

The “good edges” for maximum cardinality graph matching

Kamer Kaya¹

Sabancı University

April 15, 2016

Joint work with:

Fanny Dufossé

Inria, Lille Nord Europe
France

Bora Uçar

CNRS & ENS, Lyon
France

¹Supported by TÜBİTAK (grant #115C018)

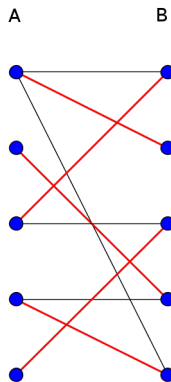
Motivation and context

- A whole family of practical, exact algorithms start with a fast heuristic to obtain a large set of initial matching.
 - Some applications are ok with a suboptimal matching (see [Lotker et al., SPAA'08], citing routers)
-
- We propose heuristics based on edge selection with non-uniform probabilities guided by parallelism and designed for parallel implementation.
 - Good theoretical and demonstrated approximation guarantee.
 - Can we use the idea for streaming graphs?

The bipartite case

Bipartite graph (A,B,E) : edges of E are between a node of set A and one of set B .

- **Matching** : set of edges with no common node
- **Perfect matching**: any node of A and B belong to the matching
→ Matrix with support
- **Total support**: any edge belong to a perfect matching

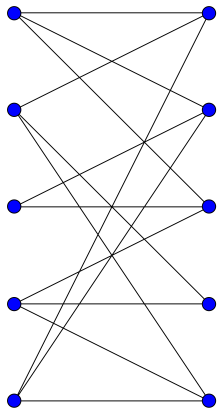


Magical property

Bipartite graph with **total support** \rightarrow doubly stochastic matrix

A

B



Matrix representation:

1	1	1	0	0
1	0	0	1	1
0	1	1	0	0
0	0	1	1	1
1	1	0	0	1

Doubly stochastic matrix:

$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0
$\frac{1}{4}$	0	0	$\frac{1}{2}$	$\frac{1}{4}$
0	$\frac{1}{2}$	$\frac{1}{2}$	0	0
0	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$
$\frac{1}{4}$	$\frac{1}{4}$	0	0	$\frac{1}{2}$

Algorithm I: naive greedy (from the literature)

```
1: for  $j = 1$  to  $n$  do  
2:    $cmatch(j) \leftarrow NIL$   
3: for  $i = 1$  to  $n$  do  
4:   Randomly pick a column  $j$  in row  $i$  (uniformly random)  
5:    $cmatch(j) \leftarrow i$ 
```

- Some columns are picked by many rows but only one of them gets to be matched.
- Some columns are not matched at all, $cmatch(\cdot)$ value remains NIL .

Algorithm I – The expected size of a matching

Assumption: all rows and columns have d nonzeros (a row picks one of its columns with $1/d$). Then the probability that the column j is not picked

$$(1 - 1/d)^d .$$

Bounded by the limit (increasing function),

$$\lim_{d \rightarrow \infty} \left(1 - \frac{1}{d}\right)^d = \frac{1}{e} .$$

The expected number of unmatched columns is less than:

$$\frac{n}{e} .$$

Therefore, the above algorithm obtains a matching of size (in expectation)

$$n \left(1 - \frac{1}{e}\right) \approx n 0.6321 .$$

Algorithm II – ONEsIDED

- The assumption that each row and column has d nonzeros is very restrictive.
- **Remedy:** Any nonnegative matrix \mathbf{A} with a **total support** can be scaled to a **doubly stochastic** matrix with two positive diagonal matrices, yielding $\mathbf{S} = \mathbf{D}_1 \mathbf{A} \mathbf{D}_2$.
- We first do a scaling and then perform matching as before.

```
1:  $\mathbf{S} \leftarrow$  doubly stochastic scaling of  $\mathbf{A}$ 
2: for  $j = 1$  to  $n$  do
3:    $cmatch(j) \leftarrow NIL$ 
4: for  $i = 1$  to  $n$  do
5:   Randomly pick column  $j$ , according to probabilities  $S_{ij}$ 
6:    $cmatch(j) \leftarrow i$ 
```

Algorithm II – The expected size of a matching

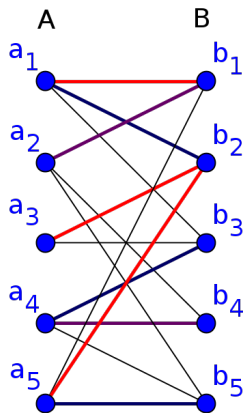
Again, the above algorithm obtains a matching of size (proof uses the arithmetic-geometric mean inequality as an additional machinery)

$$n \left(1 - \frac{1}{e} \right) \approx n \cdot 0.6321$$

in expectation as $n \rightarrow \infty$ for an $n \times n$ matrix with total support.

