the largest number



(a) Assembly-based decomposition method. (b) REB partitioning method.

Figure 7: Example decompositions for 73 subdomains for VERA progression problem 5a-2d. Each color represents a different subdomain.

of cells in a subdomain, then this efficiency is expected to be related to the largest *fraction* of cells in a subdomain. Specifically, the parallel efficiency is expected to be inversely proportional to the ratio of the largest fraction of cells to the optimal fraction of cells. In an ideally balanced decomposition, each subdomain would have $1/N$ cell fraction. By comparing the maximum cell fraction to this optimal value, one can estimate how much longer the simulation will take compared to an ideal decomposition, neglecting serial code sections and overhead.

Higher parallel efficiency indicates better utilization of the available computational resources, and lower runtime. The expectation is that higher parallel efficiency can be obtained by having a largest cell fraction closer to the optimal cell fraction. As seen in figure 8, the REB method is expected to have slightly better parallel efficiency than the other methods for many cases. It is also expected that for a low number of subdomains, the assembly-based decomposition will result in significantly lower parallel efficiency. However, for large numbers of subdomains, the assembly-based decomposition is expected to result in comparable parallel efficiency to the graph partitioning methods.

5.1.2. Communication

In MPACT, parallel boundary conditions are communicated concurrently. However, the measurement of communication time includes any time spent waiting for other subdomains to complete their calculations. Generally, the time spent sending receiving parallel boundary conditions is relatively small. This time is expected to increase with the weight of edges cut by parallel boundaries.

Figure 9 shows that the number of edges cut increases as the number of domains increases. This indicates that time sending and receiving parallel boundary conditions is expected to increase with the number of subdomains. As the number of subdomains increases, the time spent sending and receiving parallel boundary conditions is expected to increase as more data are communicated. However, the time difference between the slowest and fastest subdomains decreases, so the overall communication time measurement is expected to decrease. Because subdomains in the assembly-based decomposition are rectangular, the number of edges cut is typically lower than in the graph partitioning methods.

5.1.3. MPACT Results

As expected, the total and MOC runtimes are highly correlated with the largest fraction of cells in any subdomain, as shown in figure 10. Both total and MOC runtimes are very highly correlated with the largest fraction of cells in a subdomain, indicating that this metric can be used to estimate the relative runtimes of decompositions. The assembly-based decomposition method was not used in this correlation, as there are only a few data points available.

Utilization of computational resources is also an important aspect for a high performance simulation code; this can be measured by the parallel efficiency. The total parallel efficiency is shown in figure 11 for each decomposition method. As the core becomes more decomposed, the parallel efficiency drops off rapidly, approaching around 20%. Generally, the graph partitioning methods result in similar parallel efficiency, though the REB method appears to give very slightly higher efficiency is many cases. The assembly-based decomposition method has significantly lower parallel efficiency when there are few subdomains. However, for the moderately decomposed problem (73 subdomains) the assembly-based decomposition method results in significantly higher parallel efficiency. This was not initially expected, as the largest subdomain has a cell fraction similar to that of the graph partitioning methods.

As parallel boundary conditions have a more significant effect on the solution within each subdomain, the convergence rate decreases. This occurs as subdomains become smaller (geometrically), or as they become more "jagged." These jagged parallel boundaries cause re-entrant rays, in which a single ray in the MOC will re-enter the subdomain after leaving. These re-entrant rays will *not* occur in the assembly-based decomposition, because subdomains are forced to be rectangular. This can be observed in figure 7. The number

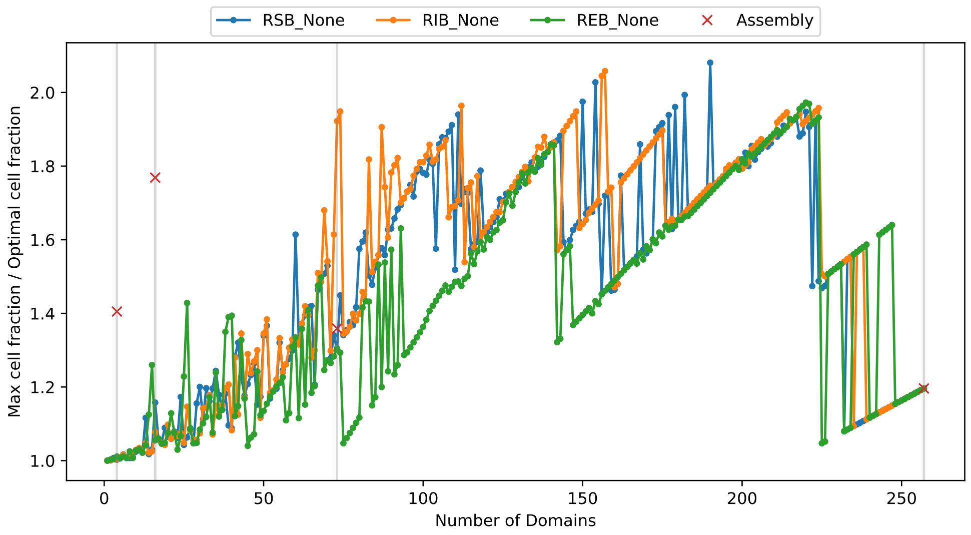


Figure 8: Ratio of largest fraction of cells to the optimal fraction of cells as a function of number of subdomains for each partitioning method without refinement.

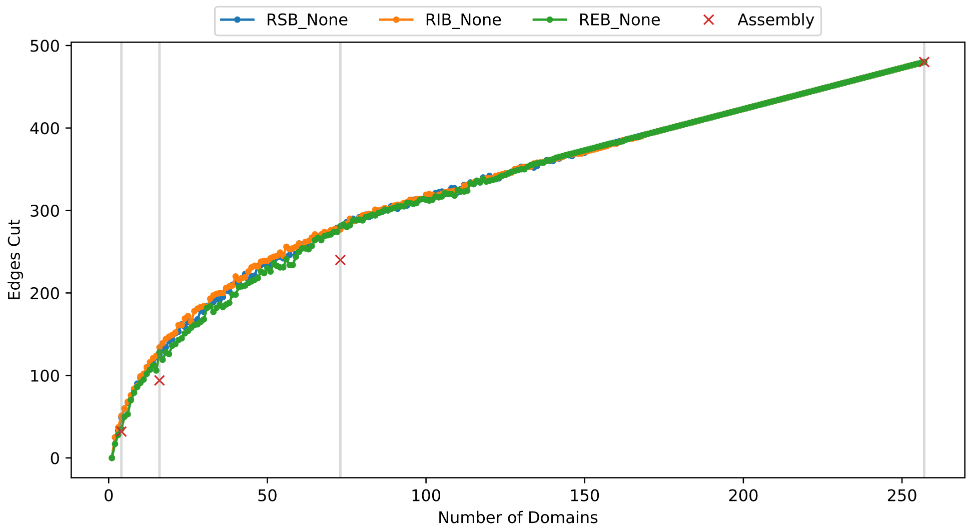


Figure 9: Number of edges cut as a function of number of domains for each partitioning method without refinement.

of MOC iterations required for convergence in each case is show in figure 12.

