INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

ICPP / 2021 / CHICAGO / USA

acm In-Cooperation

sighpc

AUGUST 9-12, 2021

An Edge-Fencing Strategy for Optimizing SSSP Computations on Large-Scale Graphs

Huashan YU

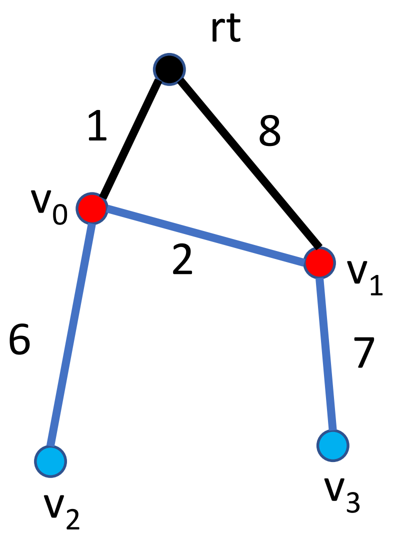
Peking University

Outline

- Edge-fencing strategy: enabling SSSP algorithms to schedule edge relaxations in a **path-centric** manner
  - Graph computations are traditionally implemented in a vertex-centric or edge-centric manner
- Skipping edges by scheduling edge relaxations according to lengths of the constructed paths
- The length distribution of a SSSP tree's shortest paths is correlated with the graph's degree distribution
- Customizing the schedule for skewed graphs
- Experimental results

SSSP computations

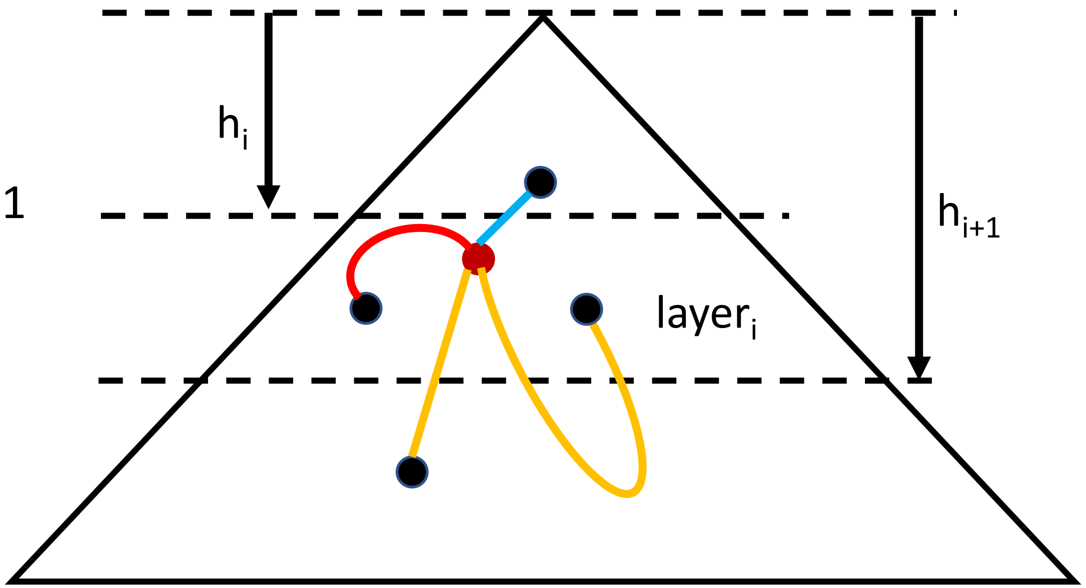
- Mature relaxation: the extended path is a shortest path
  - The relaxations on the edges incident to v_0 after the edges incident to rt have been relaxed
- Premature relaxation: the extended path is not a shortest path.
  - The relaxations on the edges incident to v_1 before its distance to rt has been updated to be 3
- Inward relaxation: the relaxed edge is not in the SSSP tree
  - The relaxation on the edge between rt and v_1