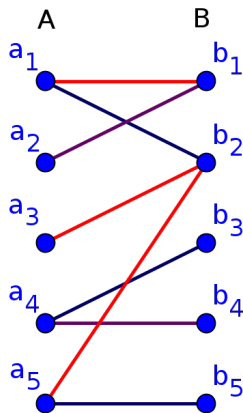
Algorithm III – TWO SIDED

- 1: $\mathbf{S} \leftarrow$ doubly stochastic scaling of \mathbf{A}
- 2: $E \leftarrow \emptyset$
- 3: **for** $i=1$ **to** n **do**
- 4: Randomly **pick a column** j , according to probabilities S_{ij}
- 5: $E \leftarrow E \cup \{(i, j)\}$
- 6: **for** $j=1$ **to** n **do**
- 7: Randomly **pick a row** i , according to probabilities S_{ij}
- 8: $E \leftarrow E \cup \{(i, j)\}$
- 9: Compute optimal matching on the edges E



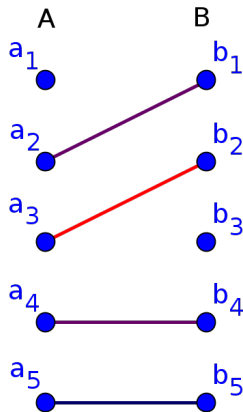
Algorithm III – TWO SIDED

- 1: $\mathbf{S} \leftarrow$ doubly stochastic scaling of \mathbf{A}
- 2: $E \leftarrow \emptyset$
- 3: **for** $i=1$ **to** n **do**
- 4: Randomly **pick a column** j , according to probabilities S_{ij}
- 5: $E \leftarrow E \cup \{(i, j)\}$
- 6: **for** $j=1$ **to** n **do**
- 7: Randomly **pick a row** i , according to probabilities S_{ij}
- 8: $E \leftarrow E \cup \{(i, j)\}$
- 9: Compute optimal matching on the edges E

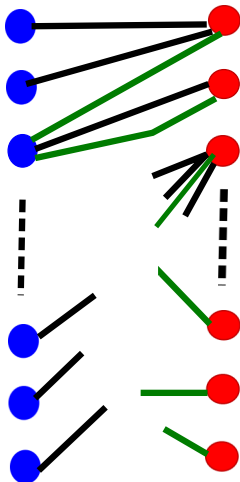


Algorithm III – TWO SIDED

- 1: $\mathbf{S} \leftarrow$ doubly stochastic scaling of \mathbf{A}
- 2: $E \leftarrow \emptyset$
- 3: **for** $i=1$ **to** n **do**
- 4: Randomly **pick a column** j , according to probabilities S_{ij}
- 5: $E \leftarrow E \cup \{(i, j)\}$
- 6: **for** $j=1$ **to** n **do**
- 7: Randomly **pick a row** i , according to probabilities S_{ij}
- 8: $E \leftarrow E \cup \{(i, j)\}$
- 9: Compute optimal matching on the edges E



Algorithm III – TWO SIDED



$2n$ edges

More complicated than before. We need an exact matching algorithm.

Any matching algorithm would do, but we can do better by taking advantage of the special structure of $2n$ edges.

- Karp-Sipser heuristic: match a degree-one vertex, if none, match a random pair.
- $O(|E|)$ time complexity, matches all but $\tilde{O}(n^{1/5})$ vertices of a random undirected graph.

KS heuristics become an exact algorithm for the type of graphs that are constructed by TWO SIDED

Parallelization on shared memory systems

Scaling algorithms: We use Sinkhorn-Knopp [’67] algorithm. Computations are similar to SpMxV; virtually any technique used in parallelizing SpMxV can be used.

