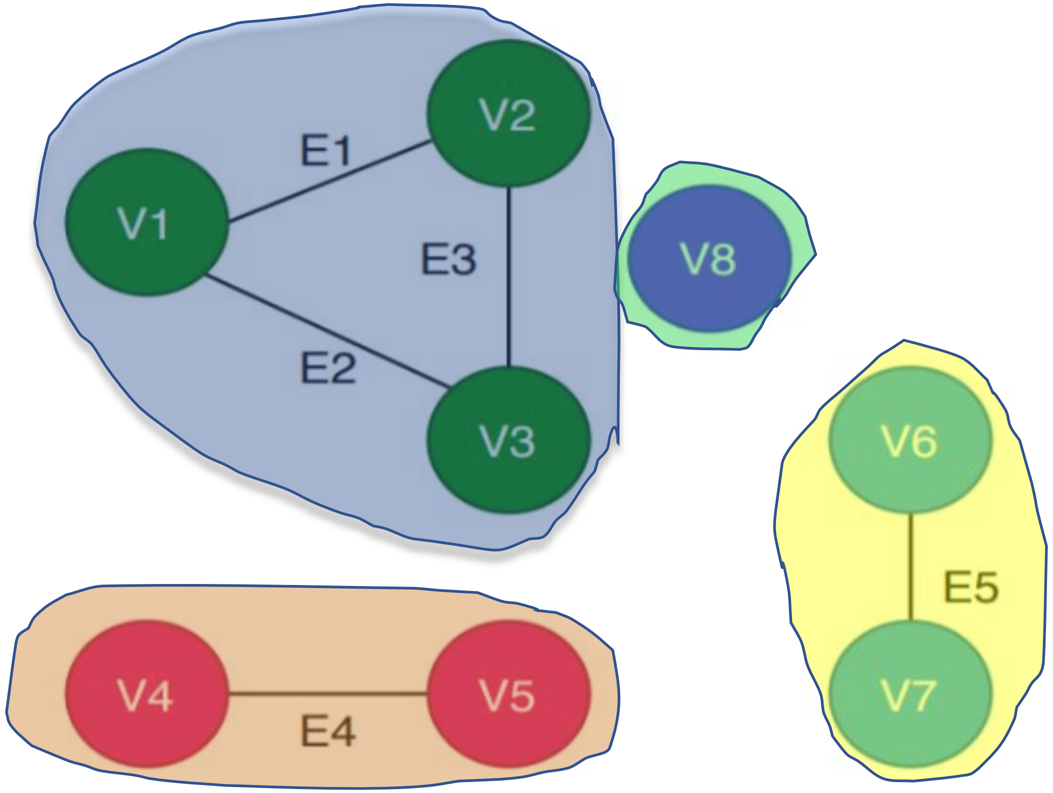
Minimum-Mapping based Connected Components Algorithm

Zhihui Du (Presenter)
Oliver Alvarado Rodriguez
Fuhuan Li
Mohammad Dindoost
David A. Bader

This research is supported by NSF grant CCF-2109988

Connected Components Problem

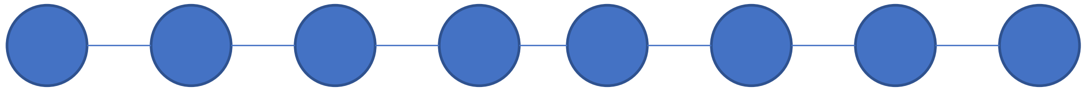
- Graph partition
  - Given undirected graph G=<V,E> $=G_1+G_2+\ldots+G_k$
    - All vertices in G_i are connected with each other (or single vertex)
    - G_i And G_j no overlap
- Importance
  - Graph Structure
  - Algorithm



