CIKM2015

19–23 October 2015 Melbourne Australia

HDRF: Stream-Based Partitioning for Power-Law Graphs.

Fabio Petroni, L. Querzoni, K. Daudjee, S. Kamali, G. Iacoboni

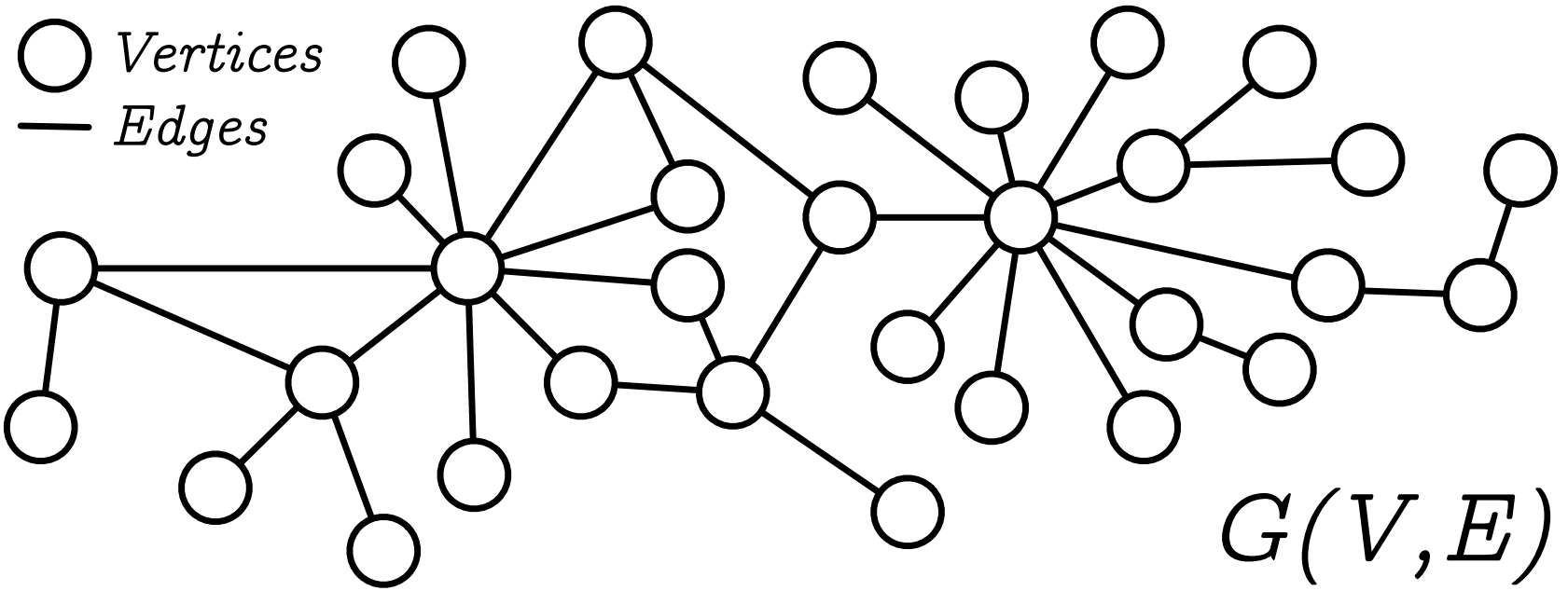
SAPIENZA
UNIVERSITÀ DI ROMA

UNIVERSITY OF WATERLOO

travel grant from
SIGIR
SPECIAL INTEREST GROUP ON
INFORMATION RETRIEVAL

Graphs

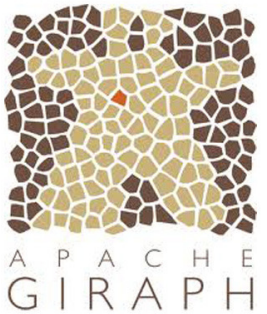
- large amount of real data can be represented as a graph.



- the problem of optimally partitioning a graph while maintaining load balance is important in several contexts.

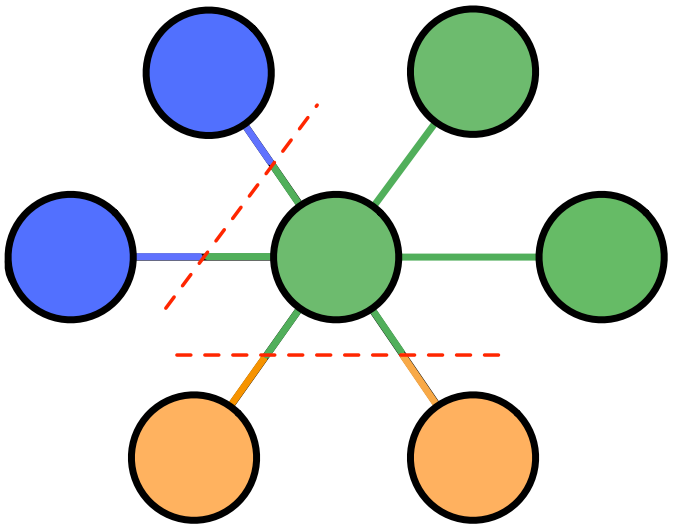
Distributed Graph Computing Frameworks

- support efficient computation on really large graphs.
  - graph analytics; data mining and machine learning tasks.

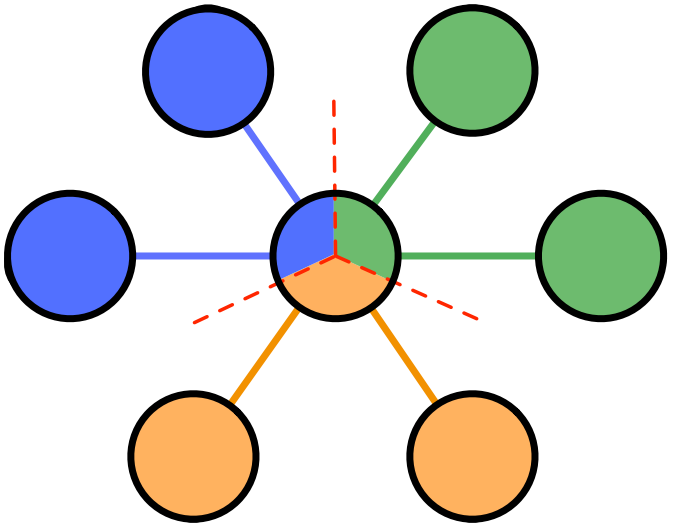


- input data partitioning can have a significant impact on the performance of the graph computation.
  - affects network usage, memory occupation, synchronization.

Balanced Graph Partitioning



edge-cut



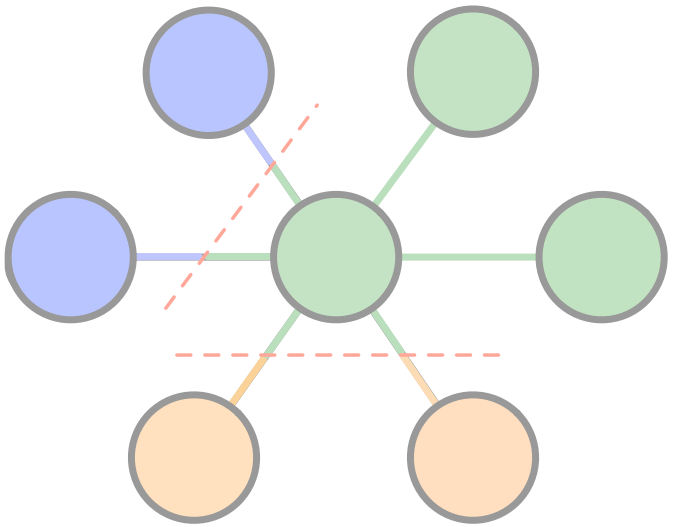
vertex-cut

- partition G into smaller components of (ideally) equal size

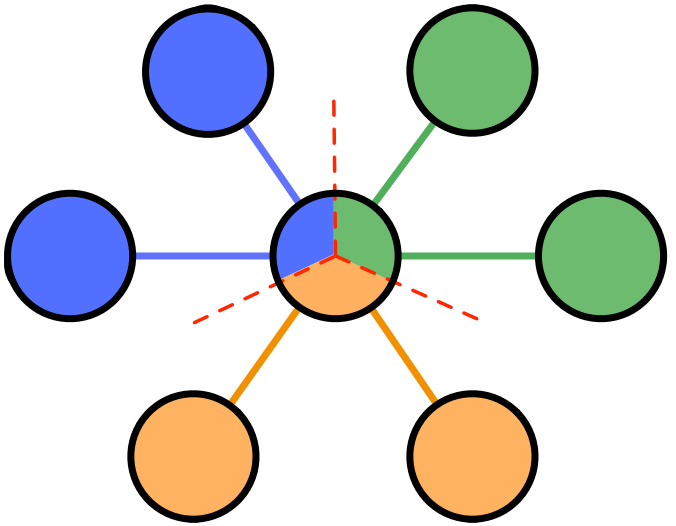
edge-cut	***vertex-cut***
vertex disjoint partitions	*edge disjoint partitions*

- a vertex can be cut in multiple ways and span several partitions while a cut edge connects only two partitions.

Balanced Graph Partitioning



edge-cut



vertex-cut

- partition G into smaller components of (ideally) equal size

edge-cut	***vertex-cut***
vertex disjoint partitions	*edge disjoint partitions*

- a vertex can be cut in multiple ways and span several partitions while a cut edge connects only two partitions.

Vertex-Cut

computation steps are associated with edges

v-cut perform better on

power law graphs

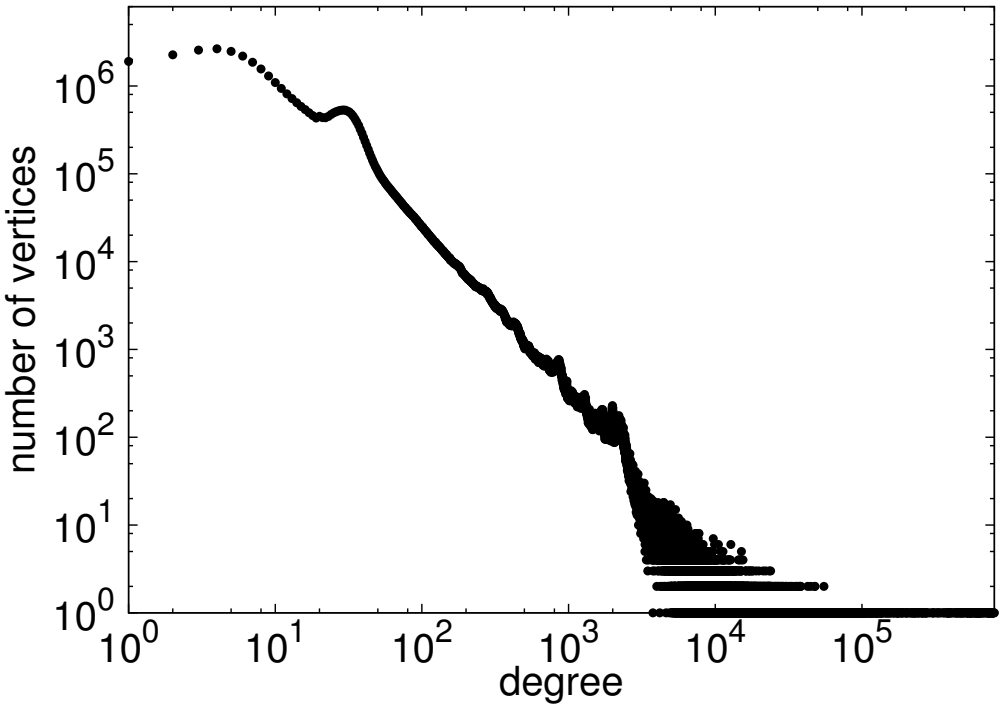
(Gonzalez et al., 2012)

create less storage and network overhead

- modern distributed graph computing frameworks use vertex-cut.

Power-Law Graphs

- characteristic of real graphs: power-law degree distribution.
  - most vertices have few connections while a few have many.



crawl
Twitter
connection
2010

log-log scale

- the probability that a vertex has degree d is $P(d) \propto d^{-\alpha}$.
- α controls the "skewness" of the degree distribution.

Balanced Vertex-Cut Graph Partitioning

- $v \in V$ vertex; $e \in E$ edge; $p \in P$ partition.
- $A(v)$ set of partitions where vertex v is replicated.
- $\sigma \geq 1$ tolerance to load imbalance.
- the size $|p|$ of partition p is its edge cardinality.

