Exploiting Domain Knowledge to Optimize Mesh Partitioning for Multi-Scale Methods

M. Hasan Jamal[1], Milind Kulkarni[1], and Arun Prakash[2]

[1] School of Electrical and Computer Engineering, Purdue University

[2] Lyles School of Civil Engineering, Purdue University

Motivation and Challenges

- Many complex scientific computing problems use multi-scale methods that utilize recursive domain-decomposition.
  - Problem decomposed into smaller subdomains
  - Subdomains independently solved at different spatial and temporal scales.
  - Solutions coupled back to get the desired solution.
  - Fine grain simulation for “area of interest” at higher computation cost.
  - Approximation with a much coarser model for remaining parts.
  - Assures necessary level of accuracy of solution at lower computation cost.
  - Tree like structure with sub-tree at lower timescale executing multiple times.

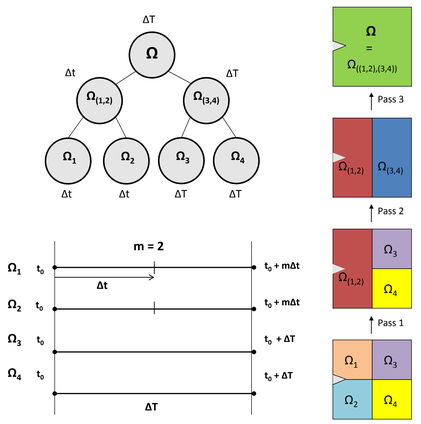


Figure 1. Recursive domain decomposition and coupling for multi-scale problems

- For a given large problem domain, exponentially large search space of possible mesh decompositions.
- Performance influenced by following parameters:
  - Variation in size of mesh elements
  - Number of elements in a partition
  - Appropriate timescale value for each partition
  - Number of partitions at each timescale
  - Number/size of partitions in the system
- Multiple orders of magnitude difference in cost for running different decompositions
- Finding an optimal mesh decomposition for multi-scale problems is non-trivial.
- A substantial amount of domain knowledge required to optimize for multiple parameters.
- METIS does not incorporate the necessary domain knowledge for multi-scale problems resulting in poorly performing decompositions.
- Not feasible for a domain scientists to partition mesh manually for complex multi-scale problems.

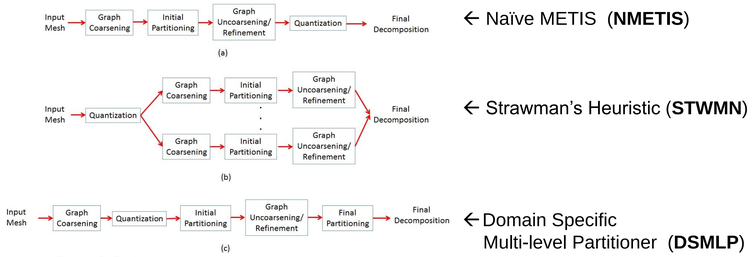
A **domain specific partitioner** **is needed that exploits domain knowledge** to produce mesh decompositions **automatically** which are **optimized** for **total number of subdomains, ratio between timescales, and the number of subdomains assigned to each timescale.**

METIS Limitations

- Number of partitions required is provided as an input parameter.
- Determining the optimal number of partitions is based on guess work.
- Multi-scale problems are sensitive to element sizes. METIS insensitive to element size.
- Mixing small and large elements in an individual partition results in that partition running at a smaller timescale leading to higher computational cost.
- METIS balances for number of elements per partition which may lead to a mixture of small and large element in all partitions resulting in running the entire problem at a lower timescale which is computationally very inefficient.

Domain Specific Multi-level Partitioners

3 Partitioning approaches



← Naïve METIS (**NMETIS**)

← Strawman’s Heuristic (**STWMN**)

←Domain Specific Multi-level Partitioner (**DSMLP**)

OUR APPROACH

Figure 2. Partitioning phases in various decomposition approaches

STWMN

- Intuitive choice a domain scientist will make
- Applicable for less complex problems (i.e. Beam Notch problem)
- Separate elements based on sizes
  - Select elements within a certain radius around the “area of interest”
  - Two initial sub-meshes
  - Partition each sub-mesh with METIS
  - Our results include radii 0.33, 0.67, 1.0, 1.25, 1.5, 1.75, and 2.0
- Number of partitions per timescale decided based on number of elements at each timescale

DSMLP

- Input mesh is represented as a graph
  - Node weights represent number of equations for an element
  - Edge weight represent number of equations at shared interface
- Domain specific cost model to determine estimate cost of executing various operations.
  - Subdomain solve
  - Coupling operations
- Coarsening phase
  - Merge nodes if cost is reduced
  - Merge only if difference in theoretical timescale of both nodes is smaller than a defined threshold.