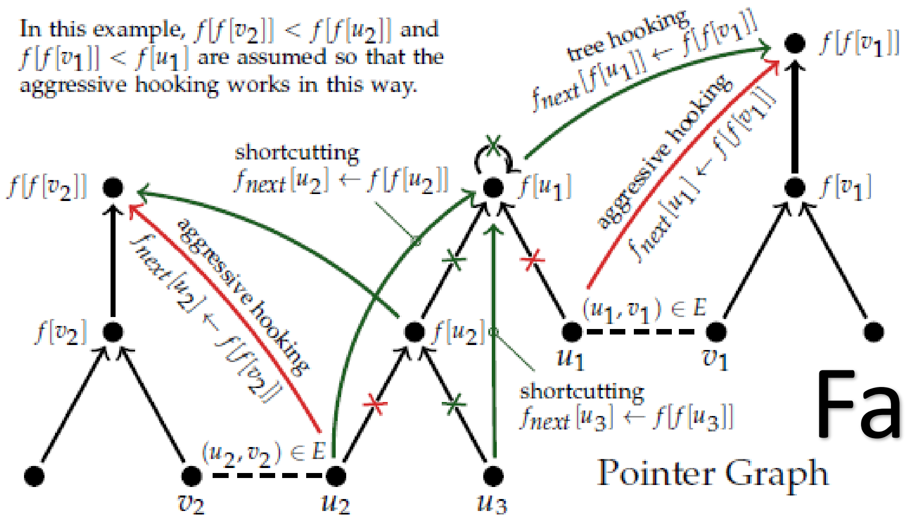
New Jersey Institute of Technology

Existing Methods (Abstract)

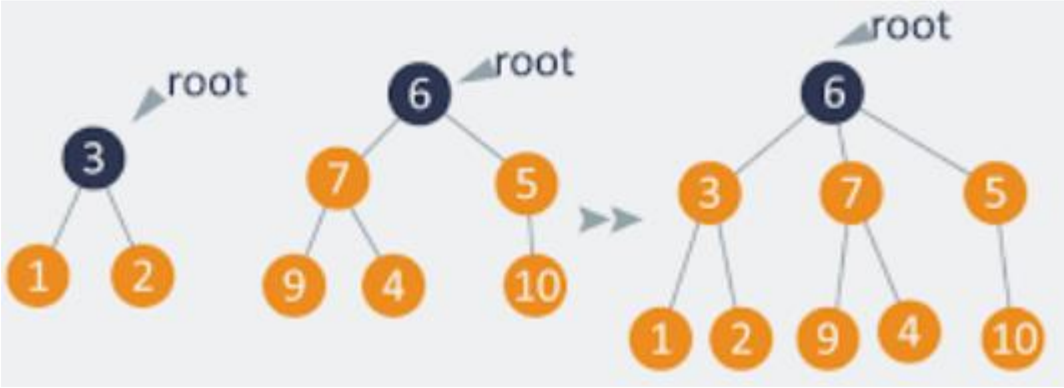
- Graph Traversal
  - O(n) iterations
  - Label Propagation/BFS operations



- Tree
  - n work-log(n) iteration
  - Hooking-Compressing operation
- Disjoin Set
  - Approximate linear time (sequential)
  - Union-Find Operations