Other than allowing only rectangular (or convex) subdomains, there is little that can be done about this decrease in convergence rate. However, by examining the parallel efficiency of *runtime per iteration*, the scaling of the solvers in parallel can be found. As shown in figure 13, the total parallel efficiency per iteration decreases as the core becomes more decomposed, limiting toward 25%. However, if only the MOC solver time is being examined, the parallel efficiency per iteration decreases at a much slower rate as shown in figure 14. This indicates that the MOC solver in MPACT is highly efficient in parallel, and that other components of MPACT are the bottleneck in parallel simulations. Furthermore, for both total and MOC runtimes, the graph partitioning methods give comparable parallel efficiencies. The assembly-based decomposition method still results in lower efficiency when using few subdomains, but for high numbers of subdomains, it is comparable with the graph partitioning methods.

In MPACT, the two main solvers contributing to runtime are the MOC solver and CMFD acceleration. From figure 15, the MOC and CMFD solvers take similar amounts of time in serial; however, as more subdomains are used, the fractional runtime of CMFD increases to almost 70% of the total. This indicates that the parallel efficiency is quite low for MPACT's CMFD linear system solvers, which heavily leverage PETSc [19] for parallelism.

Finally, the ratio of the optimal cell fraction to the maximum cell fraction is expected to be proportional to the parallel efficiency per iteration of the MOC solver. As shown in figure 16, the parallel efficiency is correlated with the the ratio of optimal-to-maximum cell fractions, though it is not cor-

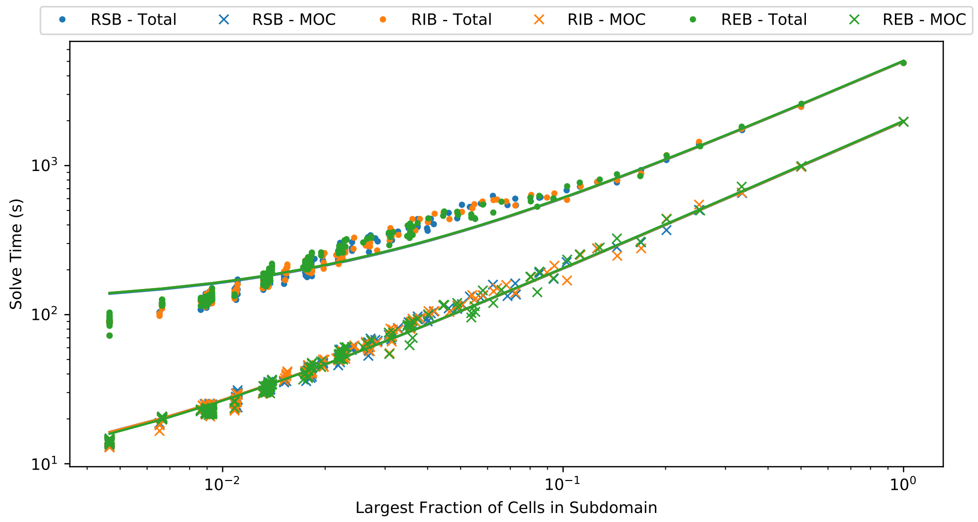


Figure 10: Correlation of total and MOC runtimes to the largest fraction of cells in a subdomain for each partitioning method.

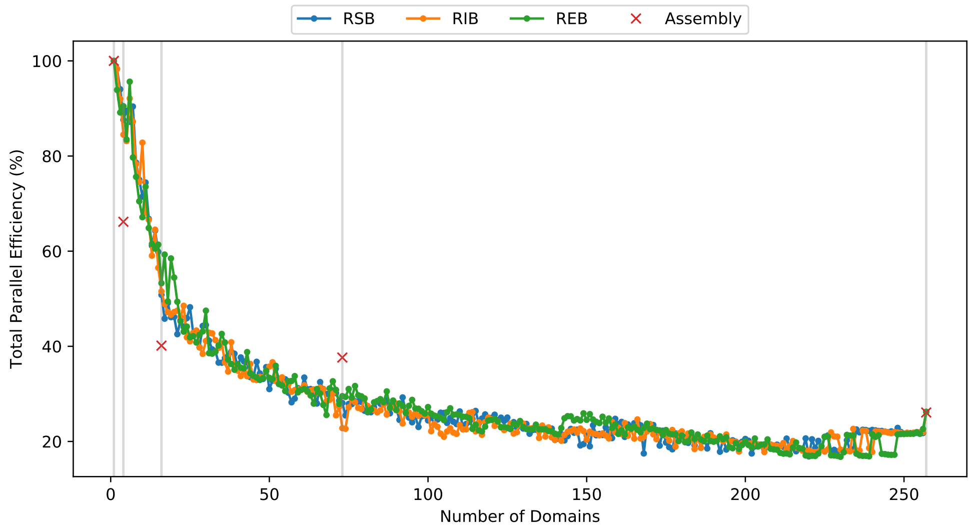


Figure 11: The total parallel efficiency for each partitioning method as a function of the number of domains.

related as strongly as the runtime with the maximum cell fraction. This indicates that this ratio can be used to estimate the parallel efficiency of the MOC solver for a decomposition.

5.2. 3D Results

As shown in section 5.1, decomposition metrics can be used to estimate the runtime and parallel efficiency without needing to run the simulations. There are different approaches to decomposition in 3D; these are discussed in more detail in section 4. Decompositions were performed without refinement for three different 3D decomposition schemes: axially and radially aligned (ARA), radially aligned (RA), and unrestricted (UR). Given fewer restrictions, the resulting decompositions were expected to be more balanced. Decompositions were performed on VERA progression problem 5a-0 in 3D [21] with 58 axial planes, but the simulations for this problem were not run.

The ratio of maximum cell fraction to optimal cell fraction can easily be converted to the maximum cell fraction by dividing by N. For brevity, only this load balance metric is shown herein. The resulting decompositions from the ARA approach are very similar to those in the 2D case. Just as in the 2D case, the maximum-to-optimal cell fraction ratio is lowest for the REB method as compared to the other partitioning methods for many cases, as seen in figure 17. This indicates that, barring any differences in the number of iterations, the REB method is expected to have slightly higher parallel efficiencies.

