Wexner Medical Center

Incremental Algorithms for Closeness Centrality

A. Erdem Sarıyüce [1,2], Kamer Kaya [1], Erik Saule [1*], **Ümit V. Çatalyürek** [1,3]

[1] Department of Biomedical Informatics
[2] Department of Computer Science & Engineering
[3] Department of Electrical & Computer Engineering
The Ohio State University

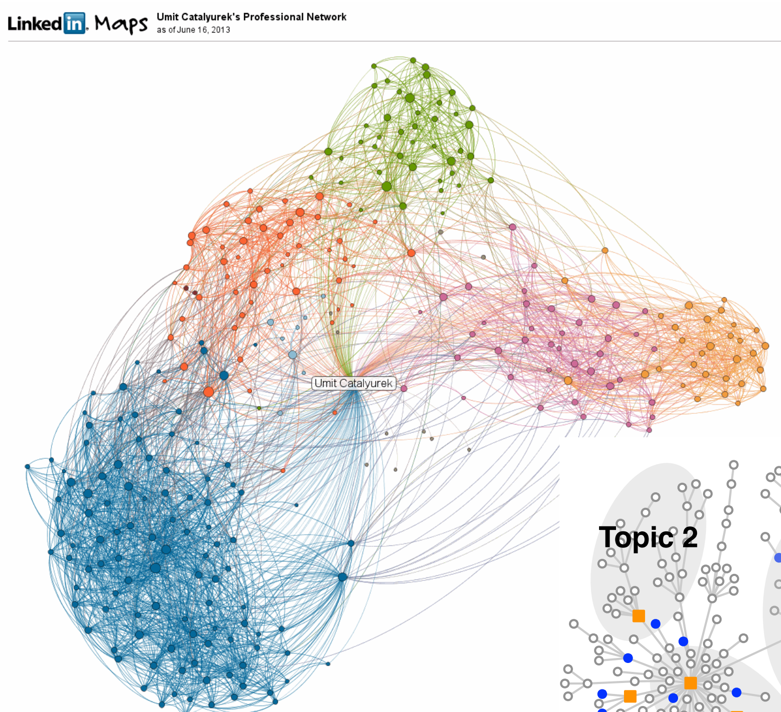
[*] Department of Computer Science
University of North Carolina Charlotte