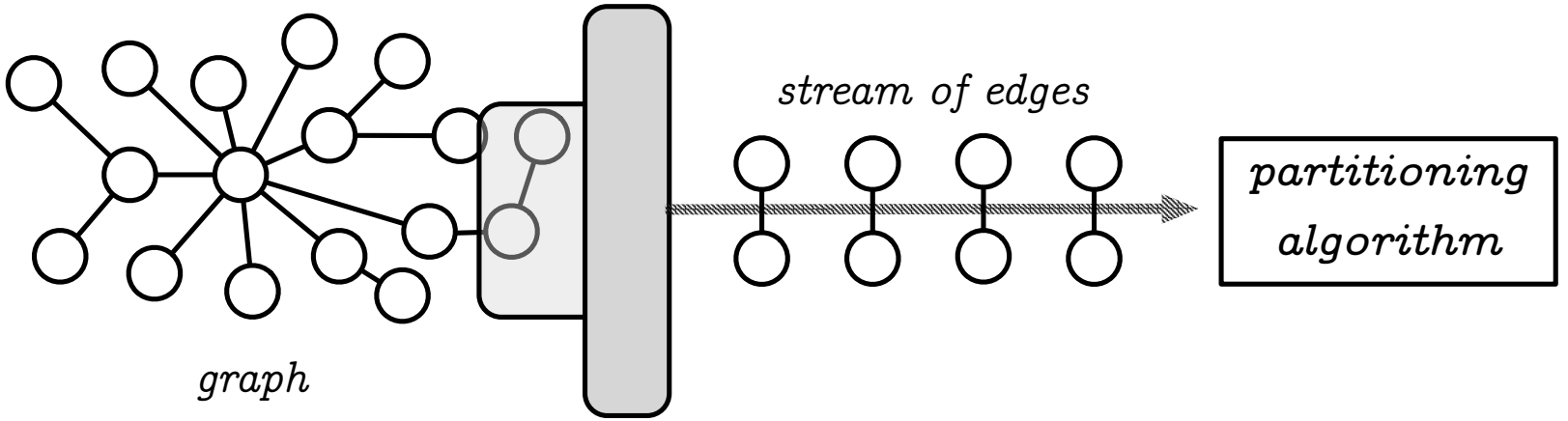
minimize replicas	*balance the load*
reduce (1) bandwidth, (2) memory usage and (3) synchronization	*efficient usage of available computing resources*

$$\min \boxed{\frac{1}{|V|} \sum_{v \in V} |A(v)|} \quad s.t. \quad \max_{p \in P} |p| < \sigma \frac{|E|}{|P|}$$

- the objective function is the **replication factor** (RF).
  - average number of replicas per vertex.

Streaming Setting

- input data is a list of edges, consumed in *streaming fashion*, requiring only a **single pass**.



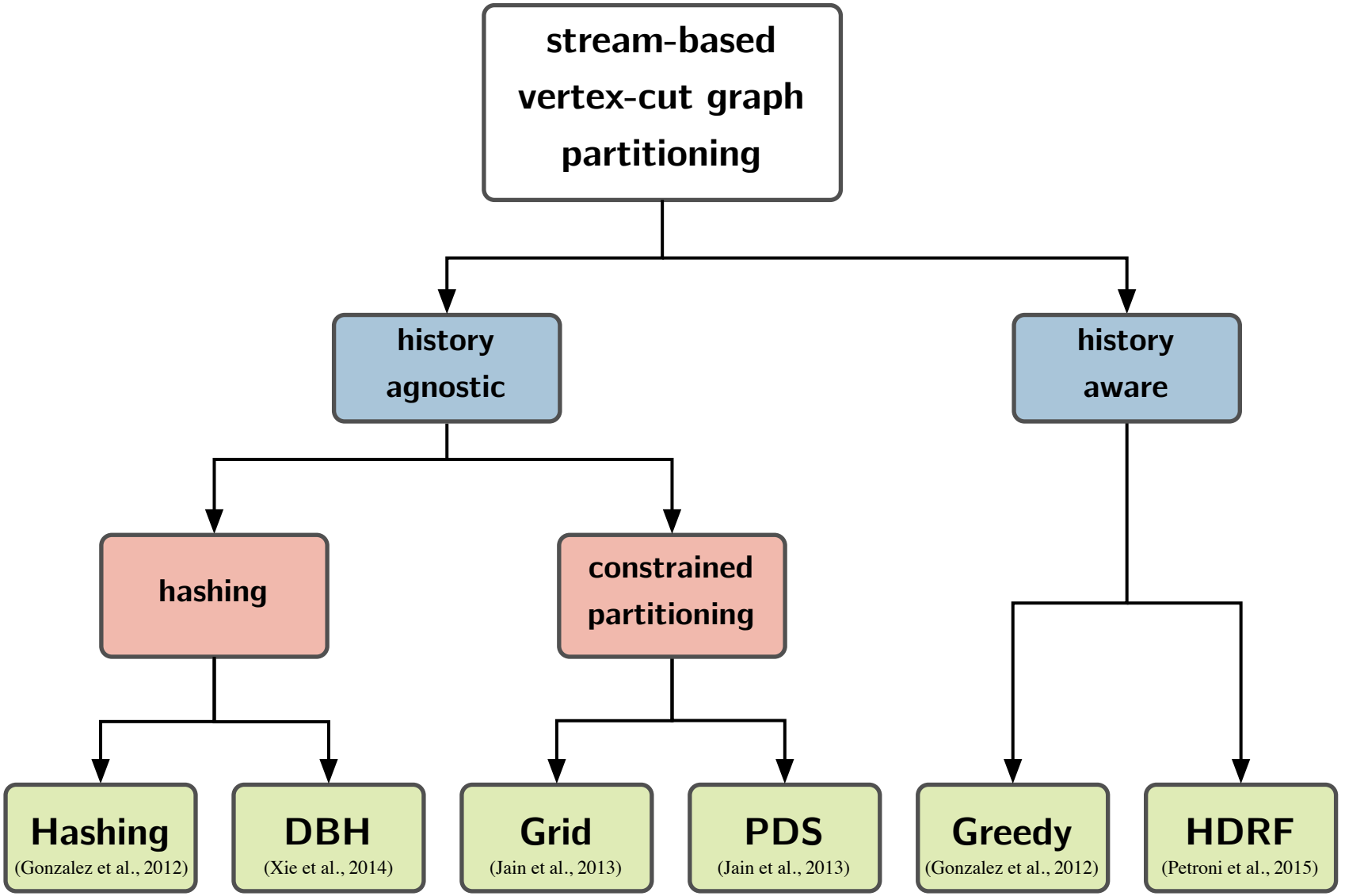
✓ handle graphs that don't fit in the main memory.

✓ impose minimum overhead in time.

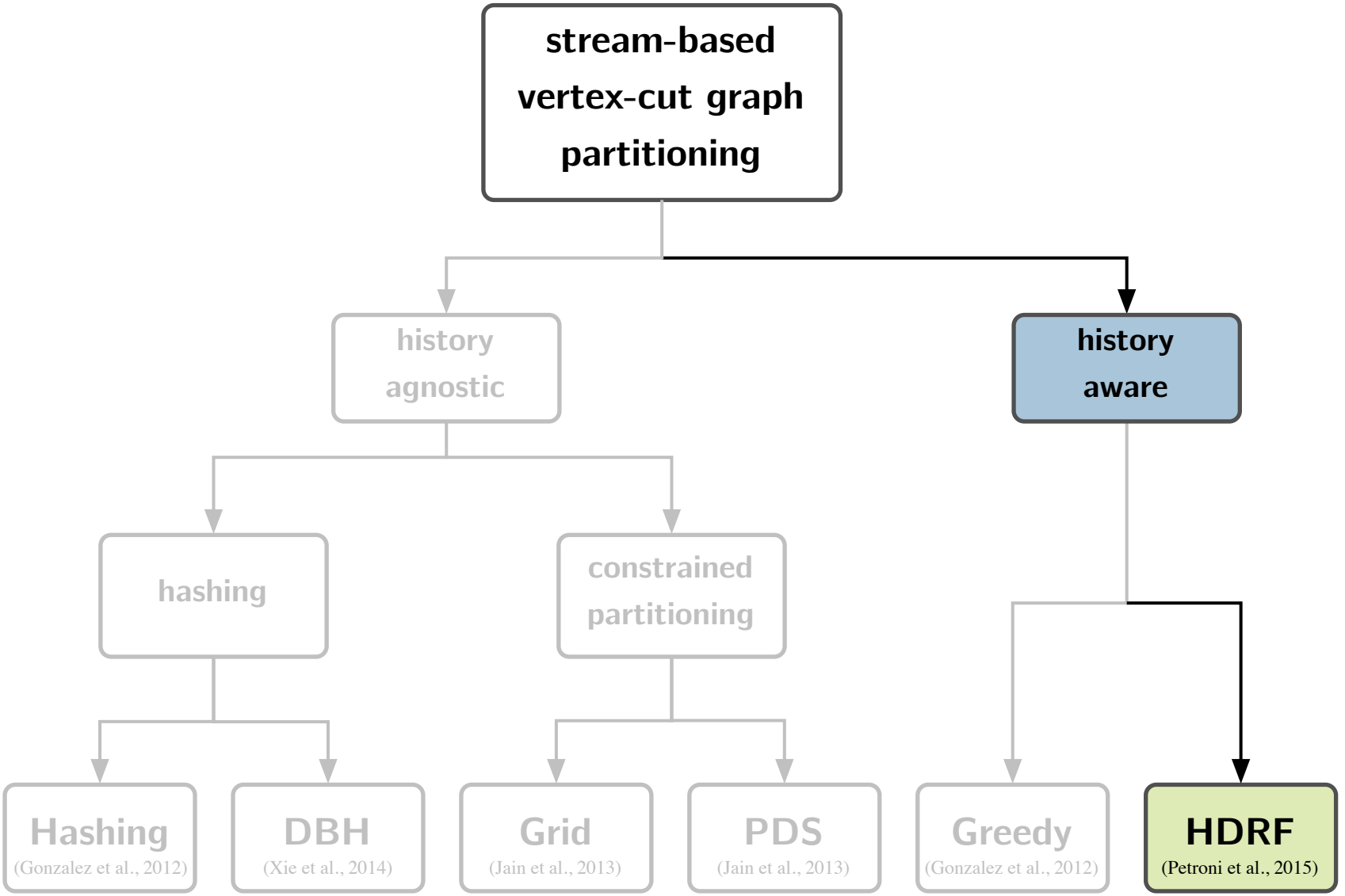
✓ scalable, easy parallel implementations.

✗ assignment decision taken cannot be later changed.

