Towards Scalable Identification of Motifs Representing Non-Determinism in HPC Simulations

Ali Y Khan[1], Sanjukta Bhowmick[1] (advisor), Michela Taufer (advisor)[2]
[1]University of North Texas, [2]University of Tennessee Knoxville

Contributions: Scalable Motif based Network Alignment | Compute Multiple Motifs per Execution | Sensitive to Changes in Network | Adapted to Graph Structure

Network Alignment

Event graphs representing large HPC simulations can differ slightly across executions, even when the execution parameters remain unchanged. Identifying these slight differences can help in identifying the regions of non-determinism in the code, and subsequently lead to more reliable and reproducible software.

Current network alignment algorithms cannot effectively identify the difference in nearly similar graphs.

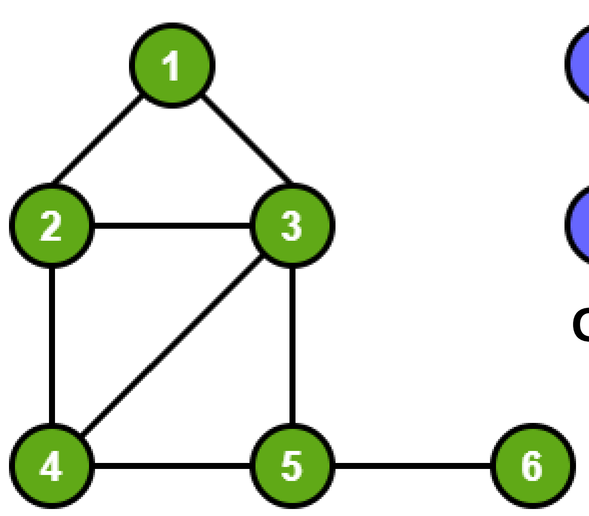
We present a motif based network alignment algorithm that focuses on identifying subtle differences in graphs.

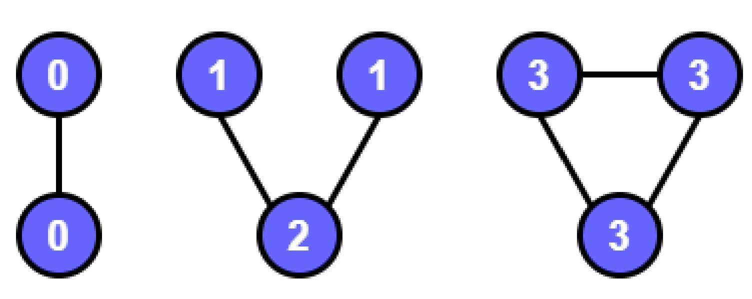
Graphlet Degree Vector

Graphlet (or motifs): structures formed of small number of vertices.

Orbit: The position of a vertex in the graphlet

Graphlet Degree Vector (GDV): Number of times a vertex appears in a given set of orbits





Motifs of size of 2 and 3.
The numbers inside the vertex give the orbit.

	Orbit 0	Orbit 1	Orbit 2	Orbit 3
Vertex 1	2	5	1	1
Vertex 3	4	7	6	3
Vertex 5	3	5	3	1

GDV values of vertices 1,3, and 5

Acknowledgements

We are grateful for the assistance from Krishna Sai Ujwal Kambhumpati, Nigel Tan, and Patrick Bell. Support of XSEDE and IBM through a Shared University Research Award is also acknowledged.
This work is supported by NSF grants 1900765 and 1900888.