In the RA scheme, a separate decomposition is performed for each axial plane, with an appropriate number of subdomains based on the number of cells in the plane. Unlike in the 2D case, the REB method seems to perform significantly worse than the other two methods for low numbers of subdo-

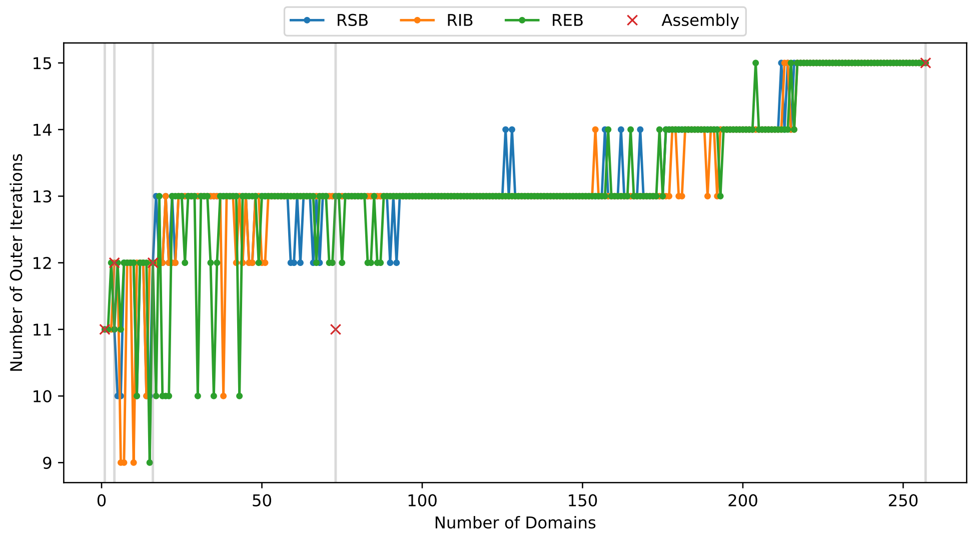


Figure 12: The number of iterations used by each decomposition method as a function of the number of subdomains.

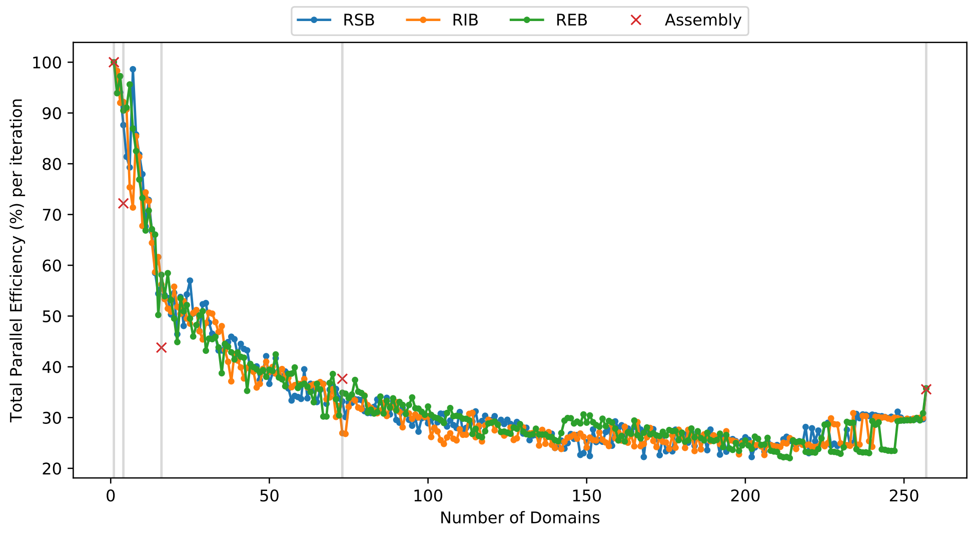


Figure 13: The total parallel efficiency per iteration for each partitioning method as a function of the number of domains.

mains. For highly decomposed cores, the REB method seems to perform slightly better than the other partitioning methods. Additionally, by comparing the magnitude of the ratios in figure 18 and figure 17, it is clear that the RA approach typically has less imbalance.

Finally, the UR approach decomposes the 3D core by directly abstracting the entire core into a graph. The REB method is significantly more imbalanced than other methods for lower numbers of subdomains; however, for highly decomposed cores, the REB method outperforms the other methods, as shown in figure 19. Additionally, for highly decomposed problems, both the RSB and RIB methods seem to have worse balance when using the unrestricted scheme compared to the radially aligned scheme.

The approach currently used in MPACT is the axially and radially aligned 3D decomposition scheme. If other approaches were to be used, the implementation of parallel communication in the 2D-1D method would need to be reworked. To justify these changes, the less restricted approaches would need to offer significant advantages over the current approach. Figures 20 to 22 examine the maximum cell fraction of each scheme relative to the current scheme. The smaller the relative maximum cell fraction, the lower the expected runtime will be.

For most applications, reactor cores in MPACT are not highly decomposed with, at maximum, on the order of 30 subdomains per axial plane. In this context, lowly decomposed cores are defined as those with fewer than 2,000 subdomains; otherwise, the core is considered highly decomposed. For the RSB and RIB methods, the RA approach is expected to give 10% better performance than the ARA approach on average. For highly decomposed cases, these partitioning

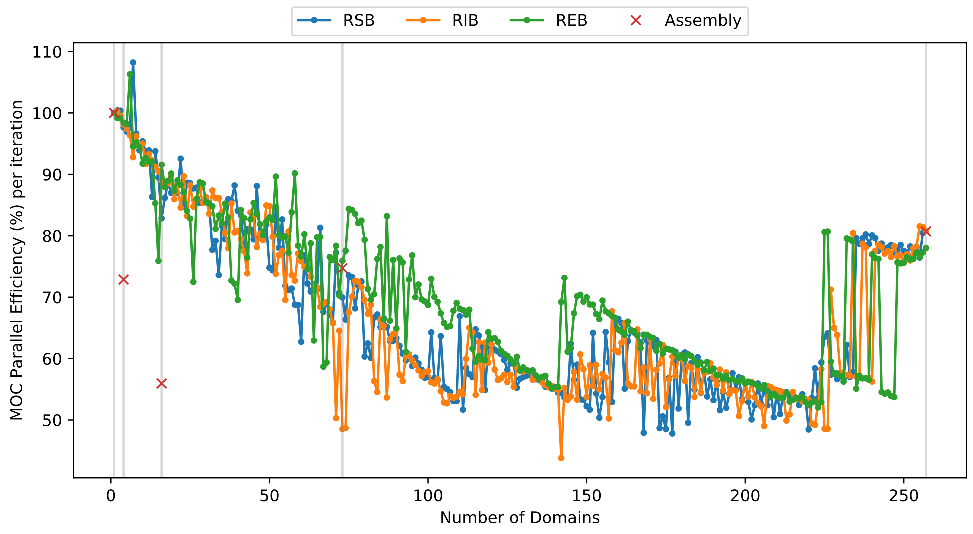


Figure 14: The MOC parallel efficiency per iteration for each partitioning method as a function of the number of domains.