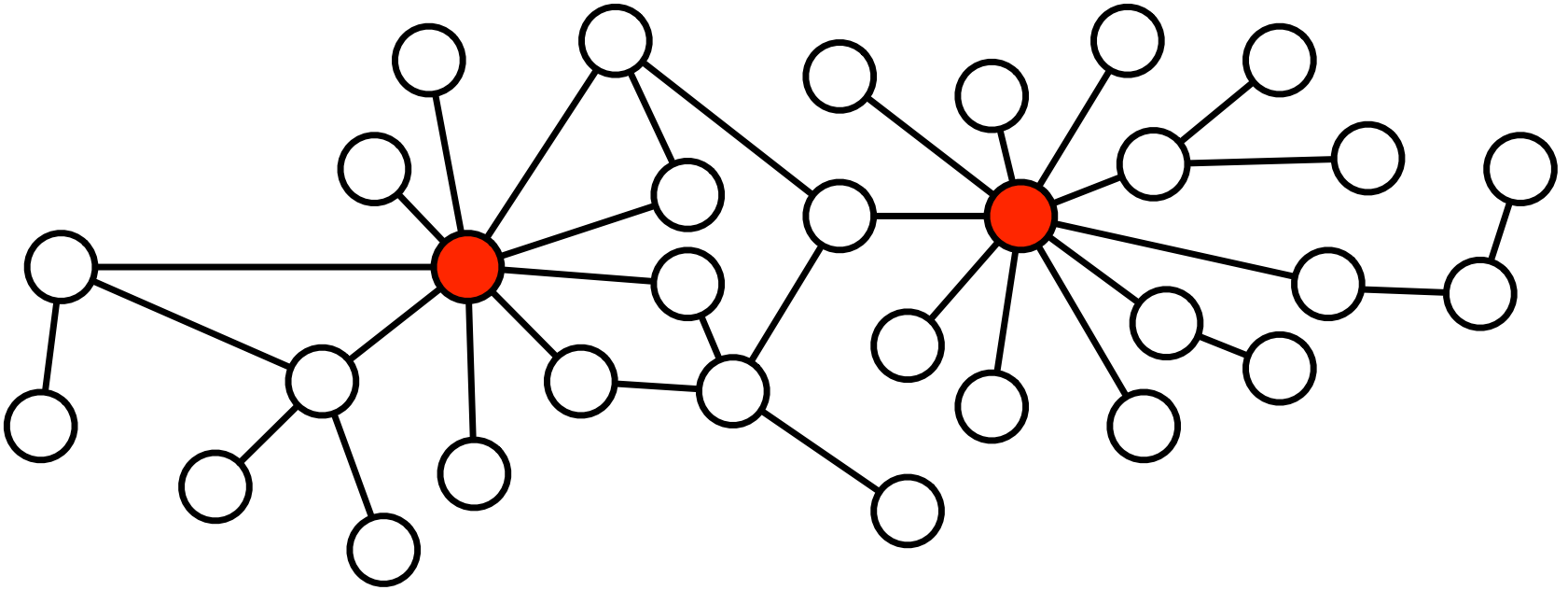
Algorithms Taxonomy



Algorithms Taxonomy

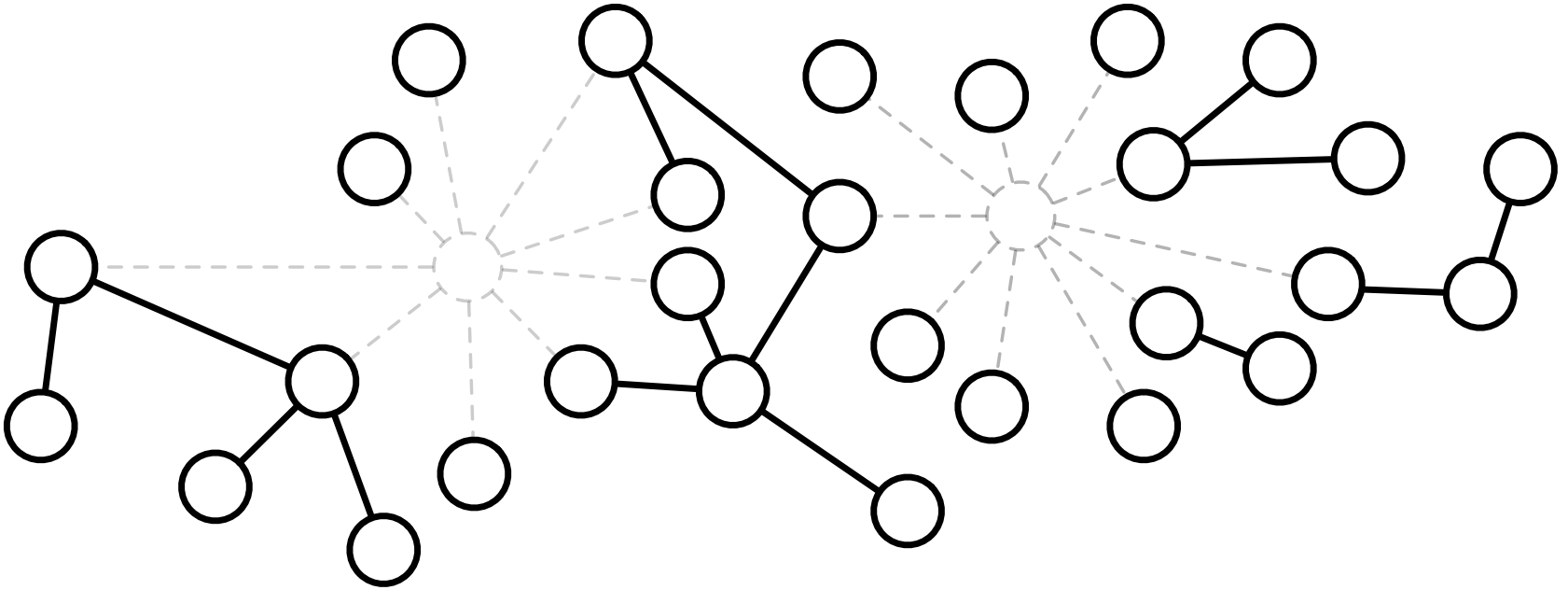


HDRF: High Degree are Replicated First



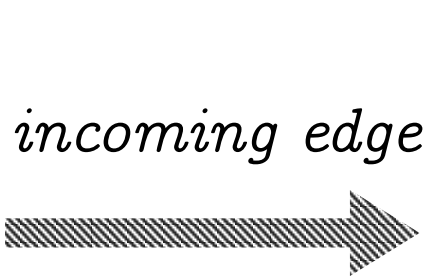
- favor the replication of high-degree vertices.
- the number of high-degree vertices in power-law graphs is very low.
- overall reduction of the replication factor.

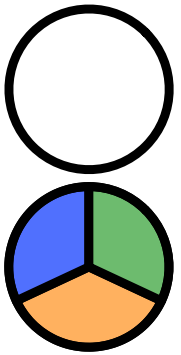
HDRF: High Degree are Replicated First



- in the context of robustness to network failure.
- if few high-degree vertices are removed from a power-law graph then it is turned into a set of isolated clusters.
- focus on the locality of low-degree vertices.

