Leveraging Big Data: Lecture 11

http://www.cohenwang.com/edith/bigdataclass2013

Instructors:

Edith Cohen

Amos Fiat

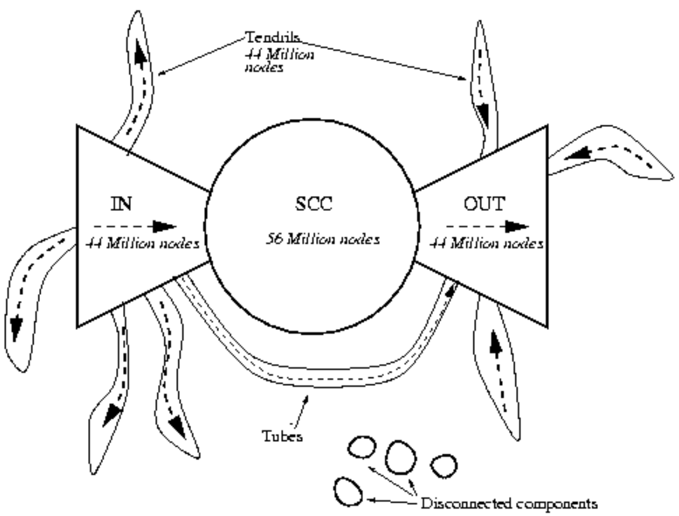
Haim Kaplan

Tova Milo

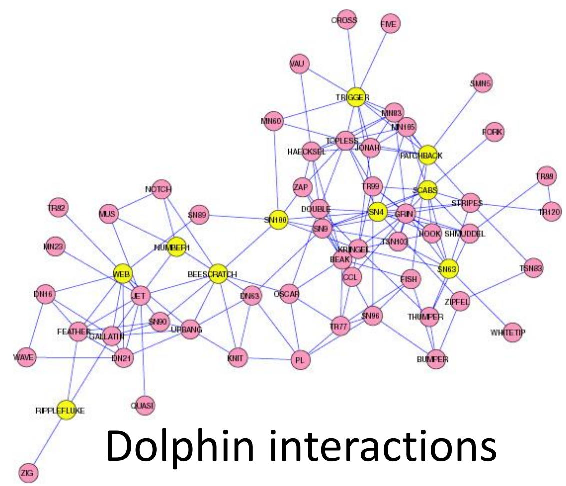
Today’s topics

- Graph datasets
- Mining graphs: which properties we look at and why
- Challenges with massive graphs
- Some techniques/algorithms for very large graphs:
  - Min-Hash sketches of reachability sets
  - All-Distances sketches

Graph Datasets: Represent relations between “things”



Bowtie structure of the Web Broder et. al. 2001



Dolphin interactions

Graph Datasets

- Hyperlinks (the Web)
- Social graphs (Facebook, Twitter, LinkedIn,…)
- Email logs, phone call logs , messages
- Commerce transactions (Amazon purchases)
- Road networks
- Communication networks
- Protein interactions
- …