Algorithm II ONESIDED:

```
1: S  $\leftarrow$  doubly stochastic scaling of A
2: for  $j = 1$  to  $n$  do
3:    $cmatch(j) \leftarrow NIL$ 
4:   for  $i = 1$  to  $n$  do
5:     Randomly pick column  $j$ ,
       according to probabilities  $S_{ij}$ 
6:      $cmatch(j) \leftarrow i$ 
```

Split the rows among the threads with a **parallel for** construct. **No synchronization or conflict detection.**

Assumption about the computer: one write survives, in case of concurrent writes to the same memory location.

Parallelization on shared memory systems

Algorithm III: TwoSided:

```
1: S  $\leftarrow$  doubly stochastic scaling of A
2:  $E \leftarrow \emptyset$ 
3: for  $i = 1$  to  $n$  do
4:   Randomly pick a column  $j$ 
5:    $E \leftarrow E \cup \{(i, j)\}$ 
6: for  $j = 1$  to  $n$  do
7:   Randomly pick a row  $i$ 
8:    $E \leftarrow E \cup \{(i, j)\}$ 
9: Karp-Sipser on the edges  $E$ .
```

With a standard Karp-Sipser, speedup is hard to achieve.

We exploit the structure of the graph :

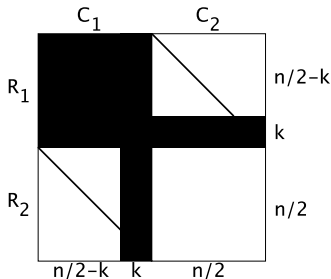
- E not kept as a regular edge set
- each connected component has at most one cycle
- if degree-one vertices are handled, we end up with cycles (any remaining $\langle i, \text{pick}[i] \rangle$ is valid along a cycle)
- if one degree-one vertex is matched, then at most one degree-one vertex is created

Theorem: Let A be an $n \times n$ matrix with total support. Then, TWO SIDED obtains a matching of size $2(1 - \Omega)n \approx 0.866n$ in expectation as $n \rightarrow \infty$.

Here Ω is the unique solution to $\Omega e^{\Omega} = 1$ also known as Lambert's $W(1)$.

Bipartite experiments I: Matching quality

- 743 matrices from UFL: 10 iterations of scaling is enough to obtain the approximation in all but 37 matrices. They needed 10 more scaling iterations.
- On these graphs can beat standard Karp-Sipser heuristics dearly.

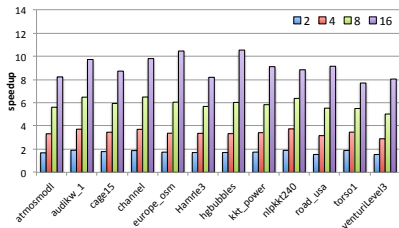


- Graphs without perfect matching: use `sprand` of Matlab (Erdős-Rényi matrices); 10 iterations of scaling were again enough to surpass the approximation guarantees.

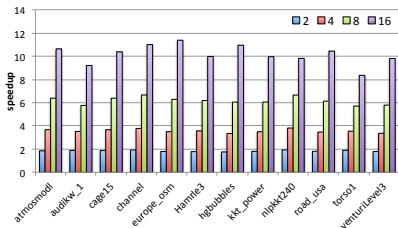
Bipartite experiments II: Running time

- Machine: two Intel [Sandybridge-EP CPUs](#) (each 8 cores) clocked at 2.00Ghz and 256GB of memory split across the two NUMA domains.
- With [2, 4, 8, 16](#) threads on a few large matrices from UFL.
- Using C and OpenMP parallelism; gcc 4.4.5 with the -O2 optimization flag; gcc atomic operations
- (dynamic, 512) OpenMP scheduling policy is employed while running all the algorithms except Karp-Sipser; it uses (guided).
- speed up values: [against a single thread execution](#) (geometric mean of 15/20 executions).
- 32 bit stores and loads are atomic (hence ONEsIDED works correctly)

Bipartite experiments II: Speed up

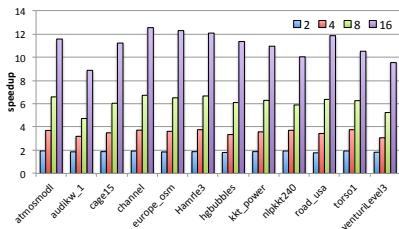


(a) SCALESK

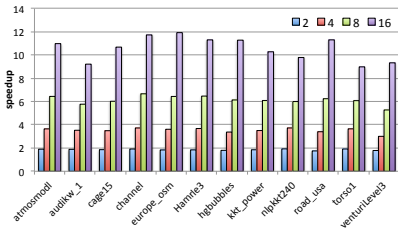


(b) ONESIDEMATCH

Fig. 3. Speedups for SCALESK (left) and ONESIDEMATCH (right) with a single scaling iteration.