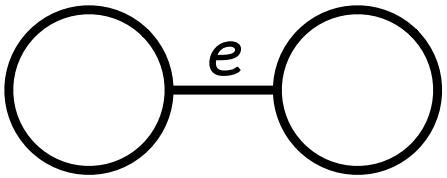
The HDRF Algorithm





vertex without replicas

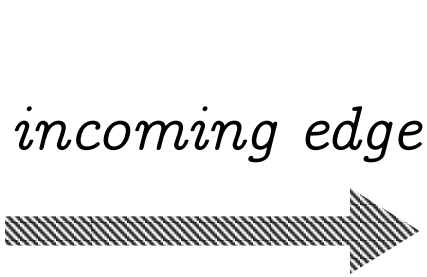
vertex with replicas

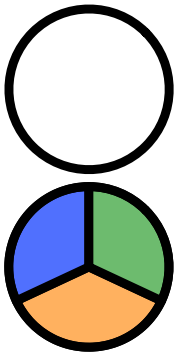


case 1

vertices not assigned to partitions

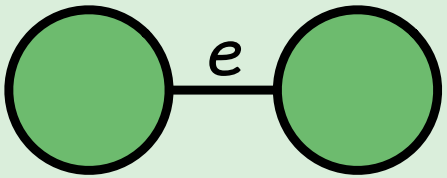
The HDRF Algorithm





vertex without replicas

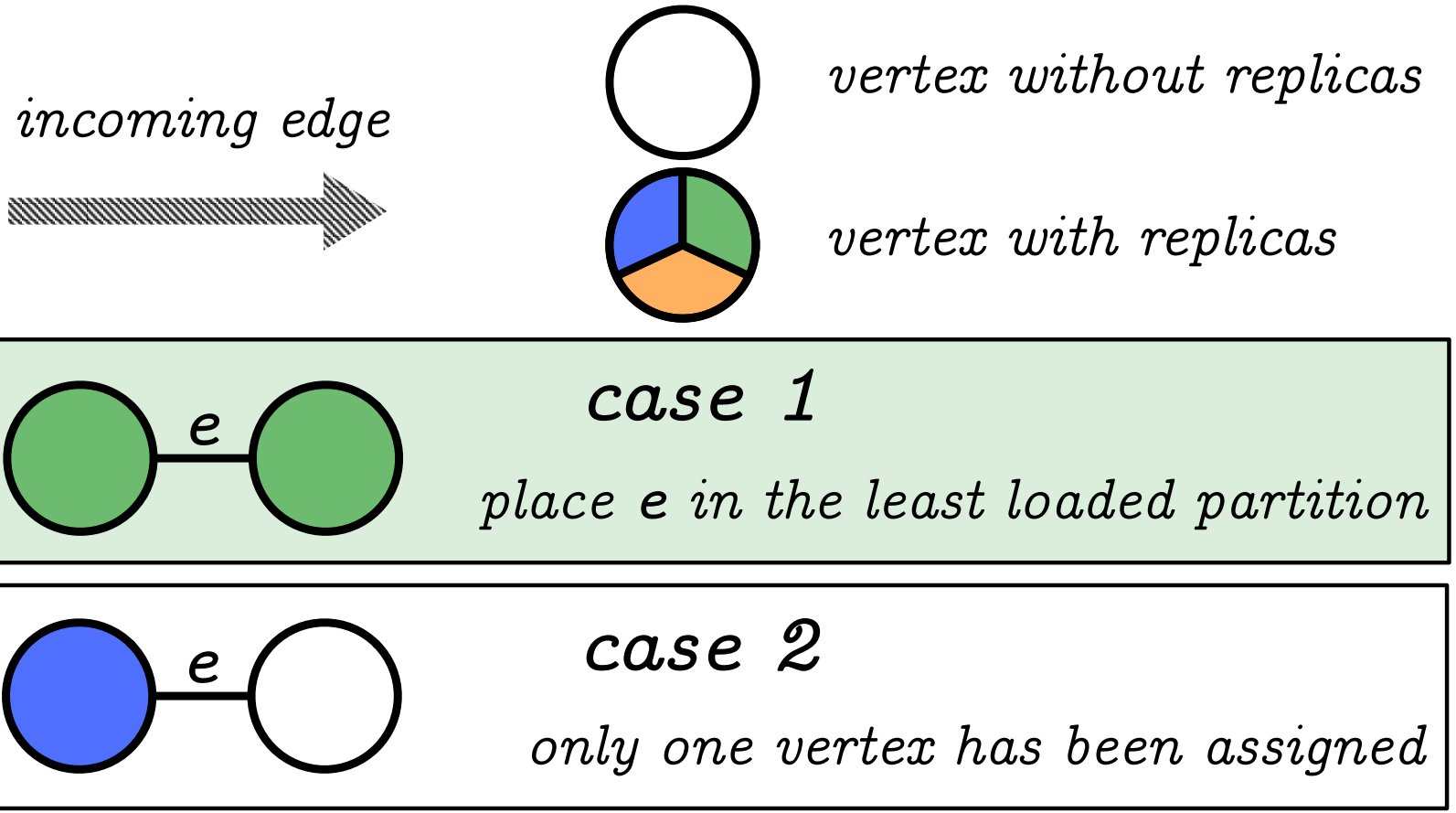
vertex with replicas



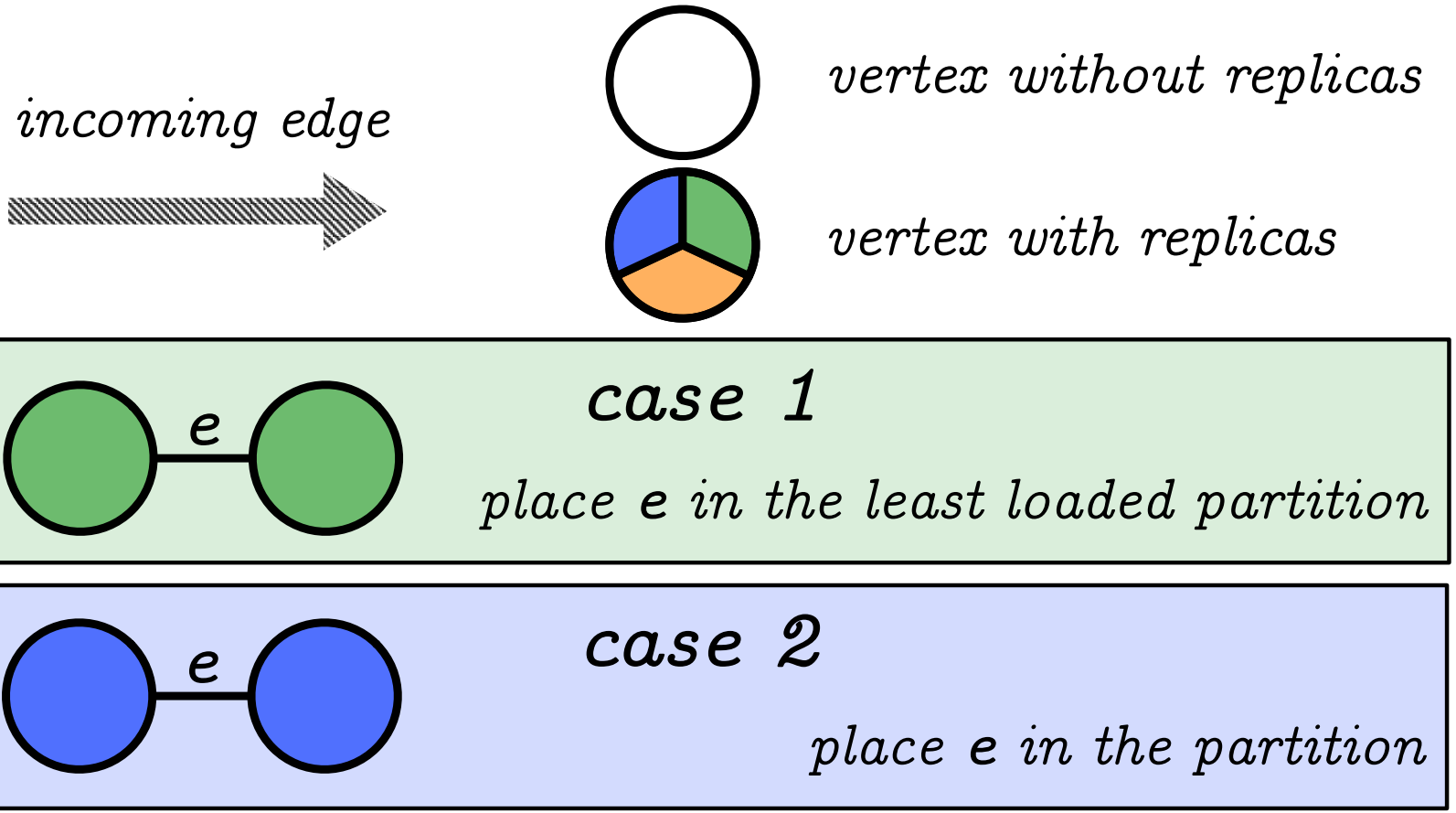
case 1

place **e** *in the least loaded partition*

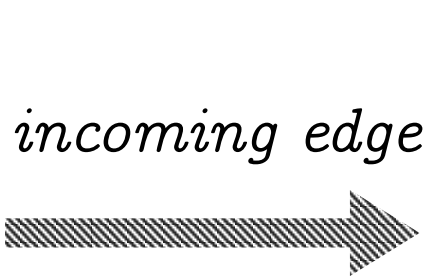
The HDRF Algorithm

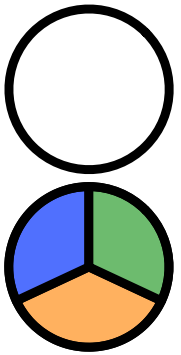


The HDRF Algorithm



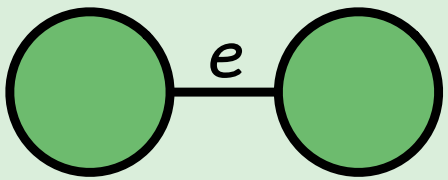
The HDRF Algorithm





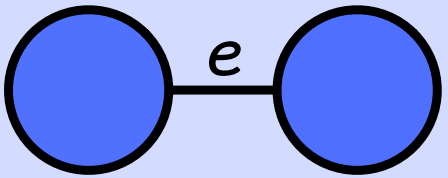
vertex without replicas

vertex with replicas



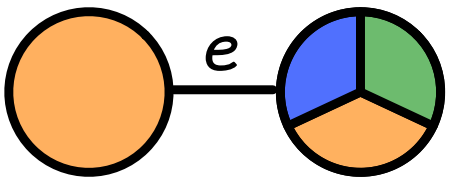
case 1

place e in the least loaded partition



case 2

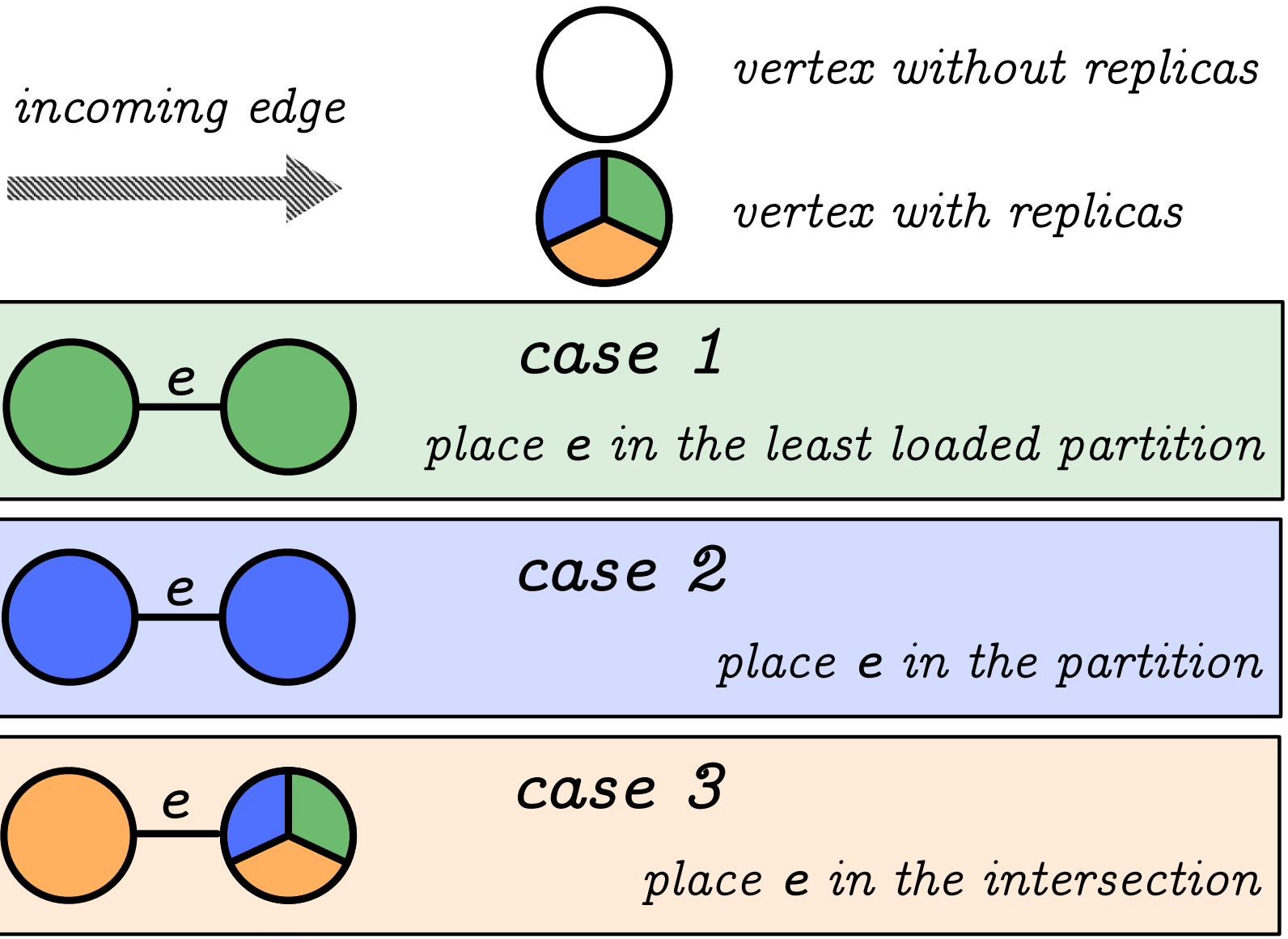
place e in the partition



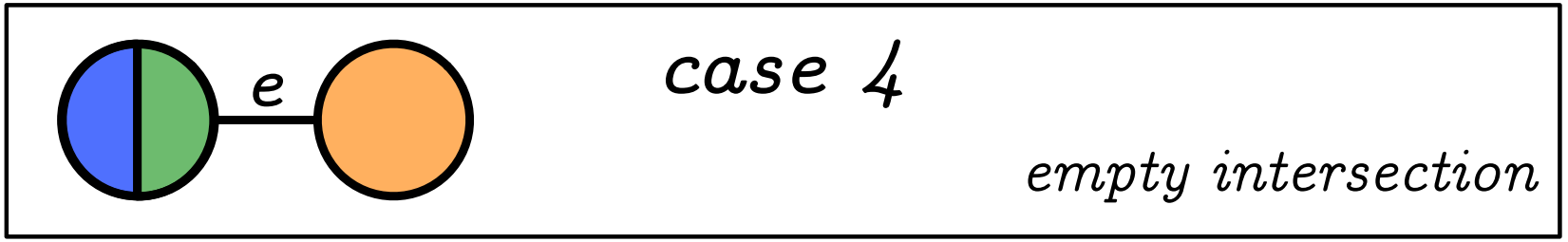
case 3

vertices assigned, common partition

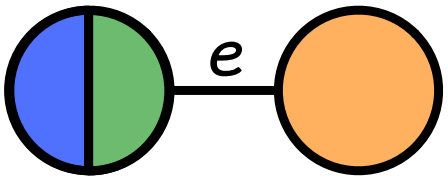
The HDRF Algorithm



Create Replicas



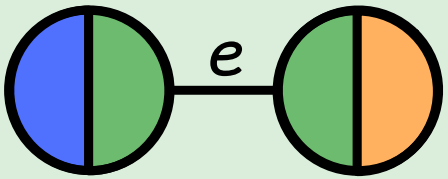
Create Replicas



case 4

empty intersection

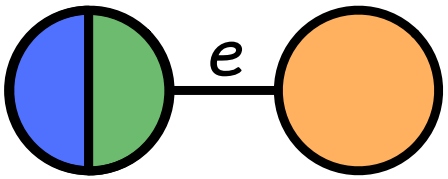
standard Greedy solution



case 4

least loaded partition in the union

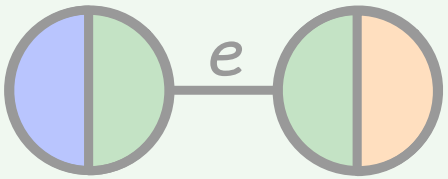
Create Replicas



case 4

empty intersection

standard Greedy solution

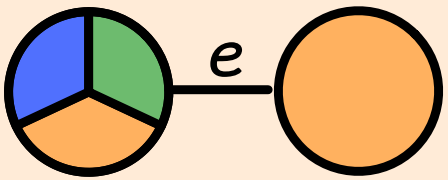


case 4

least loaded partition in the union

HDRF

$\delta(v1) > \delta(v2)$



case 4

replicate vertex with highest degree

Equivalent Formulation: Maximize The Score

- computing a score $C(e, p)$ for all partitions $p \in P$.
- assigns e to the partition that maximizes $C(e, p)$.

$$C(e, p) = C_{\text{REP}}(e, p) + C_{\text{BAL}}(p)$$

- the balance term breaks ties in replication term.
- this may not be enough to ensure load balance.

Equivalent Formulation: Maximize The Score

- computing a score $C(e, p)$ for all partitions $p \in P$.
- assigns e to the partition that maximizes $C(e, p)$.

$$C(e, p) = C_{\text{REP}}(e, p) + \lambda \cdot C_{\text{BAL}}(p)$$

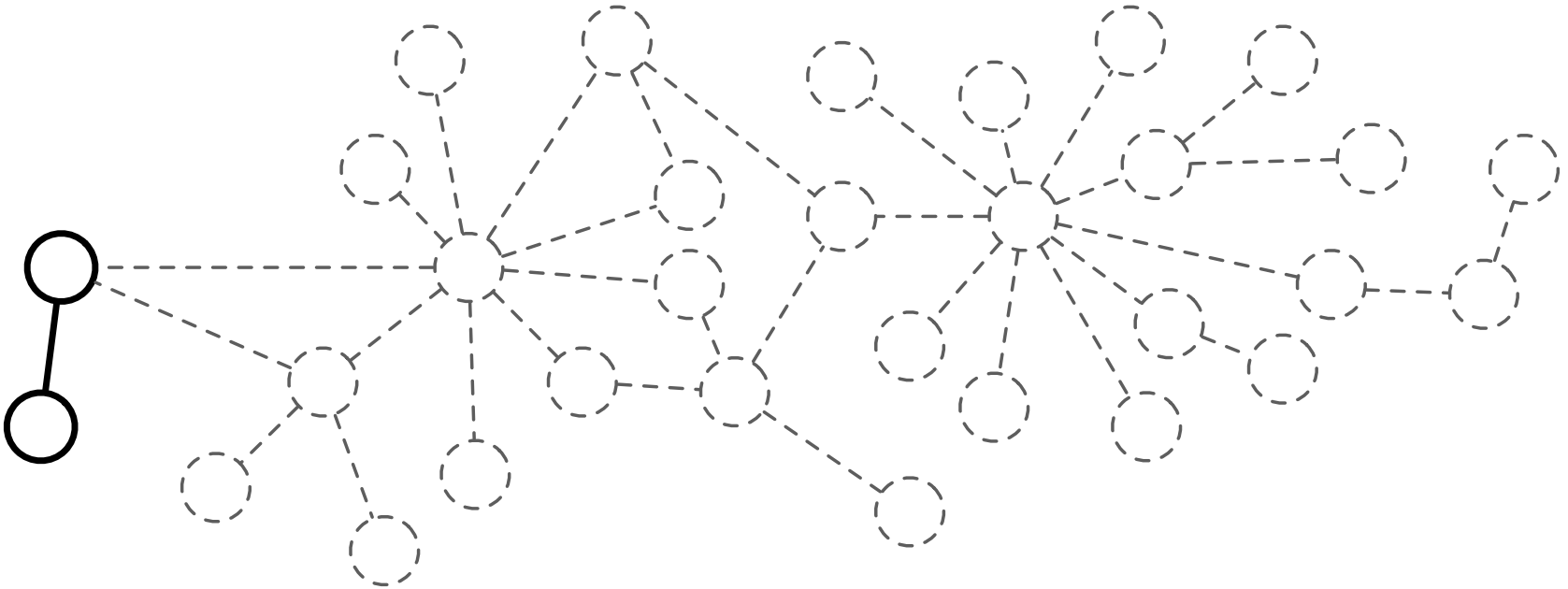
the balance term breaks ties in replication term.

this may not be enough to ensure load balance.