Properties

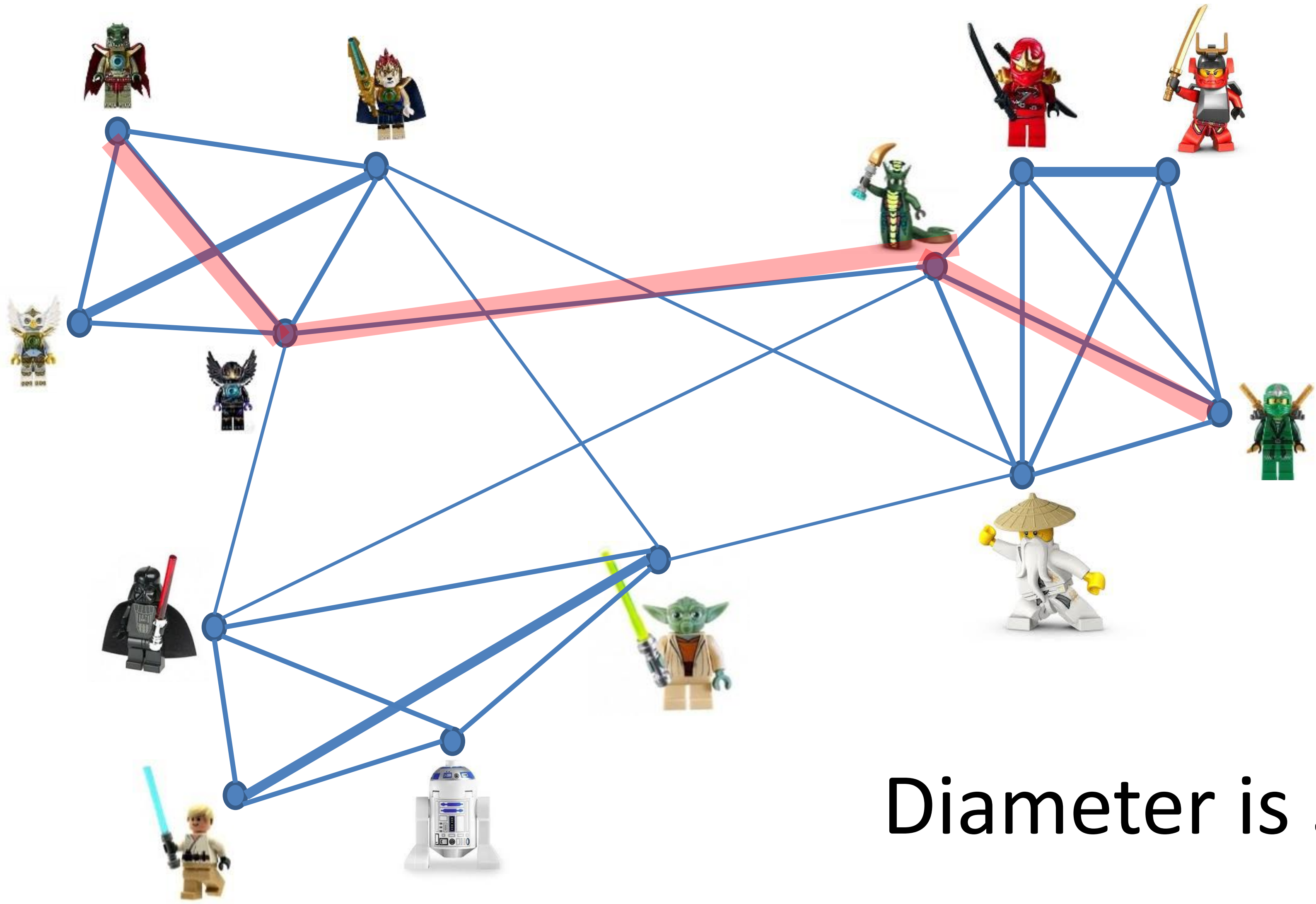
- Directed/Undirected
- Snapshot or with time dimension (dynamic)
- One or more types of entities (people, pages, products)
- Meta data associated with nodes
- Some graphs are really large: billions of edges for Facebook and Twitter graphs

Mining the link structure:

