HyPC-Map: Hybrid Parallel Community Discovery using Infomap

Md Abdul Motaleb Faysal, Shaikh M Arifuzzaman (University of New Orleans),
Cy Chan, Maximilian Bremer, Doru Thom Popovici, John Shalf (Berkeley Lab)

THE UNIVERSITY of NEW ORLEANS

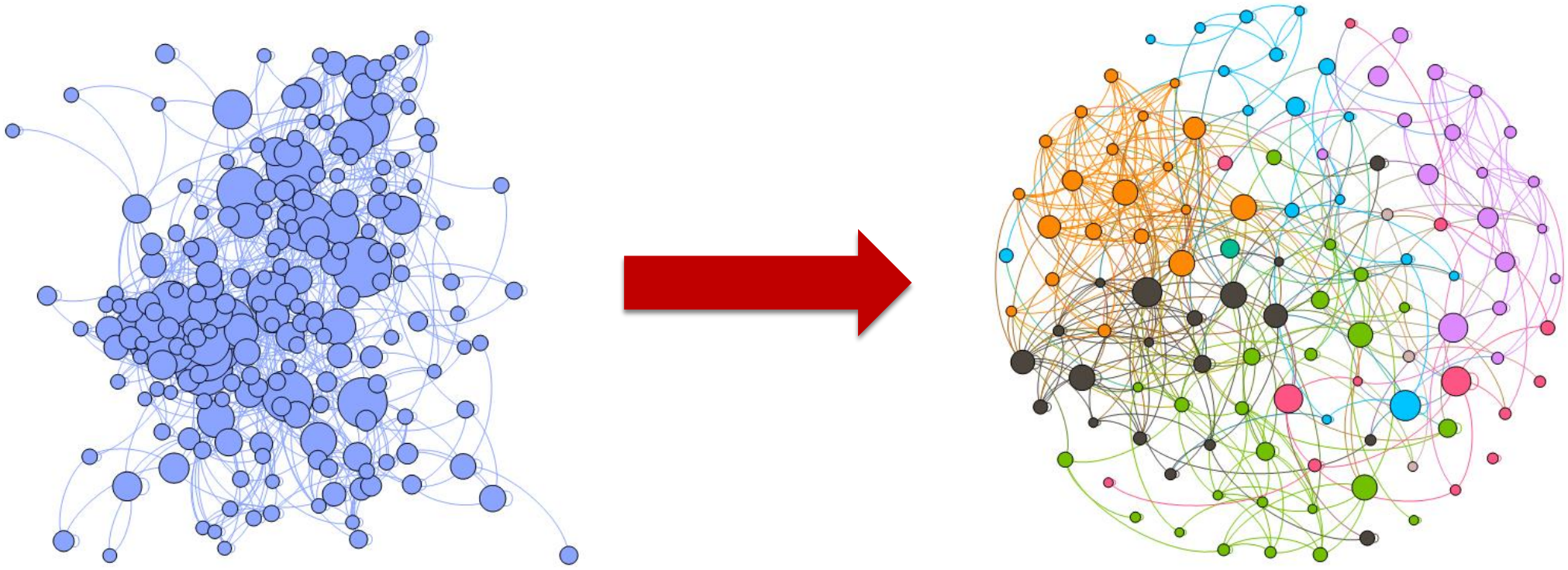
BERKELEY LAB

WORKFORCE DEVELOPMENT & EDUCATION

Community Detection

- There is no bold definition of the term *community* in the context of network analysis
- Some networks (graph) divide naturally into *groups* of vertices with dense connections internally and sparse connections across the groups
- The task of finding that group structure is generally known as community detection

Community Detection



(a) (b)

(a) A protein-protein interaction network in yeast. The vertices (blue circles) represent different proteins and the arcs (blue lines) represent interaction between those proteins. (b) Grouping of proteins by community detection based on their functional simil arities. Proteins within the same group have the same color and share similar biological functions. The visualizations are generated u sing Gephi

Applications of Community Detection

- Classifying groups in social and business networks
- Grouping similar proteins in a protein-protein interaction networks
- Detecting anomalous behavior in cyber-security domain
- Finding critical point/entity in rumor propagation or infectious disease spreading, etc.