The objective is to reduce inward relaxations and premature relaxations

Scheduling edge relaxations according to lengths of the constructed paths

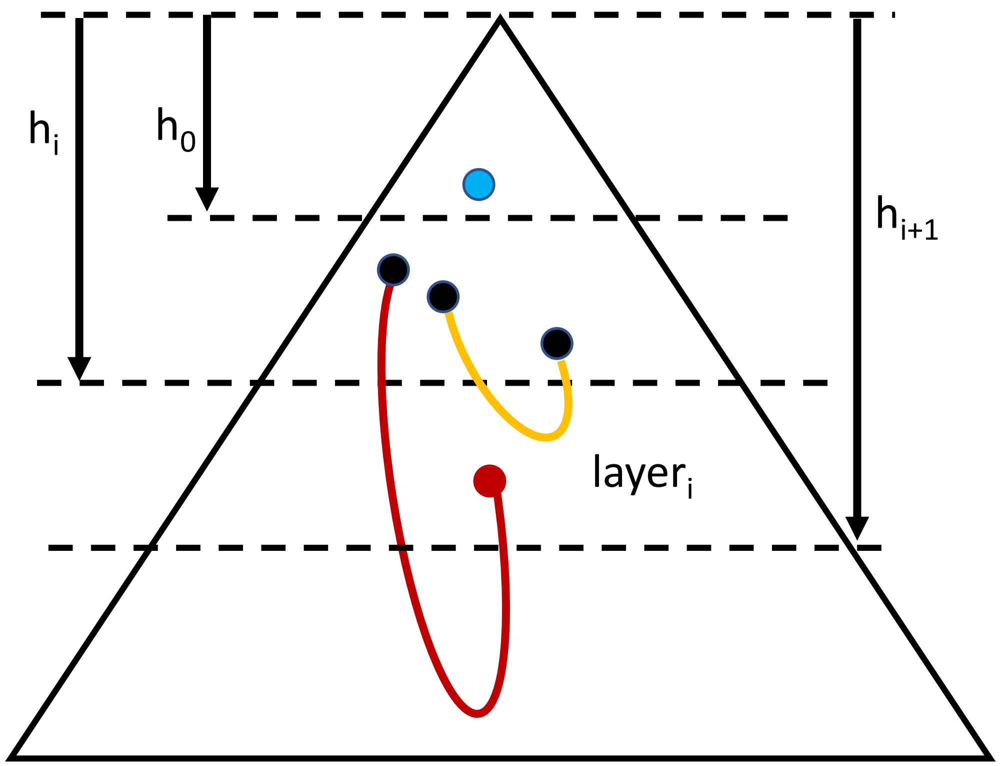
- The SSSP tree is divided into layers
- The paths with lengths greater than h_{i+1} are not allowed be extended before every vertex in $layer_i$ has been settled
- In the right figure, the blue edge, the red edge and the yellow edges are relaxed in different steps
  - The blue edge and the yellow edge are relaxed with mature relaxations
  - Premature relaxations may be executed on the red edges



Each h_i is called a fence value

Skipping inward edge relaxations

- Using the push model to relax every edge incident to vertices whose distances are less than h_0
- If i is less then k, using the push model to relax edges between $layer_i$ and vertices with the distances less than h_i
- If i is no less then k, using the pull model to relax edges between $layer_i$ and vertices with the distances less than h_i
  - The yellow edge is skipped since the two connected vertices have been settled
  - The red edge is skipped since it is too long to be in the shortest path