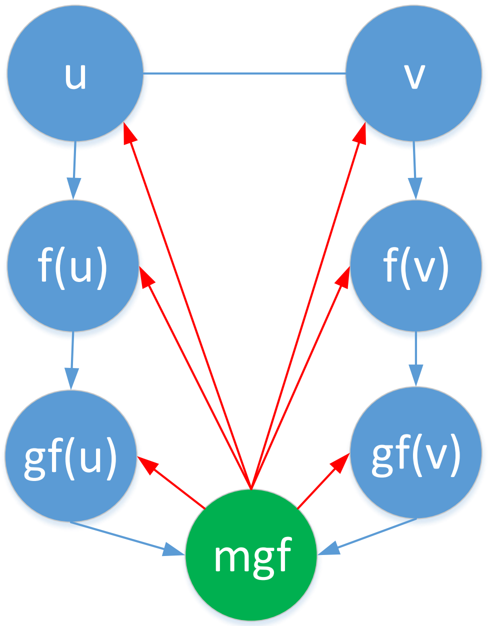


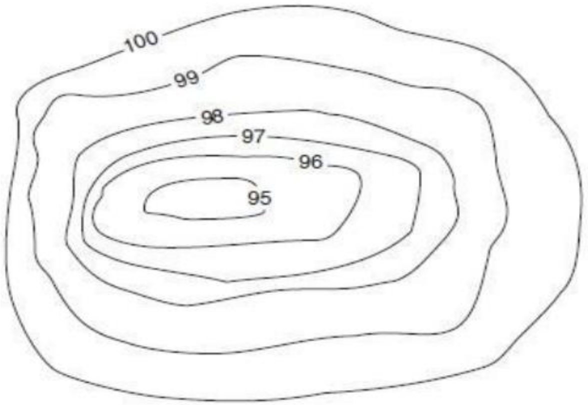
FastSV



Minimum-Mapping based Contour Algorithm

- Contour line
  - Mapping the vertices to different contour lines
  - Give edge <u,v>
    - **Search**: minimum label of their ancestors
    - **Remap**: Update labels of descendants
  - Converge in log(n) time
- Feature
  - Simple Operations
  - Easy to Parallel
- Algorithm
  - Initialize the label array
  - Repeat
    - Forall edge=<u,v>
      - Minimum-mapping based on edge <u,v>
  - Until converge