A Few Community Detection Approaches

The problem of focus

Community Detection Methods	**Drawbacks**
Spectral methods	- Not computationally efficient - Not reliable for sparse networks
Information-Theoretic Approach	**- High time complexity**
Modularity Optimization Approach	- Resolution limit problem
Stochastic Block Modeling Approach	- High time complexity

Information-Theoretic Approach

- Greater accuracy found by independent studies
- Topped in LFR benchmark by Lancichinetti et al. and named as ***Infomap***

- Limitation
  - Approximation based strategy
  - Highly sequential in nature

The Infomap

- *Infomap* uses a standard data compression technique on a dynamic process (*random walk*).

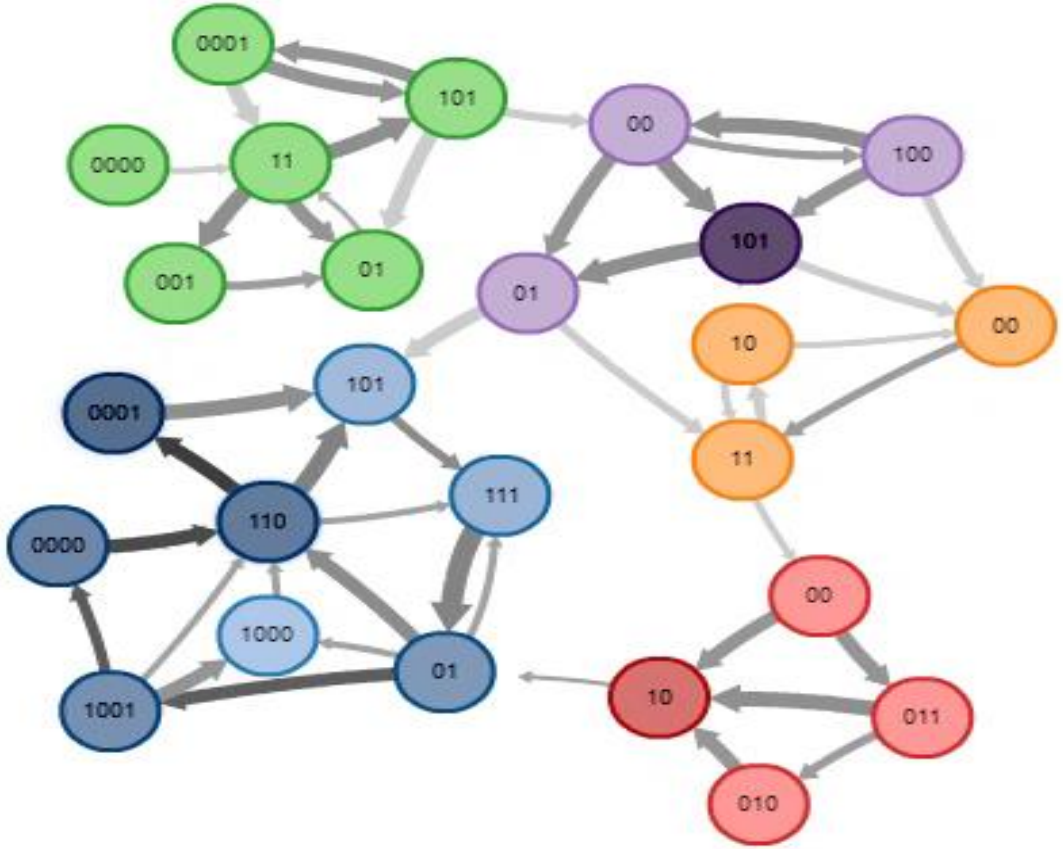
- It exploits the duality between compressing a data set and extracting significant patterns or structures in that data set

- A branch of statistics named Minimum Description Length (MDL) discusses the duality

The Infomap

- The target is to find an optimal compression code.
- Optimizing the code means detection of regularities or patterns corresponding to the structural features of that network.
- Theoretical limit of optimal compression based on the regularity of the structure is discussed in Shanon's Minimum Entropy theorem.
- The optimization function is called the mapequation.

Dynamic Process Reveals Community Structures



A dynamic visualization of how the infinite traversal path of a random walk can reveal the community structure in a network

Java applet ref: https://www.mapequation.org/apps/MapDemo.html#applet

The *MapEquation*

$$L(M) = q_{\curvearrowright} H(Q) + \sum_{i=1}^{m} p^i{}_{\circlearrowright} H(Pi)$$

$q_{\curvearrowright}$ --- Sum of exit probability of the random walk for each community

H(Q) --- Average code length for the movement between community (inter-module entropy)

$p^i{}_{\circlearrowright}$ --- Stay probability of the random walk within a community i

$H(P^i)$ --- Average code length of the random walk within the community (intra-module entropy)