Node/Network-level properties

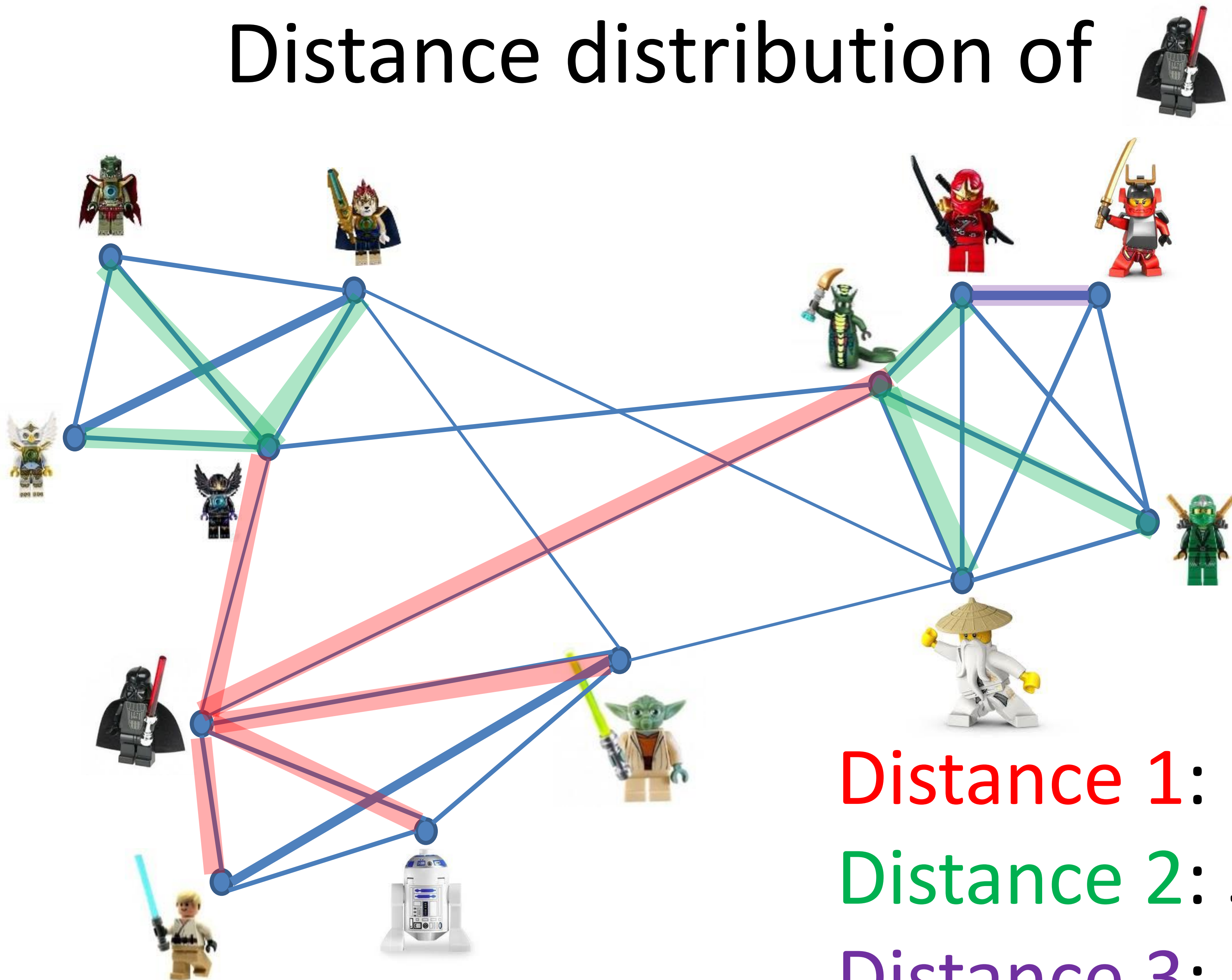
- Connected/Strongly connected components
- Diameter (longest shortest s-t path)
- Effective diameter (90% percentile of pairwise distance)
- Distance distribution (number of pairs within each distance)
- Degree distribution
- Clustering coefficient: Ratio of the number of closed triangles to open triangles.

