# Appendix:

We have conducted a series of experiments to show *SLFE*'s performance improvement over other existing systems [8, 1, 2, 7, 5]. Due to the page limitation of the paper, we are not able to fit all the experimental results into the paper: **Start Late or Finish Early: A Distributed Graph Processing System with Redundancy Reduction**. Hence, we present the supplemental data in an appendix.
There are ten graphs involved in our experiments. To make the presentation compact, we refer to those graphs with the abbreviations. Readers can find their full names and properties in the Table 1.

Table 1: The graph datasets [6, 4, 3] used in experiments.

| Real graph | —V— | —E— | AvgDegree | Type |
|---|---|---|---|---|
| pokec (PK) | 1.6M | 30.6M | 18.8 | Social |
| orkut (OK) | 3.1M | 117.2M | 38.1 | Social |
| livejournal (LJ) | 4.8M | 69M | 14.23 | Social |
| wiki (WK) | 12.1M | 378.1M | 31.1 | Hyperlink |
| delicious (DI) | 33.8M | 301.2M | 8.9 | Folksonomy |
| s-twitter (ST) | 11.3M | 85.3M | 7.5 | Social |
| friendster (FS) | 65.6M | 1.8B | 27.5 | Social |
| Synthetic graph | —V— | —E— | AvgDegree | Type |
| RMAT1 | 100M | 2B | 20 | RMAT |
| RMAT2 | 300M | 6B | 20 | RMAT |
| RMAT3 | 500M | 10B | 20 | RMAT |

To guide readers through this appendix and help link to the paper, we summaries the contents as follows:

- Figure 1 shows the absolute runtime of intra-machine scalability experiments, as we promise in the end of section 4.2.1 of the paper. We run five applications with seven real graphs that fit in a single machine's memory, while the number of cores varies from 1 to 68. *SLFE* achieves nearly linear scale-up in all cases. Moreover, we include the runtime of two state-of-the-art shared-memory systems — GraphChi [5] and Ligra [7].

- Figure 2 and Figure 3 present the preprocessing time of PowerGraph [2], PowerLyra [1], and *SLFE* on seven real graphs. These are the supplements to the Figure 9 in the paper, which only contains the preprocessing time of three synthetic graphs. *SLFE* finishes the preprocessing phase (include the RRG generation overhead) faster than PowerGraph and PowerLyra, and shows certain scale-out.

- Figure 4 has the absolute runtime of inter-machine scalability experiments on seven real graphs. This is a supplement for the Figure 10 in the paper, where the data of three synthetic graphs are exhibited. Even considering the RRG generation overhead, *SLFE* outperforms PowerLyra, PowerGraph, and Gemini [8] in almost all the cases. As we point out in the section 4.2.2 of the paper, execution time does not monotonously decrease when enlarging the cluster size. The reason behind those inflection points is that the communication overhead surpasses the benefits obtained from adding more computation resources.

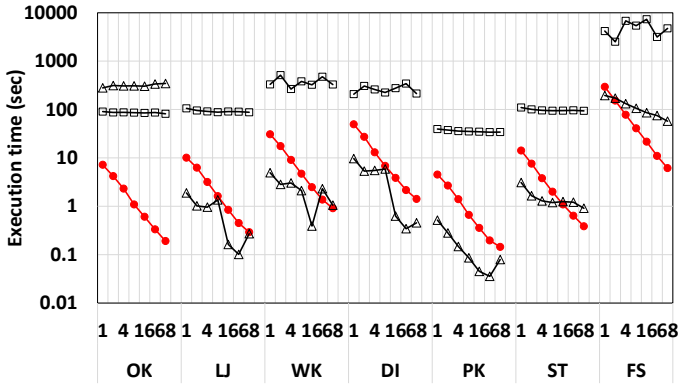
- Figure 5 has the absolute runtime of inter-machine scalability experiments on three synthetic RMAT graphs. This is also a supplement for the Figure 10 in the paper, where we only report the normalized execution time. The Redundancy Reduction Guidance (RRG) generation overhead is included in *SLFE*'s execution time (all other preprocessing time are excluded).

- Figure 6 reports *SLFE*'s trend-lines of five applications running with seven real-world graphs. Similar to the Figure 11 in the paper, Figure 6 is generated by varying the number of cores per machine as well as the number of machines in the cluster. Generally, *SLFE* gains benefits from more cores all the time, but its performance could degrade as the size of the cluster increases. Especially on those small graphs, insufficient computation with more communication overhead leads to the scaling loss.

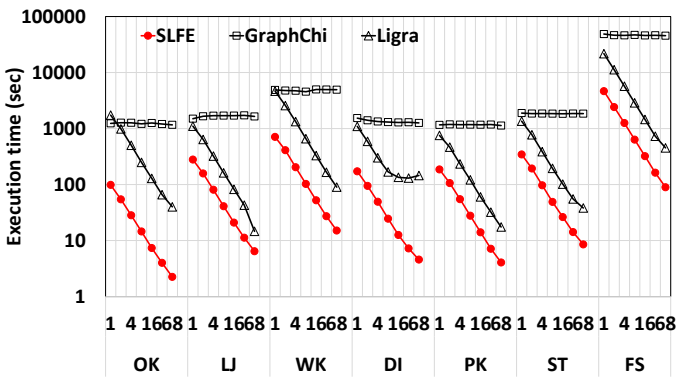
- Figure 7 to Figure 11 demonstrate how the computations reduce during the execution under the principle of"start late" and "finish early". As defined in the section 4.3.1 of the paper, the computation refers to an update on a vertex, which includes a $min/max$ or arithmetic operation and its corresponding synchronization operations. We select six figures as the representatives and place them in the Figure 12 of the paper.

- Table 2 shows the memory footprint of *SLFE* and other three distributed graph processing systems. Gemini achieves the lowest memory consumption. *SLFE* consumes memory space slightly more for the Redundancy Reduction Guidance (RRG), while it is still very competitive against Gemini.

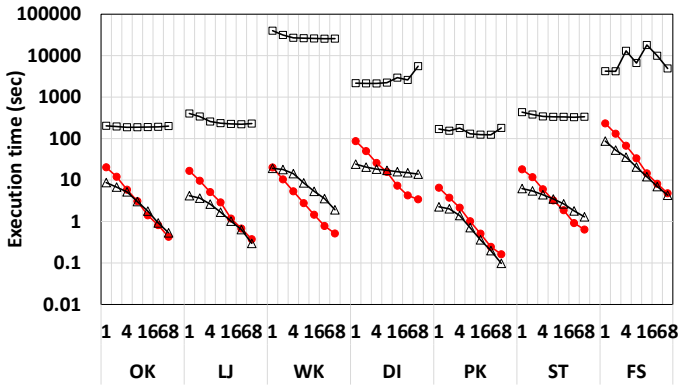
- Table 3 gives out the exact disk costs introduced by our Redundancy Reduction Guidance (RRG), as well as the ratios to total disk space needed to store the preprocessing data.