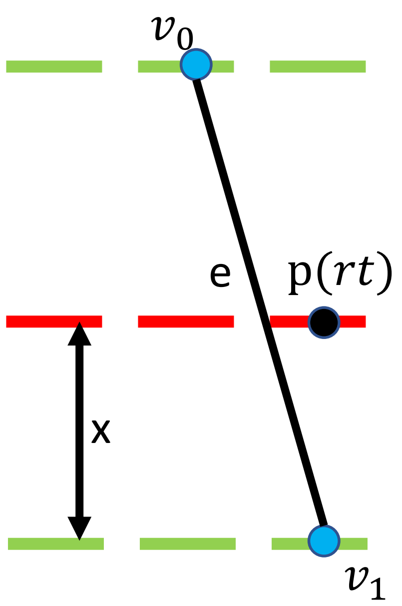
Characteristics of large-scale graphs

- Skewness in degree distribution
  - Many real networks evolve with the generic mechanism of preferential attachment
  - The edges tends to increase more quickly than the vertices
- Randomness in nature
  - The weights are vertex-independent
  - Every vertex selects its neighbors independently

G=<V E w> is assumed to be undirected, connected and skewed enough, $w(e)$ denote the edge e‘s weight and is a random value that is uniformly distributed in (0 1)

The correlation between a vertex's degree and its distance to the source vertex

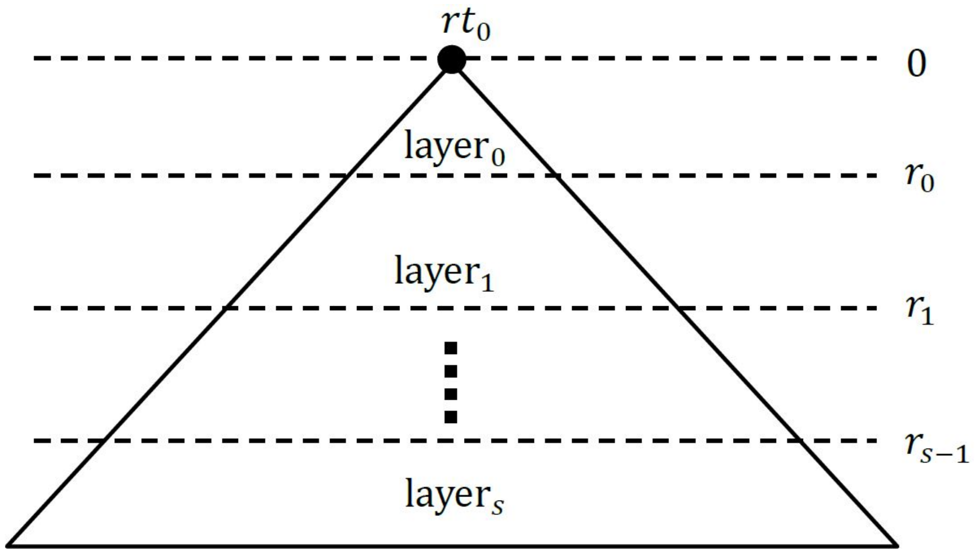
- The probability that $x \geq \text{dist}(rt,\ v_1) - \text{dist}(rt,\ \text{p}(rt))$ is about $(1-(1-x)^{\deg(v_1)\div 2}) \times (1-(1/2)^{\deg(v_1)})$
- $\text{p}(rt)$ is a vertex that both sum$\{\deg(u) : \text{dist}(rt,\ u) < \text{dist}(rt,\ \text{p}(rt))\}$ and sum$\{\deg(u) : \text{dist}(rt,\ u) > \text{dist}(rt,\ \text{p}(rt))\}$ are less than m, which is the number of graph edges
- The probability that v_1 is a neighbor of $\{u{:}\ \text{dist}(rt,\ u) \leq \text{dist}(rt,\ \text{p}(rt))\}$ is at least $1-(1/2)^{\deg(v_1)}$
  - $\text{dist}(rt,\ v_1) - \text{dist}(rt,\ \text{p}(rt))$ is less than w(e)
- The probability that w(e) is greater than x: 1 – x, assuming x is a positive less than 1