Diameter



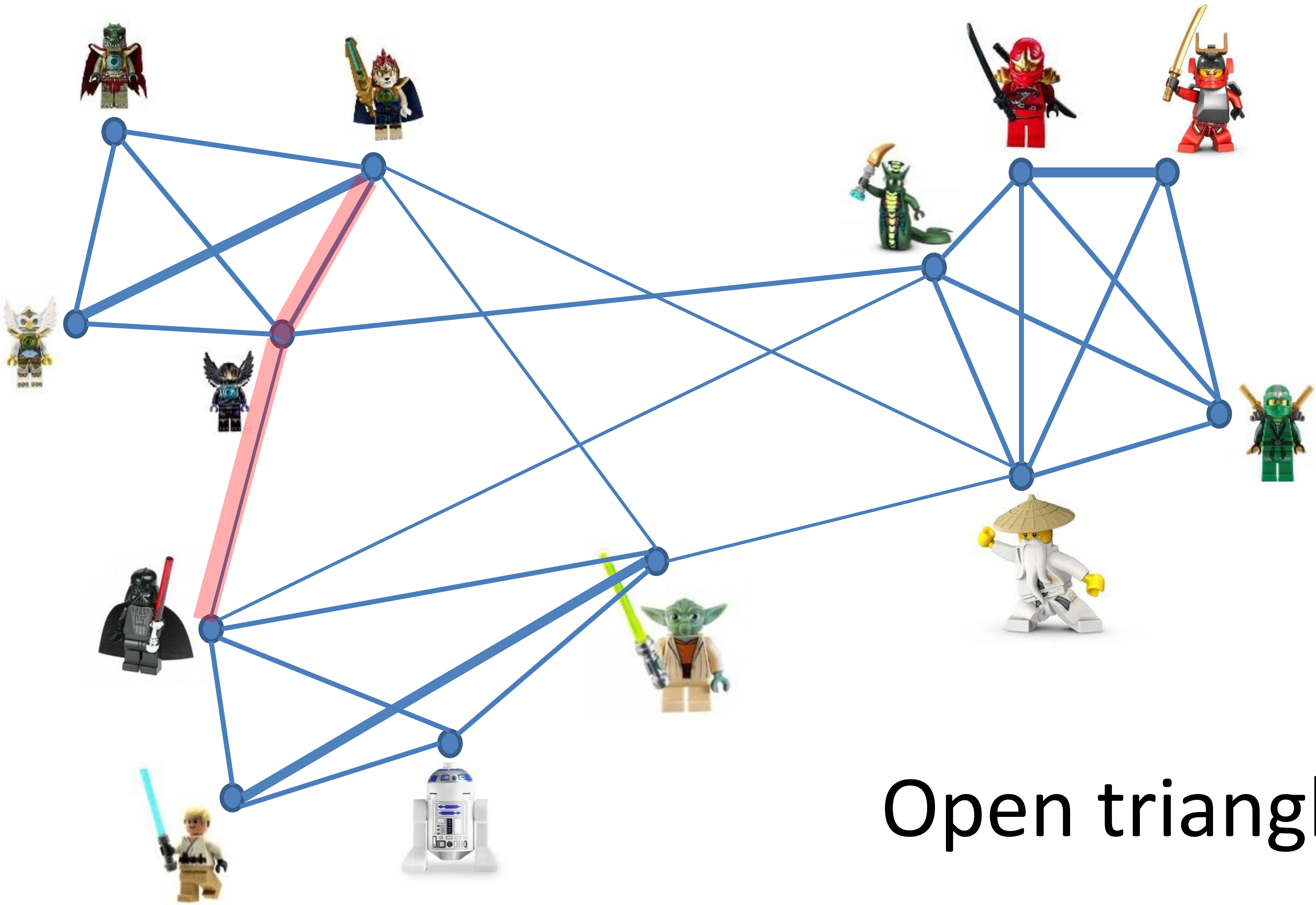
Diameter is 3

Distance distribution of



