

Approximate Graph Operations on Parallel Platforms

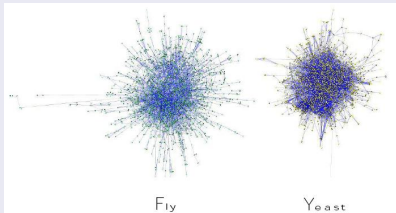
G. Kollias, M. Sathe, O. Schenk, A. Grama¹

¹Purdue University, USA (G. Kollias, A. Grama), University of Basel, Switzerland (M. Sathe, O. Schenk)

- Computing similarity of nodes in two graphs
 - Essentially ranking pairs of nodes
- Network similarity decomposition NSD
 - Algorithm
 - Sequential implementation, experiments and applications
- Parallel NSD-based computation of node similarity scores
 - Algorithm, parallel implementation, experiments
 - The alignment graph
- Parallel NSD
 - Algorithm, parallel implementation, auction matching
 - Large scale experiments
 - Strong and weak scaling results
- Conclusions and future work

Graph similarity in figures

Protein-Protein Interaction (PPI) networks for 2 species

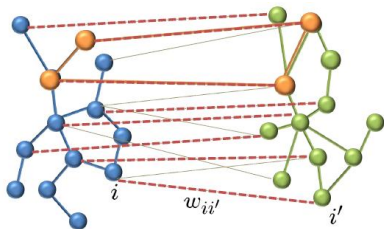


Wikipedia categories and Library of Congress subject headings



How similar are any two nodes of these networks? ^a

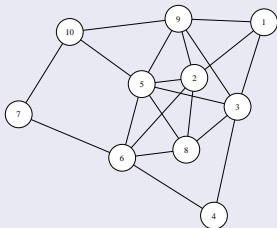
^afigures from M. Bayati, M. Gerritsen, D. Gleich, A. Saberi, and Y. Wang, Algorithms for Large, Sparse Network Alignment, ICDM 2009



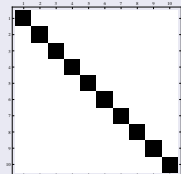
An example of similarity of a graph with itself (self-similarity)

- Applying the similarity pipeline presented here
- (i, j) square denotes a matching of node i to node j .

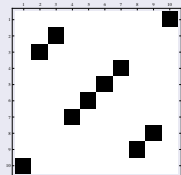
Example graph



Allow matching a node to itself



Do not allow matching a node to itself



Rank-inspired definition of similarity

Node Ranking

A node is important if it is linked by other important nodes

Graph Similarity

Two nodes are similar if they are linked by other similar node pairs



V.D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren.

A Measure of Similarity between Graph Vertices: Applications to Synonym Extraction and Web Searching.

SIAM Rev., 46(4):647–666, 2004.



R. Singh, J. Xu, and B. Berger.

Global alignment of multiple protein interaction networks with application to functional orthology detection

Proceedings of the National Academy of Sciences, 105(35):12763, 2008.

Singh's et al approach, IsoRank, is our focus, has been typically applied to undirected graphs

Notation

- A, B the the adjacency matrices of input graphs G_A, G_B .
- \tilde{A} is A^T normalized by columns, i.e. $(\tilde{A})_{ij} = a_{ji} / \sum_{i=1}^{n_A} a_{ji}$; similarly for \tilde{B} .
- $h = \text{vec}(H)$ a normalized vector where H_{ij} are independently known similarity scores between node sets, $i \in V_B$ and $j \in V_A$.
- $\text{vec}(\cdot)$ operation for building a vector from a matrix (stacking its columns); $\text{unvec}(\cdot)$ is the inverse operation.
- α : percentage of network data contribution in the algorithm.
- $\tilde{C} = A \tilde{\otimes} B = \tilde{A} \otimes \tilde{B}$
- Computed matrix X contains similarity scores: X_{ij} entry denotes how “similar” nodes $i \in V_B$ and $j \in V_A$ are.