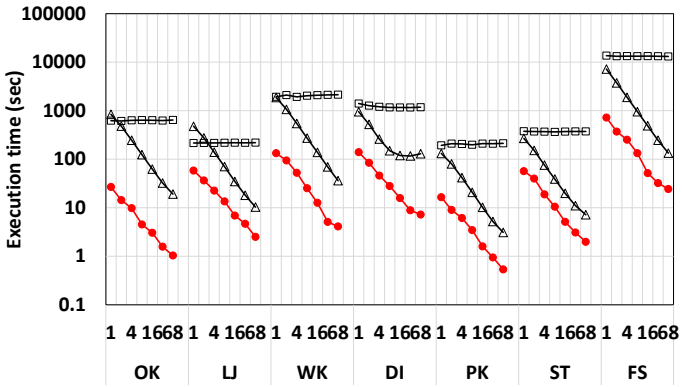

(a) Connected Component



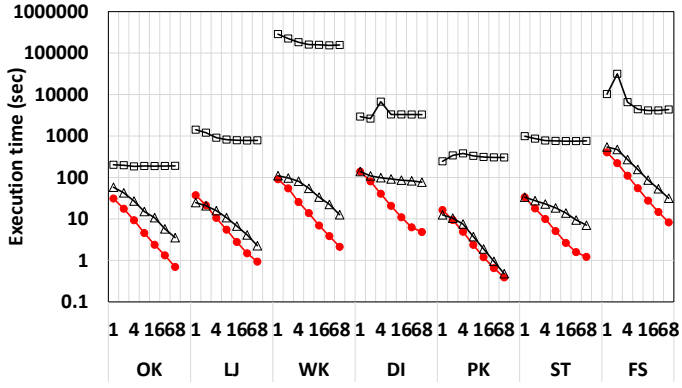

(b) PageRank




(c) Widest Path




(d) TunkRank




(e) Single Source Shortest Path

Figure 1: Intra-machine scalability (1-68 cores) of *SLFE*, GraphChi [5], and Ligra [7] on a single-machine setup.

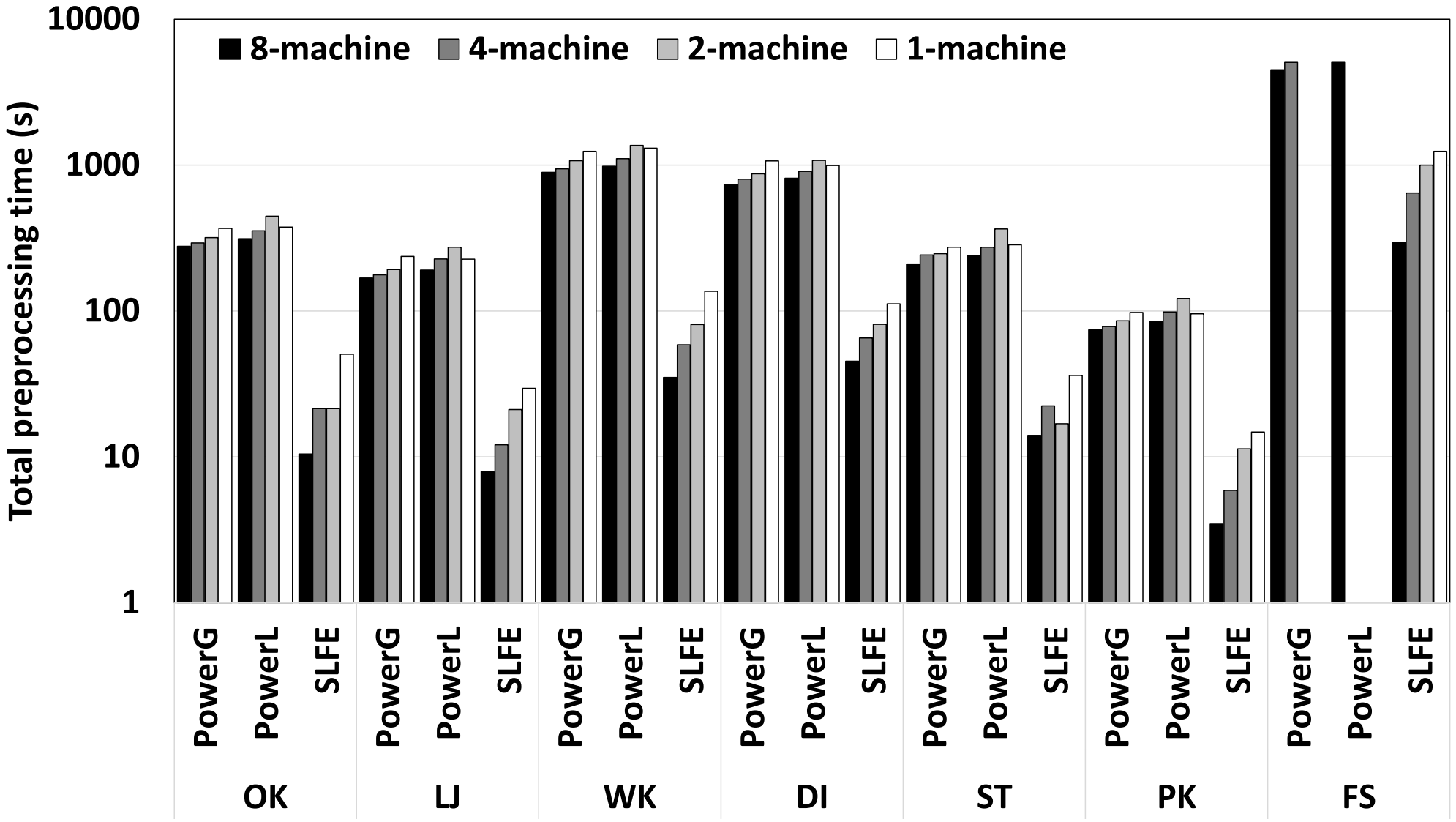



Figure 2: Total preprocessing time for seven real graphs. The preprocessing time consists of loading time, partitioning time (Random partitioner for PowerGraph [2], Ginger partitioner for PowerLyra [1], and Chunking partitioner for *SLFE* and Gemini [8]), formatting time . **Note1:** *SLFE* and Gemini utilize the same preprocessing methodology except for the Redundancy Reduction Guidance (RRG) generation, so the group of bars in the figure labeled by "SLFE" apply for Gemini as well. **Note2:** missing bars of Friendster are due to the failure of exceeding memory capacity.