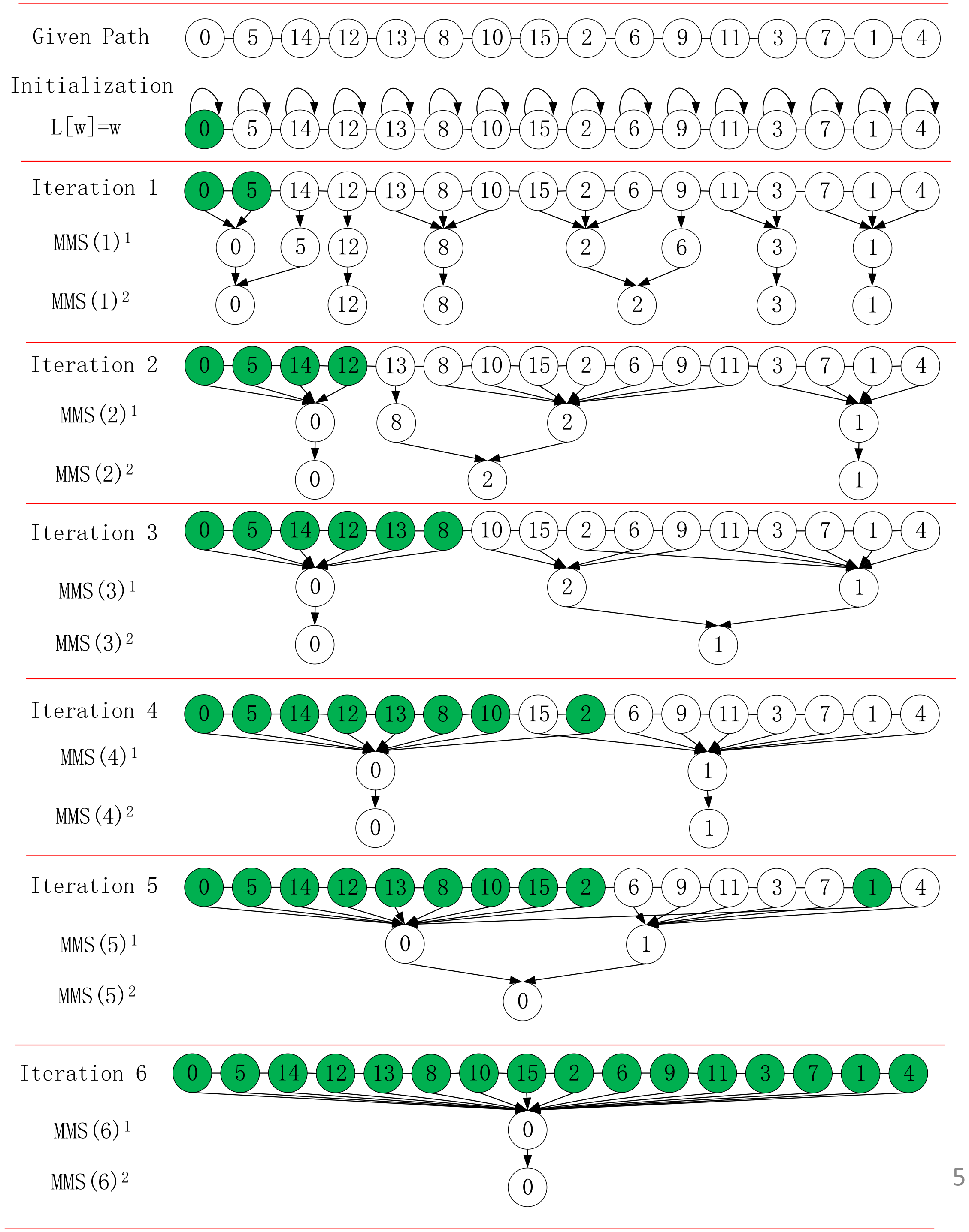




New Jersey Institute of Technology

Example

Converge in log(n) iterations



New Jersey Institute of Technology

Algorithm Implementations

- Arkouda/Chapel Implementation
  - Contour
    - Variants (search steps, update methods)
      - C-1/C-2/C-S/C-CAS/C-Syn
  - FastSV
- High-Level Graph Package Implementation
  - LAGraph (C)
    - Contour
    - FastSV
- Low-Level Graph Package
  - Graph Based Benchmark Suite –GBBS (C++)
    - Contour
    - Simplified SV

Algorithm 1: Voltage-Mapping based *Contour* Algorithm