Reminder

Kronecker product $A \otimes B$ of two matrices:

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \cdots \\ a_{2,1}B & a_{2,2}B & \\ \vdots & & \ddots \end{bmatrix} = \begin{bmatrix} a_{1,1}b_{1,1} & a_{1,1}b_{1,2} & \cdots & a_{1,2}b_{1,1} & a_{1,2}b_{1,2} & \cdots \\ a_{1,1}b_{2,1} & a_{1,1}b_{2,2} & & a_{1,2}b_{2,1} & a_{1,2}b_{2,2} & \\ \vdots & & \ddots & & & \\ a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & & a_{2,2}b_{1,1} & a_{2,2}b_{1,2} & \\ a_{2,1}b_{2,1} & a_{2,1}b_{2,2} & & a_{2,2}b_{2,1} & a_{2,2}b_{2,2} & \\ \vdots & & & & & \ddots \end{bmatrix}.$$

IsoRank algorithm

IsoRank iteration

- $x \leftarrow \alpha \tilde{C}x + (1 - \alpha)h$

until convergence.

Alternatively $X \leftarrow \alpha \tilde{B}X\tilde{A}^T + (1 - \alpha)H$ as the iteration kernel

- Because $AXB = \text{unvec}((B^T \otimes A)x)$ (property of Kronecker products)
- MAT3 (triple matrix product implementation of IsoRank idea)
- In IsoRank kernel $x \leftarrow \alpha \tilde{C}x + (1 - \alpha)h$, set $x^{(0)} = h$
- Expanding, after n steps

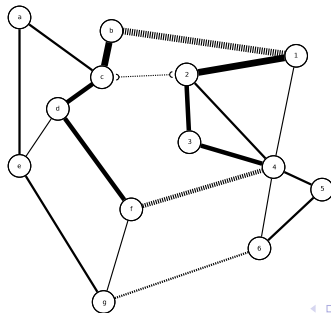
$$x^{(n)} = (1 - \alpha) \sum_{k=0}^{n-1} \alpha^k \tilde{C}^k h + \alpha^n \tilde{C}^n h$$

- Alternatively
$$X^{(n)} = (1 - \alpha) \sum_{k=0}^{n-1} \alpha^k \tilde{B}^k H(\tilde{A}^T)^k + \alpha^n \tilde{B}^n H(\tilde{A}^T)^n$$

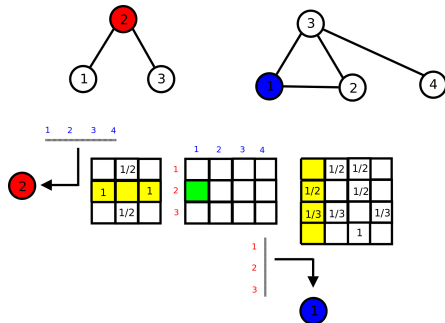
Similarity scores are sums of contributions from all k-hop neighbors (similarity score aggregation)

An example of similarity score aggregation in IsoRank

- Let G_A and G_B with nodes $\{a, b, c, d, e, f, g\}$ and $\{1, 2, 3, 4, 5, 6\}$
- Suppose nodes $(b, 1)$, $(f, 4)$ and $(g, 6)$ pairs are somewhat similar (H information)
- Sum the contributions of all 1-hop, 2-hop, 3-hop,... neighbors in the two networks (along paths like $c - b - 1 - 2$, $c - d - f - 4 - 3 - 2$, $c - a - e - g - 6 - 5 - 4 - 2$) with known or previously computed similarity)



A look into $\tilde{B}X\tilde{A}^T$ term “mechanics”



Top to bottom, left to right:

- Graphs G_B , G_A
- \tilde{B} , similarity matrix X and \tilde{A}^T

How to update similarity score between nodes 2 of G_B and 1 in G_A , i.e. X_{21} entry (green)?