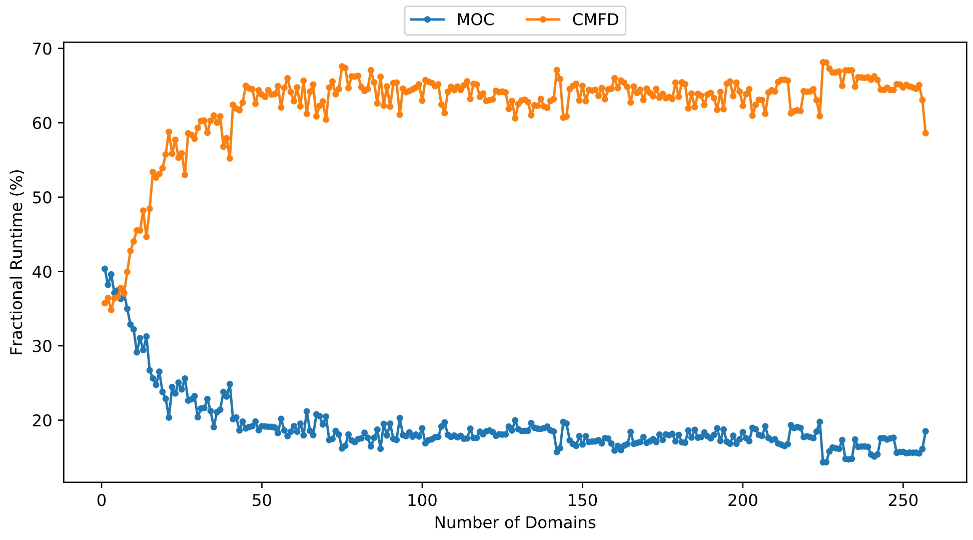


Figure 15: Fractional runtime of the MOC solver and CMFD acceleration method in MPACT for varying number of domains with the REB partitioning method.

methods are only expected to give an average of 2 – 3% better performance. On average, the UR approach is expected to give *worse* performance by more than 10% using these partitioning methods. However, for the REB partitioning method, the RA approach is only expected to give 3% better performance on average. Typically, the UR approach is still expected to result in worse performance. This may be a result of the problem examined in this work which had axial planes that were fairly well balanced. By giving the graph partitioning methods more degrees of freedom the heuristic graph partitioning methods perform worse overall. Multilevel partitioning methods reduce the degrees of freedom during coarsening and thus may be more appropriate in these cases.

6. Conclusions

Spatial decomposition is a useful technique for reducing the runtime of simulations and is necessary to run whole-core high fidelity reactor calculations. Using graph partitioning methods to decompose the spatial domain of a core has significant advantages when compared to previous decomposition methods. Graph partitioning allows for the usage of an arbitrary number of spatial subdomains, and it generalizes to different module geometries such as a hexagonal lattice. Graph partitioning methods generally provide high quality decompositions that increase parallel efficiency. These automated spatial decomposition methods improve code usability and flexibility by allowing users to easily fit simulations to any number of processors.

However, for highly decomposed cores, the convergence

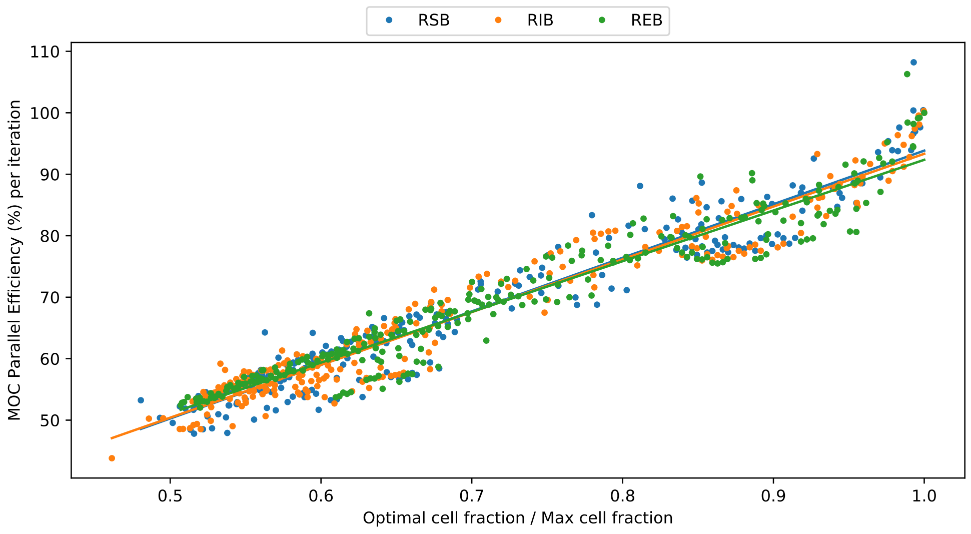


Figure 16: Correlation of the MOC parallel efficiency per iteration and the ratio of optimal and maximum cell fractions for each of the partitioning methods.

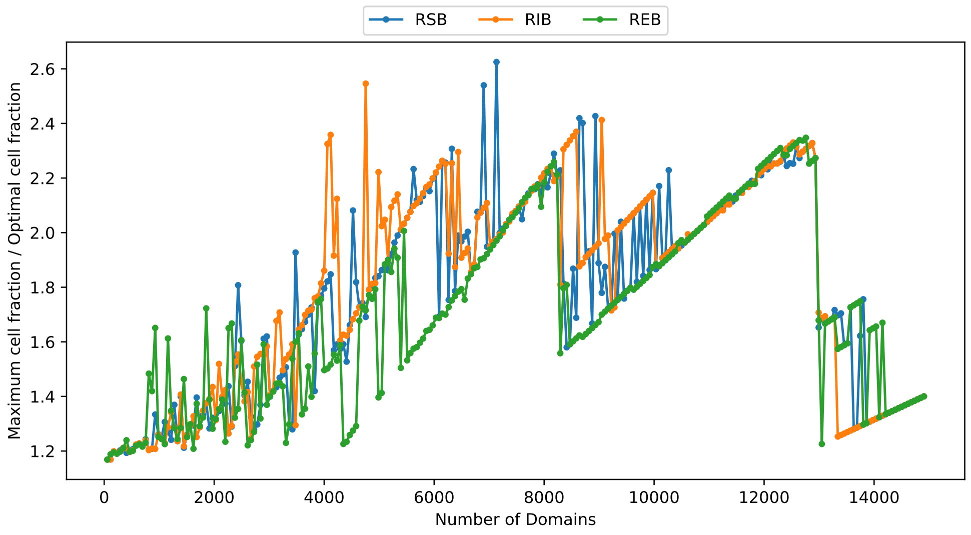


Figure 17: Maximum-to-optimal cell fraction ratio for each partitioning method as a function of number of domains in the axially and radially aligned scheme.