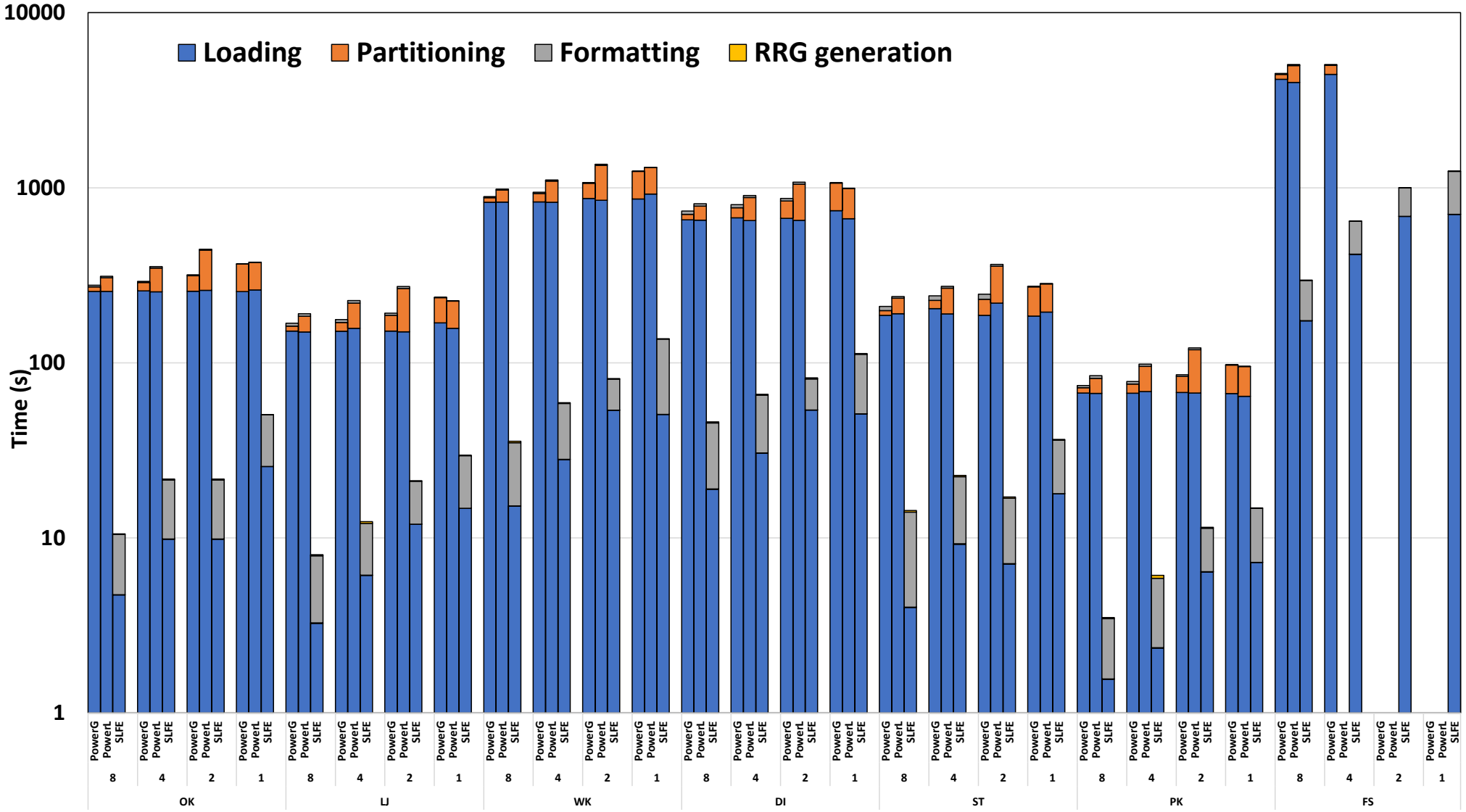
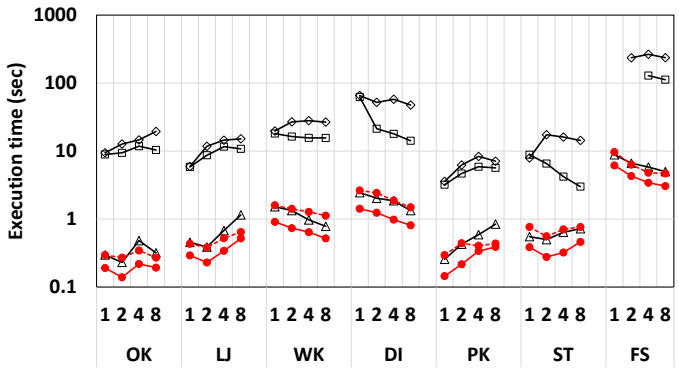
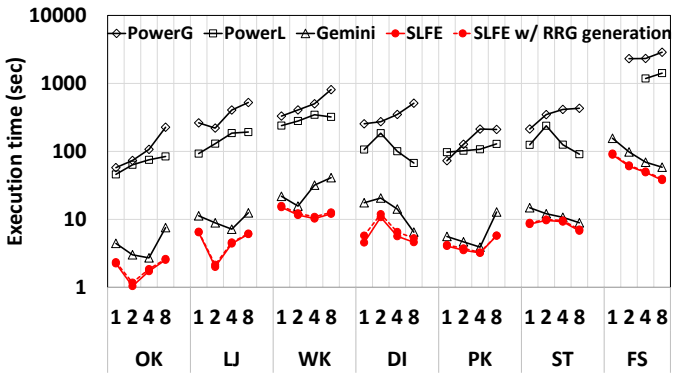


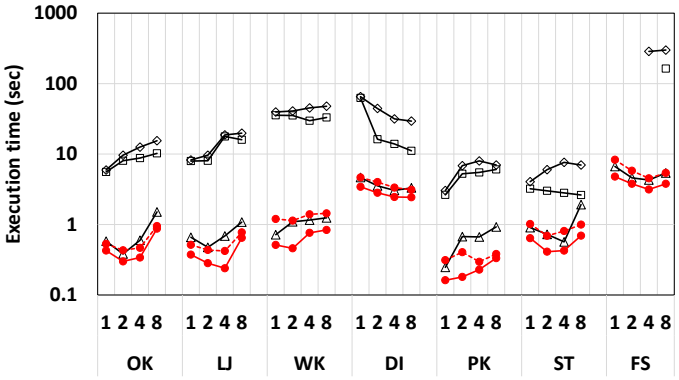

Figure 3: Breakdown of preprocessing time for seven real graphs, into loading time, partitioning time (Random partitioner for PowerGraph [2], Ginger partitioner for PowerLyra [1], and Chunking partitioner for *SLFE* and Gemini [8]), formatting time, and the Redundancy Reduction Guidance (RRG) generation time (only *SLFE* has but so little that need to zoom in to observe). **Note1:** *SLFE* and Gemini utilize the same preprocessing methodology except for the Redundancy Reduction Guidance (RRG) generation, so the group of bars in the figure labeled by "SLFE" apply for Gemini as well. **Note2:** missing bars of Friendster are due to the failure of exceeding memory capacity.