Contour(G)

/* $G = \langle E, V \rangle$ is the input graph with edge set E and vertex set V. $m = |E|$ is the total number of edges and $n = |V|$ is the total number of vertices. */

forall *i in 0..n-1* **do**

  $L[i] = i$

  $L_u[i] = i$

end

/* Initialize the label array L, L_u */

while *(There is any label change in L)* **do**

  forall $(e = \langle w, v \rangle \in E)$ **do**

    $VO^2(L_u, L, w, v)$

  end

  $L = L_u$

end

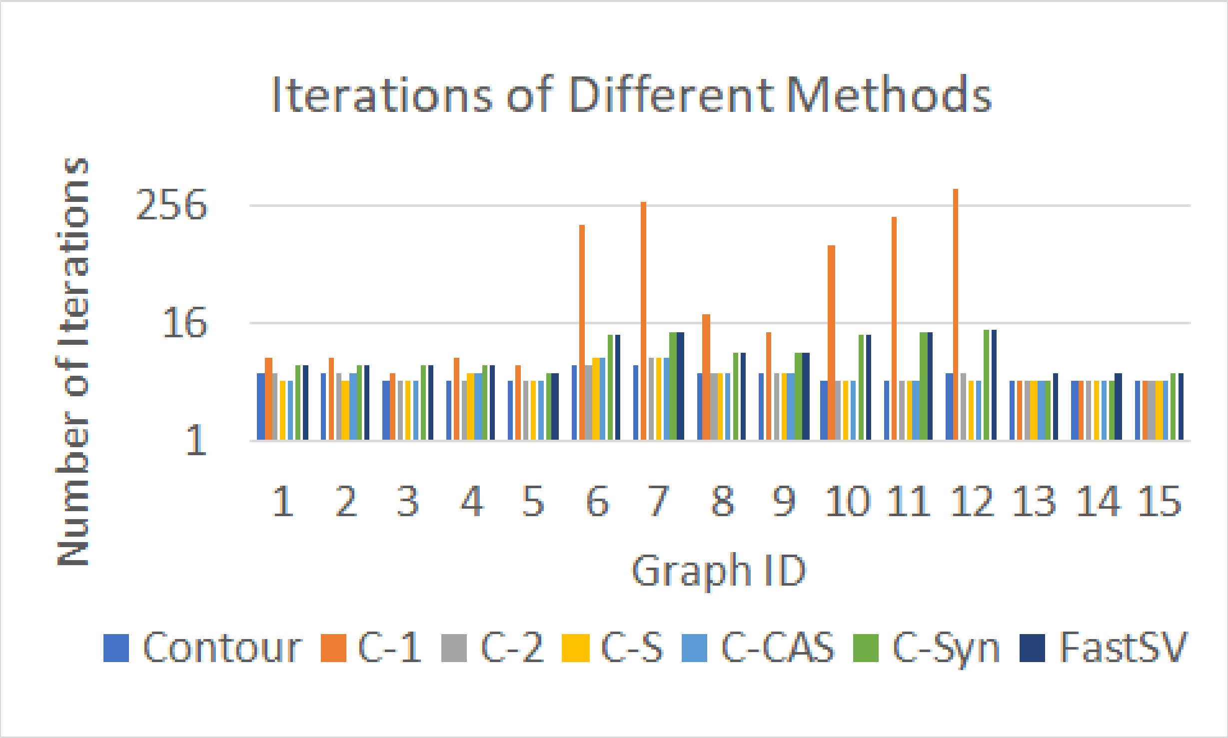
return L

New Jersey Institute of Technology

Experimental Results (number of iterations)