- Red term describes inter-modular movements
- Blue term describes movement within modules

HyPC-Map, our contributions toward parallelization of Infomap

- We combined both distributed-memory and shared-memory parallelism to gain ***25-fold*** speedup than sequential algorithm
- We incorporated clever heuristics to tackle the inherent sequential nature and low scalability of the algorithm
- We performed extensive benchmarking and analyzed memory subsystems to use cache-optimized data structures resulting in efficient compute kernels
- We achieve better speedup than state-of-the-art techniques without sacrificing the solution quality

Parallel Algorithm Challenges

- Vertex bouncing problem: When two vertices having strong affinity are distributed across two different processes, it causes a non-converging oscillation of the vertices

- Inconsistent update ordering: Being a synchronous parallel approach, different synchronization orders in different MPI processes may contribute to inconsistent modular state for a single vertex

- Inactive vertices: In general, most vertices find their communities in the earlier iterations. Computing community for all vertices in every iterations until convergences incur redundant computational cost

Solution Heuristics for the Challenges

- Solution to Vertex Bouncing Problem: Maintain numeric ordering during assignment of community id and allow only one way assignment (i.e., from lower id community to higher id community)

- Solution to Inconsistent Update Ordering: The decision to community assignment for a specific vertex is taken and broadcast by the owner process of that vertex.

- Solution to Inactive Vertex Problem: A list of active vertices is maintained where a vertex that changed its community in current iteration will be considered active along with neighboring vertices for next iteration

Experimental Datasets

Network	# Vertices	# Edges	Description
Amazon	334863	925872	Amazon co-purchased network
DBLP	317080	1049866	Computer science bibliographical network
Youtube	1134890	2987624	Youtube social network
wiki-topcats	1791489	28511807	Hyperlinks network from Wikipedia
soc-Pokec	1632803	30622564	Pokec online social network
LiveJournal	3997962	34681189	LiveJournal online social network
Orkut	3072441	117185083	Orkut online social network

The network datasets are collected from SNAP, each network exhibit good community structure

Experiment Specifications

- **Implementation platform**: Linux
- **Language**: C++
- **Frameworks**: Metis , MPI, and OpenMP