- Node 2 of G_B can pull score from its neighbors 1 and 3
- It can do so in *four* different ways because these neighbors are assumed connected with “virtual links” to every other of the *four* nodes in the other network G_A , i.e. $y^T \leftarrow \tilde{B}_{2,all} X$
- However *two* of these ways are relevant since node 1 of G_A can pull score from its *two* neighbors, namely 2 and 3, i.e. $X_{21} \leftarrow y^T \tilde{A}_{all,1}^T$

Decomposing graphs: The NSD idea

In IsoRank, using H as the initial condition ($X^{(0)} = H$), after n steps we get $X^{(n)} = (1 - \alpha) \sum_{k=0}^{n-1} \alpha^k \tilde{B}^k H (\tilde{A}^T)^k + \alpha^n \tilde{B}^n H (\tilde{A}^T)^n$

- Let $H = uv^T$ (1 component, i.e. 1 outer vector product).

Two phases for computing X then:

- ① $u^{(k)} = \tilde{B}^k u$ and $v^{(k)} = \tilde{A}^k v$ (*preprocess/compute iterates*)
- ② $X^{(n)} = (1 - \alpha) \sum_{k=0}^{n-1} \alpha^k u^{(k)} v^{(k)T} + \alpha^n u^{(n)} v^{(n)T}$ (*construct X*)

This extends to the case H is approximated by s components (sum of s outer vector products): $H \sim \sum_{i=1}^s w_i z_i^T$.

NSD key points

- Instead of triple matrix products we compute sums of outer products of vectors
- These vectors in turn are sparse matrix-vector iterates than can be computed independently

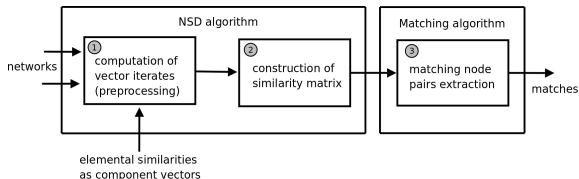
NSD-based similarity matrix construction

```
1: {Input:  $A, B, \{w_i, z_i | i = 1, \dots, s\}$ ,  
    $\alpha$  and  $n$ , Output:  $X = X^{(n)}$ }  
2: compute  $\tilde{A}, \tilde{B}$  {phase 1}  
3: for  $i = 0$  to  $s$  do  
4:    $w_i^{(0)} \leftarrow w_i, z_i^{(0)} \leftarrow z_i$   
5:   for  $k = 0$  to  $n$  do  
6:      $w_i^{(k)} \leftarrow \tilde{B} w_i^{(k-1)}$   
7:      $z_i^{(k)} \leftarrow \tilde{A} z_i^{(k-1)}$   
8:   end for  
9:   zero  $X_i^{(n)}$  {phase 2 start}  
10:  for  $k = 0$  to  $n - 1$  do  
11:     $X_i^{(n)} \leftarrow X_i^{(n)} + \alpha^k w_i^{(k)} z_i^{(k)T}$   
12:  end for  
13:   $X_i^{(n)} \leftarrow$   
     $(1 - \alpha)X_i^{(n)} + \alpha^n w_i^{(n)} z_i^{(n)T}$   
14: end for  
15:  $X^{(n)} \leftarrow \sum_{i=1}^s X_i^{(n)}$ 
```

Remarks

- Decomposition happens along the set of paths of successively larger length k , however with increasing “damping” (because of the $(1 - \alpha)\alpha^k$ factor with $\alpha \in [0, 1]$)
- The computation does not involve either explicitly building the product graph related \tilde{C} or computing triple matrix products of the form $\tilde{B}X\tilde{A}^T$ at each step. Only computing outer vector products at the end.
- (Sparse) matrix-vector products for the two graphs can be computed *quite independently*

Block diagram of the decomposition approach



- ① Input a network pair and its elemental similarities H as component vectors; Preprocess/compute vector iterates
- ② Construct similarity matrix X by summing outer products of vectors
- ③ Produce a set of pairs (matches) of nodes from one network that are “most similar” to nodes from the other: rows and columns of X as nodes of a weighted bipartite graph, X_{ij} its weights.
 - Matching algorithms used in experiments for this 3rd phase: Primal Dual Matching (PDM), Greedy Matching (1/2 approximation, GM), Hungarian, auction.