Most high-degree vertices have similar distances to the source vertex

A hierarchical graph model

- rt_0 denote the vertex that minimizes $\text{average}\{\text{dist}_f(rt_0, u): u \in V_c\}$, where V_c consists of the graph's high-degree vertices.
- The distance that minimizes $e^{r_0}/\log(\text{sum}\{\deg(u): \text{dist}_f(rt_0, u) \le r_0\})$ is selected as r_0
- r_i $(1 \le i < s)$ is derived from r_{i-1} with $\text{average}\{\deg(u): \text{dist}_f(rt_0, u) = r_i\} \times \Phi = \text{average}\{\deg(u): \text{dist}_f(rt_0, u) > r_{i-1}\} \ge 2$
  - Φ is a constant less than 1



Customizing the fence values for every input graph and every source vertex

(1) $h_1 = \min\{\text{dist}_f(rt, u): u \in \text{layer}_0\}$

(2) $h_i = h_1 + r_{i-1}$

Most high-degree vertices are in the top layers

Approximating the hierarchical graph model

- Approach I: deriving from the incident edges from some high-degree vertices
  - 4 layers
  - Sorting the vertex's incident edges according to their weights
  - r_1 is x_i that minimizes $e^{x_i}/\log i$, r_0 is $r_1 \times (1-\Phi)^2$ and r_2 is $\min\{1, r_1 \div (1-\Phi)\}$, where x_i denote weight of the i_{th} edge
- Approach I: deriving from some SSSP tree in the graph

INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

acm In-Cooperation sighpc

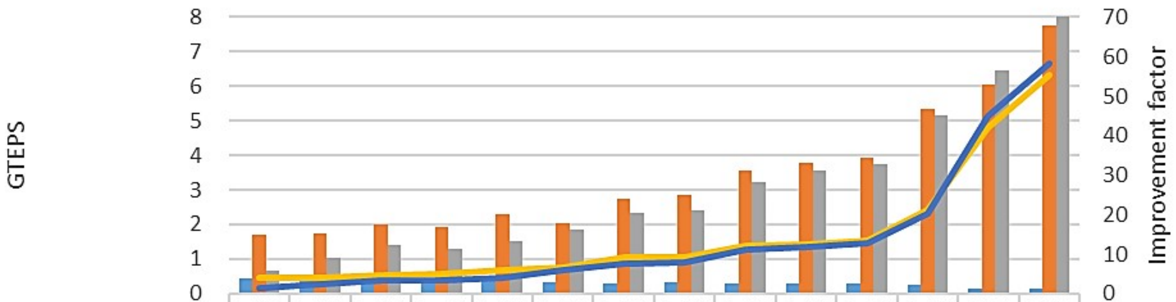
Evaluation

- Experimental environment: a Supermicro SYS-2049U-TR4 server running CentOS Linux release 7.8.2003, 64 cores (4 Intel(R) Xeon(R) Gold 5218 CPUs @ 2.30GHz) and 1536 GB of memory.
- Implementation: as the customized SSSP algorithm of Graph500 3.0
  - To compare with the Δ-stepping algorithm's implementation in Graph500
  - Exploiting the graph generator in Graph500 to generate weighted R-MAT graphs
  - Two versions were implemented for evaluating our algorithm's sensitivity to the hierarchical graph model's accuracy

Evaluation: datasets

- 14 synthetic graphs generated with the Kronecker graph generator in Graph500 3.0
  - Edgefactor = 16 and scale = 26
  - kr*x*_*y* denote the synthetic graph whose R-MAT model is configured with a=100*x* and b=c=100*y*
- 8 real graphs: selected from the publicly published networks, including social networks, Internets, Web networks and biological networks.