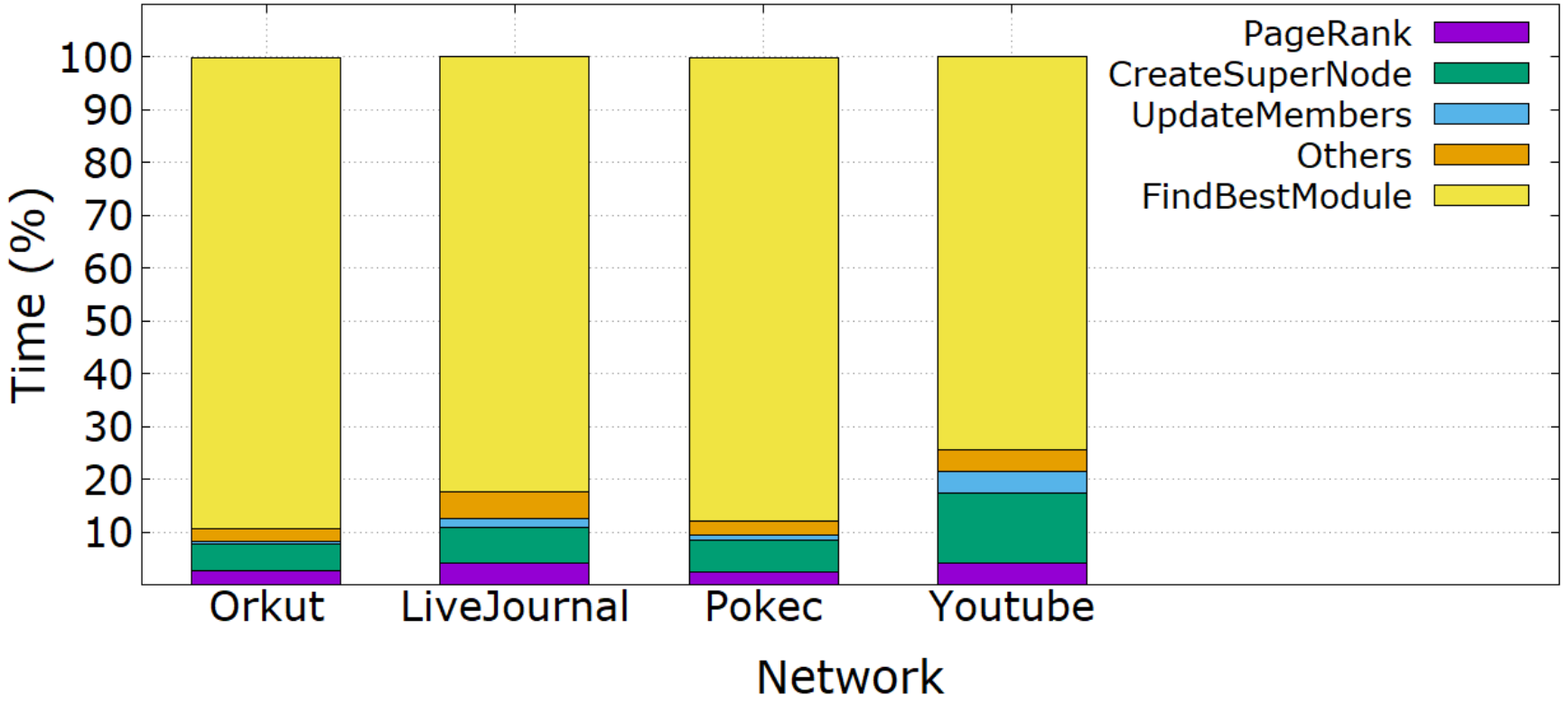
Compute Node Specifications

- LONI, Intel Xeon E5-2680v2 processor with 2.8 GHz
  - Sockets 2, 10 cores per socket, 64 GB memory
  - 56 Gb/sec InfiniBand

- NERSC Cori Haswell, Intel Xeon E5-2698v3 processor with 2.3 GHz
  - Sockets 2, 16 cores per socket, 128 GB memory
  - 45 TB/sec

- Local Compute Server, Intel(R) Xeon(R) CPU E5-2683v4 with 2.1 GHz
  - Sockets 2, 16 cores per socket, 512 GB memory

Kernel Breakdown and Optimization



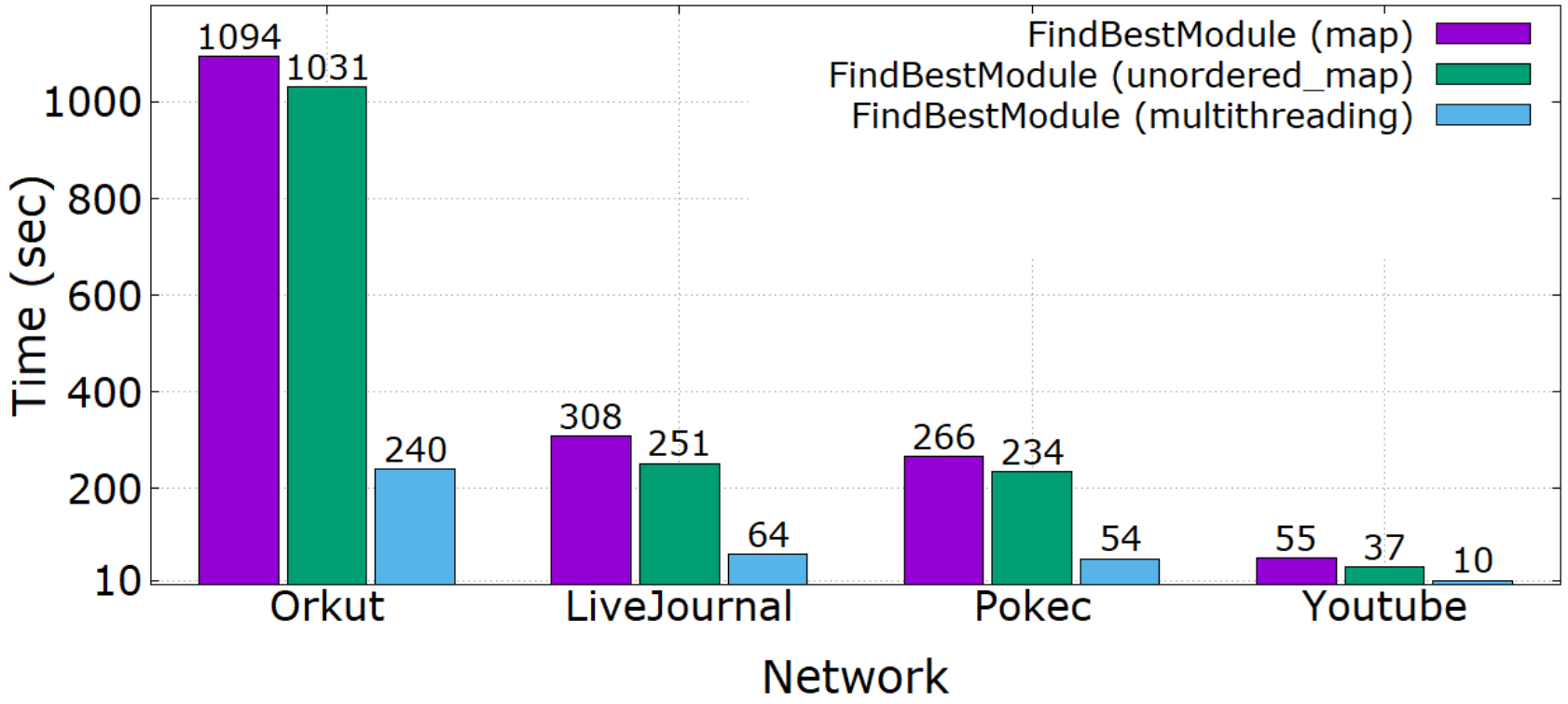
FindBestModule is responsible for the major portion of the run time taking as much as 90% of the overall computation time (Orkut network). Optimization of this kernel means optimizing the whole algorithm significantly