In the sequel IsoRank refers to the implementation of the IsoRank idea followed by the application of Hungarian and PDM algorithms to resulting X (as available in Singh's binary code).

Protein-Protein Interaction (PPI) networks (sequential)

Species	Nodes	Edges
celeg (worm)	2805	4572
dmela (fly)	7518	25830
ecoli (bacterium)	1821	6849
hpylo (bacterium)	706	1414
hsapi (human)	9633	36386
mmusc (mouse)	290	254
scere (yeast)	5499	31898

Species pair	NSD (secs)	MAT3 (secs)	PDM (secs)	GM (secs)	IsoRank (secs)
celeg-dmela	3.15	64.20	152.12	7.29	783.48
celeg-hsapi	3.28	69.74	163.05	9.54	1209.28
celeg-scere	1.97	44.61	127.70	4.16	949.58
dmela-ecoli	1.86	37.79	86.80	4.78	807.93
dmela-hsapi	8.61	211.19	590.16	28.10	7840.00
dmela-scere	4.79	131.22	182.91	12.97	4905.00
ecoli-hsapi	2.41	47.48	79.23	4.76	2029.56
ecoli-scere	1.49	35.86	69.88	2.60	1264.24
hsapi-scere	6.09	152.02	181.17	15.56	6714.00

- We computed the similarity matrices X for various possible pairs of species using only PPI data (network data)
- $\alpha = 0.80$, uniform initial conditions (outer product of suitably normalized 1's for each pair), 20 iterations, 1 component.

Finding

1-3 orders of magnitude speedup of NSD-based approaches compared to comparable MAT3-based (with PDM, GM) and IsoRank ones (no parallelization yet).



G. Kollias, S. Mohammadi, and A. Grama.

Network Similarity Decomposition (NSD): A Fast and Scalable Approach to Network Alignment.
IEEE Transactions on Knowledge and Data Engineering, 2011.

Parallel NSD-based similarity matrix construction

Parallel NSD: Root process

```

compute  $\tilde{A}, \tilde{B}$ 
for  $i = 1$  to  $s$  do
     $w_i^{(0)} \leftarrow w_i, z_i^{(0)} \leftarrow z_i$ 
    for  $k = 0$  to  $n$  do
         $w_i^{(k)} \leftarrow \tilde{B} w_i^{(k-1)}$ 
         $z_i^{(k)} \leftarrow \tilde{A} z_i^{(k-1)}$ 
    end for
end for
for  $i = 1, \dots, s, k = 0, \dots, n$  do
    Partition  $w_i^{(k)}$  in  $p$  fragments,  $w_{i,1}^{(k)}, \dots, w_{i,p}^{(k)}$ 
    Partition  $z_i^{(k)}$  in  $q$  fragments,  $z_{i,1}^{(k)}, \dots, z_{i,q}^{(k)}$ 
end for
Send to every process  $(r, u)$  in the process grid  $p \times q$  its
corresponding  $w_{i,r}^{(k)}, z_{i,u}^{(k)}$  fragments,
 $\forall i = 1, \dots, s, k = 0, \dots, n$  ( $r = 1, \dots, p,$ 
 $u = 1, \dots, q$ )
    
```

Parallel NSD: Worker process (r, u)

```

Receive corresponding  $w_{i,r}^{(k)}, z_{i,u}^{(k)}$  fragments,
 $\forall i = 1, \dots, s, k = 0, \dots, n$  from the root process
for  $i = 1$  to  $s$  do
    zero  $X_{i,ru}^{(n)}$ 
    for  $k = 0$  to  $n - 1$  do
         $X_{i,ru}^{(n)} \leftarrow X_{i,ru}^{(n)} + \alpha^k w_{i,r}^{(k)} z_{i,u}^{(k)T}$ 
    end for
     $X_{i,ru}^{(n)} \leftarrow (1 - \alpha) X_{i,ru}^{(n)} + \alpha^n w_{i,r}^{(n)} z_{i,u}^{(n)T}$ 
end for
 $X_{ru}^{(n)} \leftarrow \sum_{i=1}^s X_{i,ru}^{(n)}$ 
    
```