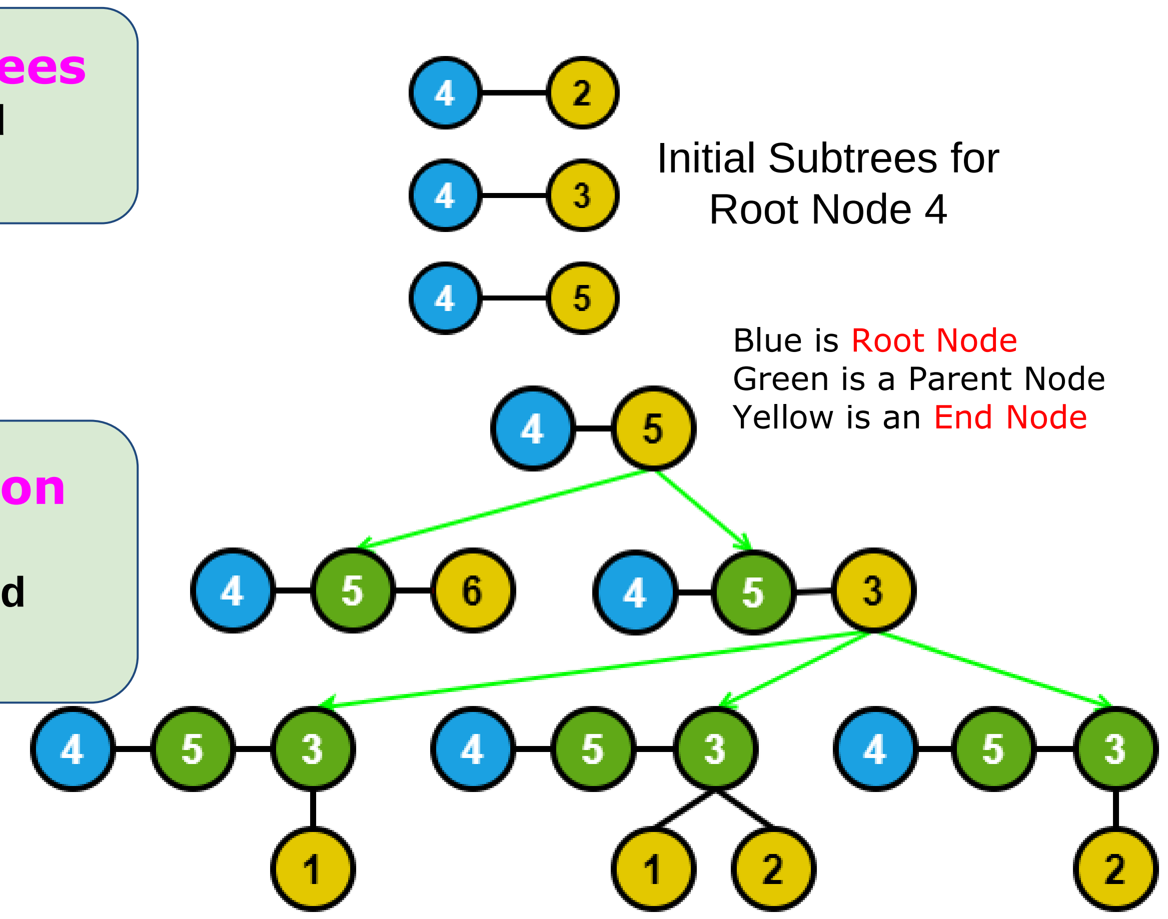
Algorithm for Computing GDV

1. Create Initial Subtrees
Pick root node from graph and create root-neighbor subtrees

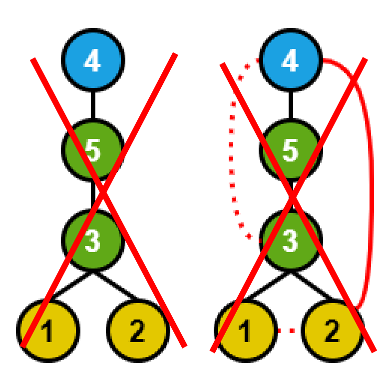
2. Subtree Enumeration
Create copies of subtree and expand all combinations of end nodes

3. Motif Generation
Add all combinations of valid backedges to generate motifs
a) subtree is valid if neighbor of root > end node
b) back edge is valid if connects 2 lowest nodes in cycle

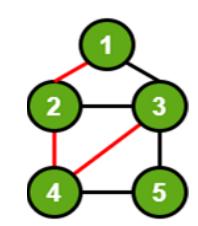
4. Calculate GDV
Use distance and degree vectors to identify corresponding orbit and update GDV
Distance Vector: number of node at distance x from node
Degree Vector: sorted list of degree of each node

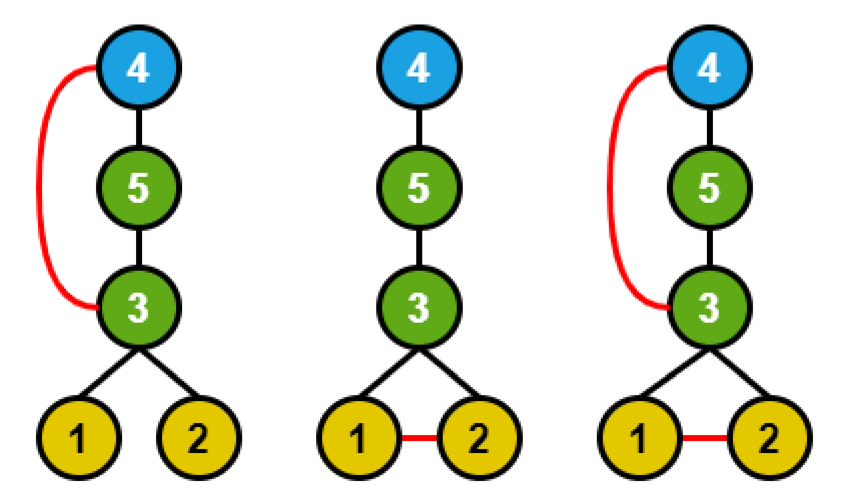


Subtree Expansion of Initial Subtree (4,5)



Not Valid Motifs





Valid Motifs from Subtree

	M1	M2	M3
Degree Vector	[1,1,2,2,4]	[1,2,2,2,3]	[2,2,2,2,4]
Distance Vector	[1,2,2,0,0]	[1,1,1,2,0]	[1,2,2,0,0]
Orbit #	**32**	**27**	**43**
Motif #	14	13	18

Orbit calculation for motifs M1, M2, M3

Challenges

Redundant Computation: Naive motif generation creates exponentially redundant computation
Solution: We use the two constraints for **valid subtrees and backedges** to generate subtrees exactly once