Micro-benchmarking reveals benefit of cache-friendly data structure

Number of entries	Insertion map (µs)	Insertion unordered_ map (µs)	Read map (µs)	Read unordered_ map (µs)
2048	1904	1284	70	58
4096	3991	2586	110	123
8192	8499	5139	230	239
16384	16927	9764	462	465
32768	34916	19197	887	902
65536	75827	37914	1689	1810
131072	166398	76855	3936	3608

Insertion time for array based **unordered_map** is nearly half as the insertion time for RB-tree based STL **map**

Using OpenMP parallelism for kernel optimization



Switching to **unordered_map** indeed displays benefit over **map.** However, implementing multithreaded parallelism for the **FindBestModule** kernel has delivered the most significant performance upgrade

Performance Analysis

- Quantitative Analysis Metrics
  - →Speedup
  - →Strong Scaling
- Quality Analysis Metrics
  - →Modularity
  - →Conductance
  - →Normalized Mutual Information

Performance comparison with state-of-the-arts

Work Name	Type	Strength	Weakness
Infomap	Sequential	High accuracy	Computationally expensive
RelaxMap	Shared-memory parallelism	High accuracy	Scalability limited to single node
Gossipmap	Asynchronous distributed-memory parallelism	Asynchronous	Scalability up to 128 parallel units
Distributed Infomap [20]	Synchronous distributed-memory parallelism	Scales to 512 processors	Speedup up to $\sim$ 5X
Distributed Infomap [29]	Synchronous distributed-memory parallelism	Scales to $\sim$ 4k processors	Speedup up to $\sim$ 6X
HyPC-Map	Synchronous hybrid memory parallelism	High accuracy & speedup	