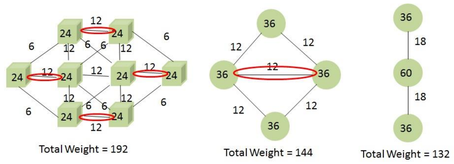


Figure 3. Illustration of coarsening phase exploiting domain knowledge

- Initial Partitioning
  - Use cost model to help determine optimized workload distribution between timescales
  - 2 quantization levels in this study
- Refinement
  - Move elements between timescales to reduce large interface cost
  - Ensure accuracy and numerical stability by theoretical timescale information of elements
- Final decomposition
  - By utilizing the cost model recursively bisect the graph until the cost is decreasing.

Input Mesh #1: Beam with Notch (BN)

- 36,552 tetrahedral elements
- 30m x 5m x 1m
- Notch depth = 1m

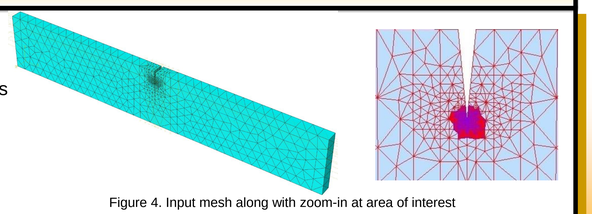


Figure 4. Input mesh along with zoom-in at area of interest

Input Mesh #2: Cuboid with fractures (FC)

- 100,720 tetrahedral elements
- 4.8m x 1.6m x 1.6m
- Three cubes to form a cuboid
- Two cube with needle shaped fractures
- One center cube with disk shaped fracture

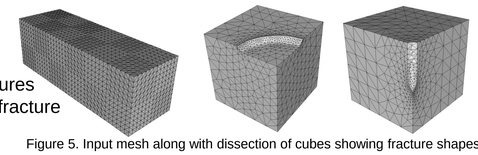


Figure 5. Input mesh along with dissection of cubes showing fracture shapes

Results

Experimental Setup

- Intel Xeon E5-4650 with four 8-core processors (32 cores total) running at 2.7GHz
- Cilk runtime system for parallel execution

Performance Comparison

- NMETIS and STWMN are explored with different number of partitions.
- STWMN is explored with different quantization functions.

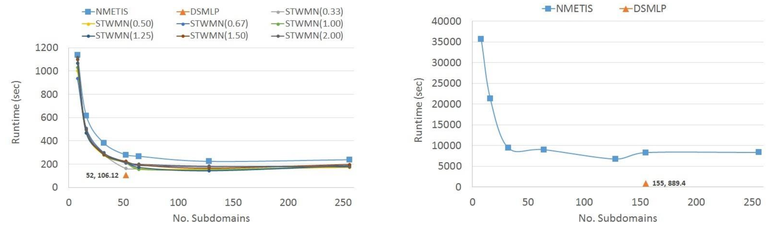


Figure 6. Execution time for solving decompositions produced with various partitioners for BN and FC

DSMLP is 35% and 111% to 656% faster in execution time for BN and FC respectively, over best produced decomposition by STWMN and NMETIS.

Parallel Performance

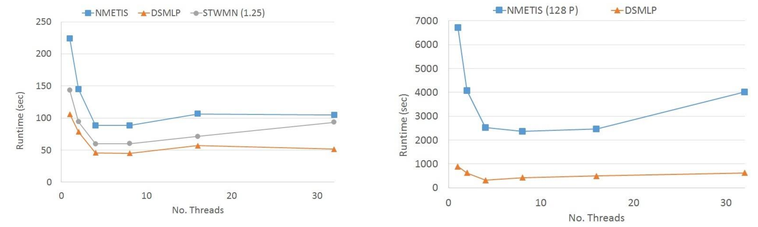


Figure 7. Parallel Execution time for BN and FC

DSMLP delivers better parallel performance than NMETIS and STWMN

Table 1. Speedups of STWMN and DSMLP schedules over baseline NMETIS

	No. Thread = 1		No. Thread = 8	
	STWMN	DSMLP	STWMN	DSMLP
BN	1.75	2.11	1.47	1.97
FC	N/A	7.56	N/A	5.53

Table 2. Inherent parallelism in coupling schedules

Input	Schedule	Work (sec)	Critical Path (sec)	Parallelism
BN	METIS	224	87.70	2.55
	STWMN(1.25)	143.1	59.13	2.42
	DSMLP	106.12	54.56	1.95
FC	METIS	6725	2671.7	2.52
	DSMLP	889.4	450.6	1.97

DSMLP schedule achieves best speedups in single and multiple threaded execution

Parallelism saturates beyond 8 threads due to structure of input

DSMLP less scalable than NMETIS and STWMN because it optimizes for work. NMETIS generates unnecessary work.