# Parallel Graph Computations: Foundation and Practice

Wenfei Fan

University of Edinburgh

## Backgrounds

What is the right parallel graph computation model?

- **GRAPE**: automatically parallelizing graph computation
- **AAP**: adaptive asynchronous parallel

How to partition the graph under various circumstances?

- Dynamically re-partitioning a graph when adding or removing processors
- Partitioning the graph based on application scenario
- Incrementalizing existing graph partitioners

How to efficiently carry out local computation?

- Graph contraction: making big graphs small
- Incrementalize existing graph algorithms

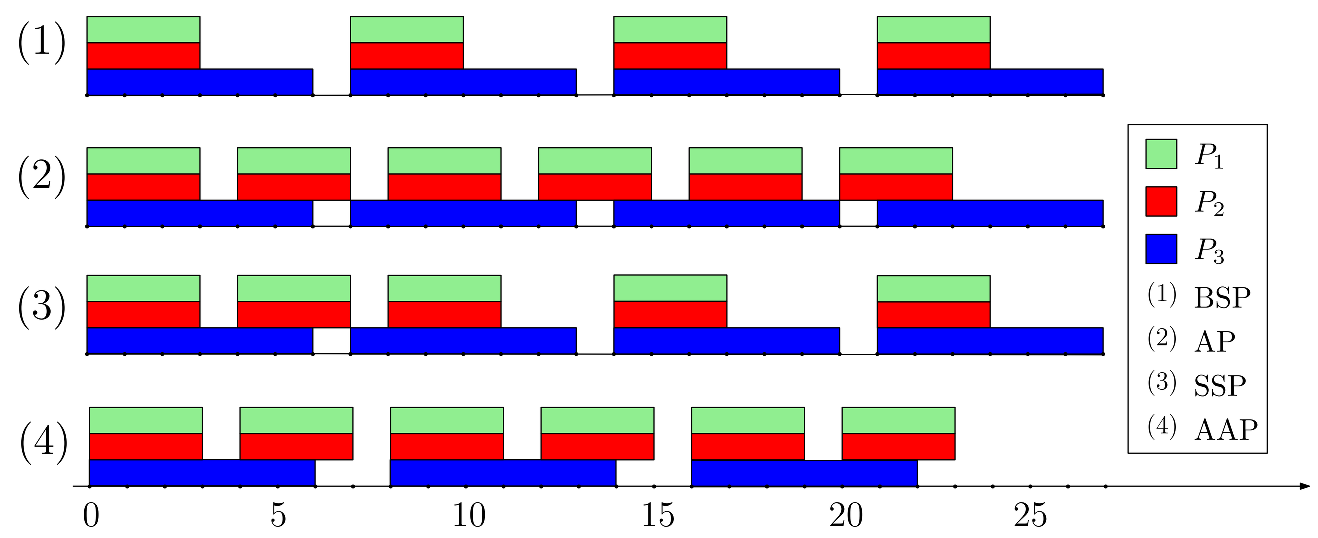
## Parallel graph computation model

**GRAPE** programming model: plug and play

- Partial evaluation **PEval**
- Incremental evaluation **IncEval**
- Assemble partial results

**AAP**: each worker maintains parameters and dynamically adjusts:

- its progress relative to other workers
- the staleness of its accumulated changes




Unique properties

- Dynamic model switch: reduce the total rounds
- Programming simplicity
- Ready to *optimally* simulate other platforms:
  - **MapReduce**, **BSP**,**AP**, **SSP** and **PRAM**
- Convergence guarantee: Church-Rosser

Outperform the state-of-the-art systems

- Outperforming **BSP**, **AP**,**SSP** by 4.3X, 14.7X and 4.7X on average
- 30X times faster than Petuum on collaborative filtering

Industrialized as GraphScope in Alibaba:
Open source: https://github.com/alibaba/GraphScope

## Partitioning

Dynamic scaling

- Updating the partition $\Pi(n+k)$ from $\Pi(n)$ when add $k$ servers with minimal migration cost, subject to balance factor $b$ and replication factor $f$
- Tri-criteria optimization problem: **NP**-complete
- Approximation algorithms: extending consistent hashing on graphs

Application driven partitioning

- Finding partition $\Pi(n)$ to minimize the cost $\mathcal{C}(\mathcal{A})$ of algorithm $\mathcal{A}$
- Learned cost model: unifying computation and communication
- Hybrid partitioners: combining edge-cut and vertex-cut

Incrementalization of partition algorithms

- Tri-criteria optimization problem: **NP**-complete
- A generic approach based on fixpoint model

Partition transparency for algorithm $\mathcal{A}$

- Conditions: monotonic + robustness under vertex cut
- Example: **SSSP**, **WCC**, **PageRank**, etc.