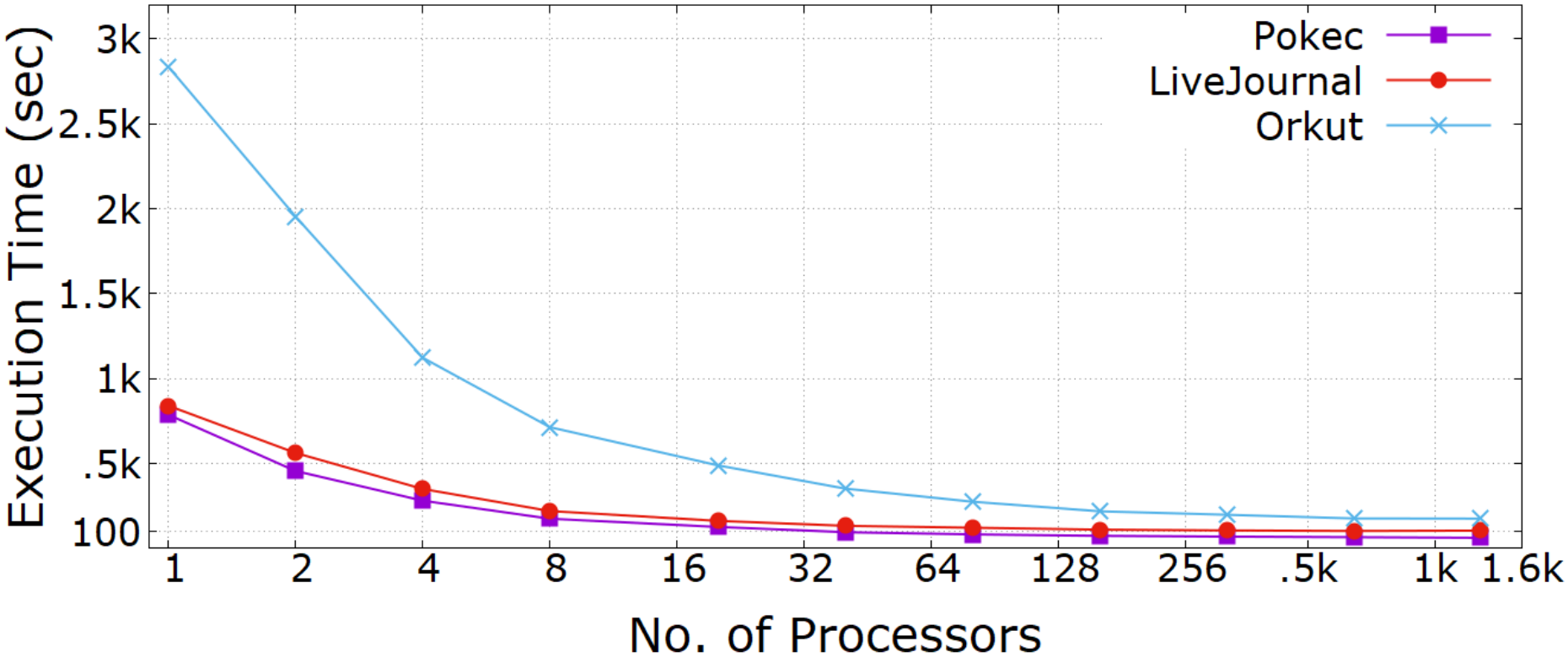
While the state-of-the arts display speedup less than one order of magnitude, HyPC-Map gains 25-fold speedup

Speedup Comparison

Network	Speedup (vs sequential HyPC-Map)	Speedup (vs original Infomap)
Amazon	2.78	8.79
DBLP	3.66	7.00
Youtube	4.58	9.43
LiveJournal	8.19	25.11
Wiki-topcats	10.52	16.06
Soc-pokec	12.52	20.67
Orkut	16.16	21.42

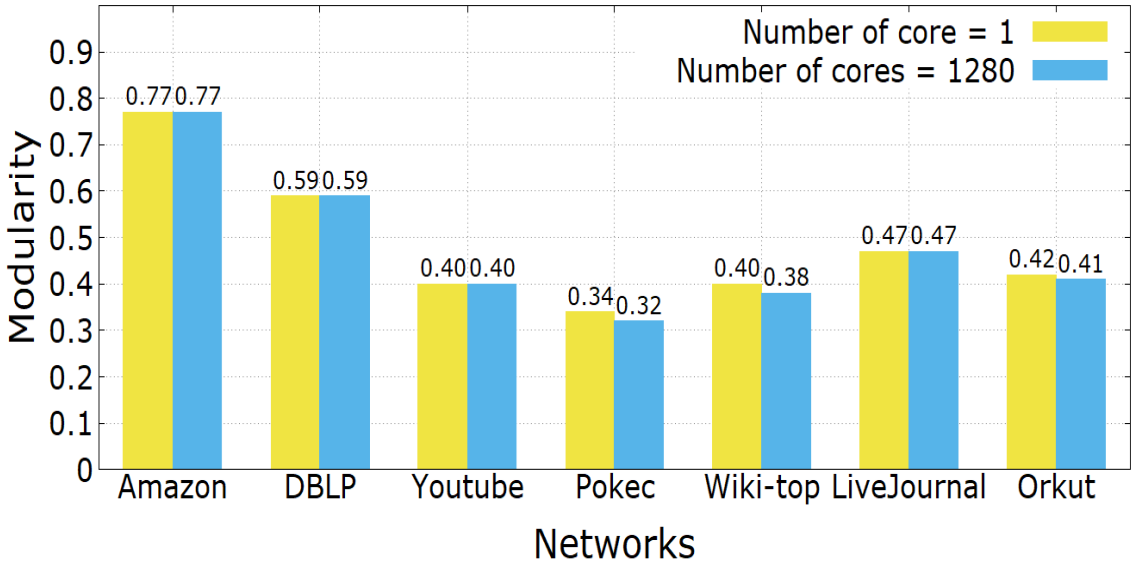
Speedup comparison with original Infomap developed by Rosvall (column 3), and with sequential HyPC-Map (column 2) on various social and information networks