rate decreases due to jagged subdomain boundaries. This caused graph partitioning methods to significantly reduce runtime for problems that were not highly decomposed but actually increase runtime for highly decomposed problems. This is because the assembly-based decomposition method used rectangular (non-jagged) subdomains. This indicates that it may be advantageous to create a high quality decomposition method that enforces rectangular subdomains. However, this approach will not generalize to other lattice types, such as hexagonal lattices.

In the current MPACT implementation, there is no significant difference in runtimes when using any of the three partitioning methods discussed, although REB typically results in slightly lower MOC runtimes. The 2D results indicate that the maximum fraction of cells in a subdomain is highly correlated with the runtime of the simulation. Furthermore, 2D results indicate that the parallel efficiency of MOC is highly correlated with the ratio of the optimal cell fraction per subdomain to the maximum cell fraction in a subdomain.

In 3D, MPACT currently requires spatial domains to be axially and radially aligned. If these restrictions are lifted, then other approaches can be used to perform the 3D spatial decomposition. Three 3D decomposition approaches were investigated in this work: radial and axial aligned subdomains, radially aligned subdomains, and an approach with no alignment restrictions. The radially aligned approach is expected to out perform the current approach by an average of 10% for typical cases. However, the unrestricted approach is actually expected to perform worse than the current approach on average. This analysis was performed on a 3D core, which

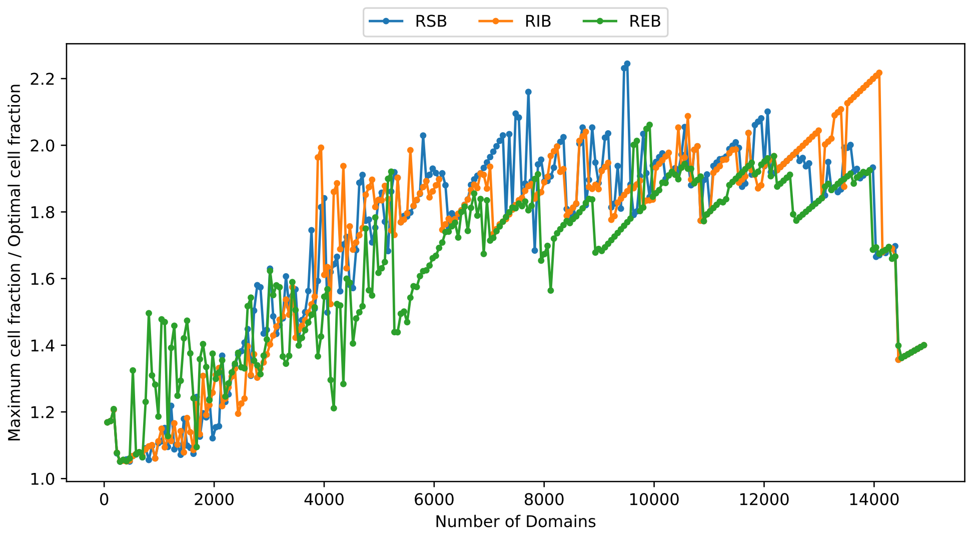


Figure 18: Maximum-to-optimal cell fraction ratio for each partitioning method as a function of number of domains in the radially aligned scheme.

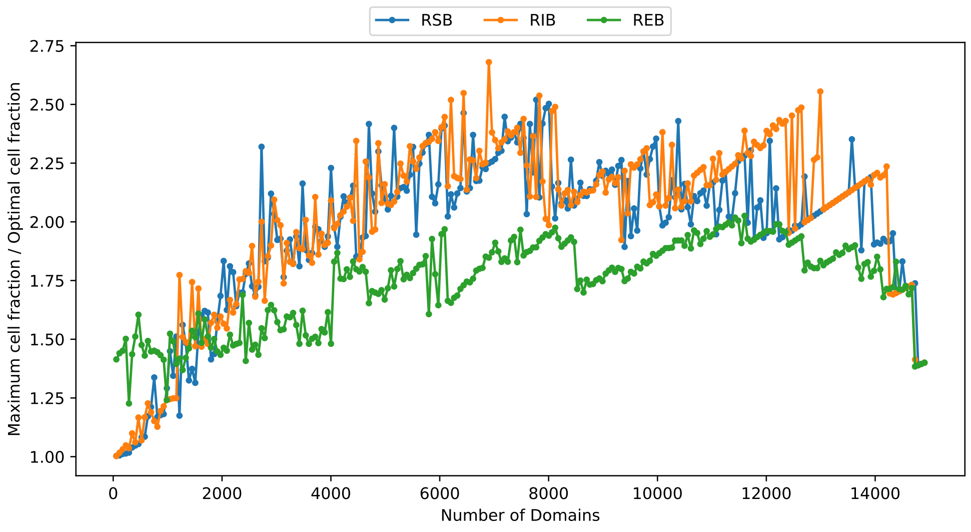


Figure 19: Maximum-to-optimal cell fraction ratio for each partitioning method as a function of number of domains in the unrestricted scheme.

was relatively homogeneous in the axial direction; if a core design were to be more axially heterogeneous, then a more significant increase in performance might be expected.

The parallel efficiency is a measure of how well computational resources are utilized in parallel applications. MPACT's overall parallel efficiency decreases rapidly as more domains are used due to two factors: increased number of iterations, and the inefficiency of parallel CMFD solves. For the 2D VERA progression problem 5a, the overall parallel efficiency of MPACT dropped to nearly 20%. The parallel efficiency of only the MOC computations in MPACT drops to between 40–60%, while the parallel efficiency of the CMFD computation drops to <20%. This causes the CMFD computation to dominate runtime in spatially decomposed cases, and motivates the implementation of a more efficient parallel linear system operator in MPACT rather than using a third-party library.