- λ controls the importance of the balance term.
- balanced partitions even when classical greedy fails.

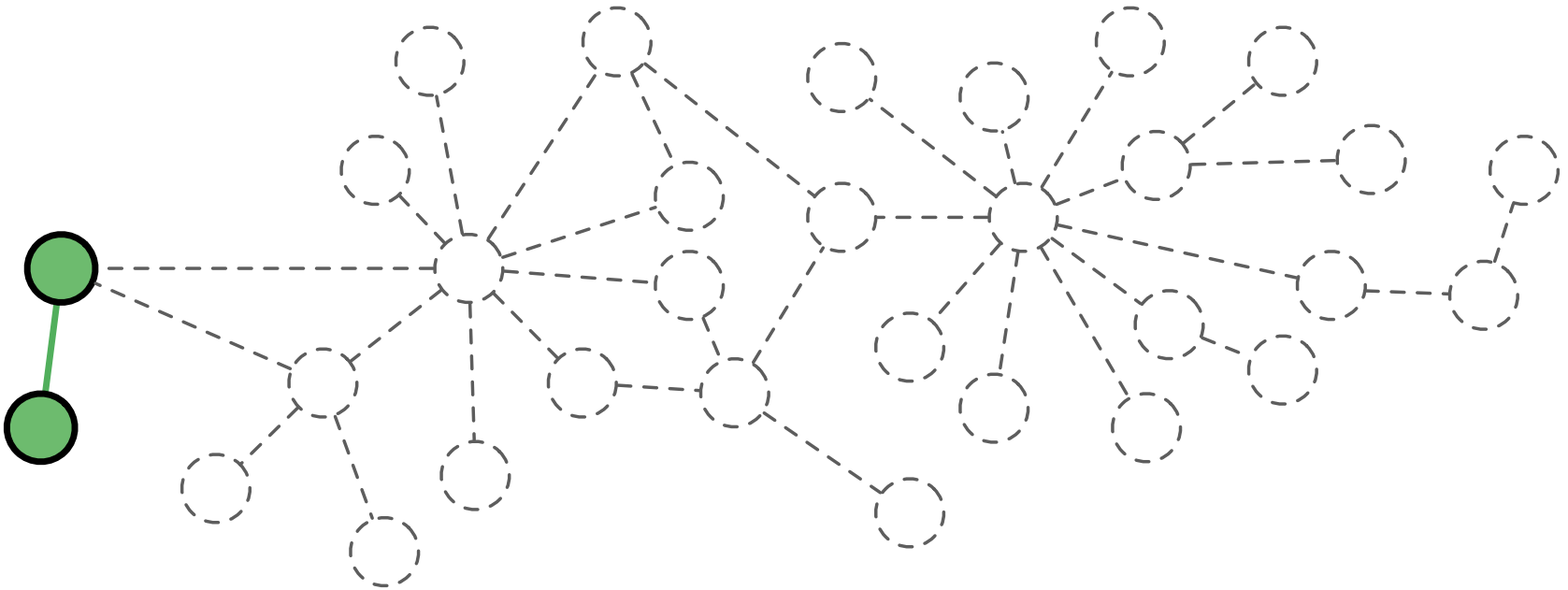
Ordered Stream Of Edges

- without λ



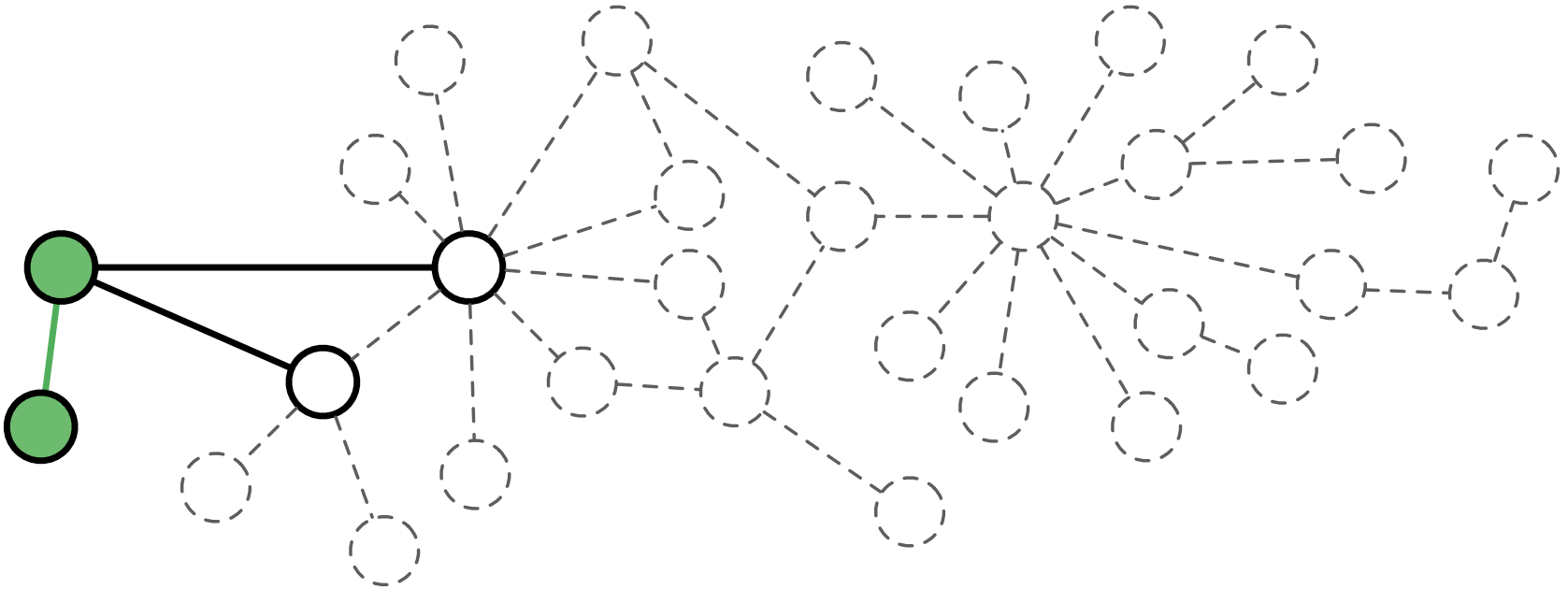
Ordered Stream Of Edges

- without λ



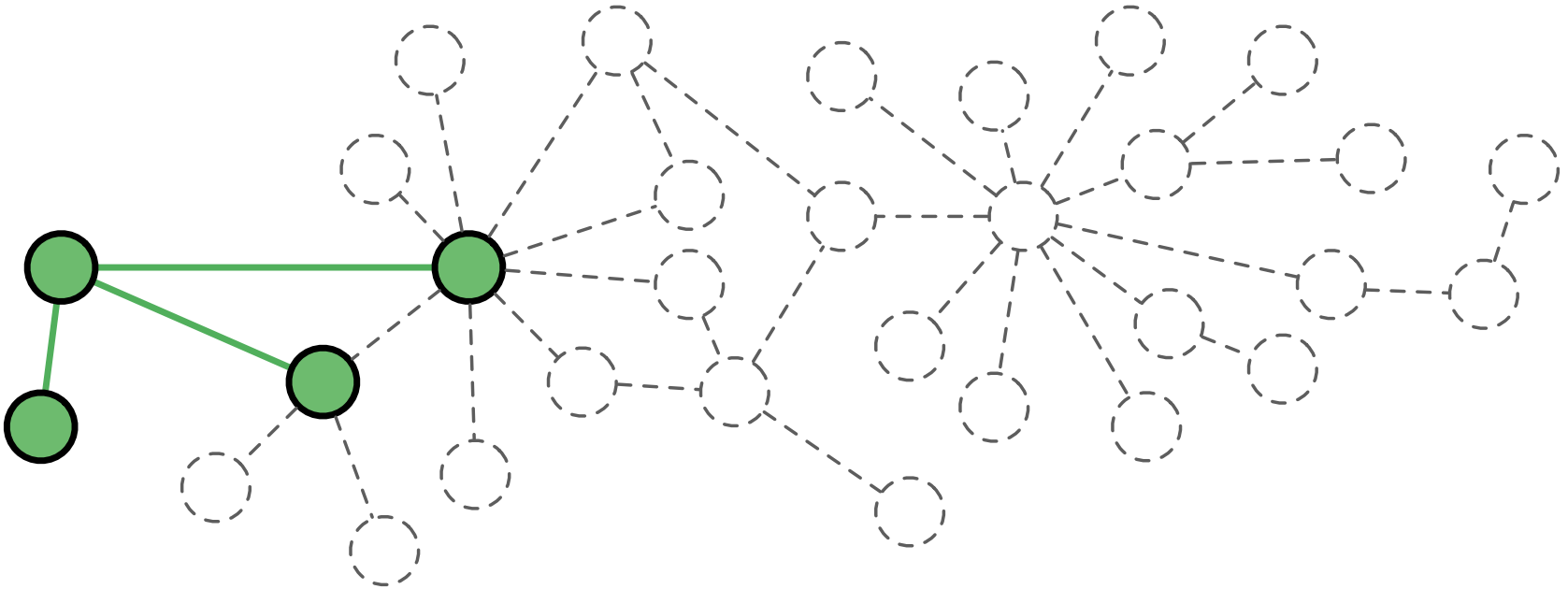
Ordered Stream Of Edges

- without λ



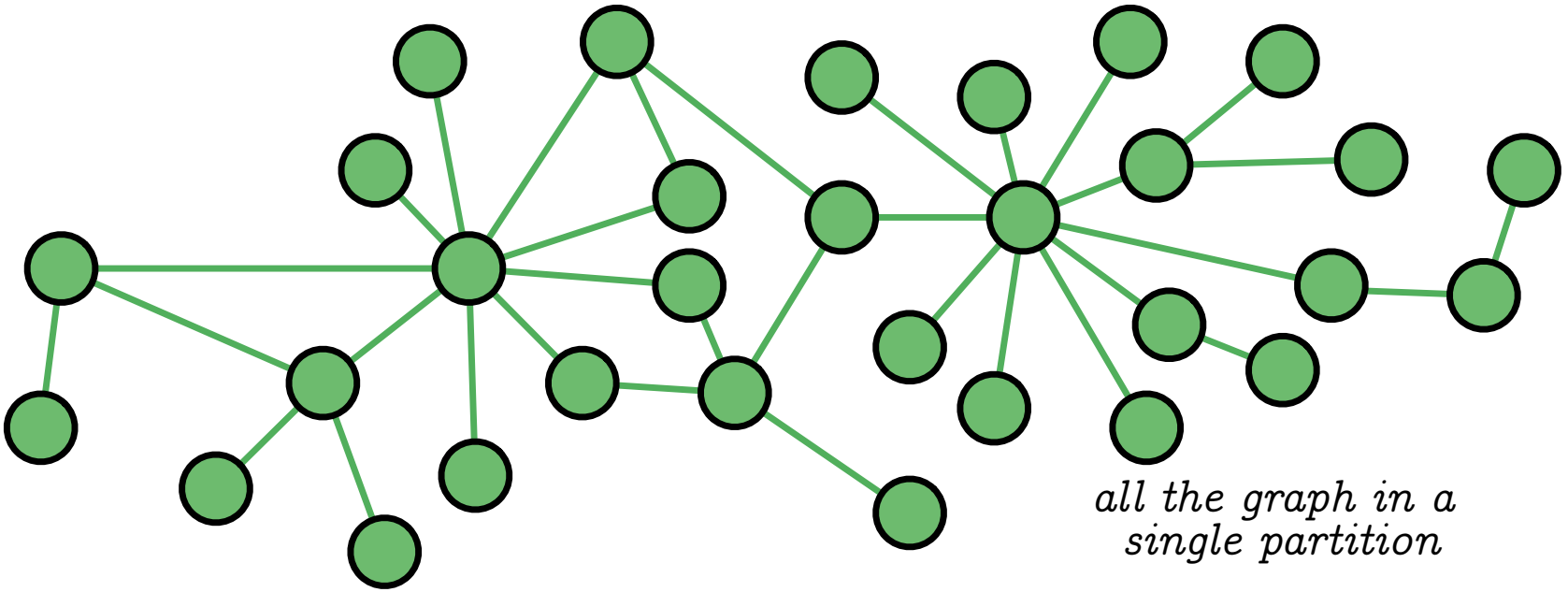
Ordered Stream Of Edges

- without λ



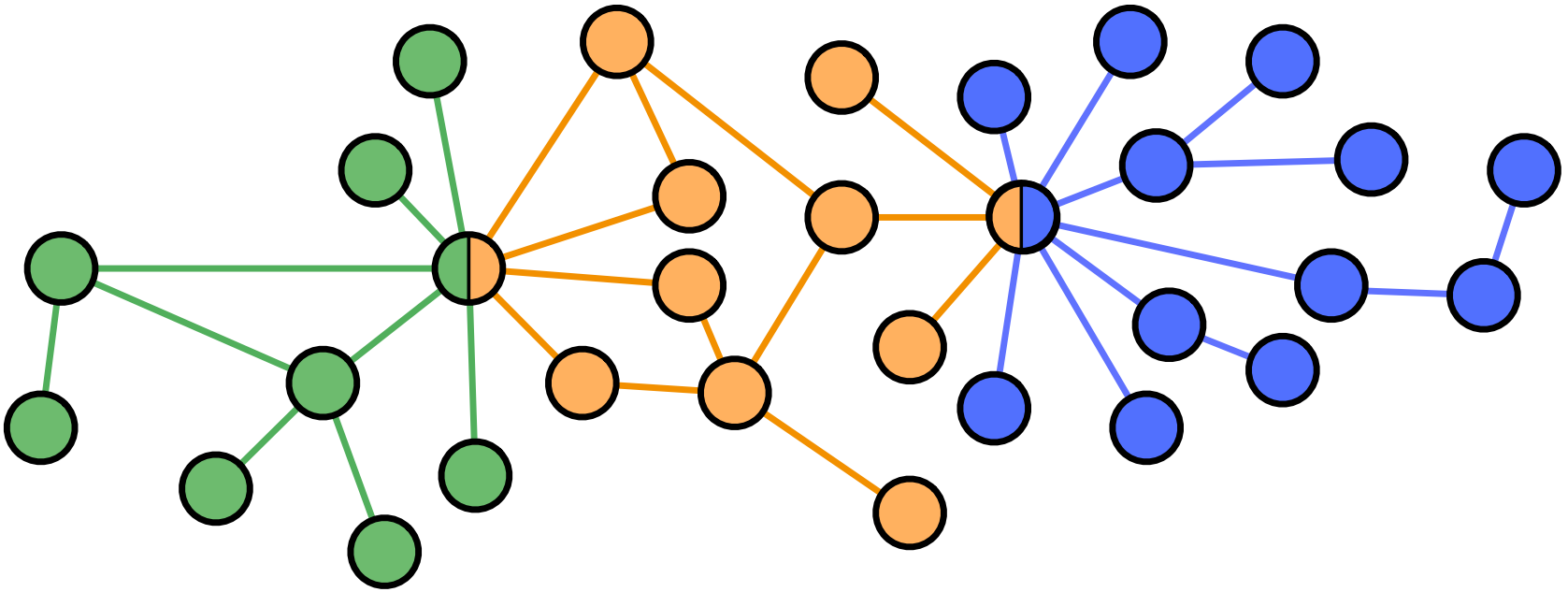
Ordered Stream Of Edges

- without λ



Ordered Stream Of Edges

- with λ



Experiments - Settings

- standalone partitioner.
  - VGP, a software package for one-pass vertex-cut balanced graph partitioning.