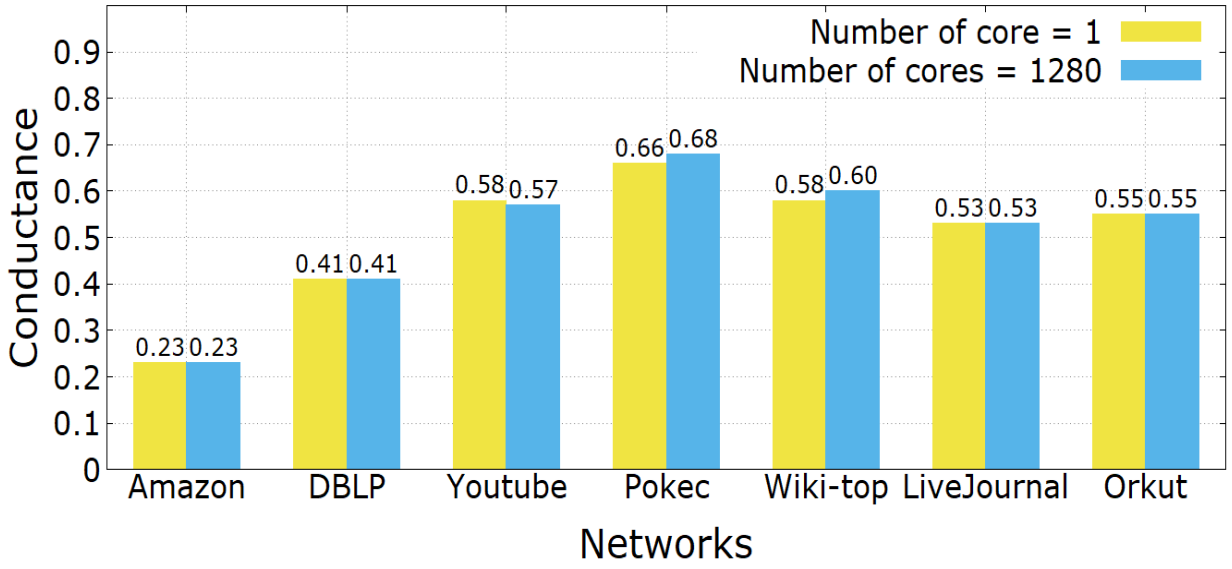
Scalability Analysis



Run time reduces from 2835 seconds to 175 seconds for the Orkut network

Scalability of the Discovered Community





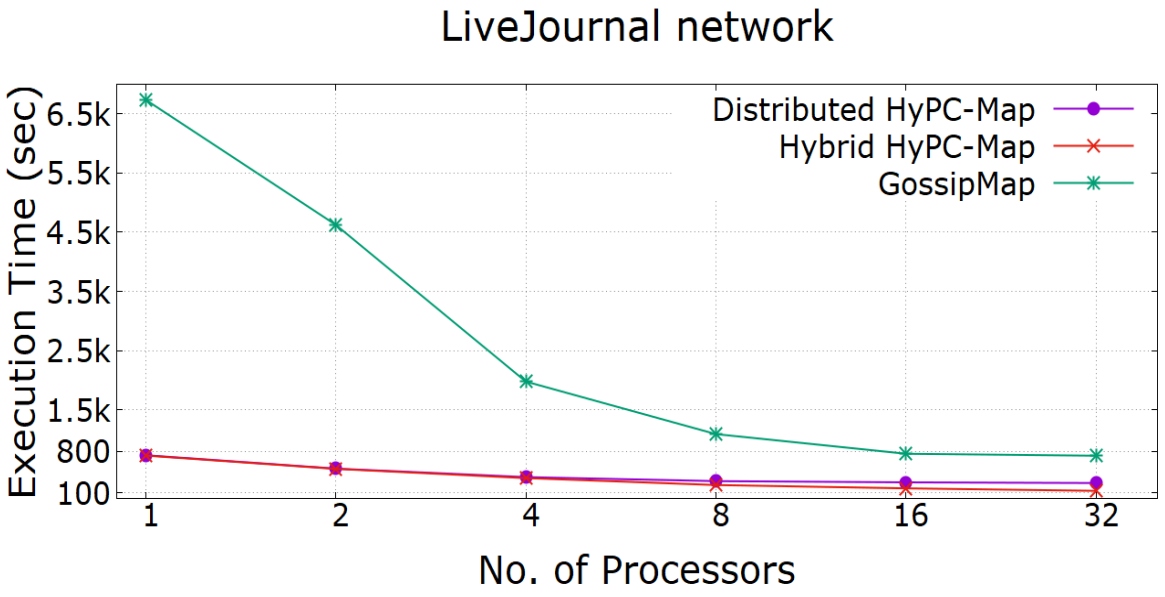
The quality of the discovered community in terms of the quality metrics **Modularity** and **Conductance**. In most of the experimental datasets, the impact in the quality metrics range from 0 – 2%