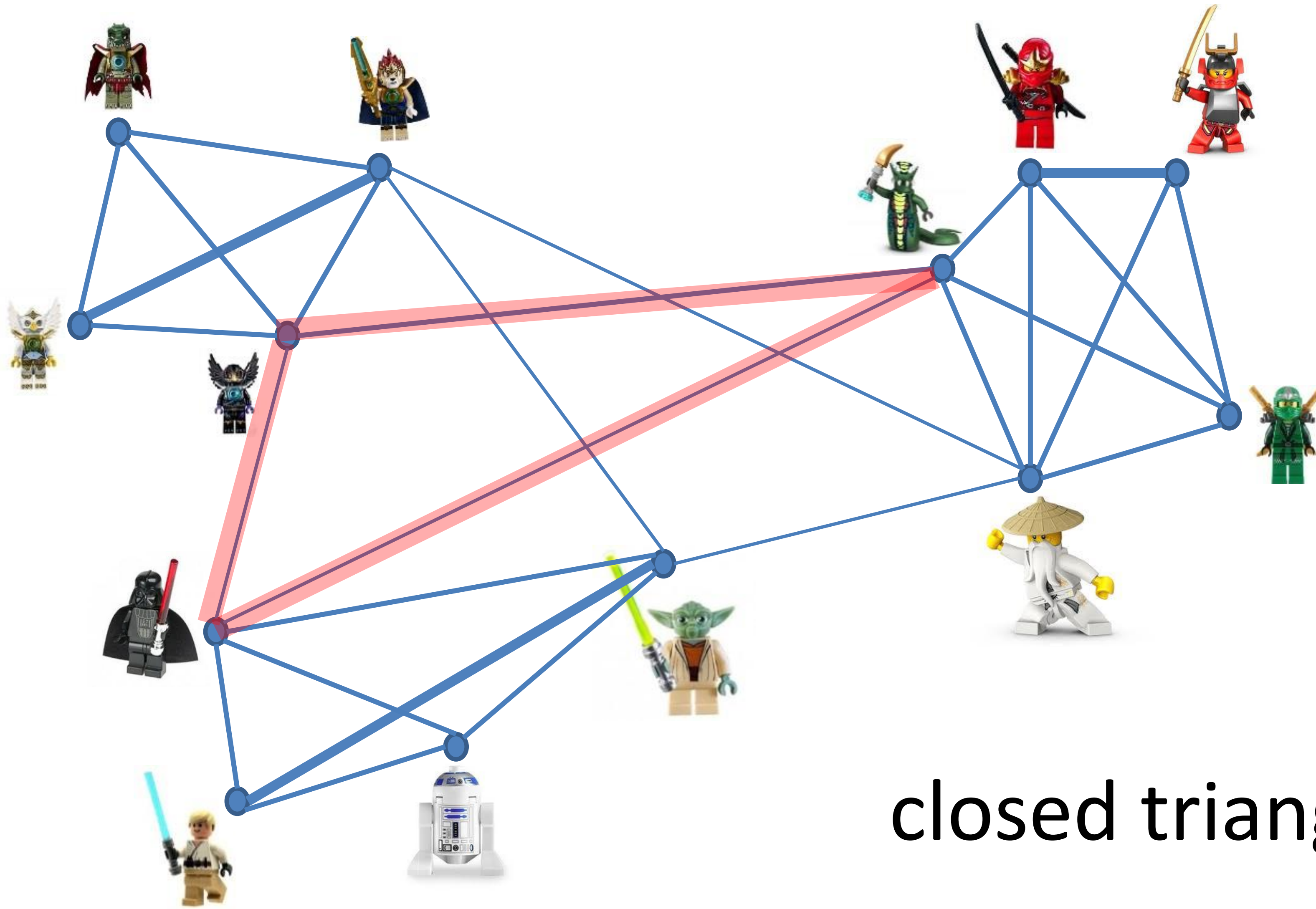
Distance 1: 5
Distance 2: 5
Distance 3: 1