  - measure the performance: replication and balancing.
- GraphLab.
  - HDRF has been integrated in GraphLab PowerGraph 2.2.
  - measure the impact on the execution time of graph computation in a distributed graph computing frameworks.
- stream of edges in random order.

Experiments - Datasets

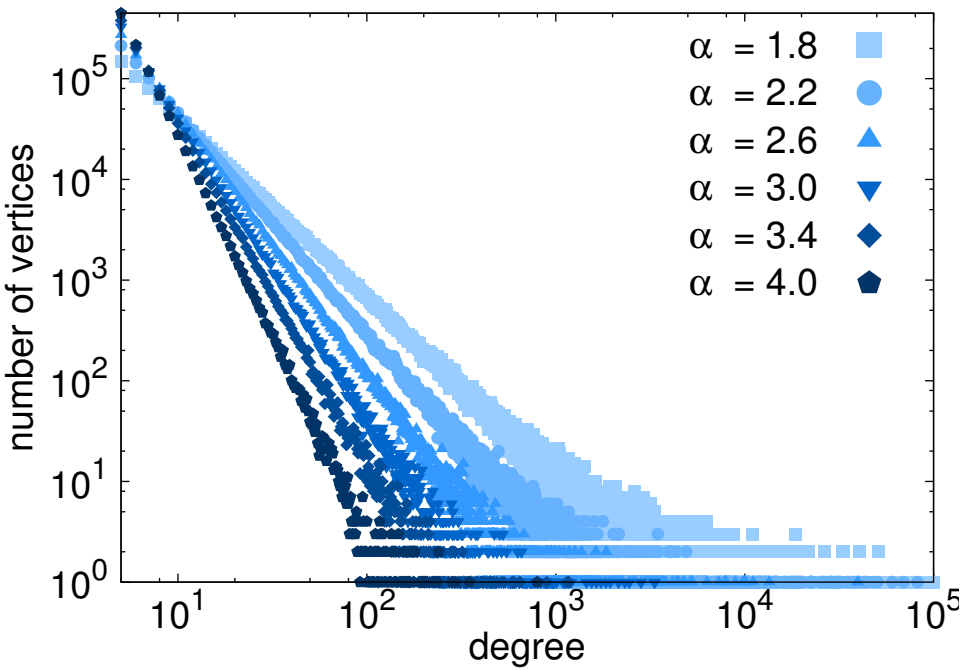
- real-word graphs.

Dataset	$\|V\|$	$\|E\|$
MovieLens 10M	80.6K	10M
Netflix	497.9K	100.4M
Tencent Weibo	1.4M	140M
twitter-2010	41.7M	1.47B

- synthetic graphs.

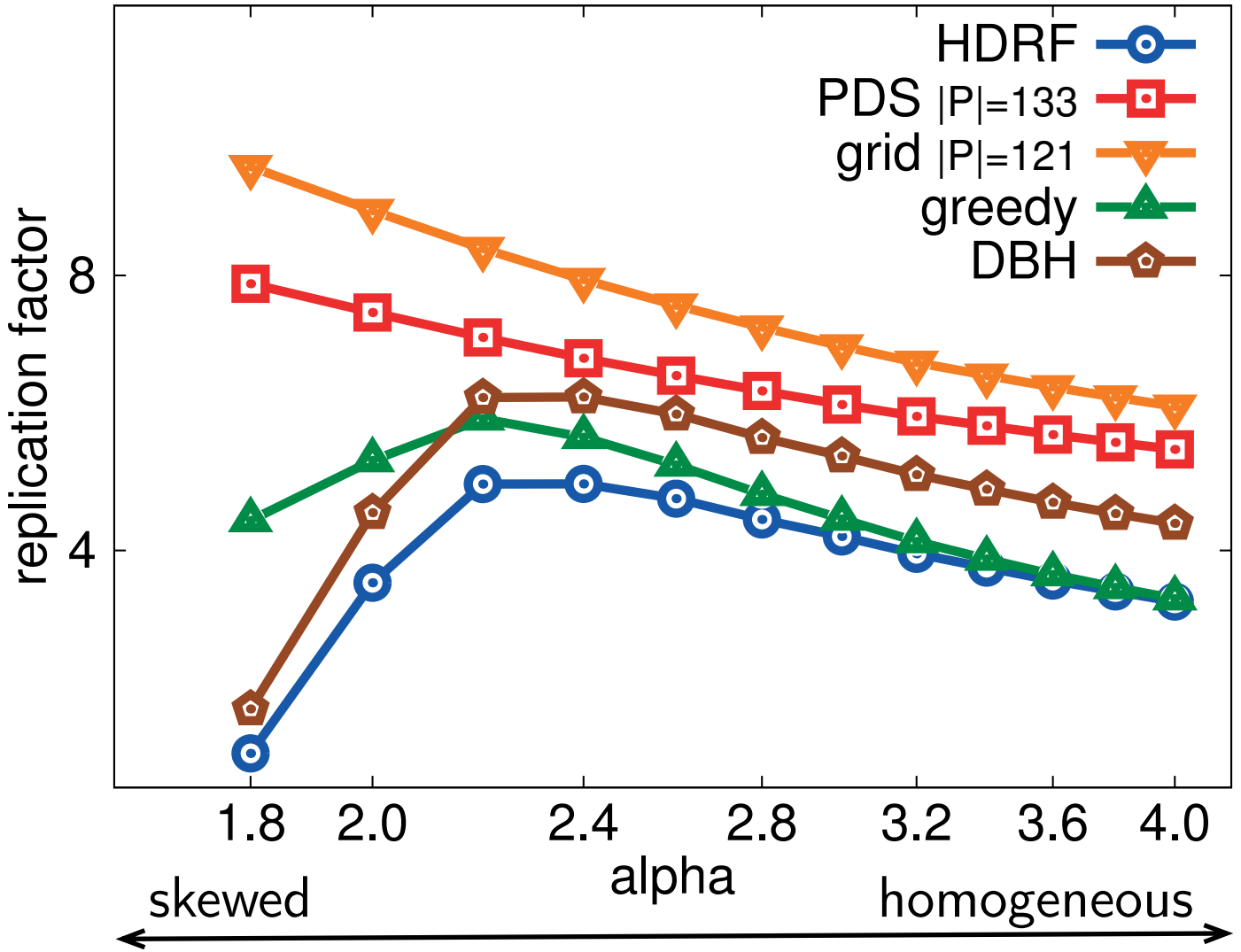
1M vertices

60M to 3M edges



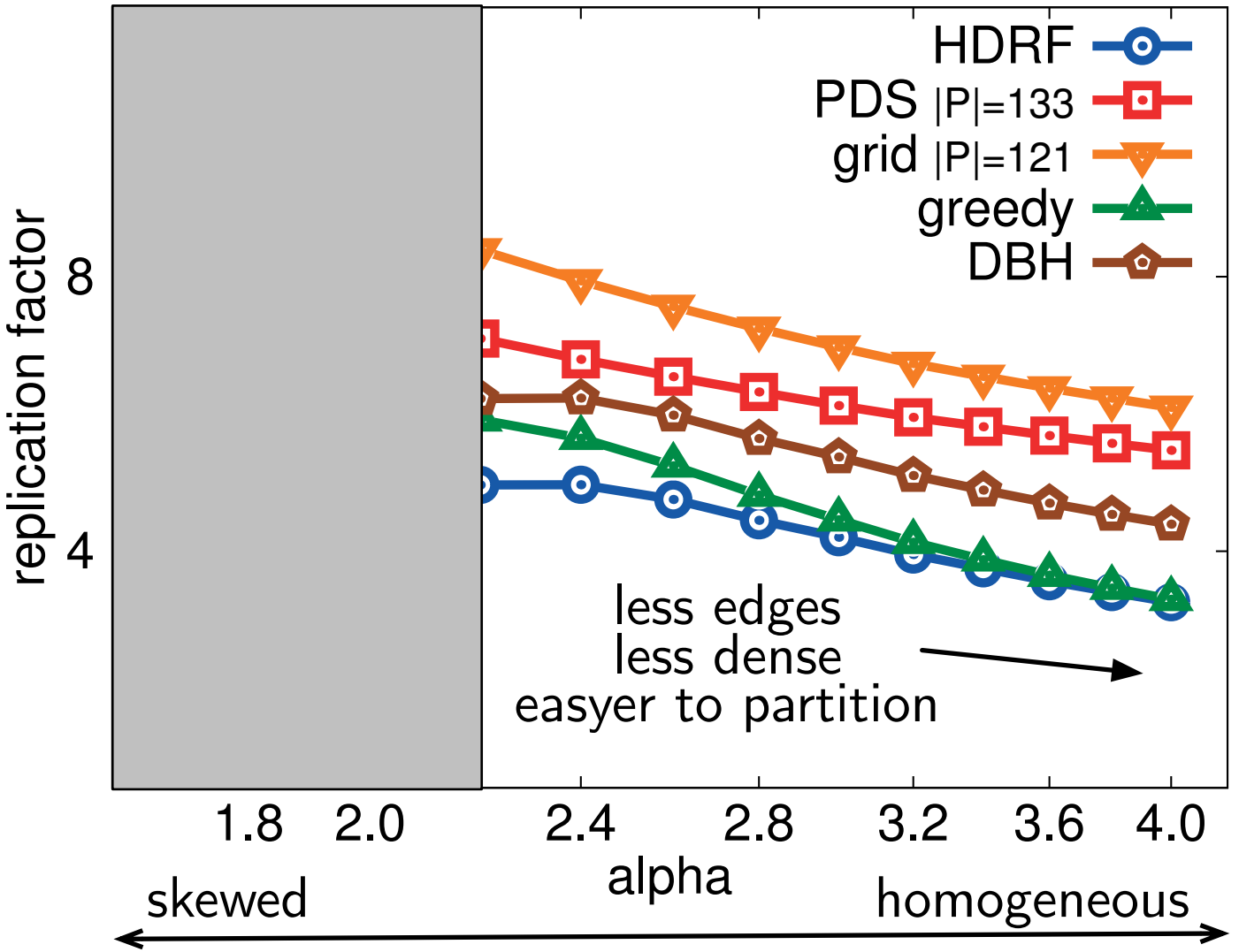
Results - Synthetic Graphs Replication Factor