Testing scalability of the quality by Normalized Mutual Information (NMI)

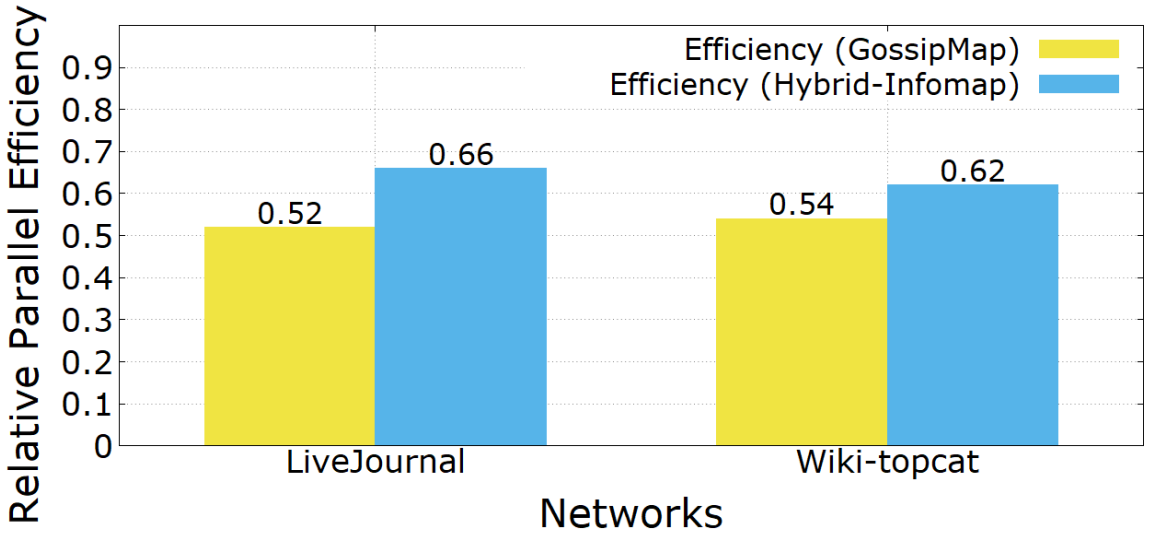
Network	# Vertices	# Edges	NMI							
			1	20	40	80	160	320	640	1280
SG_50000	50000	1011755	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90
SG_500000	500000	10160671	0.85	0.85	0.86	0.86	0.87	0.87	0.87	0.88
SG_2000000	2000000	40670978	0.84	0.84	0.84	0.85	0.86	0.85	0.87	0.87

NMI is a measure of the scalability of the solution that requires ground truth community. Therefore, static synthetic network datasets with ground-truth partition from MIT graphchallenge are used. Apart from some noise, the NMI results show scalability for different number of processing cores

Comparison with Gossipmap



(a)



(b)

(a) Run time comparison with Gossipmap where the compute kernels work in purely distributed mode (purple) and in hybrid mode (red). The benefit of using an efficient compute kernel is visible from the plot. (b) Higher relative parallel efficiency exhibited by HyPC-Map

Comparison with HipMCL

Network	HyPC-Map (sec) 1 Compute Node	HipMCL (sec) 1 Compute Node	HipMCL (sec) 4 Compute Nodes	HipMCL (sec) 16 Compute Nodes
Amazon	3.50	85.18	50.61	20.24
DBLP	3.90	278.64	166.42	57.35
Youtube	21.14	MLE	9251.05	2545.89
soc-Pokec	82.05	MLE	37014.52	10792.05
Orkut	235.0	MLE	MLE	35715.63

Execution performance comparison between HipMCL and HyPC-Map. Here, MLE: Memory Limit Exceeded. HipMCL does not work well in clustering scale-free networks that exhibit power-law degree distribution

Q & A