A. Partition Refinement

A.1. Partition Refinement Methods

The Kernighan-Lin algorithm [22] is often described as one of the earliest developed graph partitioning algorithms; however, the algorithm does not actually create a partitioning of the graph, it improves, or refines, the quality by reducing the number of edges cut between existing partitions [4]. Therefore, in this work, this method and a modified version of it are called *refinement methods*.

As suggested by Pothen [10], the RSB method or other partitioning methods can create a high quality initial parti-

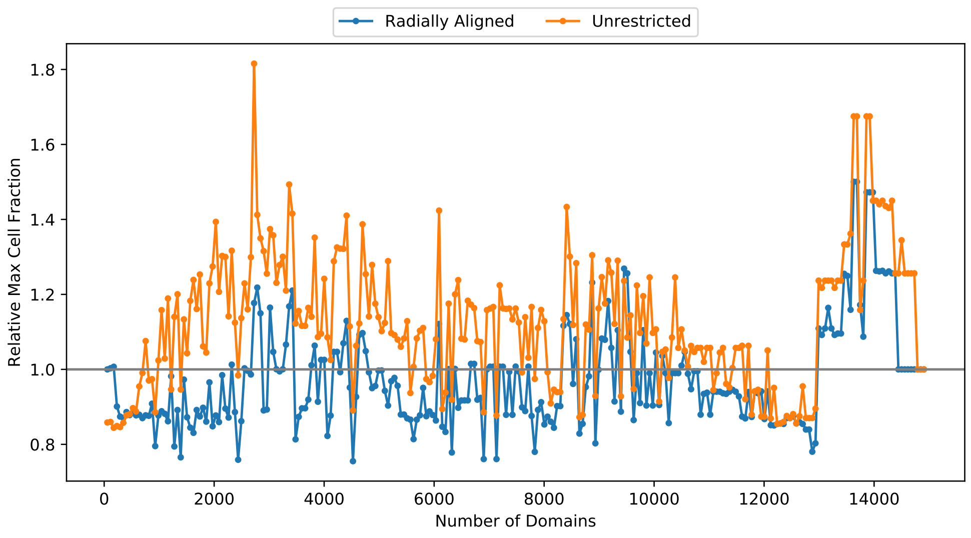


Figure 20: Maximum cell fraction relative to the ARA approach for the RSB partitioning method.

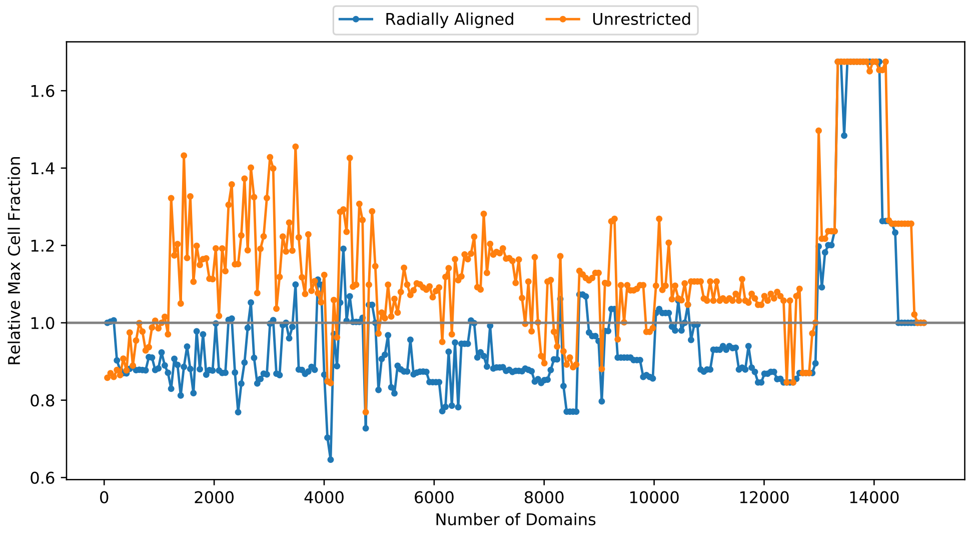


Figure 21: Maximum cell fraction relative to the ARA approach for the RIB partitioning method.

tioning to use in the Kernighan-Lin algorithm. Significant improvements have been made to the efficiency of the original Kernighan-Lin algorithm [23]; however, the graphs of concern in this work are relatively small, and graph decomposition time is negligible when compared to the overall simulation runtime. Two partition refinement methods were examined: the Kernighan-Lin algorithm [22] and a modification to the Kernighan-Lin algorithm which takes some geometric information of the graph into account [3].

The investigated partition refinement algorithms reduce the weight of edges cut between two partitions by swapping vertex pairs between the partitions iteratively. The original Kernighan-Lin algorithm operates entirely on the connectivity of the graph, while the modified Spatial Kernighan-Lin algorithm uses both connectivity and geometric information from the graph.

For each vertex in the graph, D is defined as

$$D_i \equiv E_{E,i} - E_{I,i}, \tag{5}$$

where $E_{E,i}$ is the sum of edge weights from vertex i connecting with vertices outside the partition containing vertex i, and $E_{I,i}$ is the sum of edge weights from vertex i connecting with vertices within the partition containing vertex i. The reduction in communication or "gain," from swapping a pair of vertices (a, b) is defined as

$$g_{(a,b)} \equiv D_a + D_b - 2c_{a,b}. \tag{6}$$

The Kernighan-Lin algorithm is a greedy algorithm, in that it will swap a pair (a, b) with maximal g at the current step. The idea is that by doing this iteratively, the algorithm will lead to a minimized cut-size; in reality, the algorithm

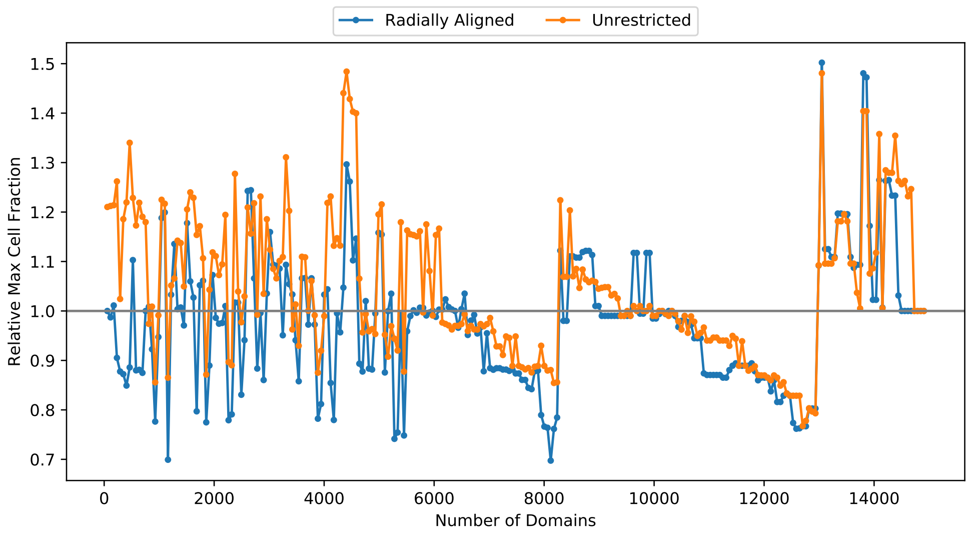


Figure 22: Maximum cell fraction relative to the ARA approach for the REB partitioning method.

will often get stuck in local minima that do not have a global minimized cut-size. Additionally, there may be multiple pairs (a, b) with the same maximal gain value: the algorithm will only consider one of these pairs.