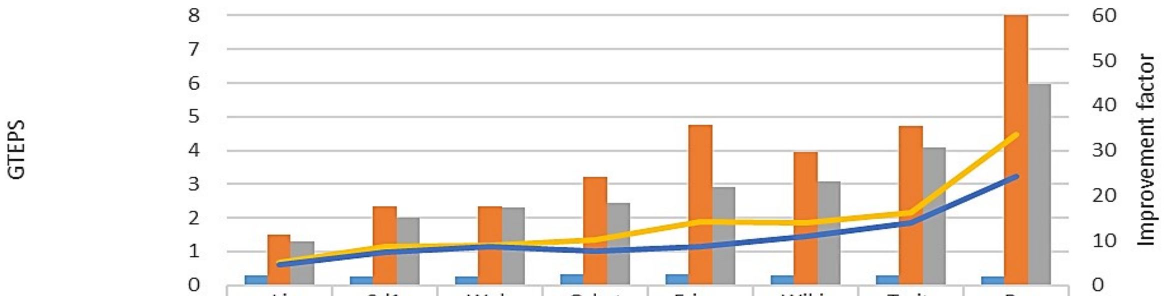
Evaluation: Parallel Performance



	kr25_25	kr30_25	kr35_25	kr45_15	kr40_25	kr55_10	kr55_15	kr50_20	kr60_15	kr50_25	kr57_19	kr65_15	kr70_15	kr75_10
Δ-st. GTEPS	0.44	0.44	0.42	0.40	0.39	0.31	0.30	0.31	0.29	0.30	0.29	0.25	0.144	0.140
T-based GTEPS	1.70	1.75	2.00	1.93	2.31	2.04	2.73	2.84	3.56	3.78	3.94	5.36	6.029	7.744
E-based GTEPS	0.67	1.02	1.41	1.31	1.53	1.85	2.31	2.39	3.24	3.56	3.75	5.14	6.450	8.174
T-based factor	3.83	3.98	4.71	4.81	5.97	6.52	9.06	9.20	12.21	12.54	13.41	21.15	41.87	55.27
E-based factor	1.51	2.31	3.32	3.26	3.96	5.91	7.67	7.76	11.11	11.79	12.75	20.27	44.78	58.34

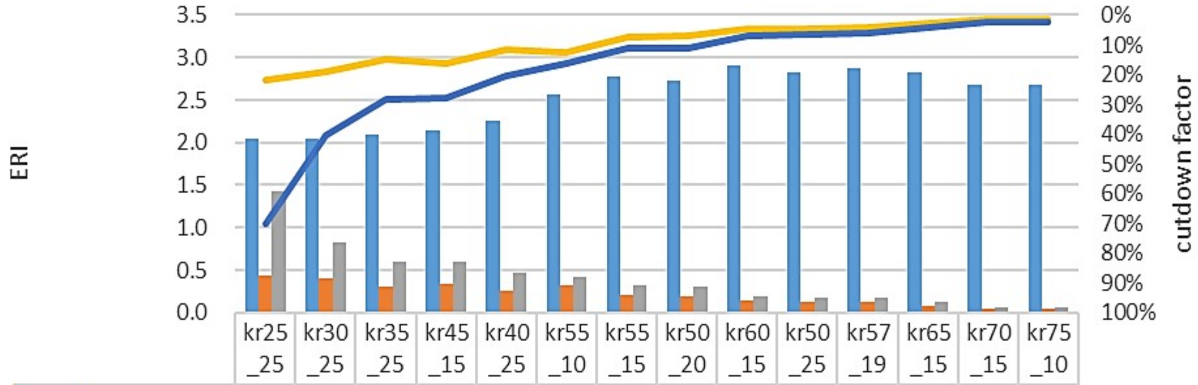
- a 3.83× - 55.27× improvement in GTEPS (Billion Edges Traversed Per Second) over the Δ-stepping algorithm's implementation in Graph500

- the performance is relatively insensitive to the hierarchical graph model's accuracy



	Lj	Sd1	Web	Orkut	Frien.	Wiki.	Twit.	Bn
Δ-st. GTEPS	0.288	0.271	0.264	0.319	0.333	0.286	0.294	0.245
T-based GTEPS	1.494	2.329	2.353	3.219	4.743	3.951	4.737	8.233
E-based GTEPS	1.310	2.006	2.290	2.440	2.903	3.081	4.070	5.951
T-based factor	5.182	8.608	8.907	10.087	14.235	13.792	16.137	33.542
E-based factor	4.541	7.414	8.669	7.646	8.711	10.755	13.865	24.246