IEEE BigData 2013, Santa Clara, CA

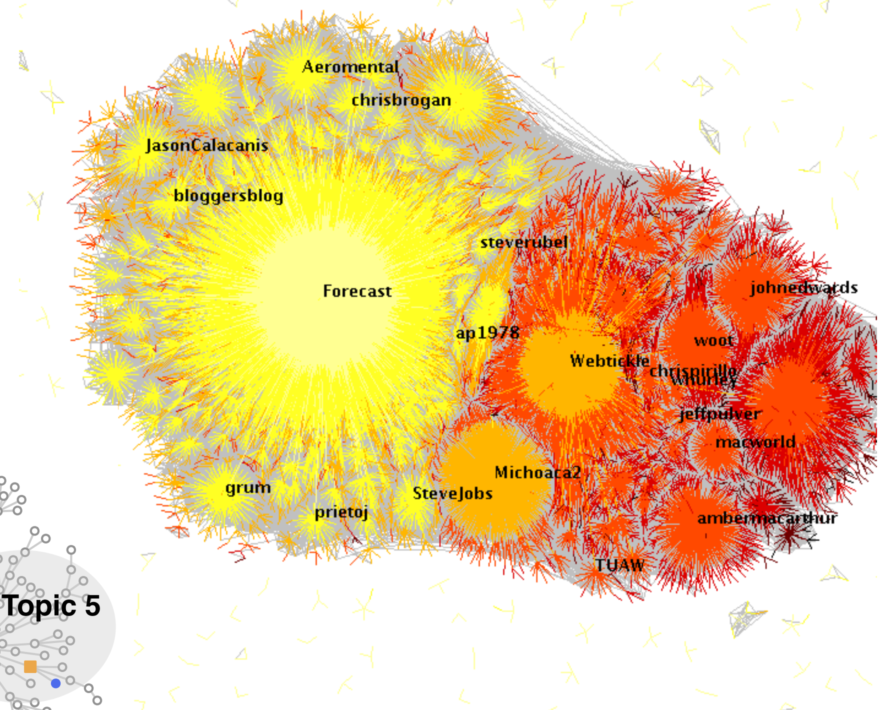
Massive Graphs are everywhere

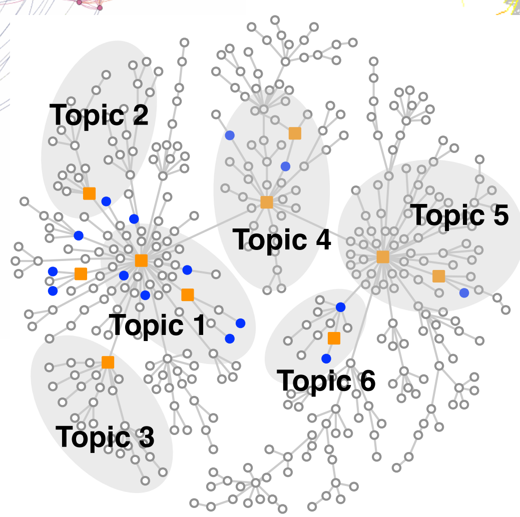
Wexner Medical Center

- Facebook has a billion users and a trillion connections
- Twitter has more than 200 million users



Twitter Social Network, 20K nodes 250K edges



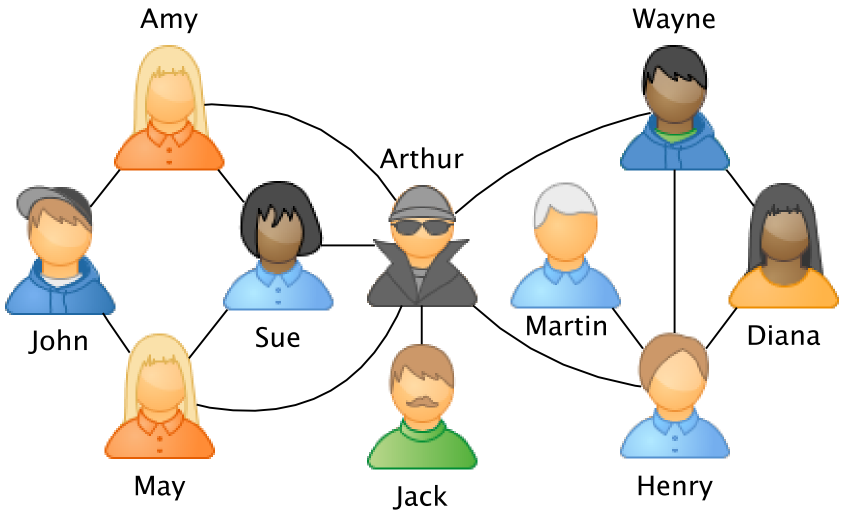


citation graphs

Image Copyright UMBC eBiquity Research Group

Large(r) Networks and Centrality

- Who is more important in a network? Who controls the flow between nodes?
  - Centrality metrics answer these questions
  - Closeness Centrality (CC) is an intriguing metric
- How to handle changes?
  - Incremental algorithms are essential



Closeness Centrality (CC)

- Let *G=(V, E)* be a graph with vertex set *V* and edge set *E*
  - Farness (far) of a vertex is the sum of shortest distances to each vertex

$$\texttt{far}[u] = \sum_{\substack{v \in V \\ d_G(u,v) \neq \infty}} d_G(u, v)$$

  - Closeness centrality (cc) of a vertex :

$$\texttt{cc}[u] = \frac{1}{\texttt{far}[u]}$$

- Best algorithm: All-pairs shortest paths
  - O(|V|.|E|) complexity for unweighted networks
- For large and dynamic networks
  - From scratch computation is infeasible
  - Faster solutions are essential