(a) KARPISERMT

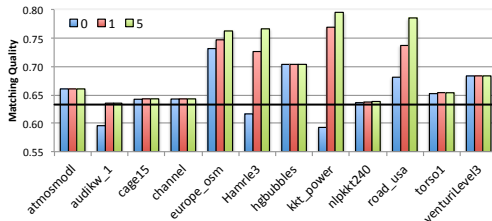


(b) TWOSIDEMATCH

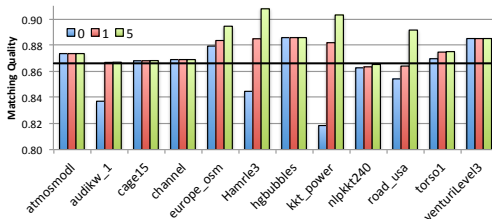
Fig. 4. Speedups for KARPISERMT (left) and TWOSIDEMATCH (right) with a single scaling iteration.

Bipartite experiments III: Quality wrt the scaling iterations

Matching quality of ONEsIDED



Matching quality of TWOSIDED



The horizontal lines are at 0.632 and 0.866 (approximation guarantees)

The bipartite case: some other codes

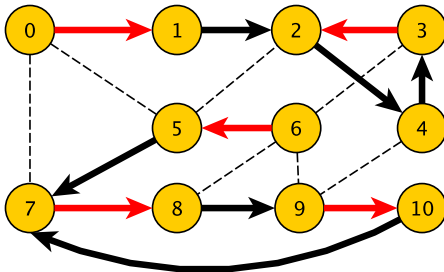
Name	Run time in seconds									
	One thread					16 threads				
	ksV	inc	nd	1SD	2SD	ksV	inc	nd	1SD	2SD
atmosmod1	0.20	14.80	0.07	0.12	0.30	0.03	1.63	0.01	0.01	0.03
audikw.1	0.98	206.00	0.22	0.55	0.82	0.10	16.90	0.02	0.06	0.09
Hamrle3	0.15	0.12	0.04	0.09	0.25	0.02	0.02	0.01	0.01	0.02
hugebubbles	8.36	4.65	1.28	3.92	10.04	0.95	0.53	0.18	0.37	0.94
kkt_power	0.55	0.68	0.09	0.19	0.45	0.08	0.16	0.01	0.02	0.05
road_usa	6.23	3.57	0.96	2.16	5.42	0.70	0.38	0.09	0.22	0.50
wc_50K_64	7.21	3200.00	1.46	4.39	5.80	1.17	402.00	0.12	0.55	0.59

Name	Quality of the obtained matching									
	ksV	inc	nd	1SD	2SD	ksV	inc	nd	1SD	2SD
atmosmod1	1.00	1.00	1.00	0.66	0.87	0.98	1.00	0.97	0.66	0.87
audikw.1	1.00	1.00	1.00	0.64	0.87	0.95	1.00	0.97	0.64	0.87
Hamrle3	1.00	0.84	0.84	0.75	0.90	0.97	0.84	0.86	0.75	0.90
hugebubbles	0.99	0.96	0.96	0.70	0.89	0.96	0.96	0.90	0.70	0.89
kkt_power	0.85	0.79	0.79	0.78	0.89	0.87	0.79	0.86	0.78	0.89
road_usa	0.94	0.81	0.81	0.76	0.88	0.94	0.81	0.81	0.76	0.88
wc_50K_64	0.50	0.50	0.50	0.81	0.98	0.70	0.50	0.51	0.81	0.98

ksV from Azad et al.; inc and nd from Blelloch et al. All heuristics are very fast (except inc which is fast on some other instances). ONEsIDED and TWOsIDED are run with 2 scaling iterations.