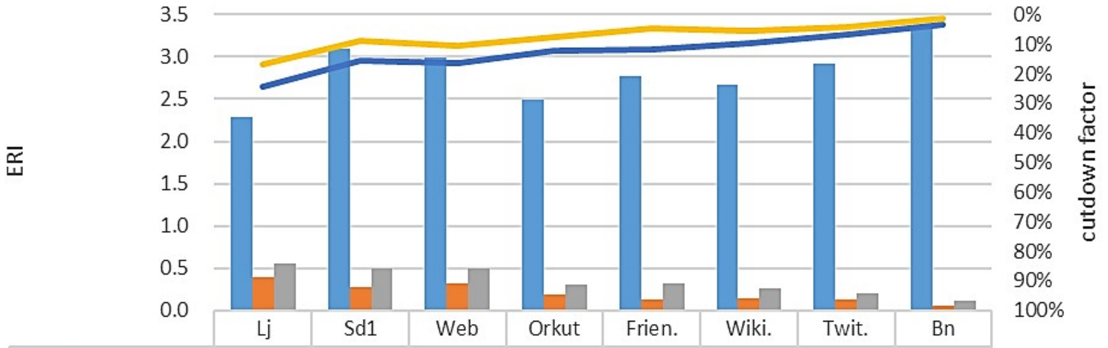
Evaluation: Efficiency



	kr25_25	kr30_25	kr35_25	kr45_15	kr40_25	kr55_10	kr55_15	kr50_20	kr60_15	kr50_25	kr57_19	kr65_15	kr70_15	kr75_10
Δ-st. ERI	2.04	2.05	2.09	2.14	2.26	2.57	2.78	2.73	2.90	2.82	2.87	2.83	2.68	2.68
T-based ERI	0.44	0.39	0.31	0.35	0.26	0.32	0.21	0.19	0.14	0.13	0.12	0.08	0.04	0.04
E-based ERI	1.43	0.82	0.59	0.59	0.47	0.42	0.31	0.31	0.20	0.18	0.17	0.12	0.06	0.06
T-based factor	0.218	0.193	0.148	0.161	0.116	0.126	0.075	0.071	0.047	0.046	0.042	0.028	0.016	0.016
E-based factor	0.704	0.402	0.282	0.277	0.207	0.163	0.113	0.112	0.068	0.064	0.059	0.042	0.024	0.024

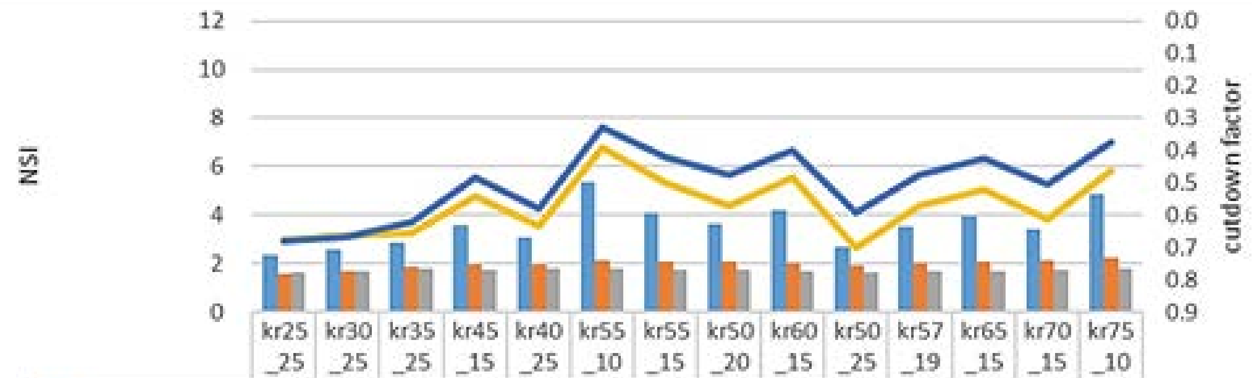
- the edge relaxations are reduced to 0.04× - 0.44× graph edges