CC Algorithm

Wexner Medical Center

Algorithm 1: CC: Basic centrality computation

Data: $G = (V, E)$
Output: $cc[.]$
for each $s \in V$ **do**
    ▷SSSP(G, s) with centrality computation
    $Q \leftarrow$ empty queue
    $d[v] \leftarrow \infty, \forall v \in V \setminus \{s\}$
    Q.push(s), $d[s] \leftarrow 0$
    $\texttt{far}[s] \leftarrow 0$
    while *Q is not empty* **do**
        $v \leftarrow Q$.pop()
        for all $w \in \Gamma_G(v)$ **do**
            if $d[w] = \infty$ **then**
                Q.push(w)
                $d[w] \leftarrow d[v] + 1$
                $\texttt{far}[s] \leftarrow \texttt{far}[s] + d[w]$
    $\texttt{cc}[s] = \frac{1}{\texttt{far}[s]}$
return $\texttt{cc}[.]$

Single Source Shortest Path (SSSP) is computed for each vertex

Breadth-First Search with farness computation

cc value is assigned

Incremental Closeness Centrality

Wexner Medical Center

- Problem definition: Given a graph G=(V, E), closeness centrality values of vertices cc and an inserted (or removed) edge u-v; find the closeness centrality values cc' of the graph G' = (V, E U {u,v}) (or G' = (V, E \ {u,v}))

- Computing cc values from scratch after each edge change is very costly
  - Need a faster algorithm

Filtering Techniques

Wexner Medical Center

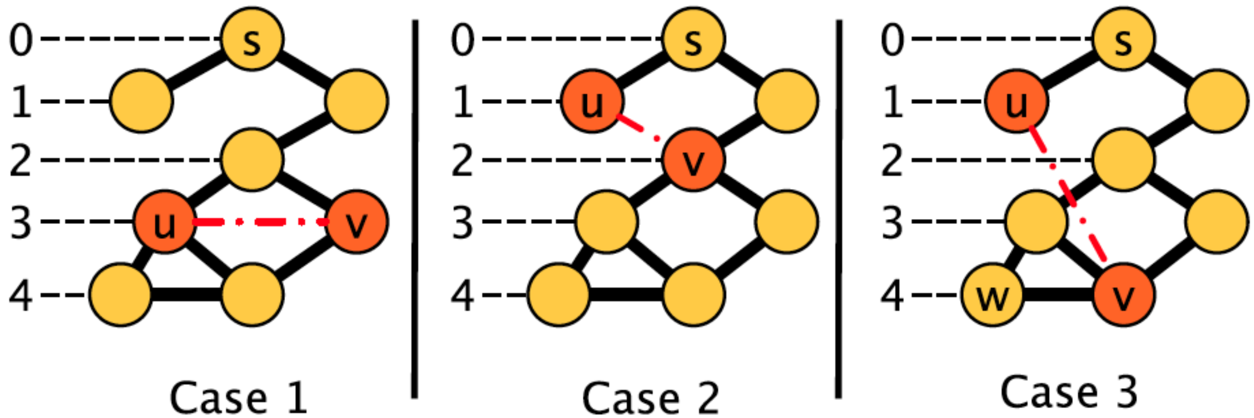
- We aim to <u>reduce number of SSSPs</u> to be executed

- Three filtering techniques are proposed
  - Filtering with level differences
  - Filtering with biconnected components
  - Filtering with identical vertices

- And an additional SSSP hybridization technique

Filtering with level differences

- Upon edge insertion, breadth-first search tree of each vertex will change. Three possibilities:



- Case 1 and 2 will not change cc of s!
  - No need to apply SSSP from them
- Just Case 3
  - How to find such vertices?
    - BFSs are executed from u and v and level diff is checked