Table 1. Dataset 1 - Real-World and Synthetic Graphs

Graph Type	Graph ID	Graph Name	m	n
Real-World Graph	1	loc-brightkite_edges	214078	58228
	2	soc-Epinions1	405740	75879
	3	amazon0601	2443408	403394
	4	com-youtube.ungraph	2987624	1134890
	5	soc-LiveJournal1	68993773	4847570
	6	kmer_V1r	232705452	214005017
	7	kmer_A2a	180292586	170728175
	8	uk-2002	261787258	18484117
	9	uk-2005	783027125	39454746
Synthetic Graph	10	rgg_n_2_21_s0	14487995	2097148
	11	rgg_n_2_22_s0	30359198	4194301
	12	rgg_n_2_24_s0	132557200	16777215
	13	kron_g500-logn16	2456071	55321
	14	kron_g500-logn18	10582686	210155
	15	kron_g500-logn20	44619402	795241

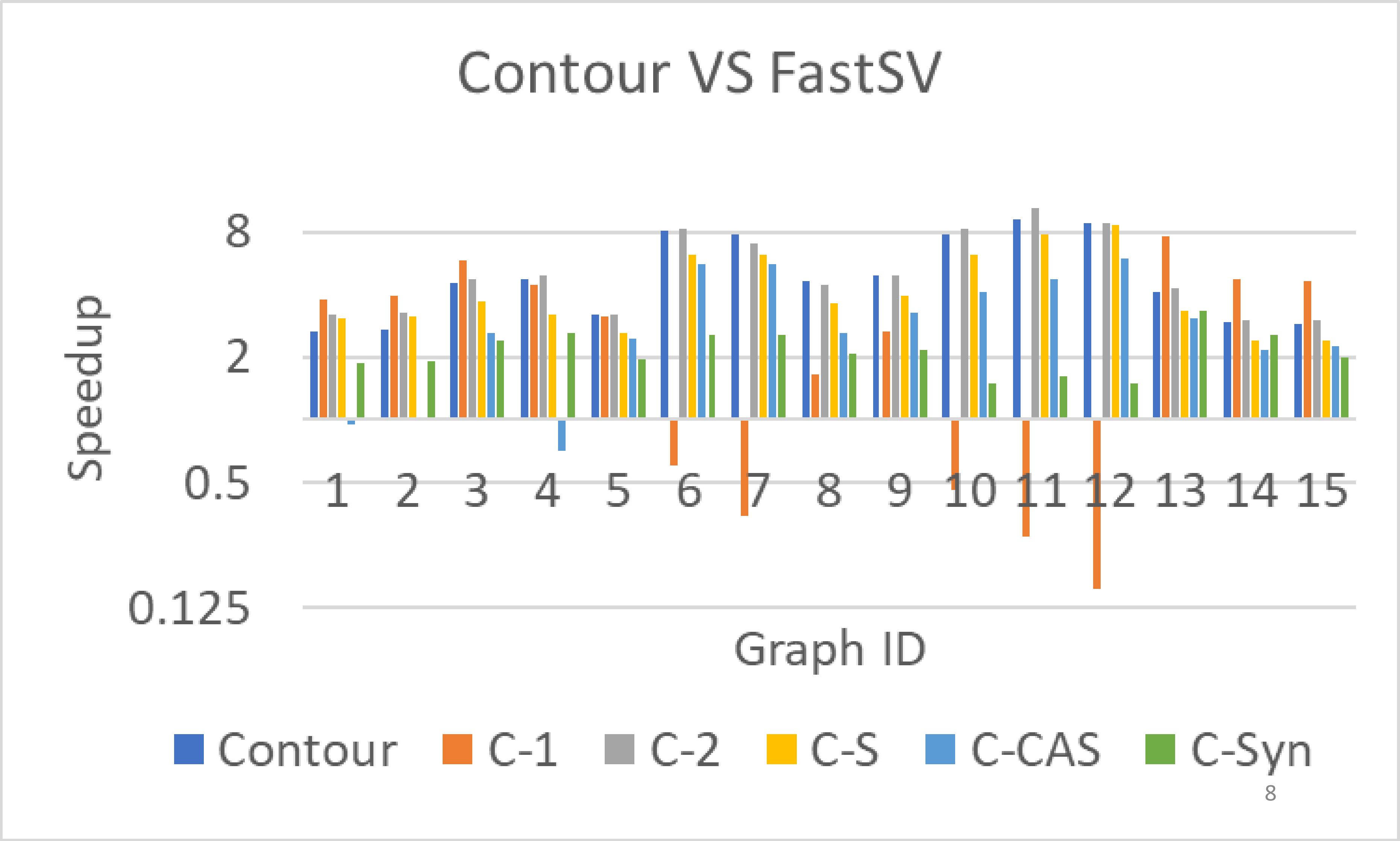


C-2: search two steps
C-S: Simplified minimum-mapping
C-CAS: compare-and-swap operation for update

C-1: search one step
C-Syn: synchronization before updates
FastSV: state-of-the-art tree-based method