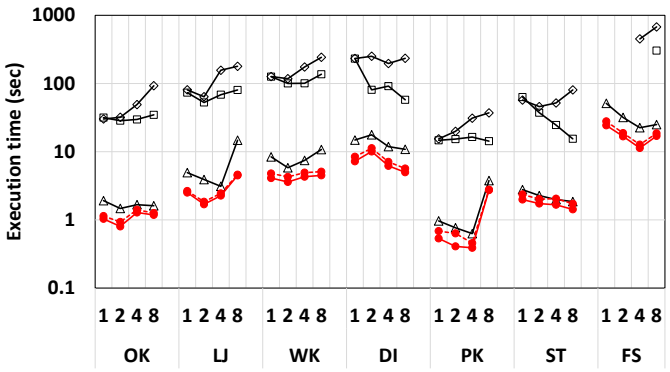

(a) Connected Component



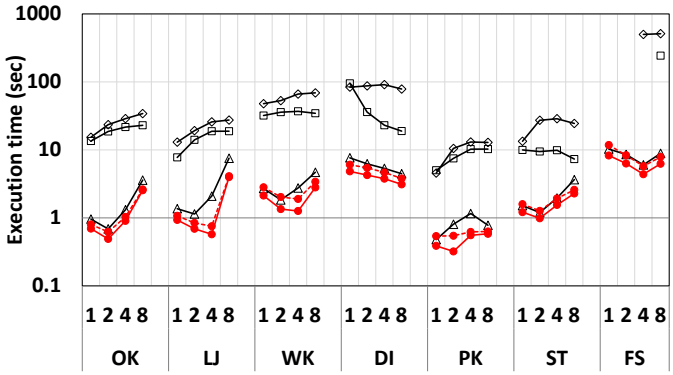

(b) PageRank




(c) Widest Path

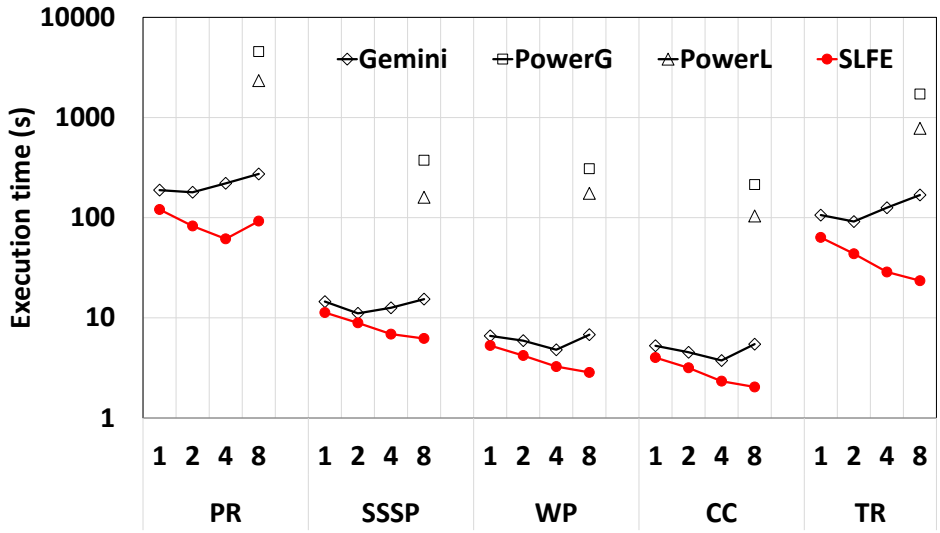



(d) TunkRank

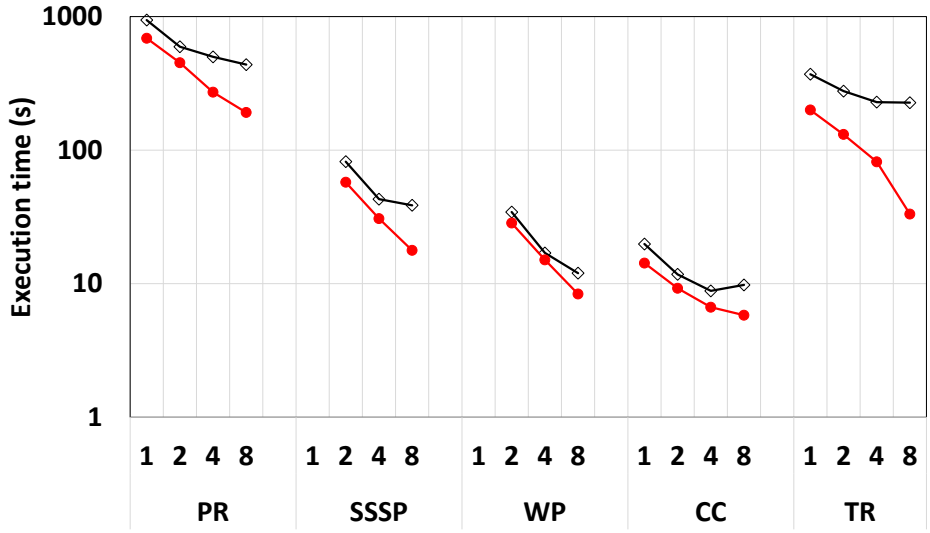



(e) Single Source Shortest Path

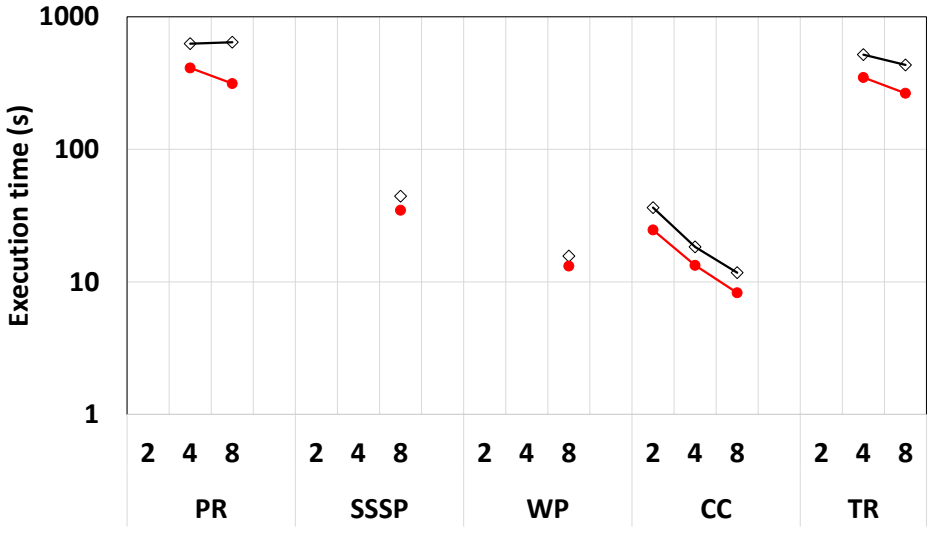
Figure 4: Inter-machine scalability (1-8 machines) of PowerGraph [2], PowerLyra [1], Gemini [8] and *SLFE* on seven real graphs. Execution time does not include preprocessing time, and *SLFE* has two series, one includes the Redundancy Reduction Guidance (RRG) generation overhead, another not. **Note:** missing points are due to the failure of execution.




(a) RMAT 1

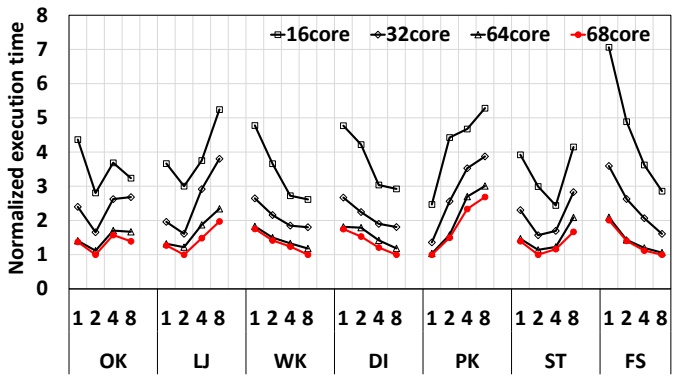



(b) RMAT 2

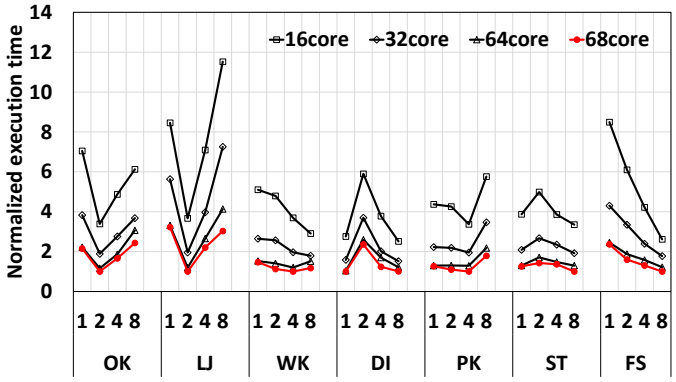



(c) RMAT 3

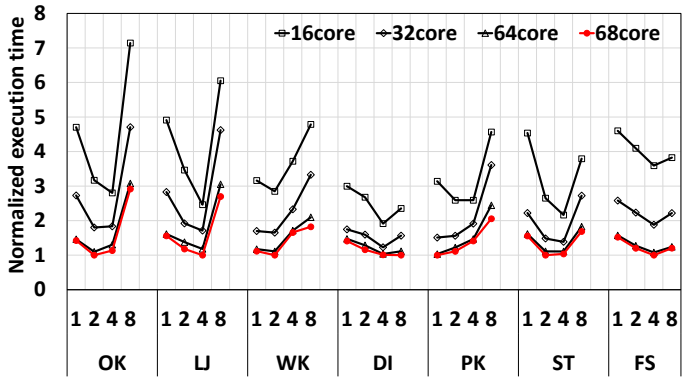
Figure 5: Inter-machine scalability (1-8 machines) of PowerGraph [2], PowerLyra [1], Gemini [8] and *SLFE* on three synthetic RMAT graphs. Execution time does not include preprocessing time. **Note:** missing points are due to the failure of exceeding memory capacity.