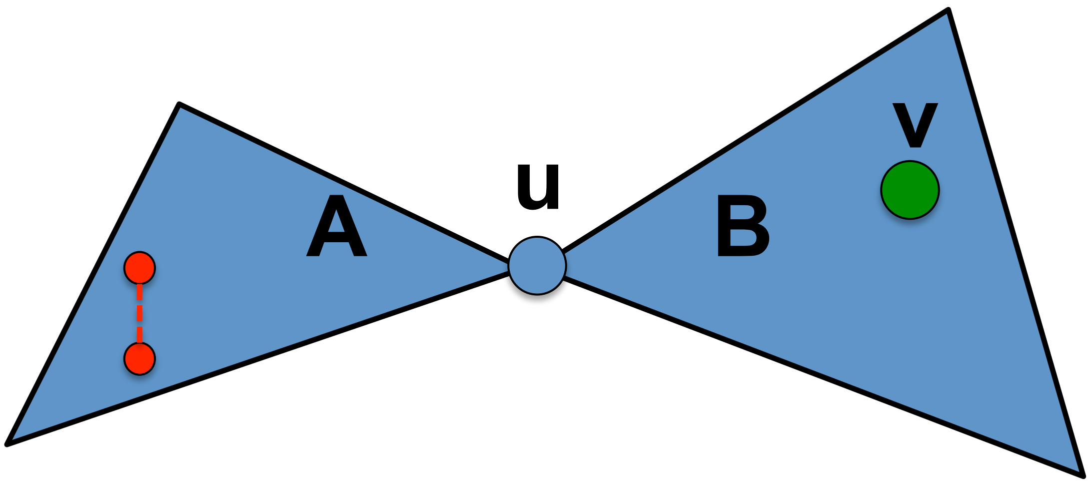
Filtering with level differences

Algorithm 2: Simple work filtering

Data: $G = (V, E)$, $\texttt{cc}[.]$, uv
Output: $\texttt{cc}'[.]$
$G' \leftarrow (V, E \cup \{uv\})$
$du[.] \leftarrow \text{SSSP}(G, u)$ ▷ `distances from` u `in` G
$dv[.] \leftarrow \text{SSSP}(G, v)$ ▷ `distances from` v `in` G
for each $s \in V$ **do**
  if $|du[s] - dv[s]| \leq 1$ **then** → Case 1 and 2
    $\texttt{cc}'[s] = \texttt{cc}[s]$
  else → Case 3
    ▷ `use the computation in Algorithm 1 with` G'
return $\texttt{cc}'[.]$

Filtering with biconnected components

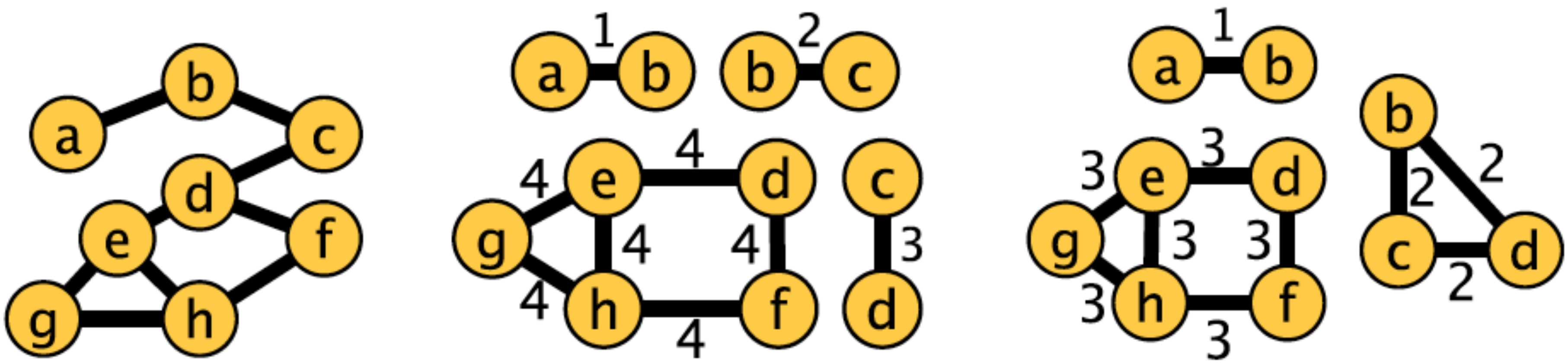
- What if the graph have articulation points?