- 128 partitions



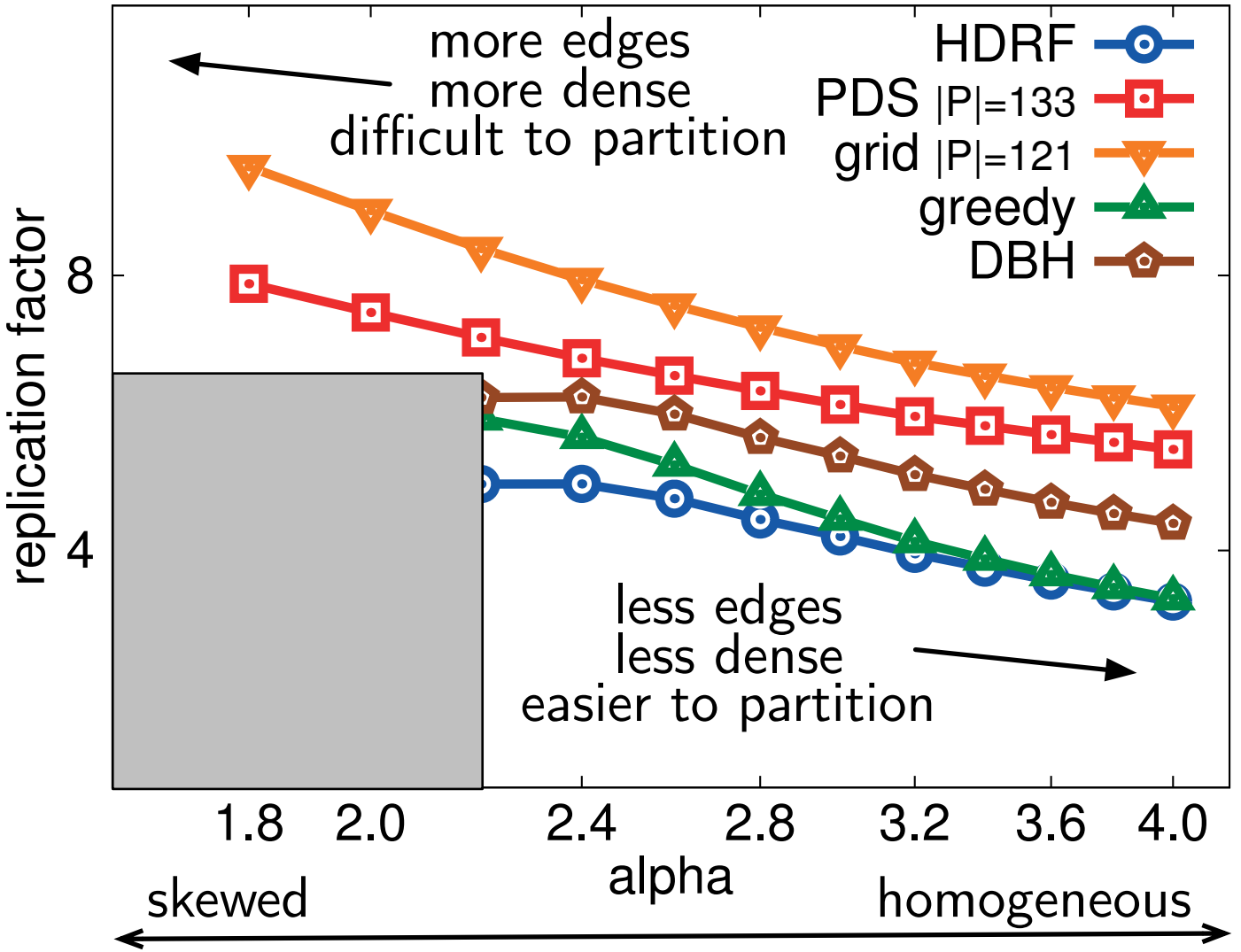
Results - Synthetic Graphs Replication Factor

- 128 partitions



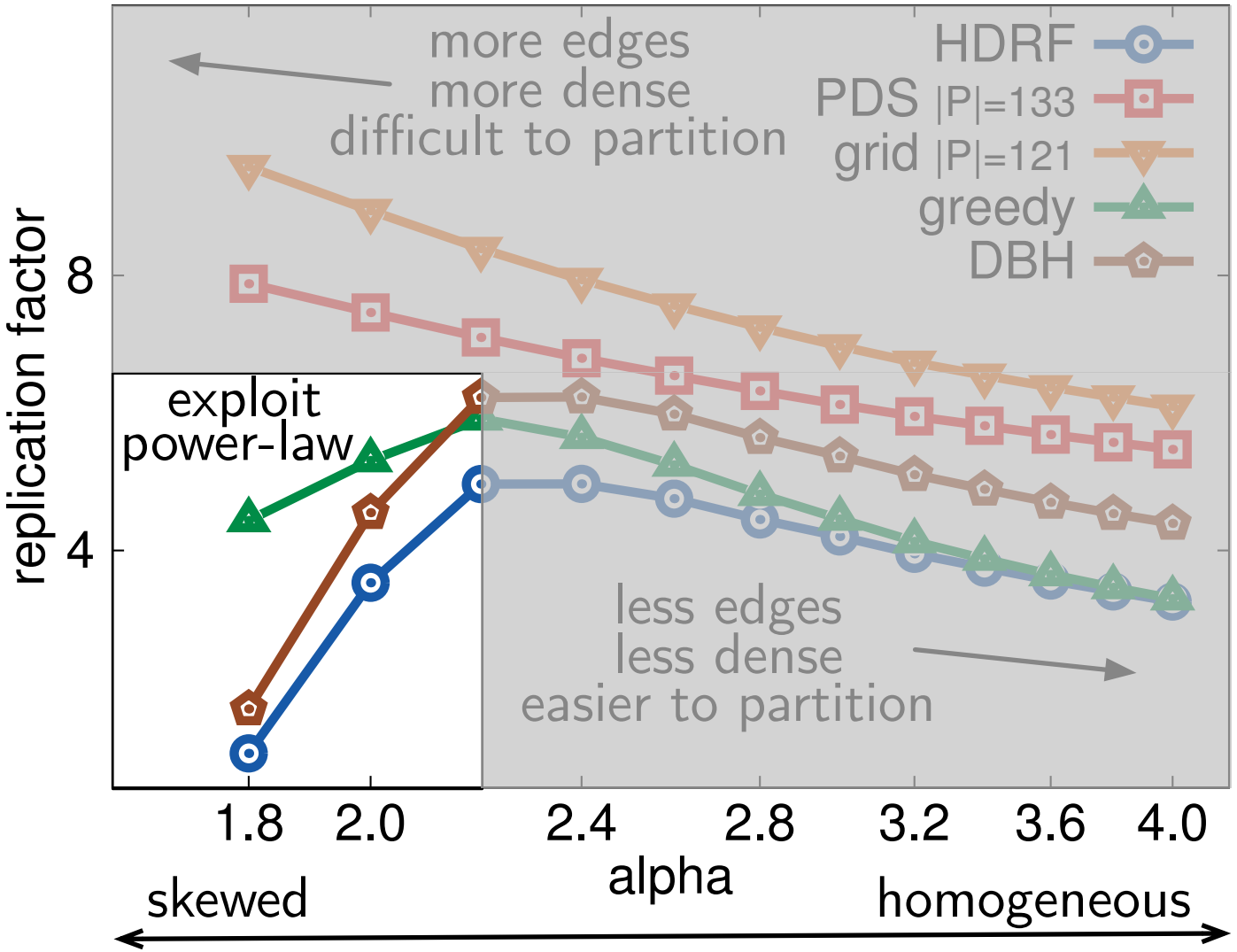
Results - Synthetic Graphs Replication Factor

- 128 partitions



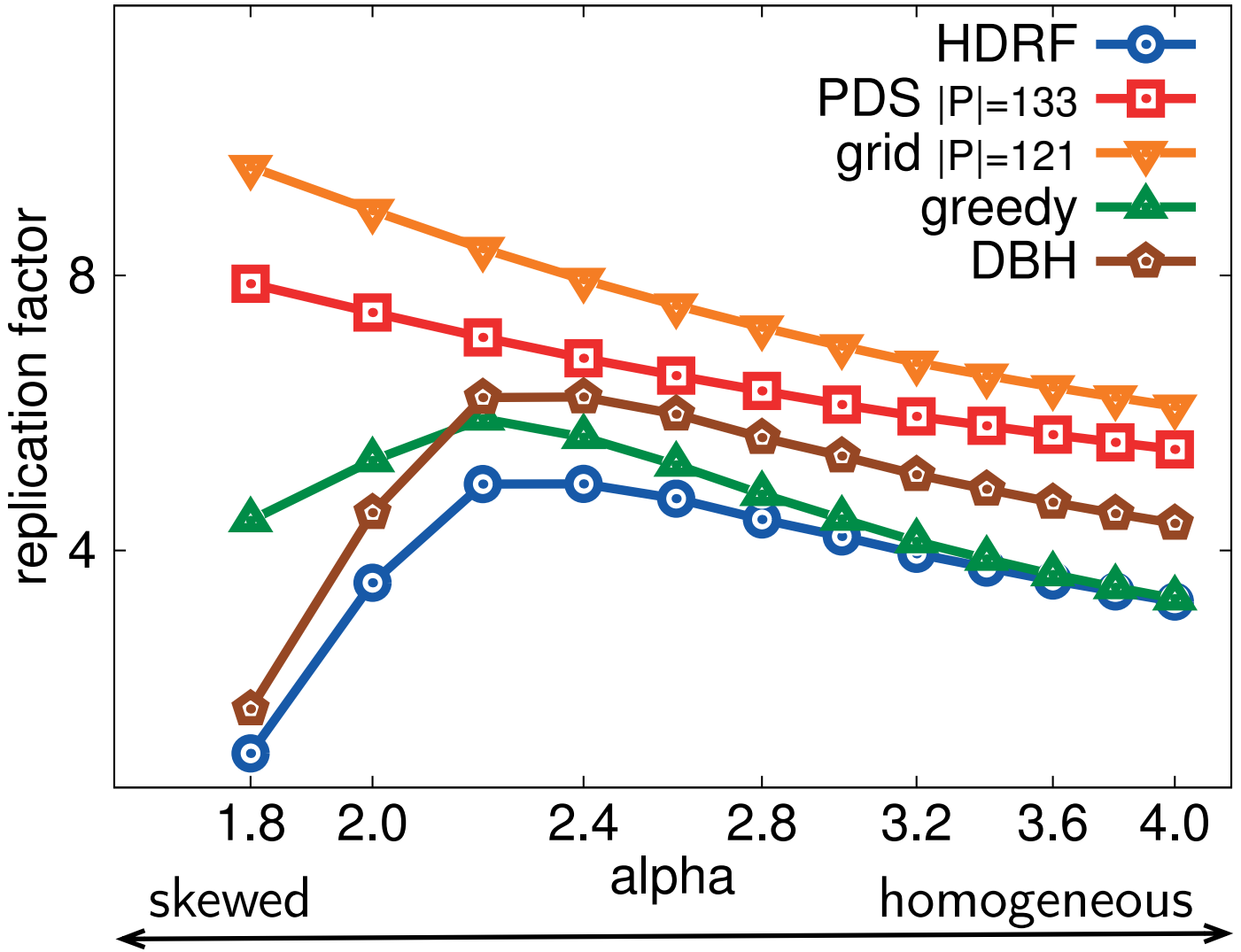
Results - Synthetic Graphs Replication Factor