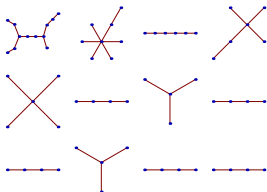
num cores	dmela-hsapi		hsapi-scere	
	t_{Iters}	t_{SimMat}	t_{Iters}	t_{SimMat}
4	0.211	28.103	0.194	21.062
9	0.210	15.914	0.213	11.865
16	0.219	9.851	0.215	7.478
25	0.202	7.072	0.195	5.283
36	0.311	6.080	0.209	4.493
49	0.193	5.809	0.240	4.233
64	0.207	4.915	0.253	3.576

- Java implementation, MPJ for message passing, csparsej for sparse matvec's
- 2 heterogeneous clusters (40 + 24 cores)
- 1 component, 20 iters, $\alpha = 0.8$ (times in secs)

The alignment graph

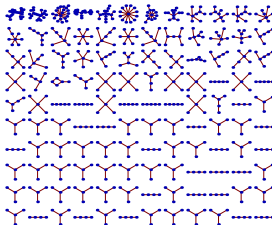
Build the alignment graph

- Nodes are pairs of matching nodes (b_i/a_j) from the original networks G_B, G_A
- (b_i/a_j) is connected to (b_p/a_q) iff b_i, b_p and a_j, a_q are neighbors in G_B, G_A
- Each conserved edge implies matching the corresponding edges connecting the elements of the matching pairs at its endpoints in the input networks.
- These subgraphs are essentially matchings of substructures in the input networks (CCS, Common Connected Subgraphs).



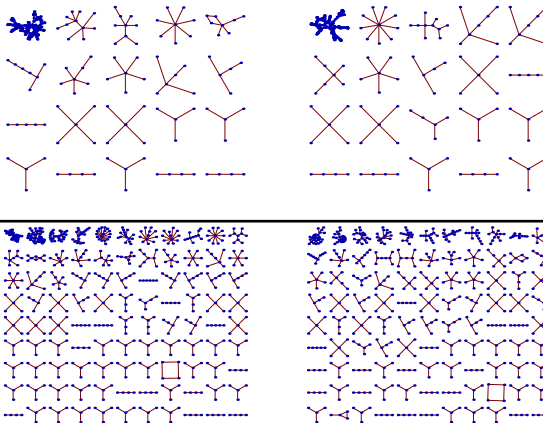
Topological quality of computed matchings?

- Number of edges in the alignment graph (conserved edges)
- Size of the connected subgraphs in the alignment graph



CCS for the alignment graph based on H (sequence similarity) and X (MAT3 computed) for a PPI network pair (dmela-scere)

CCS and H approximation



- NSD on the dominant $s = 5$ and $s = 10$ components of H as computed by NMF (upper row) and $s = 500$ and $s = 1000$ components as computed by SVD (lower row). 20 iters, $\alpha = 0.8$ (dmela-scere)
- More components offer more opportunities for CCS development and thus more conserved edges; smaller CCS are favored over very large ones

Parallel NSD: Matching G_A , G_B network nodes in parallel

Parallel NSD similarity matrix construction and parallel auction combined

- 1: $\{\square = \text{root process, no labels} = \text{all processes } r\}$
- 2: \square load adjacency matrices A , B and component vectors w_i , z_i ;
- 3: \square compute \tilde{A} , \tilde{B} ;
- 4: broadcast \tilde{A} , w_i , z_i ;
- 5: distribute \tilde{B} by row blocks {each process r gets its \tilde{B}_r part};
- 6: {for all components i and steps k ($z_i^{(0)} = z_i$, $w_i^{(0)} = w_i$)}
- 7: compute vector iterates $z_i^{(k)} \leftarrow \tilde{A} z_i^{(k-1)}$
- 8: compute vector iterates $w_{i,r}^{(k)} \leftarrow \tilde{B}_r w_i^{(k-1)}$, gather $w_i^{(k)}$ (*// matvec*);
- 9: compute *row-wise* the local similarity matrix X_r (*embarrassingly //*)
- 10: {NSD-based, *sparsify* if needed (sort row entries, keep largest ones)};
- 11: compute weighted matchings by *// auction*
- 12: {matching permutation lands on root};
- 13: \square compute number of conserved edges;