- Change in A can change cc of any vertex in A and B
- Computing the change for u is **enough** for finding changes for any vertex v in B (constant factor is added)

Filtering with biconnected components

- Maintain the biconnected decomposition

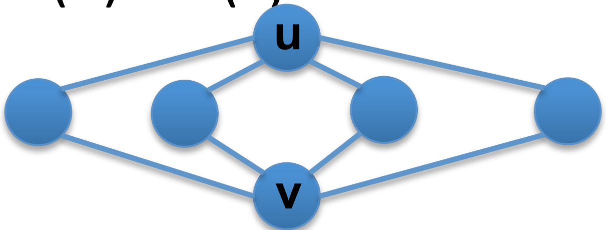


(a) G (b) Π (c) Π'

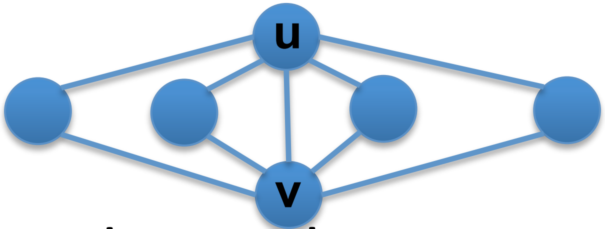
edge **b-d** added

Filtering with identical vertices

- Two types of identical vertices:
  - Type I: u and v are identical vertices if their neighbor lists are same, i.e., $\Gamma(u) = \Gamma(v)$



  - Type II: u and v are identical vertices if their neighbor lists are same and they are also connected, i.e., $\{u\} \cup \Gamma(u) = \{v\} \cup \Gamma(v)$



- If u and v are identical vertices, their cc are the same
  - Same breadth-first search trees!

Filtering with identical vertices

- Let V_{ID} be a subset of V and it's a vertex class containing type-I or type-II identical vertices. Then cc values of all the vertices in V_{ID} are equal
  - Applying SSSP from only one of them is enough!

- Type-I and type-II identical vertices are found by simply hashing the neighbor lists

SSSP Hybridization

Wexner Medical Center

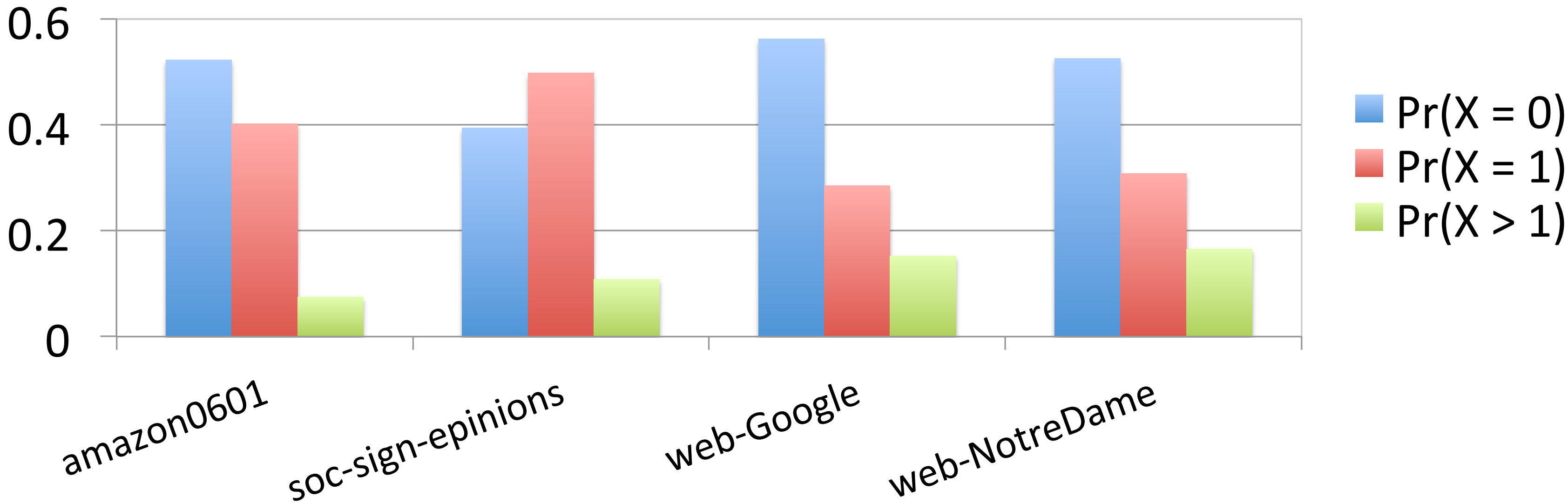
- BFS can be done in two ways:
  - Top-down: Uses the vertices in distance **k** to find the vertices in distance **k+1**
  - Bottom-up: After all distance **k** vertices are found, all other unprocessed vertices are processed to see if they are neighbor
  - Top-down is expected to be better for small **k** values
  - Following the idea of Beamer et al. [SC'12], we apply hybrid approach
    - Simply compare the # of edges to be processed at level **k**
    - Choose the cheaper option