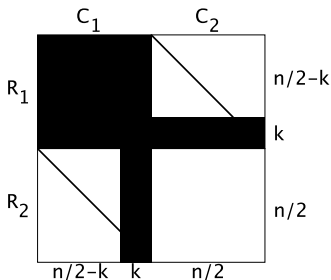
For general, undirected graphs

- ONESIDED is not meaningful; We use TWOSIDED
 - each vertex selects one neighbor
 - we apply the specialized KS developed for the bipartite case above



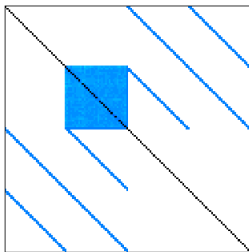
For general, undirected graphs

- We do not have theoretical bounds for the general case
- But empirical results resemble the bipartite case: For the worst case graphs (below)
 - with 3200 vertices and $k = 32$: KS obtains 0.6375 and TwoSided obtains 0.9950.
 - with 6400 vertices and $k = 128$: KS obtains 0.6141 and TwoSided obtains 0.9900



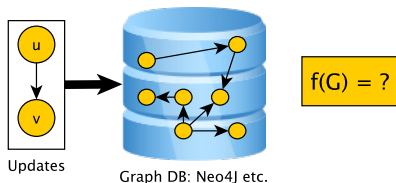
The general case

- For real life graphs, we experimented on the UFL matrices with more than 100 and less than 10000 rows (on these 864 matrices, we applied the algorithms on the pattern of $\mathbf{A} + \mathbf{A}^T$):
 - KS's average quality is 0.9737 and TwoSided's average is 0.8917 with maximum 20 scaling iterations
 - With TwoSided, 0.866 is reached for most of the matrices (some needed more iterations)
 - KS's quality was less than 0.866 for only one matrix: GHS_indef/exdata_1



The streaming case

- Suppose you have a massive-scale graph G in a graph DB.
- The graph is updated w. edge additions and deletions.
- We want to know the value of a function on G (e.g., the cardinality of maximum matching in G)

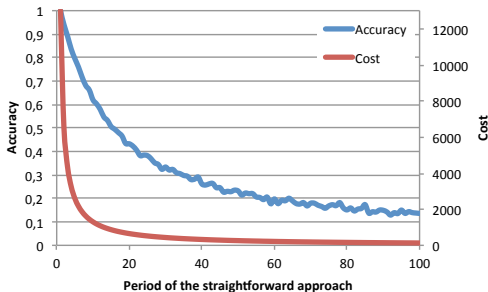


Problem: You don't want to probe and make the DB busy for each update.

The streaming case

Naive solution: Probe the DB after every k updates (i.e., periodic probing).

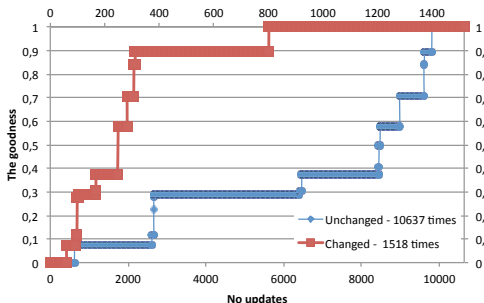
- Example: the matrix Hamrle2 with 22,162 nonzeros (edges in the corresponding bipartite graph)
 - 9,111 of the edges are used for the initial graph structure
 - 13,051 of the edges are used for the updates (i.e., insertions)



Observation: The cost is proportional to $\frac{1}{k}$ so it decreases fast. However, this is also correct for the accuracy (the ratio of correct information).

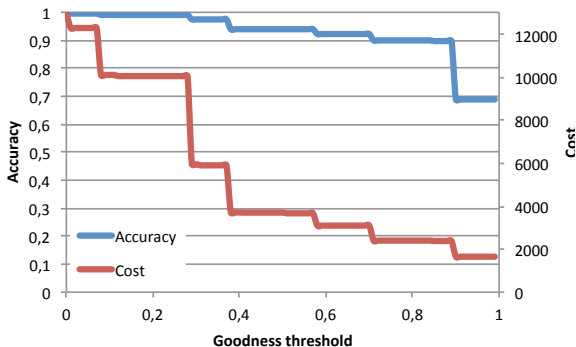
The streaming case

Goodness approach: Probe the DB only for the edges with goodness larger than τ (i.e., the goodness threshold).