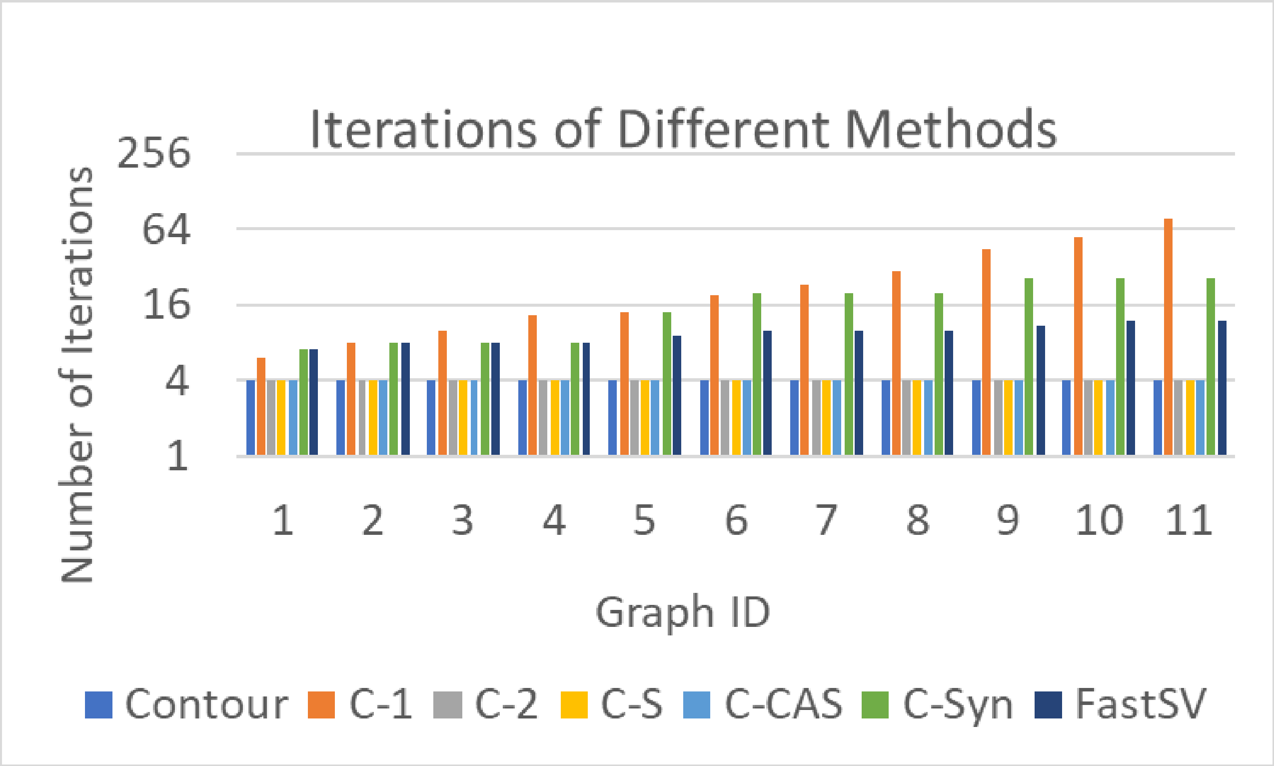
NJIT New Jersey Institute of Technology

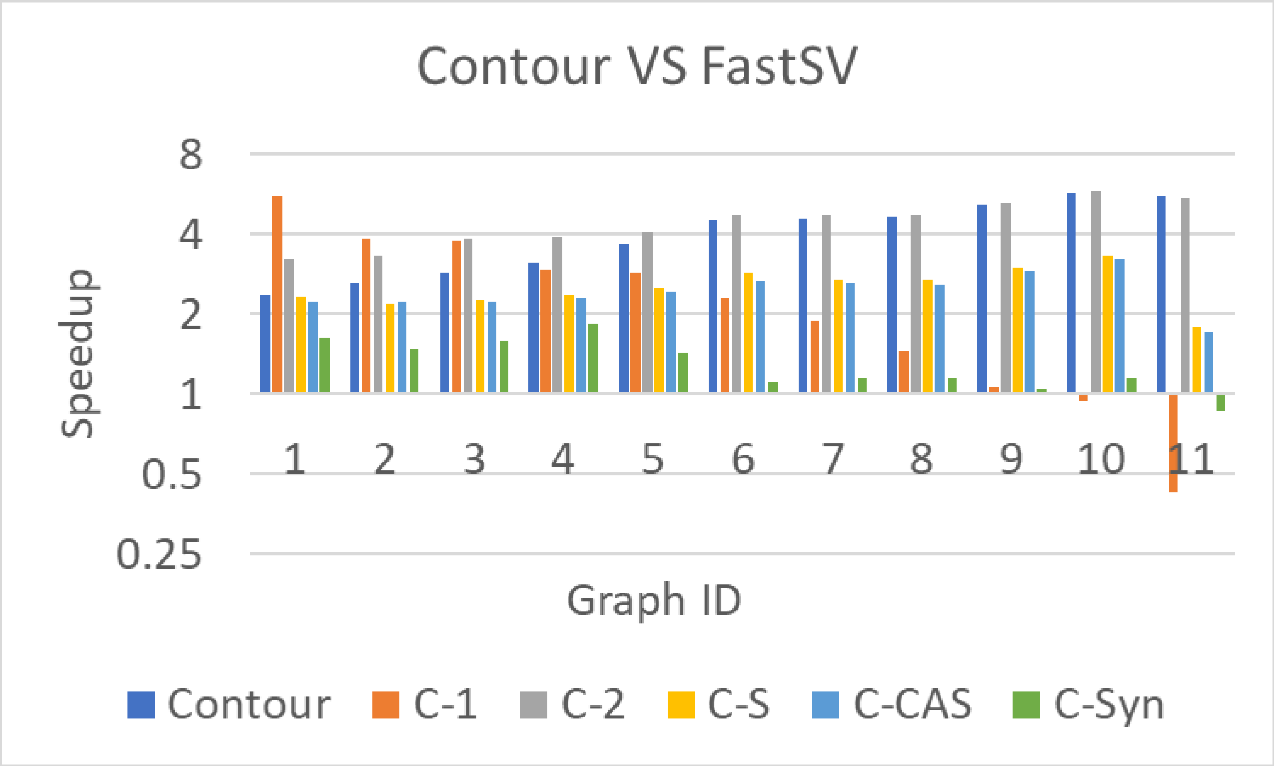
Experimental Results (performance)



New Jersey Institute of Technology

Multi-Locale Results





New Jersey Institute of Technology

Other Implementations (delaunay_n20)

- Chapel
  - FastSV 0.290952s
  - Contour 0.038154s
- LAGraph implementation (C+GraphBLAS)
  - FastSV needs 0.0226186s
  - Contour just like FastSV
- Graph Based Benchmark Suite (GBBS)
  - Optimized SV: 0.022097s
  - Contour: 0.012859s

Conclusion

- Contour algorithm
  - simple, easy to parallelize and high-performance for connected components
  - Converge in O(log(n)) iterations
- How Chapel can affect the performance
  - Compared with High-Level LAGraph(GraphBLAS) package (C) (Vector, Matrix)
    - LAGraph/GraphBLAS cannot exploit fine and flexible parallelism like Chapel
    - Chapel has a performance overhead
  - Compared with the Graph-Based Benchmark Suite (GBBS) package (C++)
    - GBBS cannot support distributed parallelism like Chapel
    - Chapel's overhead is relatively high

Acknowledgement

We appreciate the help from the Chapel and Arkouda community when we integrated the algorithms into Arkouda. This research was funded in part by NSF grant number CCF-2109988.

NJIT
New Jersey Institute of Technology

Thank You!

Q&A