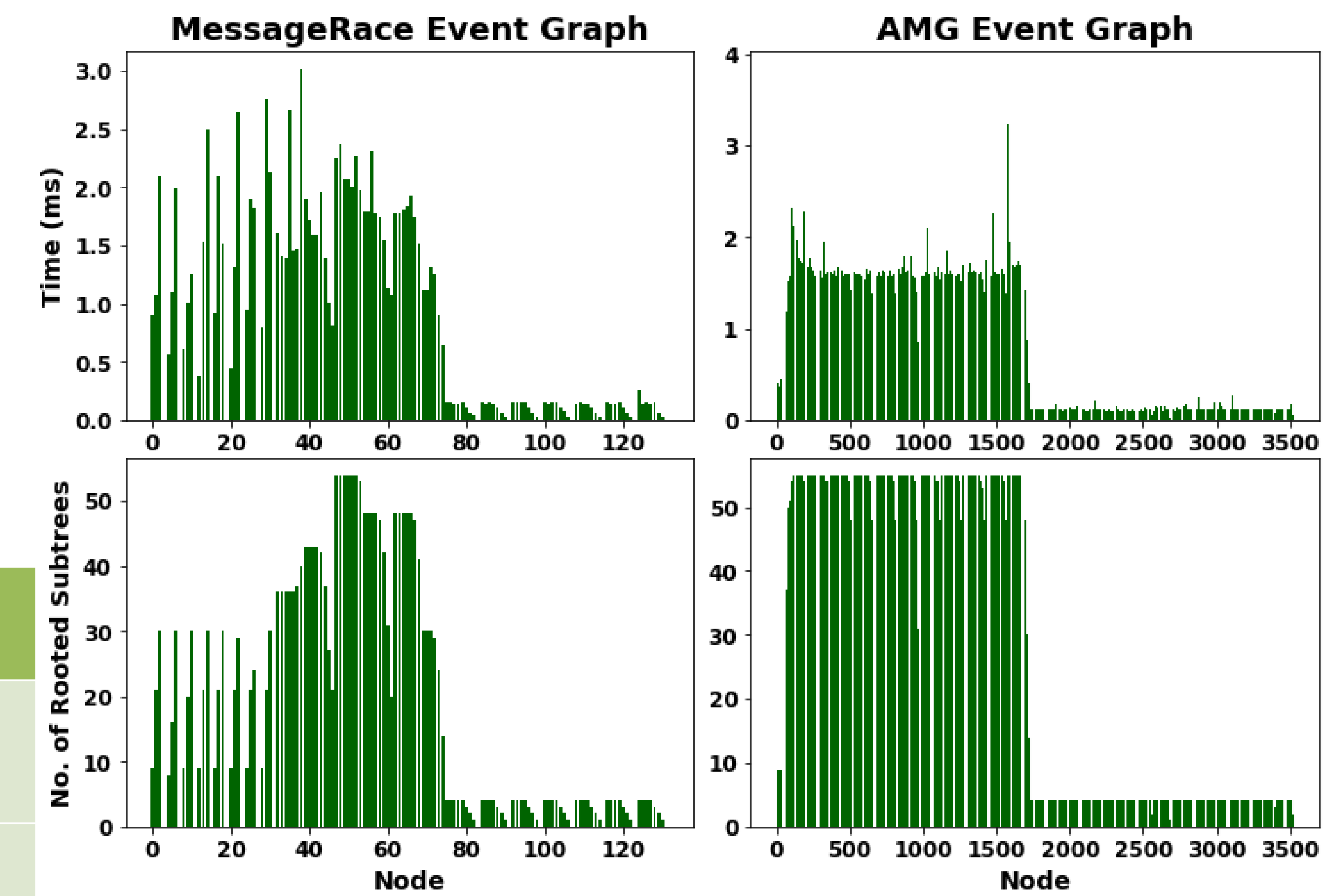
Memory Usage: Memory for storing subtrees from each root can increase exponentially
Solution: We store the trees as **Prufer sequences,** which can be stored as strings. We will also use **limited storage queue** to prevent new tree generation when full

Load Balancing: Not all root vertices perform equal amount of computation, leading to computational load imbalance.
Solution: We apply **dynamic scheduling** as well as **vertex splitting**, processing the neighbors in batches for high degree vertices.

Preliminary Results

Environment Settings: Jetstream2 Single Node (32 CPU cores, 125 GB RAM)
Comparing event graphs of AMG (algebraic multigrid application from CORAL Benchmark Suite) vs. Message Race (N process with N-1 process sending a message to the root).
Nodes ordered according to increasing core number.

Execution time for generating subtrees per node and number of subtrees at each node



References

[1] D. Chapp, N. Tan, S. Bhowmick and M. Taufer. Identifying Degree and Sources of Non-Determinism in MPI Applications Via Graph Kernels, in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 12, pp. 2936-2952, 1 Dec. 2021, doi:10.1109/TPDS.2021.3081530.

[2] Shi, T., Zhai, M., Xu, Y., & Zhai, J. (2020, November). Graphpi: High performance graph pattern matching through effective redundancy elimination. In SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 1-14). IEEE.

PhDPosters.com