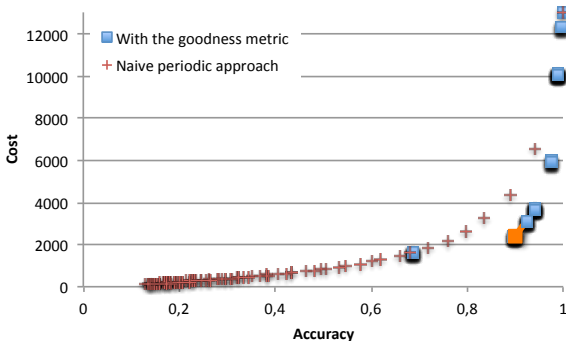
Observation: The goodness values of the edges that increase the maximum cardinality tend to be larger than those of the ones which do not have an impact on the cardinality (averages are 0.82 and 0.35).

The streaming case



Observation: The cost decreases with increasing threshold. Fortunately, the accuracy does not decrease as fast as the naive periodic approach.

The streaming case: comparing with the naive approach



Observation: With the goodness approach, one can have 90% accuracy with 2387 DB probes. With the naive approach the accuracy is 80% with 2611 DB probes.

Thanks!

Thanks for your attention.

<http://people.sabanciuniv.edu/kaya>

More information

- Bipartite case preliminary version: Bipartite Matching Heuristics with Quality Guarantees on Shared Memory Parallel Computers, in proc. IPDPS 2014.
- The final work of the bipartite case with proofs is Fanny Dufossé, Kamer Kaya, Bora Uçar: Two approximation algorithms for bipartite matching on multicore architectures. J. Parallel Distrib. Comput. 85: 62-78 (2015)

Magical transformation

Sinkhorn-Knopp algorithm:

- 1: **while** true **do**
- 2: Scale rows
- 3: Scale columns

- Not necessary convergence in finite time
- Can be applied on matrices with no total support

1	1	1	0	0
1	0	0	1	1
0	1	1	0	0
0	0	1	1	1
1	1	0	0	1

Magical transformation

Sinkhorn-Knopp algorithm:

- 1: **while** true **do**
- 2: Scale rows
- 3: Scale columns

- Not necessary convergence in finite time
- Can be applied on matrices with no total support

$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0	0
$\frac{1}{3}$	0	0	$\frac{1}{3}$	$\frac{1}{3}$
0	$\frac{1}{2}$	$\frac{1}{2}$	0	0
0	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{1}{3}$	0	0	$\frac{1}{3}$

Magical transformation

Sinkhorn-Knopp algorithm:

- 1: **while** true **do**
- 2: Scale rows
- 3: Scale columns

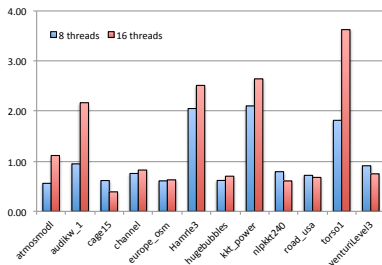
- Not necessary convergence in finite time
- Can be applied on matrices with no total support

$\frac{1}{3}$	$\frac{2}{7}$	$\frac{2}{7}$	0	0
$\frac{1}{3}$	0	0	$\frac{1}{2}$	$\frac{1}{3}$
0	$\frac{3}{7}$	$\frac{3}{7}$	0	0
0	0	$\frac{2}{7}$	$\frac{1}{2}$	$\frac{1}{3}$
$\frac{1}{3}$	$\frac{2}{7}$	0	0	$\frac{1}{3}$

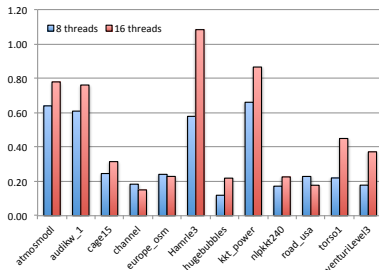
Bipartite experiments : Machine specs

- Machine: two Intel Sandybridge-EP CPUs (each 8 cores) clocked at 2.00Ghz and 256GB of memory split across the two NUMA domains.
- CPUs: Each CPU has eight-cores (16 cores in total) and HyperThreading is enabled.
- Cores: Each core has its own 32kB L1 cache and 256kB L2 cache. The 8 cores on a CPU share a 20MB L3 cache.
- OS: 64-bit Debian with Linux 2.6.39-bpo.2-amd64.
- Using C and OpenMP parallelism; gcc 4.4.5 with the -O2 optimization flag
- (dynamic,512) OpenMP scheduling policy is employed while running all the algorithms except KarpSipser; it uses (guided).
- With 2, 4, 8, 16 threads.
- For atomic operations, gcc's built-in functions are used.