The Spatial Kernighan-Lin algorithm is an adaptation of the Kernighan-Lin algorithm which accounts for the multiple pairs with maximal gain. This algorithm prioritizes vertex pairs which are geometrically distant from one another. The idea behind this modification is that to minimize the edge-cut, the cut should be as straight as possible. By prioritizing distant vertex pairs, the bisector is typically "straightened" out; this process can be likened to pulling on the ends of a string in order to straighten it.

A.2. Partition Refinement Results

In appendix A.1, refinement methods are introduced as a method for further reducing communication. Figure 23 shows that both refinement methods are able to slightly reduce the number of edges cut compared to the cases without refinement for RSB. Similar trends are observed for the other partitioning methods. While these refinement methods offer a slight reduction in communication, the parallel efficiency due to load imbalance may be negatively affected by applying refinement, as shown in figure 24. Communication is not expected to have as significant of an effect as balance, so for the simulations in MPACT, partitioning methods were used without refinement.

Algorithm 5 Kernighan-Lin Algorithm [22]

```
 1: procedure KERNIGHAN-LIN($G(V, E)$, $A$, $B$)
 2:     $g_m = 1$
 3:     while $g_m > 0$ do
 4:         Compute $D\ \forall\ V$ (Equation (5))
 5:         Let $a_v, b_v, g_v$ be empty sets
 6:         for $n = 1$ to $N/2$ do
 7:             Find unmarked pair $(a, b)$ such that:
                    1. $a \in A$ and $b \in B$
                    2. $g$ is maximized (Equation (6))
 8:             if relative weights would be conserved
                  if $a$ and $b$ are swapped then
 9:                 Append $a$ to $a_v$, $b$ to $b_v$, and $g$ to $g_v$
10:                 Update $D$ values as if $a, b$ have been swapped
11:             else
12:                 End search
13:             end if
14:         end for
15:         Find $k$ maximizing $g_m = \sum_{i=1}^{k} g_v(i)$
16:         if $g_m > 0$ then
17:             Exchange vertices in $a_v(1:k)$ and $b_v(1:k)$
18:         end if
19:     end while
20: end procedure
```

Acknowledgments

This material is based upon work supported under an Integrated University Program Graduate Fellowship.

This research was supported by the Consortium for Advanced Simulation of Light Water Reactors www.casl.gov, and Energy Innovation Hub (http://www.energy.gov/hubs) for Modeling and Simulation of Nuclear Reactors under U.S. Department of Energy Contract No. DE-AC05-00OR22725.

This research made use of the resources of the High Performance Computing Center at Idaho National Laboratory, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517.

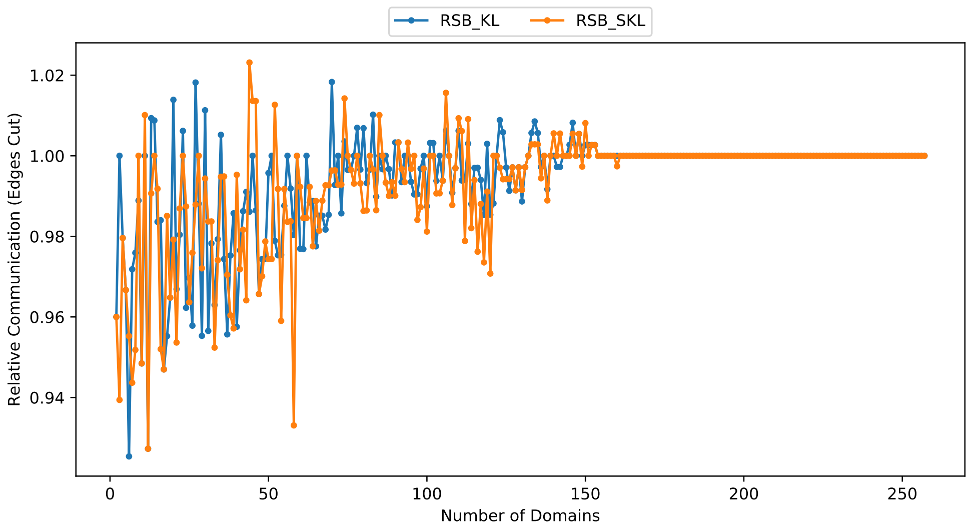


Figure 23: Communication relative to the RSB method without refinement as a function of number of domains for each refinement method using the RSB partitioning method.

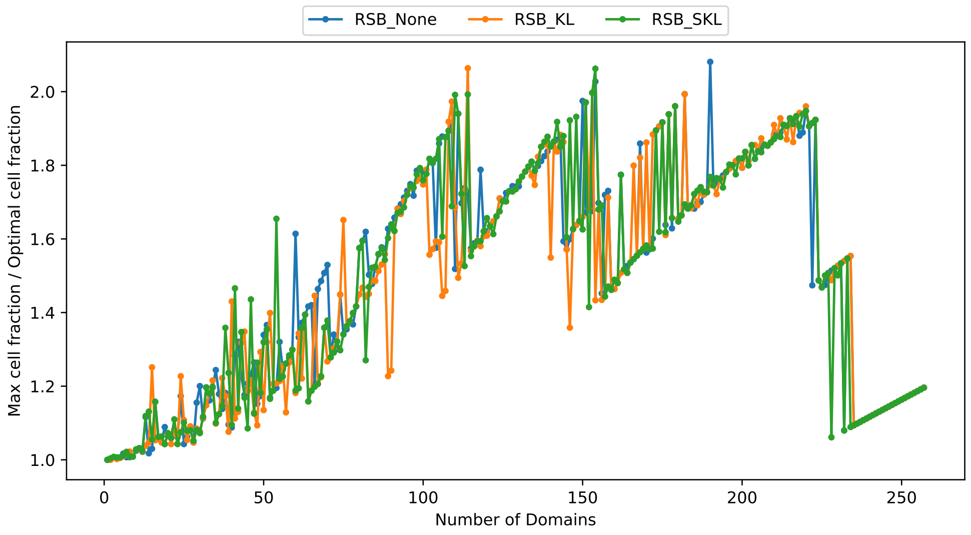


Figure 24: Ratio of maximum cell fraction to optimal cell fraction for the RSB partitioning method with each refinement method.