Auction algorithm and experimental setup

Auction algorithm (implemented in parallel)

```
1:  $M = \emptyset$ ; {current matching}
2:  $I = \{i : 1 \leq i \leq m\}$ ; {set of unassigned persons}
3:  $p_j \leftarrow 0$  for  $j = 1, \dots, n$ ; {prices for the objects}
4: while  $I \neq \emptyset$  do
5:   {find object  $j$  with best and second best profit};
6:    $j_i \leftarrow \arg \max_j \{x_{ij} - p_j\}$ ;
7:    $v_i \leftarrow \max_j \{x_{ij} - p_{j_i}\}$ ;
8:    $w_i \leftarrow \max_{j \neq j_i} \{x_{ij_i} - p_{j_i}\}$ ;
9:   {update price with the bid  $v_i - w_i + \epsilon$ }
10:   $p_{j_i} \leftarrow p_{j_i} + v_i - w_i + \epsilon$ ;
11:  {assign person to the desired object};
12:   $M \leftarrow M \cup \{i, j_i\}$ ,  $I \leftarrow I \setminus \{i\}$ ;
13:  {free previous owner  $k$  if available}
14:   $M \leftarrow M \setminus \{k, j_i\}$ ,  $I \leftarrow I \cup \{k\}$ ;
15: end while
```

- Based on the metaphor of buyers and objects
- Favors 1D distribution of row block in the parallel setting
- Actually a special *adaptive* parallel auction variant is implemented providing faster convergence

Setup

- constant nonzero entries/ X row: *strong scaling* for auction
- constant nonzero entries/core: *weak scaling* for auction
- strong scaling for similarity score computations
- Up to 3,072 cores (CRAY XE6, consists of 2-hexacore processors)
- Hybrid programming model (C/MPI/OpenMP)

Summary of base timing results (1/2)

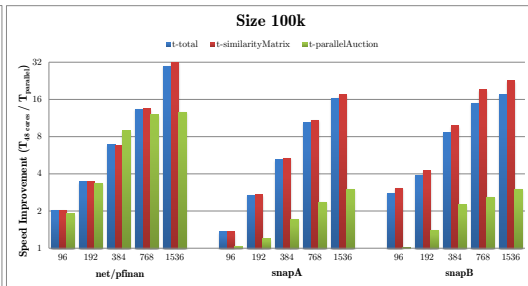
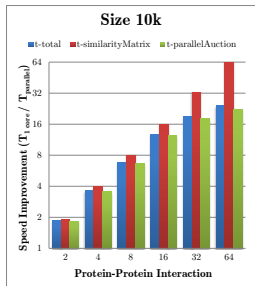
Pair	Graph	#Vertices	#Edges	Total time (s)	#Cores
protein-protein	yeast fruitfly	5,499 7,518	31,898 25,830	75	1
net/pfinan	net4-1 pfinan512	88,343 74,752	1,265,035 335,872	796	48
snapA	soc-slashdot090221 soc-slashdot090216	82,144 81,871	549,202 545,671	2,688	48
snapB	soc-slashdot0902 soc-slashdot0811	82,168 77,360	948,464 905,468	1,497	48
usroads	usroads usroads-48	129,164 126,146	165,435 161,950	281	384
dnvs	halfb fullb	224,617 199,187	6,306,219 5,953,632	880	384
b3	m133-b3 shar_te2-b3	200,200 200,200	800,800 800,800	1,593	384
coAuthors	coAuthorsDBLP coAuthorsCiteseer	299,067 227,320	977,676 814,134	659	768
notreDame	NotreDame_www web-NotreDame	325,729 325,729	929,849 1,497,134	764	768
stanford	Stanford web-Stanford	281,903 281,903	2,312,497 2,312,497	615	768