- 128 partitions



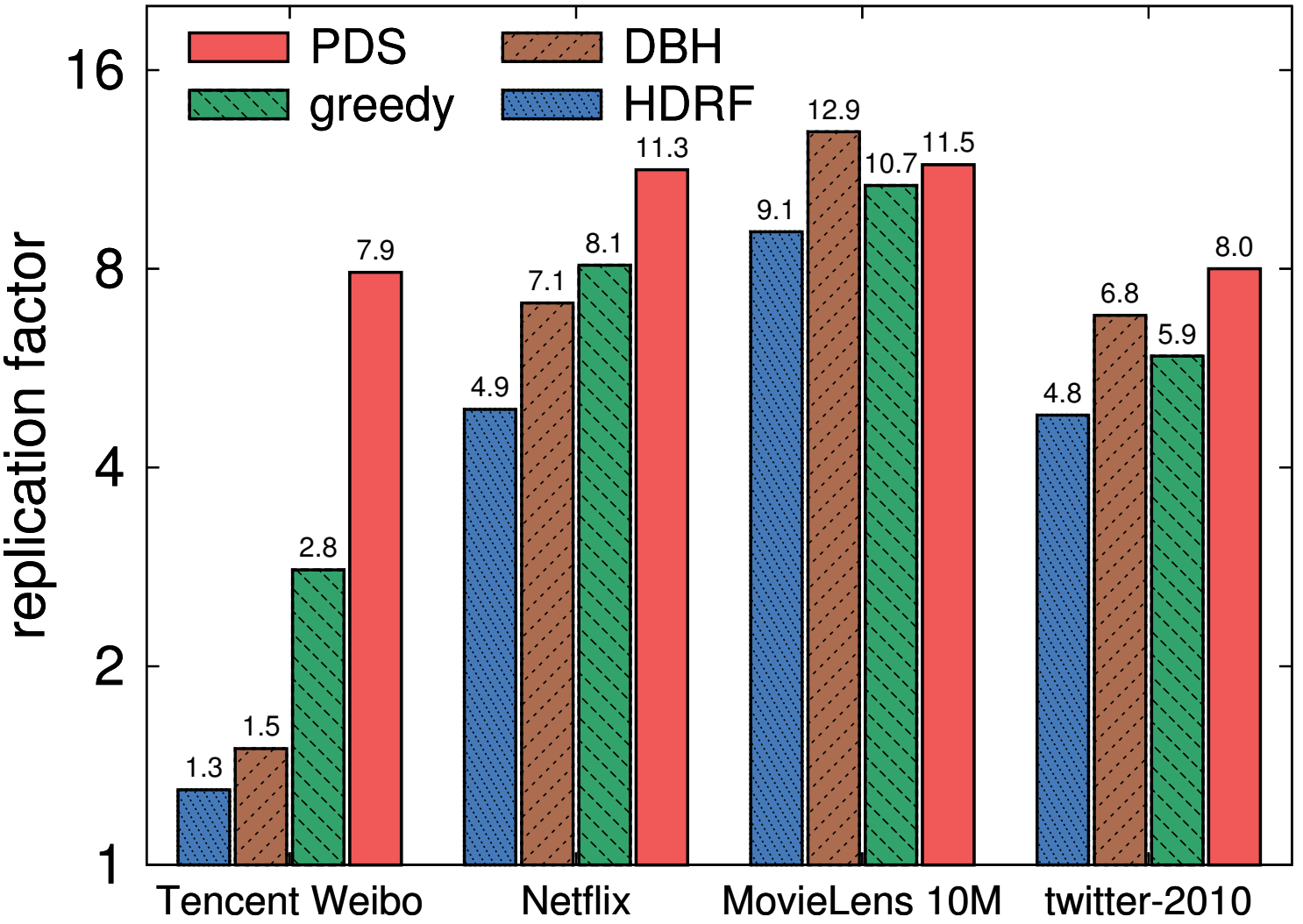
Results - Synthetic Graphs Replication Factor

- 128 partitions



Results - Real-Word Graphs Replication Factor

- 133 partitions



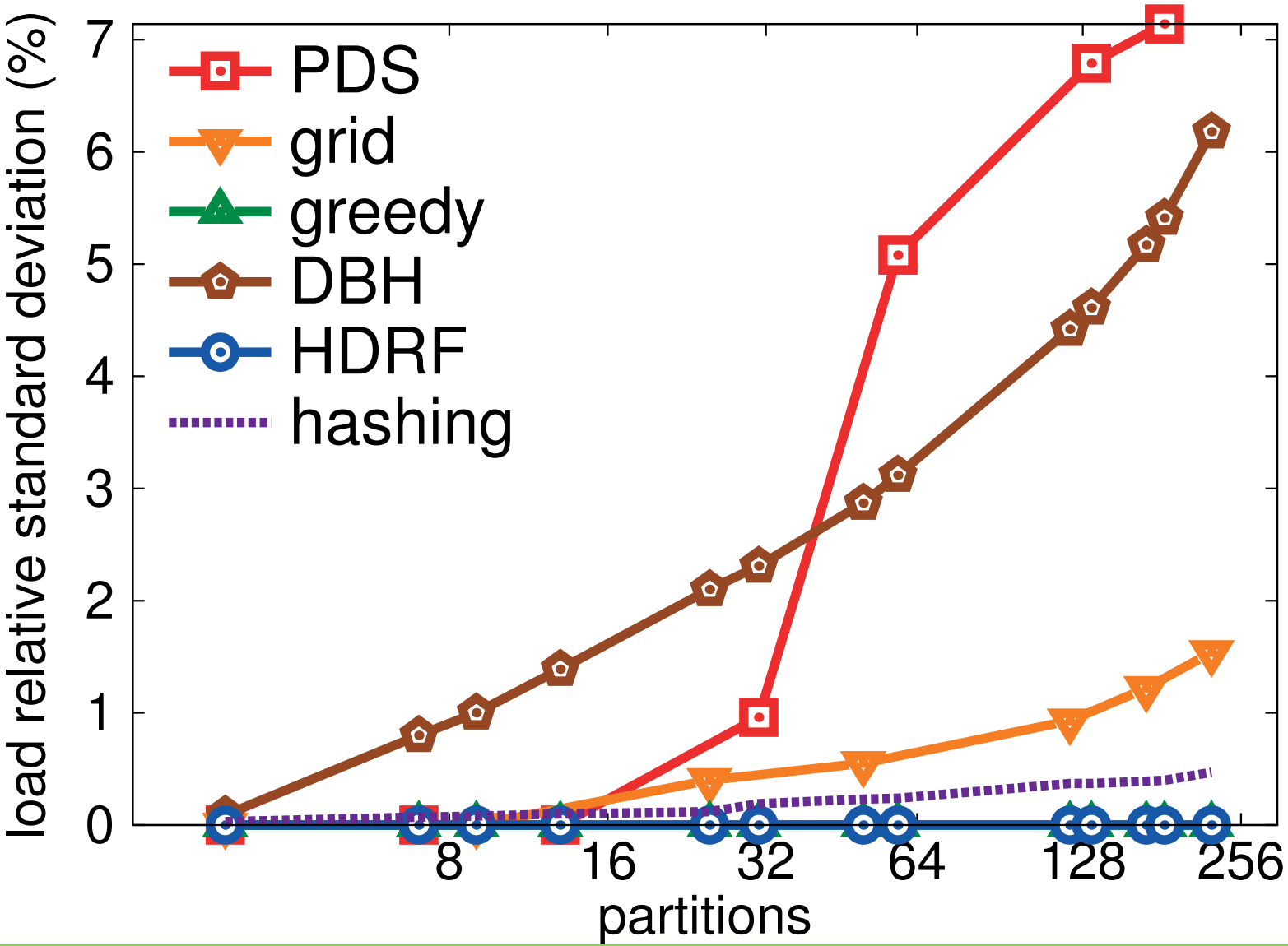
Results - Real-Word Graphs Replication Factor

- 133 partitions

HDRF achieves a replication factor about **40%** smaller than **DBH,** more than **50%** smaller than **Greedy,** almost **3×** smaller than **PDS,** more than **4×** smaller than **Grid** and almost **14×** smaller than **Hashing.**

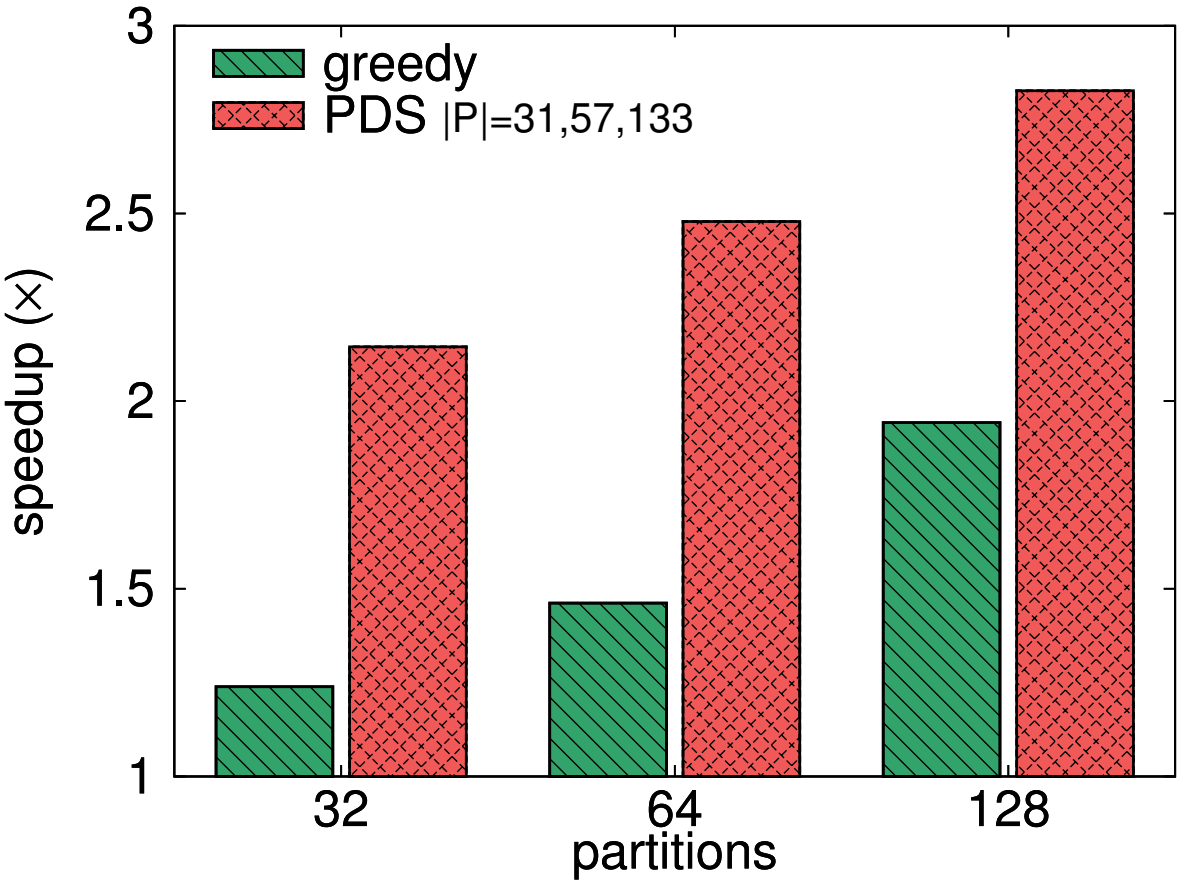
Results - Load Relative Standard Deviation

- MovieLens 10M



Results - Graph Algorithm Runtime Speedup

- *SGD* algorithm for collaborative filtering on *Tencent Weibo*.



- the speedup is proportional to both:
  - the advantage in replication factor.
  - the actual network usage of the algorithm.

Conclusion

- HDRF is a one-pass vertex-cut graph partitioning algorithm.
- based on a greedy vertex-cut approach that leverages information on vertex degrees.
- we provide a theoretical analysis of HDRF with an average-case upper bound for the vertex replication factor.
- experimental study shows that:
  - HDRF provides the smallest replication factor with close to optimal load balance.
  - HDRF significantly reduces the time needed to perform computation on graphs.
- the stand-alone software package for one-pass v-cut balanced partitioning at https://github.com/fabiopetroni/VGP.

Thank you!

Questions?

Fabio Petroni

Sapienza University of Rome, Italy

Current position:

PhD Student in Engineering in Computer Science

Research Interests:

data mining, machine learning, big data

petroni@dis.uniroma1.it