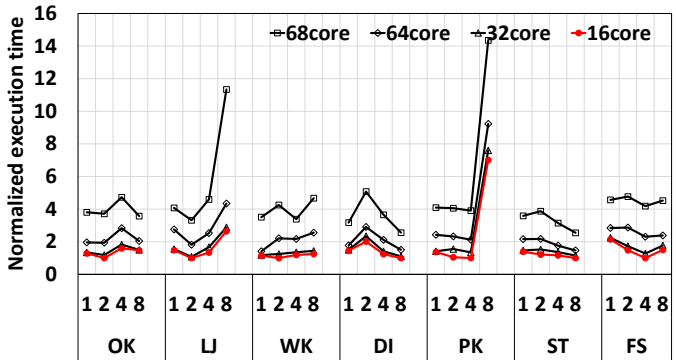

(a) Connected Component



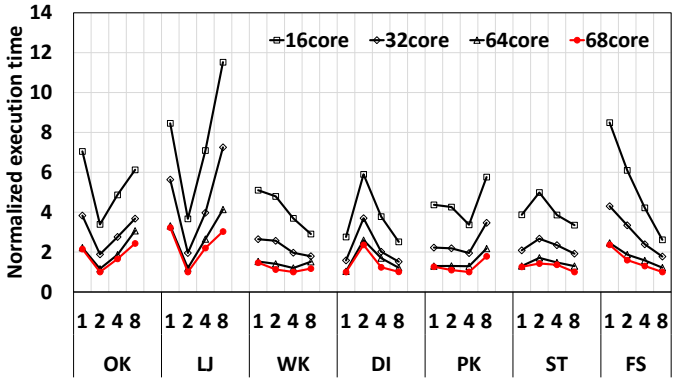

(b) PageRank




(c) Widest Path

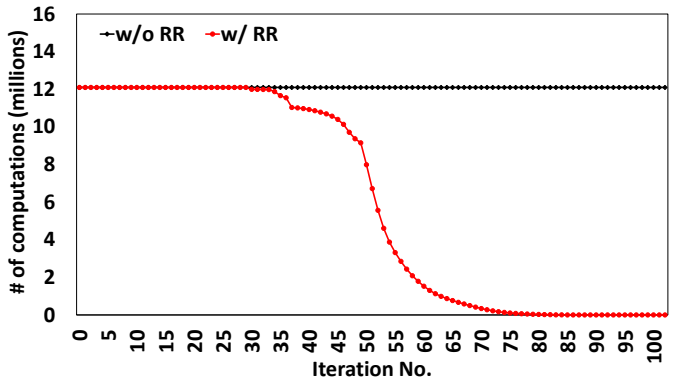



(d) TunkRank

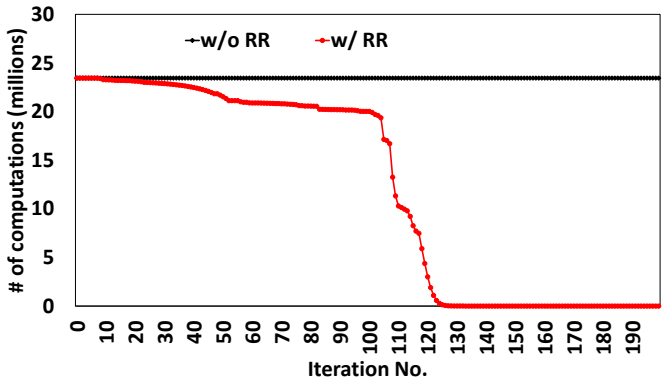



(e) Single Source Shortest Path

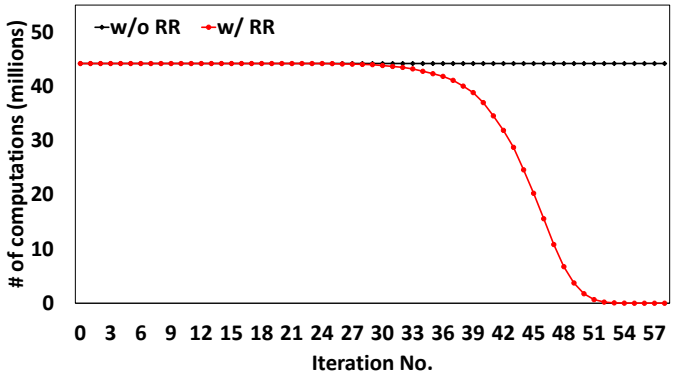
Figure 6: Trend-line analysis of *SLFE* (1-8 machines with 16, 32, 64, and 68 cores per machine) on seven real graphs.



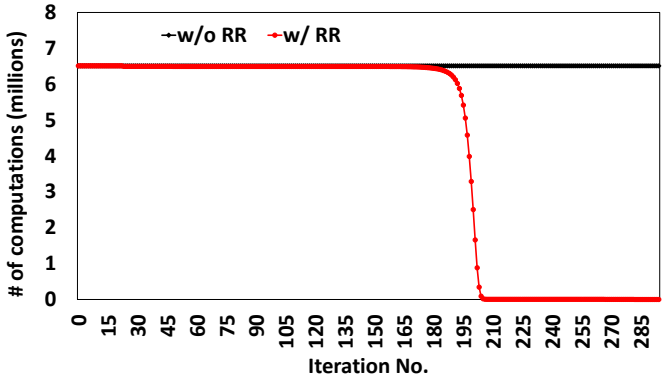

(a) Orkut



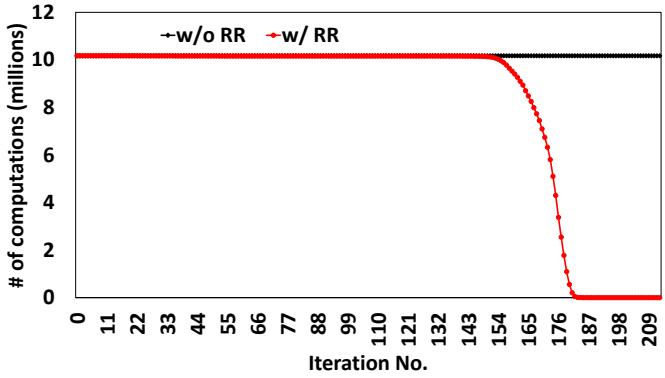

(b) Wiki




(c) Delitags

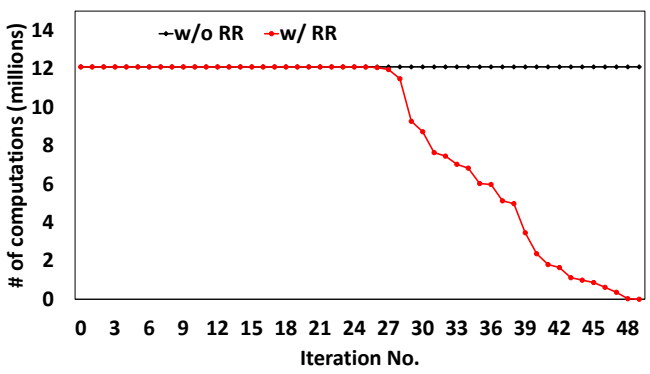



(d) Pokec

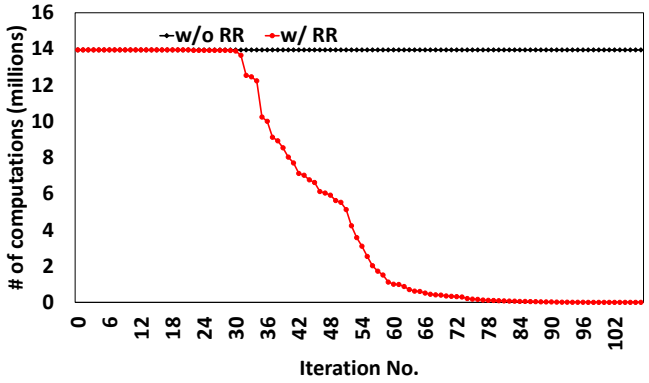



(e) Twitter

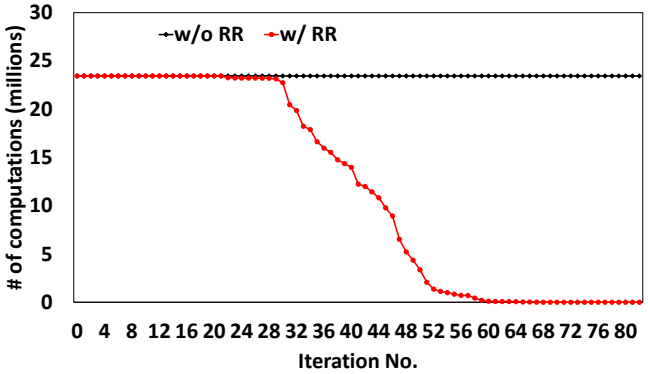
Figure 7: Dynamic counts of computations in each iteration of PageRank (charts for Liverjournal and Friendster are in the paper). The gap between those two curves indicates the amount of computations reduced by our Redundancy Reduction (RR) technique.