Experiments

- The techniques are evaluated on different sizes and types of large real-world social networks

Graph		
name	$\|V\|$	$\|E\|$
hep-th	8.3K	15.7K
PGPgiantcompo	10.6K	24.3K
astro-ph	16.7K	121.2K
cond-mat-2005	40.4K	175.6K
soc-sign-epinions	131K	711K
loc-gowalla	196K	950K
web-NotreDame	325K	1,090K
amazon0601	403K	2,443K
web-Google	875K	4,322K
wiki-Talk	2,394K	4,659K
DBLP-coauthor	1,236K	9,081K

Probability Distribution



- Bars show the distribution of random variable of level differences into three cases when an edge is inserted

Speedups

- Random insertions for 10 graphs
- Real insertions for DBLP-coauthor graph
- Speedups are w.r.t. full cc computation

~100 times better

real temporal data shows larger speedups

Graph	Time (secs) CC	CC-B	CC-BL	CC-BLI	CC-BLIH	Speedups CC-B	CC-BL	CC-BLI	CC-BLIH	Filter time (secs)
hep-th	1.413	0.317	0.057	0.053	0.048	4.5	24.8	26.6	29.4	0.001
PGPgiantcompo	4.960	0.431	0.059	0.055	0.045	11.5	84.1	89.9	111.2	0.001
astro-ph	14.567	9.431	0.809	0.645	0.359	1.5	18.0	22.6	40.5	0.004
cond-mat-2005	77.903	39.049	5.618	4.687	2.865	2.0	13.9	16.6	27.2	0.010
Geometric mean	9.444	2.663	0.352	0.306	0.217	3.5	26.8	30.7	43.5	0.003
soc-sign-epinions	778.870	257.410	20.603	19.935	6.254	3.0	37.8	39.1	124.5	0.041
loc-gowalla	2,267.187	1,270.820	132.955	135.015	53.182	1.8	17.1	16.8	42.6	0.063
web-NotreDame	2,845.367	579.821	118.861	83.817	53.059	4.9	23.9	33.9	53.6	0.050
amazon0601	14,903.080	11,953.680	540.092	551.867	298.095	1.2	27.6	27.0	50.0	0.158
web-Google	65,306.600	22,034.460	2,457.660	1,701.249	824.417	3.0	26.6	38.4	79.2	0.267
wiki-Talk	175,450.720	25,701.710	2,513.041	2,123.096	922.828	6.8	69.8	82.6	190.1	0.491
DBLP-coauthor	115,919.518	18,501.147	288.269	251.557	252.647	6.2	402.1	460.8	458.8	0.530
Geometric mean	13,884.152	4,218.031	315.777	273.036	139.170	3.2	43.9	50.8	99.7	0.146

biconnected decomposition brings 3x speedup

level differences filtering provides 14x speedup

1.15x speedup with identical vertices

Hybridization brings 2x

Conclusion

- First algorithms for incremental closeness centrality computation
- Update time of a real temporal data is reduced from 1.3 days to 4.2 mins
- Fundamental building block for streaming workloads and centrality management problem
- Future Work:
  - Sampling-based solutions
  - Parallelization
    - A.E. Sarıyuce, E. Saule, K. Kaya, Ümit V. Çatalyürek. **STREAMER: a Distributed Framework for Incremental Closeness Centrality Computation**, IEEE Cluster 2013.

Thanks

Wexner Medical Center

- For more information
  - Email umit@bmi.osu.edu
  - Visit http://bmi.osu.edu/~umit or http://bmi.osu.edu/hpc
- Acknowledgement of Support