$\mathcal{A}$ works for any partition without requiring any change to $\mathcal{A}$

## Graph contraction

Graph contraction: making big graphs small

- Genericness: support different classes of queries at the same time
- Losslessness: compute exact answer

Contraction scheme $\langle f_C, \mathcal{S}, f_D \rangle$

- Contraction function $f_C$: subgraphs to supernodes
- Synopsis synopsis functions $\mathcal{S}$: annotate supernodes
- Decontraction function $f_D$: restore supernodes to subgraphs

Algorithms:

- Contraction following deterministic order
- Bounded Incremental contraction

## Incrementalization

Incremental algorithms: $\mathcal{A}(G \oplus \Delta G) = \mathcal{A}(G) \oplus \mathcal{A}_\Delta(G, \Delta G)$

Incrementalize batch algorithms

- Generating an incremental algorithm $\mathcal{A}_\Delta$ from the batch algorithm $\mathcal{A}$
- Reusing the original logic and data structures
- Providing performance guarantee

Fixpoint model for batch algorithm $\mathcal{A}$:

$$(D^{t+1}_{\mathcal{A}},\ H^{t+1}_{\mathcal{A}}) = f_{\mathcal{A}}(D^{t}_{\mathcal{A}},\ Q,\ G,\ H^{t}_{\mathcal{A}})$$

- Status $D_{\mathcal{A}}$, status varibale $H_{\mathcal{A}}$, and step function $f_{\mathcal{A}}$

Incrementalization

- Initial scope function $h$: initializing status $(D^0_{\mathcal{A}}, H^0_{\mathcal{A}})$ from $\Delta G$ and previous result
- Reusing the same $D_{\mathcal{A}}$, $H_{\mathcal{A}}$ and $f_{\mathcal{A}}$
- Reiterating from $(D^0_{\mathcal{A}}, H^0_{\mathcal{A}})$ until convergence

Incrementalizability: every fixpoint algorithm $\mathcal{A}$ is incrementalizable
Relative boundedness: correct and bounded $h$ + Church-rosser $\mathcal{A}$

## Heterogeneous application

Heterogeneous entity resolution: match entities across relations and graphs

- Convert relations into canonical graphs $G_{\mathcal{D}}$
- Parallelize parametric simulation across $G$ and $G_{\mathcal{D}}$
- Integrate simulation with ML models to assess semantic closeness

## Delivered

13 top-tier publications :

- 2022: VLDB [1], TKDE [2]
- Past: 2021 [3, 4, 5, 6, 7], 2020 [8, 9, 10, 11], 2019 [12], and 2018 [13]

To develop a package of solutions for parallel graph computations

## Publications

[1] Wenfei Fan. Big graphs: Challenges and opportunities. In *PVLDB*, 2022.

[2] Wenfei Fan, Liang Geng, Ruochun Jin, Ping Lu, Resul Tugay, and Wenyuan Yu. Linking entities across relations and graphs. In *ICDE*, pages 634–647, 2022.

[3] Wenfei Fan, Yuanhao Li, Muyang Liu, and Can Lu. Making graphs compact by lossless contraction. In *SIGMOD*, 2021.

[4] Wenfei Fan, Chao Tian, Qiang Yin, Ruiqi Xu, Wenyuan Yu, and Jingren Zhou. Incrementalizing graph algorithms. In *SIGMOD*, 2021.

[5] Wenfei Fan, Tao He, Longbin Lai, Xue Li, Yong Li, Zhao Li, Zhengping Qian, Chao Tian, Lei Wang, Jingbo Xu, et al. Graphscope: a unified engine for big graph processing. *Proceedings of the VLDB Endowment*, 14(12):2879–2892, 2021.

[6] Jingbo Xu, Zhanning Bai, Wenfei Fan, Longbin Lai, Xue Li, Zhao Li, Zhengping Qian, Lei Wang, Yanyan Wang, Wenyuan Yu, et al. Graphscope: a one-stop large graph processing system. *Proceedings of the VLDB Endowment*, 14(12):2703–2706, 2021.

[7] Wenfei Fan, Muyang Liu, Ping Lu, and Qiang Yin. Graph algorithms with partition transparency. *TKDE*, 2021.

[8] Wenfei Fan, Ping Lu, Jingbo Xu, Wenyuan Yu, Qiang Yin, Xiaojian Luo, Jingren Zhou, and Ruochun Jin. Adaptive asynchronous parallelization of graph algorithms. *TODS (Invited)*, 2020.

[9] Grace Fan, Wenfei Fan, Yuanhao Li, Ping Lu, Chao Tian, and Jingren Zhou. Extending graph patterns with conditions. In *SIGMOD*, 2020.

[10] Wenfei Fan, Ruochun Jin, Muyang Liu, Ping Lu, Xiaojian Luo, Ruiqi Xu, Qiang Yin, Wenyuan Yu, and Jingren Zhou. Application driven graph partitioning. In *SIGMOD*, 2020.

[11] Wenfei Fan, Muyang Liu, Chao Tian, Ruiqi Xu, and Jingren Zhou. Incrementalization of graph partitioning algorithms. *PVLDB*, 13(8):1261–1274, 2020.

[12] Wenfei Fan, Chunming Hu, Muyang Liu, Ping Lu, Qiang Yin, and Jingren Zhou. Dynamic scaling for parallel graph computations. *PVLDB*, 12(8):877–890, 2019.

[13] Wenfei Fan, Ping Lu, Xiaojian Luo, Jingbo Xu, Qiang Yin, Wenyuan Yu, and Ruiqi Xu. Adaptive asynchronous parallelization of graph algorithms. In *SIGMOD*, pages 1141–1156, 2018.