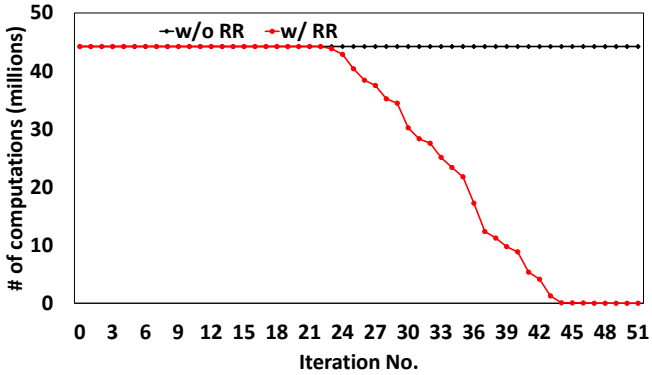

(a) Orkut



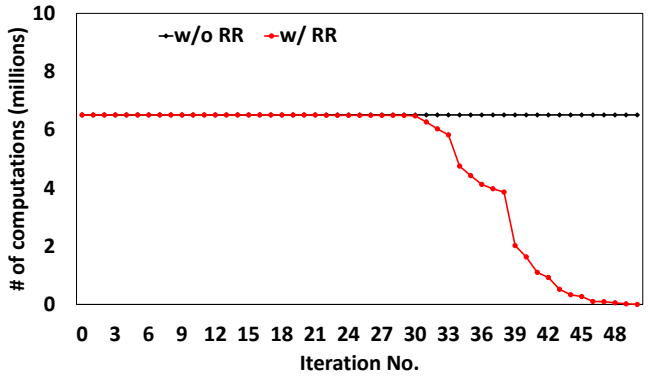

(b) Livejournal



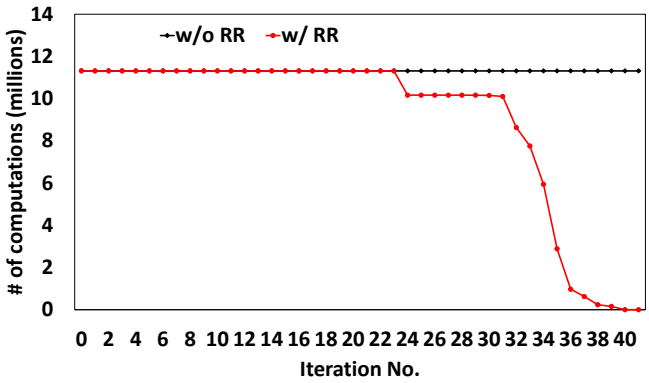

(c) Wiki



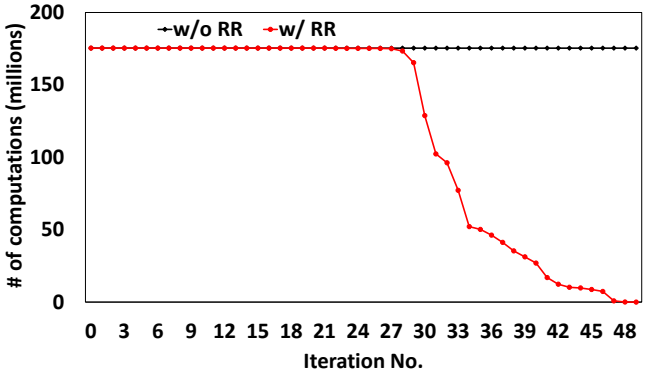

(d) Delitags




(e) Pokec

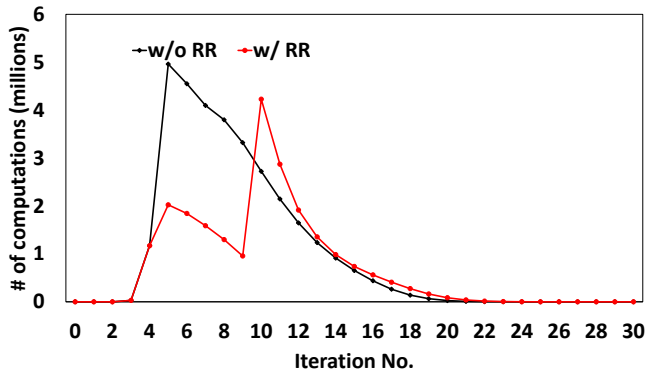



(f) Twitter

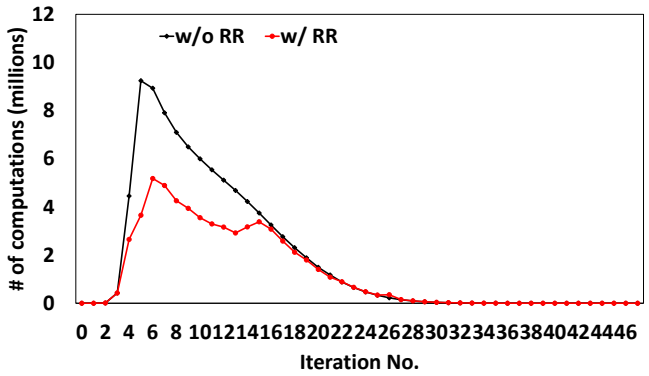



(g) Friendster

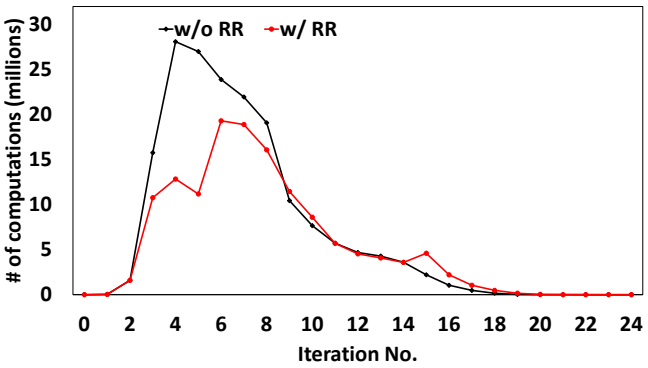
Figure 8: Dynamic counts of computations in each iteration of TunkRank. The gap between those two curves indicates the amount of computations reduced by our Redundancy Reduction (RR) technique.