Bipartite experiments : Imbalance in TwoSIDED








(a)








(b)

Figure: The imbalances on the number of processed edges for (a) the edge selection and (b) KARPSSIPSEMT within TwoSIDEDMATCH for 8 and 16 threads. The values are the averages of 20 executions.







References I







-  J. Aronson, M. Dyer, A. Frieze, S. Suen, Randomized greedy matching II, Random Struct. Algor. 6 (1) (1995) 55–73.
-  J. Aronson, A. Frieze, B. G. Pittel, Maximum matchings in sparse random graphs: Karp-Sipser revisited, Random Struct. Algor. 12 (2) (1998) 111–177.
-  A. Azad, M. Halappanavar, S. Rajamanickam, E. G. Boman, A. Khan, A. Pothen, Multithreaded algorithms for maximum matching in bipartite graphs, in: IEEE 26th International Parallel Distributed Processing Symposium (IPDPS), Shanghai, China, 2012, pp. 860–872.
-  D. A. Bader, K. Madduri, GTgraph: A synthetic graph generator suite (Feb. 2006).
<http://www.cse.psu.edu/~madduri/software/GTgraph/gen.pdf>
-  D. A. Bader, H. Meyerhenke, P. Sanders, D. Wagner (Eds.), Graph Partitioning and Graph Clustering - 10th DIMACS Implementation Challenge Workshop, Georgia Institute of Technology, Atlanta, GA, USA, February 13-14, 2012. Proceedings, Vol. 588 of Contemporary Mathematics, American Mathematical Society, 2013.







-  M. Birn, V. Osipov, P. Sanders, C. Schulz, N. Sitchinava, Efficient parallel and external matching, in: F. Wolf, B. Mohr, D. an Mey (Eds.), Euro-Par 2013 Parallel Processing, Vol. 8097 of LNCS, Springer Berlin Heidelberg, 2013, pp. 659–670.
-  G. E. Blelloch, J. T. Fineman, J. Shun, Greedy sequential maximal independent set and matching are parallel on average, in: Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), ACM, 2012, pp. 308–317.
-  R. A. Brualdi, H. J. Ryser, Combinatorial Matrix Theory, Vol. 39 of Encyclopedia of Mathematics and its Applications, Cambridge University Press, Cambridge, UK; New York, USA; Melbourne, Australia, 1991.
-  Ü. V. Çatalyürek, M. Deveci, K. Kaya, B. Uçar, Multithreaded clustering for multi-level hypergraph partitioning, in: 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS), Shanghai, China, 2012, pp. 848–859.







-  D. Chakrabarti, Y. Zhan, C. Faloutsos, R-MAT: A recursive model for graph mining, in: Proceedings of the SIAM International Conference on Data Mining, 2004, pp. 442–446.
-  T. A. Davis, Y. Hu, The University of Florida sparse matrix collection, ACM T. Math. Software 38 (1) (2011) 1:1–1:25.
-  M. Deveci, K. Kaya, B. Uçar, Ü. V. Çatalyürek, GPU accelerated maximum cardinality matching algorithms for bipartite graphs, in: F. Wolf, B. Mohr, D. an Mey (Eds.), Euro-Par 2013 Parallel Processing, Vol. 8097 of LNCS, Springer Berlin Heidelberg, 2013, pp. 850–861.
-  M. Deveci, K. Kaya, Ü. V. Çatalyürek, B. Uçar, A push-relabel-based maximum cardinality matching algorithm on GPUs, in: 42nd International Conference on Parallel Processing, Lyon, France, 2013, pp. 21–29.
-  I. S. Duff, K. Kaya, B. Uçar, Design, implementation, and analysis of maximum transversal algorithms, ACM T. Math. Software 38 (2) (2011) 13:1–13:31.