Summary of base timing results (2/2)

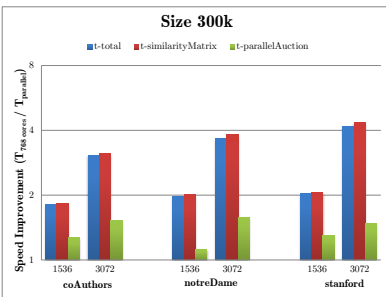
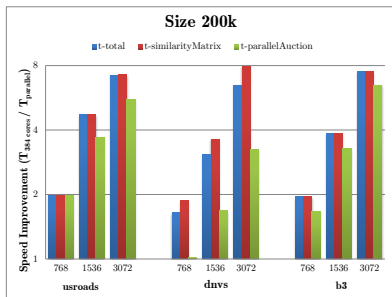
Pair	Graph	#Vertices	#Edges	Total time (s)	#Cores
amazon	amazon0505	410,236	3,356,824	558	3,072
	amazon0601	403,394	3,387,388		
delaunay	delaunay_n19	524,288	1,572,823	938	3,072
	delaunay_n18	262,144	786,396		
authorsSelf	coAuthorsCiteseer	227,320	814,134	226	3,072
	coAuthorsCiteseer	227,320	814,134		
coPapers	coPapersDBLP	540,486	15,245,729	2,167	3,072
	coPapersCiteseer	434,102	16,036,720		
papersSelf	coPapersCiteseer	434,102	16,036,720	1,630	3,072
	coPapersCiteseer	434,102	16,036,720		
dbpedia1	dbpedia-3.0_300k	300,000	1,320,138	17,382	128
	dbpedia-3.5.1_500k	500,000	10,546,881		
eu/in	eu-2005_300k	300,000	10,835,193	18,122	128
	in-2004_500k	500,000	8,506,508		
dbpedia2	dbpedia-3.0_500k	500,000	2,680,807	16,838	256
	dbpedia-3.5.1_1500k	1,500,000	26,794,451		
euSelf	eu-2005	862,664	19,235,140	10,939	256
	eu-2005	862,664	19,235,140		

Summary of speed improvements (1/2)

- Absolute values for the time ratio nominator part correspond to base timings
- x-axis is the number of cores
- Speed improvements for both the two basic phases of the parallel similarity pipeline (similarity scores matrix construction followed by matching by auction) and the total procedure are depicted



Summary of speed improvements (2/2)



Pair	amazon	delaunay	coPapers	papersSelf	authorsSelf
t_similarityMatrix	481.73	935.04	2, 156.20	1, 620.30	222.56
t_parallelAuction	76.18	2.61	10.37	9.47	3.01
t_total	557.91	937.65	2, 166.57	1, 629.77	225.57

Quality measurement indices

- Conserved Edges (CE)
- Rate: Ratio of conserved edges over the minimum of the edges in the two networks

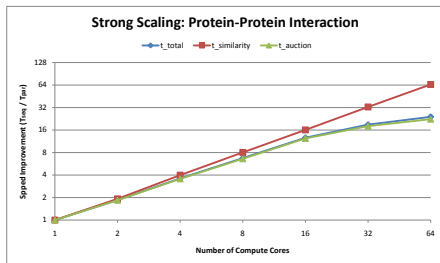
Pair	#CE	Rate
protein-protein	745	0.03
net/pfinan	74,778	0.22
snapA	14,296	0.02
snapB	77,617	0.09
usroads	2,666	0.02
dnvs	1,750,799	0.29
b3	29,217	0.15
coAuthors	85,437	0.11
notreDame	113,992	0.12
stanford	107,968	0.05

Pair	#CE	Rate
amazon	46,278	0.01
delaunay	112,152	0.14
authorsSelf	814,134	1.00
coPapers	3,520,545	0.23
papersSelf	16,036,720	1.00
dbpedia1	1,100	0.004
eu/in	80,884	0.04
dbpedia2	2,082	0.007
euSelf	219,759	0.26