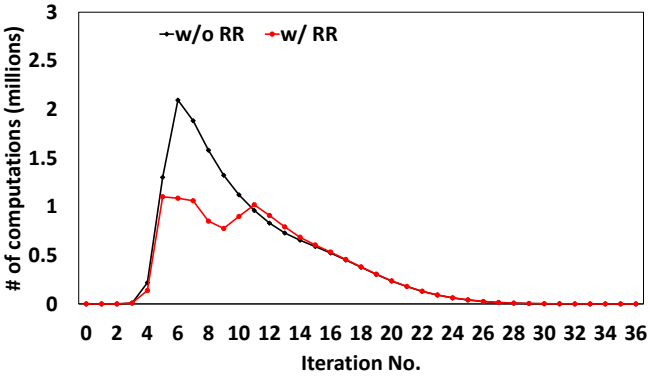

(a) Orkut



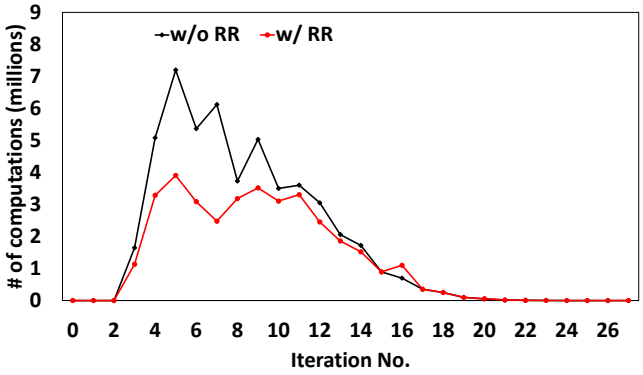

(b) Wiki




(c) Delitags

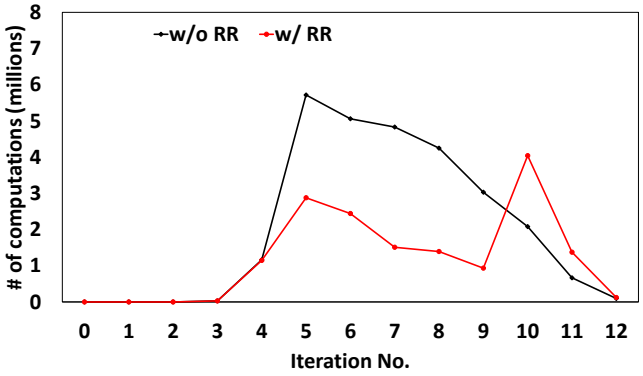



(d) Pokec

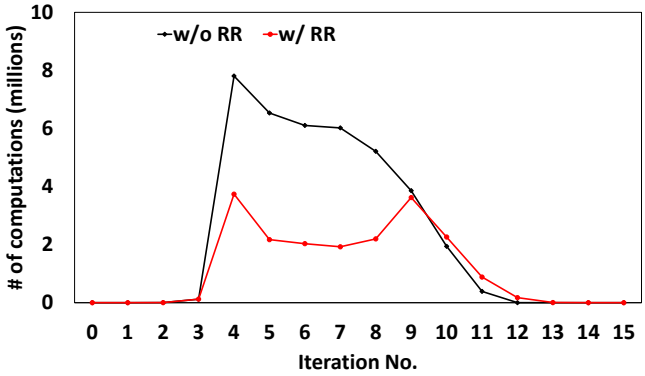



(e) Twitter

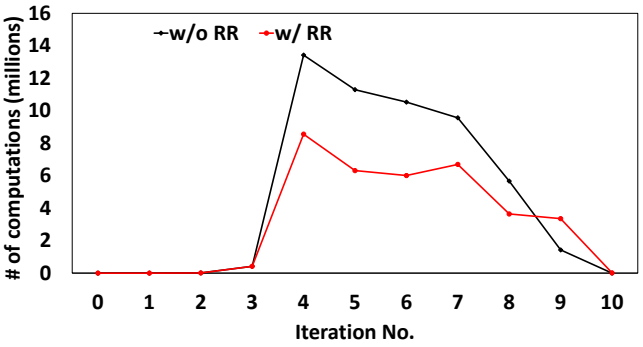
Figure 9: Dynamic counts of computations in each iteration of Single Source Shortest Path (charts for Liverjournal and Friendster are in the paper). The gap between those two curves indicates the amount of computations reduced by our Redundancy Reduction (RR) technique.