References IV

-  A. L. Dulmage, N. S. Mendelsohn, Coverings of bipartite graphs, Can. J. Math. 10 (1958) 517–534.
-  M. E. Dyer, A. M. Frieze, Randomized greedy matching, Random Struct. Algor. 2 (1) (1991) 29–45.
-  P. Erdős, A. Rényi, On random matrices, Publications of the Mathematical Institute of the Hungarian Academy of Sciences 8 (3) (1964) 455–461.
-  B. Fagginger Auer, R. Bisseling, A GPU algorithm for greedy graph matching, in: Facing the Multicore Challenge II, Vol. 7174 of LNCS, Springer-Verlag Berlin, 2012, pp. 108–119.
-  M. Halappanavar, J. Feo, O. Villa, A. Tumeo, A. Pothen, Approximate weighted matching on emerging manycore and multithreaded architectures, Int. J. High Perform. C. 26 (4) (2012) 413–430.
-  J. E. Hopcroft, R. M. Karp, An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, SIAM J. Comput. 2 (4) (1973) 225–231.

-  M. Karoński, B. Pittel, Existence of a perfect matching in a random $(1 + e^{-1})$ -out bipartite graph, J. Comb. Theory Ser. B 88 (2003) 1–16.
-  R. M. Karp, U. V. Vazirani, V. V. Vazirani, An optimal algorithm for on-line bipartite matching, in: 22nd annual ACM symposium on Theory of computing (STOC), Baltimore, MD, USA, 1990, pp. 352–358.
-  R. M. Karp, M. Sipser, Maximum matching in sparse random graphs, in: 22nd Annual IEEE Symposium on Foundations of Computer Science (FOCS), Nashville, TN, USA, 1981, pp. 364–375.
-  K. Kaya, J. Langguth, F. Manne, B. Uçar, Push-relabel based algorithms for the maximum transversal problem, Comput. Oper. Res. 40 (5) (2013) 1266–1275.
-  P. A. Knight, The Sinkhorn–Knopp algorithm: Convergence and applications, SIAM J. Matrix Anal. A. 30 (1) (2008) 261–275.
-  P. A. Knight, D. Ruiz, A fast algorithm for matrix balancing, IMA Journal of Numerical Analysis 33 (3) (2013) 1029–1047.

-  P. A. Knight, D. Ruiz, B. Uçar, A symmetry preserving algorithm for matrix scaling, SIAM J. Matrix Anal. A. 35 (3) (2014) 931–955.
-  J. Langguth, F. Manne, P. Sanders, Heuristic initialization for bipartite matching problems, J. Exp. Algorithmics 15 (2010) 1.1–1.22.
-  Z. Lotker, B. Patt-Shamir, S. Pettie, Improved distributed approximate matching, in: Proceedings of the Twentieth Annual Symposium on Parallelism in Algorithms and Architectures (SPAA), ACM, Munich, Germany, 2008, pp. 129–136.
-  J. Magun, Greedy matching algorithms, an experimental study, J. Exp. Algorithmics 3 (1998) 6.
-  N. McKeown, The iSLIP scheduling algorithm for input-queued switches, IEEE/ACM Trans. Netw. 7 (2) (1999) 188–201.
-  A. Meir, J. W. Moon, The expected node-independence number of random trees, Indagationes Mathematicae 76 (1973) 335–341.

-  M. Poloczek, M. Szegedy, Randomized greedy algorithms for the maximum matching problem with new analysis, in: IEEE 53rd Annual Sym. on Foundations of Computer Science (FOCS), New Brunswick, NJ, USA, 2012, pp. 708–717.
-  A. Pothén, Sparse null bases and marriage theorems, Ph.D. thesis, Dept. Computer Science, Cornell Univ., Ithaca, New York (1984).
-  A. Pothén, C.-J. Fan, Computing the block triangular form of a sparse matrix, ACM T. Math. Software 16 (4) (1990) 303–324.
-  D. Ruiz, A scaling algorithm to equilibrate both row and column norms in matrices, Tech. Rep. TR-2001-034, RAL (2001).
-  R. Sinkhorn, P. Knopp, Concerning nonnegative matrices and doubly stochastic matrices, Pacific J. Math. 21 (1967) 343–348.
-  D. W. Walkup, Matchings in random regular bipartite digraphs, Discrete Math. 31 (1) (1980) 59–64.