Strong scaling experiments with (the small) PPI networks

dmela and scere

Cores	1	2	4	8	16	32	64
t_similarityMatrix	11.73	6.10	2.94	1.46	0.73	0.36	0.18
t_parallelAuction	62.68	34.02	17.61	9.49	5.07	3.47	2.80
t_totalSimilarityProcess	74.52	40.23	20.65	11.05	5.90	3.94	3.10
Conserved edges	625	691	688	737	745	668	658



- roughly 3 secs for getting matching pairs ($a = 0.8$, 20 iterations, 1 component, 64 cores); cf. over an hour (4905 secs) for the inherently sequential IsoRank
- nice scaling properties for the critical parallel auction phase

Strong scaling experiments

eu-2005_300k and in-2004_500k

Cores	128	256	512	1024
t_generateIterates	5.07	5.04	5.33	6.13
t_generateRow	16,450.88	8,152.49	4,030.19	1,224.77
t_sort	1,577.80	788.39	394.80	197.03
t_similarityMatrix	18,045.54	8,949.46	4,429.16	1,423.65
t_parallelAuction	55.16	28.82	16.32	11.90
t_totalSimilarityProcess	18,121.54	8,998.95	4,466.56	1,457.53
Conserved edges	80,884	80,884	80,884	80,884

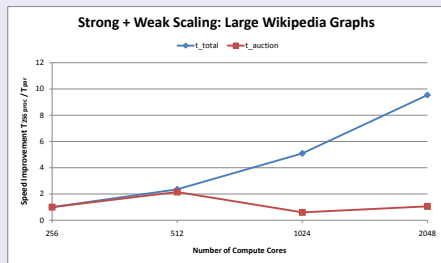
dbpedia-3.0_300k and dbpedia-3.5.1_500k

Cores	128	256	512	1024
t_generateIterates	11.00	11.19	11.53	12.36
t_generateRow	15,703.82	7,475.45	3,254.46	1,228.59
t_sort	1,606.47	802.44	400.97	200.69
t_similarityMatrix	17,327.27	8,286.56	3,659.58	1,431.17
t_parallelAuction	31.97	19.78	14.37	14.93
t_totalSimilarityProcess	17,382.15	8,329.41	3,697.45	1,470.62
Conserved edges	1,010/1,100	1,018/1,097	1,014/1,088	1,014/1,088

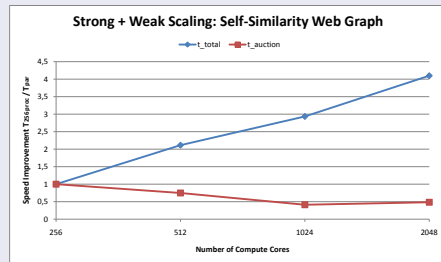
- Timings for various phases, number of conserved edges
- $a = 0.8$, 20 iterations, 10 random components for all large scale experiments

Weak scaling experiments (for auction)

dbpedia-3.0_500k,
dbpedia-3.5.1_1500k



Self-similarity for eu-2005_300k



- The expected almost ideal scaling properties of parallel matrix construction phase compensates for deviations from optimality of the auction phase in the total time speedup.

Conclusions

- Graph similarity computations can be **decomposed** with respect to
 - the identical link patterns occurring in the graphs
 - the rank-one terms building up the initial condition
 - Interpretation insight
- Graph similarity computations can be **uncoupled**
 - Each graph is processed independently, sparse matrix-vector kernel
 - Merging through outer products only at the end
 - Speedup

Highlights of NSD

- Massive performance gains from Parallel NSD-based similarity matrix construction, no scale up limit (embarrassingly //).
- Parallel NSD is an integrated, performant approach for processing very large networks

- Evaluate the effect of more components on the quality of the similarity scores for various application domains
- Develop and implement heuristics for parallel matching, suitable for different application domains

Weighted matching algorithm for matrices represented as sum of outer products of vectors?