Algorithm 6 Spatial Kernighan-Lin Algorithm [3]

```
1: procedure SPATIAL KERNIGHAN-LIN($G(V, E), A, B$)
2:     $g_m = 1$
3:     while $g_m > 0$ do
4:         Compute $D \forall V$ (Equation (5))
5:         Let $a_v, b_v, g_v$ be empty sets
6:         for $n = 1$ to $N/2$ do
7:             Allow $(f_a, f_b)$ to be sets from $A, B$ satisfying:
                 1. $a \in A, b \in B$
                 2. $g$ is maximized (Equation (6))
                 3. $a$ and $b$ are on the boundary between $A$ and $B$
8:             Find pair $(f'_a, f'_b)$ such that distance is maximized
9:             if No pair found then
10:                Search using standard Kernighan-Lin rules
11:            end if
12:            if relative weights would be conserved
                 if $a$ and $b$ are swapped then
13:                Append $a$ to $a_v$, $b$ to $b_v$, and $g$ to $g_v$
14:                Update $D$ values as if $a, b$ have been swapped
15:            else
16:                End search
17:            end if
18:        end for
19:        Find $k$ maximizing $g_m = \sum_{i=1}^{k} g_v(i)$
20:        if $g_m > 0$ then
21:            Exchange vertices in $a_v(1 : k)$ and $b_v(1 : k)$
22:        end if
23:    end while
24: end procedure
```

References

[1] J. Askew, A Characteristics Formulation of the Neutron Transport Equation in Complicated Geometries (1972).

[2] MPACT Team, MPACT Theory Manual v2.2.0, Tech. rep., Consortium for Advanced Simulation of Light Water Reactors (2016).

[3] A. Fitzgerald, Z. Dodson, S. Stimpson, B. Kochunas, Automated Decomposition of a Structured Grid, in: Transactions of the American Nuclear Society, Vol. 117, Washington D.C., 2017, pp. 731–734.

[4] U. Elsner, Graph partitioning A Survey, Tech. rep., Technische Universitat Chemnitz (1997).
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.2060

[5] Y. Yao, B. Richards, Parallel CFD Computation on Unstructured Grids, in: Parallel Computational Fluid Dynamics, Elsevier, 1998, pp. 289–296. doi:10.1016/B978-044482849-1/50035-X.
URL http://linkinghub.elsevier.com/retrieve/pii/B978044482849150035X

[6] S. Kosaka, E. Saji, Transport theory calculation for a heterogeneous multi-assembly problem by characteristics method with direct neutron path linking technique, Journal of Nuclear Science and Technology 37 (12) (2000) 1015–1023. doi:10.1080/18811248.2000.9714987.

[7] S. Stimpson, B. Collins, Flexible Spatial Partitions in MPACT Through Module-Based Data Passing, in: Transactions of the American Nuclear Society, Vol. 116, San Francisco, 2017, pp. 654–657.

[8] G. M. Morton, A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing, Tech. rep., IBM Ltd., Ottawa, Canada (1966).

[9] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM Journal on Scientific Computing 20 (1) (1998) 359–392.

[10] A. Pothen, H. D. Simon, K.-P. P. Liu, Partitioning Sparse Matrices with Eigenvectors of Graphs, SIAM Journal on Matrix Analysis and Applications 11 (3) (1989) 430–452.

[11] H. D. Simon, Partitioning of unstructured problems for parallel processing, Computing Systems in Engineering 2 (2-3) (1991) 135–148. doi:10.1016/0956-0521(91)90014-V.

[12] D. A. Spielman, S. H. Teng, Spectral partitioning works: Planar graphs and finite element meshes, Linear Algebra and Its Applications 421 (2-3 SPEC. ISS.) (2007) 284–305. doi:10.1016/j.laa.2006.07.020.

[13] S. H. Hsieh, G. H. Paulino, J. F. Abel, Recursive spectral algorithms for automatic domain partitioning in parallel finite element analysis, Computer Methods in Applied Mechanics and Engineering 121 (1-4) (1995) 137–162. doi:10.1016/0045-7825(94)00704-Q.

[14] M. Fiedler, Algebraic Connectivity of Graphs, Czechoslovak Mathematical Journal 23 (2) (1973) 298–305.

[15] N. Floros, J. S. Reeve, J. Clinckemaillie, S. Vlachoutsis, G. Lonsdale, Comparative efficiencies of domain decompositions, Parallel Computing 21 (11) (1995) 1823–1835. doi:10.1016/0167-8191(95)00031-7.

[16] C. Farhat, A simple and efficient automatic fem domain decomposer, Computers and Structures 28 (5) (1988) 579–602. doi:10.1016/0045-7949(88)90004-1.

[17] M. A. Nasra, D. T. Nguyen, An algorithm for domain decomposition in finite element analysis, Computers & Structures 39 (3/4) (1991) 277–289.

[18] K. S. Smith, Nodal method storage reduction by nonlinear iteration, in: Transactions of the American Nuclear Society, Vol. 44, 1983, pp. 265–266.

[19] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, B. Smith, H. Zhang, PETSc Users Manual Revision 3.10, Tech. rep., Argonne National Laboratory (ANL), Argonne, IL (United States) (jun 2018).

[20] B. Collins, S. Stimpson, B. W. Kelley, M. T. Young, B. Kochunas, A. Graham, E. W. Larsen, T. Downar, A. Godfrey, Stability and accuracy of 3D neutron transport simulations using the 2D/1D method in MPACT, Journal of Computational Physics 326 (2016) 612–628. doi:10.1016/j.jcp.2016.08.022.
URL http://dx.doi.org/10.1016/j.jcp.2016.08.022

[21] A. T. Godfrey, VERA Core Physics Benchmark Progression Problem Specifications, Tech. Rep. 4 (2014).
URL https://www.casl.gov/sites/default/files/docs/CASL-U-2012-0131-004.pdf

[22] B. Kernighan, S. Lin, An efficient heuristic for partitioning graphs, Bell Systems Technical Journal 49 (1970) 291–308.

[23] C. M. Fiduccia, R. M. Mattheyses, A Linear-Time Heuristic for Improving Network Partitions, Tech. rep., General Electric Research and Development Center, Schenectady, NY (1982).