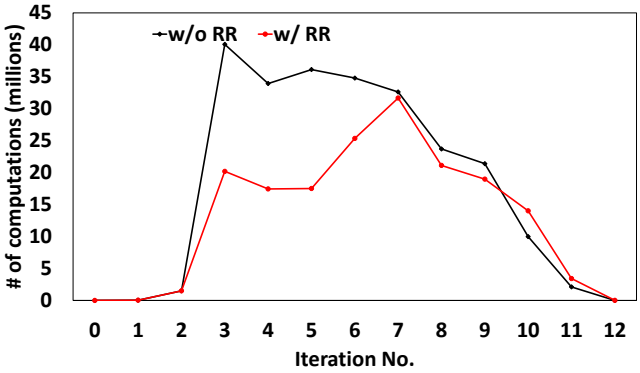

(a) Orkut



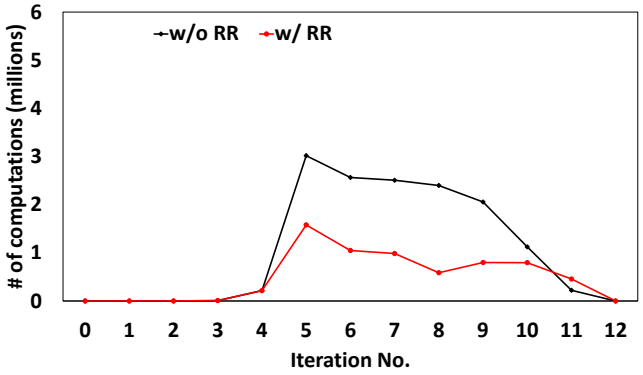

(b) Livejournal



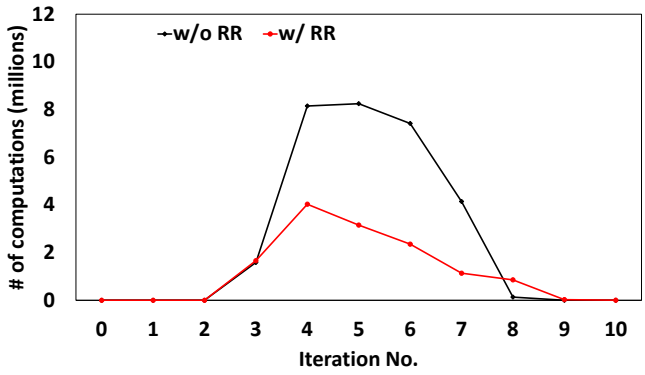

(c) Wiki



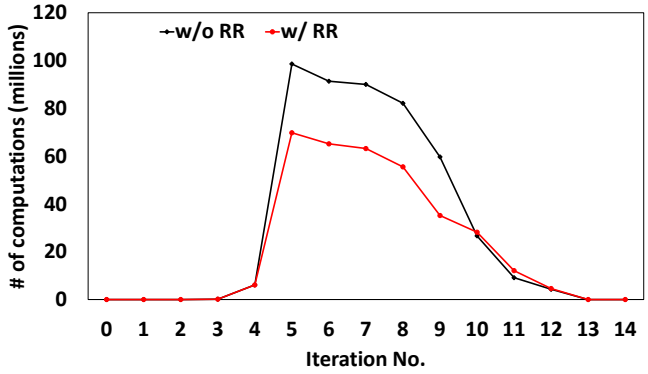

(d) Delitags




(e) Pokec

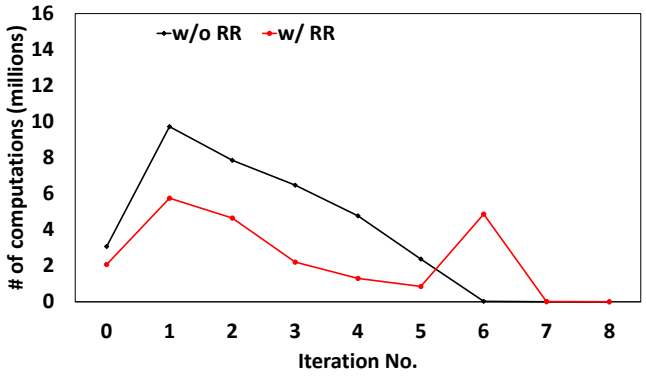



(f) Twitter

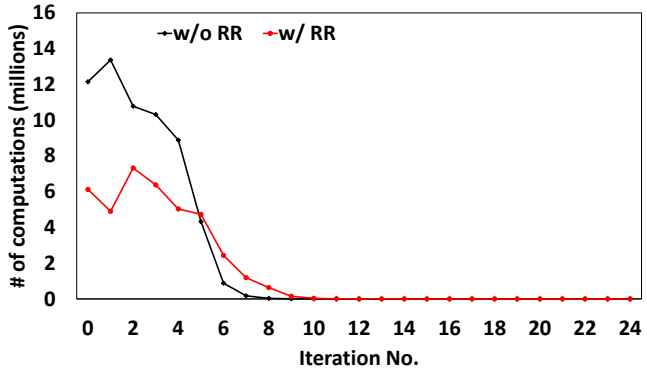



(g) Friendster

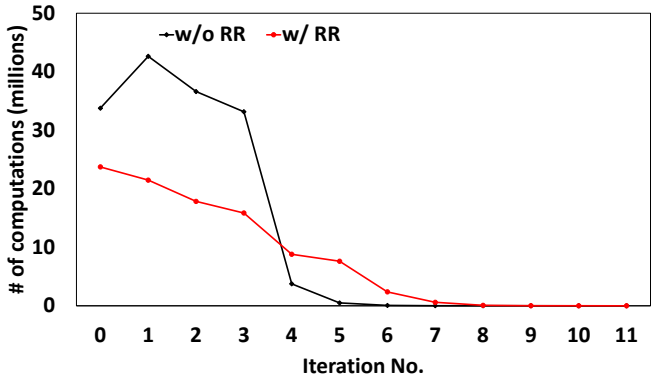
Figure 10: Dynamic counts of computations in each iteration of Widest Path. The gap between those two curves indicates the amount of computations reduced by our Redundancy Reduction (RR) technique.



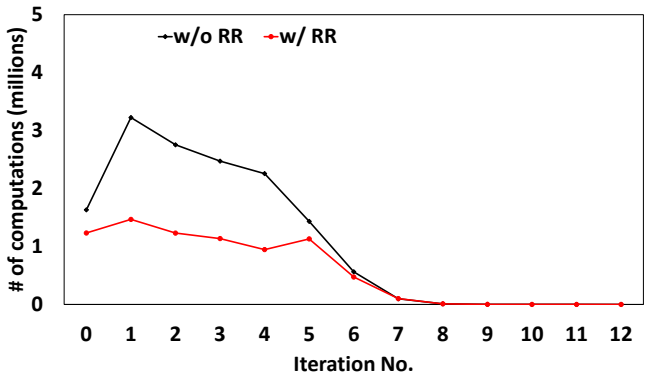

(a) Orkut



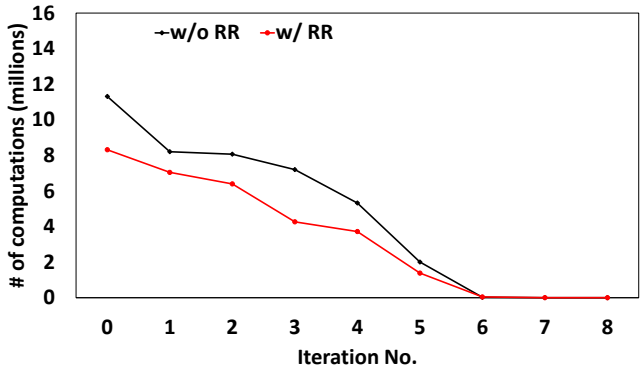

(b) Wiki




(c) Delitags




(d) Pokec




(e) Twitter

Figure 11: Dynamic counts of computations in each iteration of Connected Component (charts for Liverjournal and Friendster are in the paper). The gap between those two curves indicates the amount of computations reduced by our Redundancy Reduction (RR) technique.

Table 2: Memory footprint of *SLFE*, PowerLyra [1], PowerGraph [2], and Gemini [8] on seven real graphs. "-" denotes the execution failure due to exceeding memory capacity.

| | | Orkut(OK) | Livejournal(LJ) | Wiki(WK) | Delitags(DI) | Pokec(PK) | Twitter(ST) | Friendster(FS) |
|---|---|---|---|---|---|---|---|---|
| SLFE | 1-machine(GB) | 1.57 | 1.16 | 4.98 | 5.18 | 0.51 | 1.67 | 26.42 |
| | 8-machine(GB) | 3.36 | 3.57 | 8.41 | 8.66 | 2.1 | 4.5 | 43.82 |
| PowerL | 1-machine(GB) | 15.33 | 9.19 | 45.61 | 38.91 | 4 | 11.37 | - |
| | 8-machine(GB) | 22.18 | 16.06 | 63.14 | 61.7 | 6.76 | 17.25 | 305.38 |
| PowerG | 1-machine(GB) | 10.58 | 6.55 | 33.96 | 29.72 | 2.87 | 8.5 | - |
| | 8-machine(GB) | 13.05 | 11.24 | 39.17 | 46.47 | 5.26 | 14.85 | 203.96 |
| Gemini | 1-machine(GB) | 1.54 | 1.14 | 4.9 | 5 | 0.5 | 1.61 | 25.78 |
| | 8-machine(GB) | 3.24 | 3.38 | 7.9 | 7.31 | 2.03 | 4.04 | 38.8 |

Table 3: Redundancy Reduction Guidance (RRG) disk space overhead on seven real graphs.

| | Orkut(OK) | Livejournal(LJ) | Wiki(WK) | Delitags(DI) | Pokec(PK) | Twitter(ST) | Friendster(FS) | GMean |
|---|---|---|---|---|---|---|---|---|
| RRG disk space | 0.035 | 0.052 | 0.09 | 0.43 | 0.017 | 0.08 | 0.85 | |
| RRG/total disk space generated from preprocessing on 1-machine | 1.21% | 2.75% | 0.94% | 4.86% | 2.09% | 3.1% | 1.82% | 2.1% |
| RRG/total disk space generated from preprocessing on 8-machine | 1.07% | 2.06% | 0.8% | 3.24% | 1.66% | 1.97% | 1.54% | 1.62% |

# References


[1] R. Chen, J. Shi, Y. Chen, and H. Chen. Powerlyra: Differentiated graph computation and partitioning on skewed graphs. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, pages 1:1–1:15, New York, NY, USA, 2015. ACM.

[2] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin. Powergraph: Distributed graph-parallel computation on natural graphs. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 17–30, Hollywood, CA, 2012. USENIX.

[3] F. Khorasani, R. Gupta, and L. N. Bhuyan. Scalable simd-efficient graph processing on gpus. In *Proceedings of the 24th International Conference on Parallel Architectures and Compilation Techniques*, PACT '15, pages 39–50, 2015.

[4] J. Kunegis. Konect: The koblenz network collection. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, pages 1343–1350, New York, NY, USA, 2013. ACM.

[5] A. Kyrola, G. Blelloch, and C. Guestrin. Graphchi: Large-scale graph computation on just a pc. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 31–46, Hollywood, CA, 2012. USENIX.

[6] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[7] J. Shun and G. E. Blelloch. Ligra: A lightweight graph processing framework for shared memory. In *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '13, pages 135–146, New York, NY, USA, 2013. ACM.

[8] X. Zhu, W. Chen, W. Zheng, and X. Ma. Gemini: A computation-centric distributed graph processing system. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 301–316, GA, 2016. USENIX Association.