- **Edge relaxation intensity** (ERI): average number of relaxations on each graph edge



	Lj	Sd1	Web	Orkut	Frien.	Wiki.	Twit.	Bn
Δ-st. ERI	2.29	3.09	3.00	2.49	2.77	2.67	2.92	3.36
T-based ERI	0.39	0.28	0.31	0.19	0.13	0.15	0.13	0.05
E-based ERI	0.55	0.49	0.49	0.30	0.32	0.26	0.19	0.11
T-based factor	0.170	0.089	0.104	0.075	0.046	0.055	0.044	0.014
E-based factor	0.242	0.158	0.164	0.121	0.117	0.096	0.067	0.034

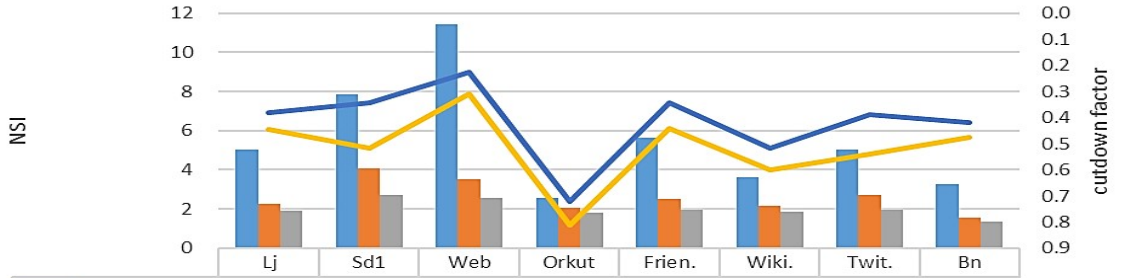
Evaluation: parallelism



	kr25_25	kr30_25	kr35_25	kr45_15	kr40_25	kr55_10	kr55_15	kr50_20	kr60_15	kr50_25	kr57_19	kr65_15	kr70_15	kr75_10
Δ-st. NSI	2.35	2.54	2.83	3.59	3.08	5.35	4.07	3.62	4.19	2.69	3.50	3.98	3.43	4.83
T-based NSI	1.58	1.69	1.86	1.95	1.96	2.10	2.04	2.07	2.02	1.89	2.00	2.07	2.10	2.24
E-based NSI	1.60	1.70	1.76	1.73	1.79	1.77	1.72	1.72	1.68	1.59	1.67	1.69	1.73	1.81
T-based factor	0.68	0.66	0.66	0.54	0.63	0.39	0.50	0.57	0.48	0.70	0.57	0.52	0.61	0.46
E-based factor	0.68	0.67	0.62	0.48	0.58	0.33	0.42	0.47	0.40	0.59	0.48	0.42	0.50	0.38

- the bulk synchronizations required in parallel settings are 1.55× - 4.06× maximum of edges in one shortest path created during the computation

- **Normalized synchronization intensity** (NSI): average phases for the SSSP tree's every hop-level



	Lj	Sd1	Web	Orkut	Frien.	Wiki.	Twit.	Bn
Δ-st. NSI	5.04	7.84	11.41	2.54	5.66	3.64	5.04	3.28
T-based NSI	2.24	4.06	3.52	2.06	2.50	2.19	2.72	1.55
E-based NSI	1.92	2.70	2.58	1.83	1.95	1.87	1.95	1.38
T-based factor	0.45	0.52	0.31	0.81	0.44	0.60	0.54	0.47
E-based factor	0.38	0.34	0.23	0.72	0.34	0.52	0.39	0.42

Thank you