Triangles



Open triangle

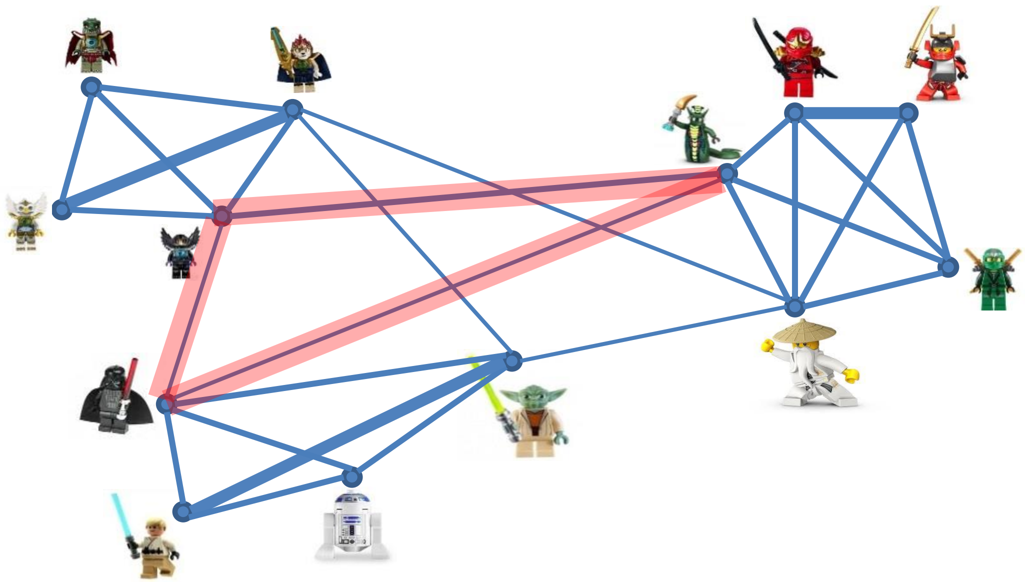
Triangles



closed triangle

Triangles

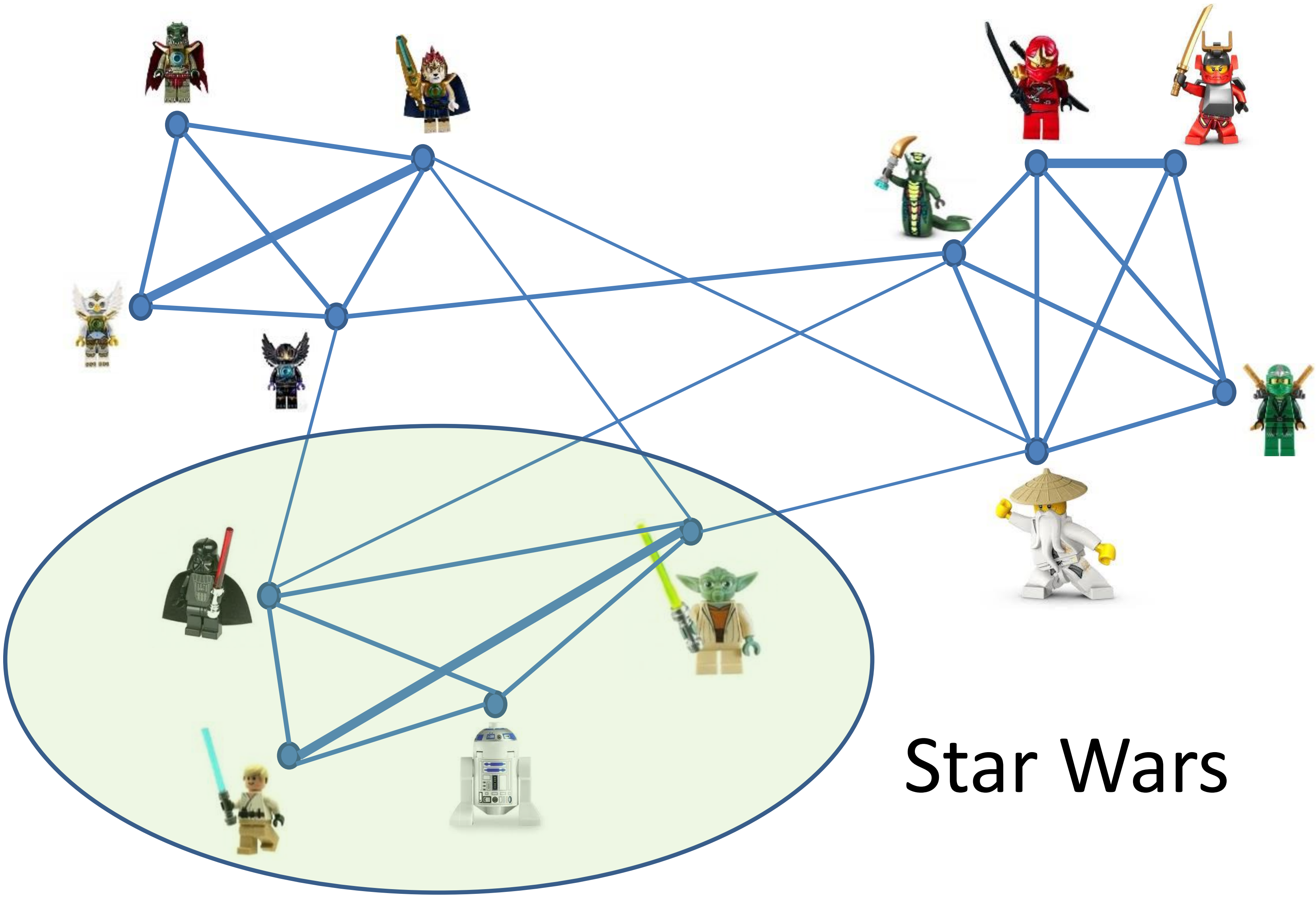
- Social graphs have many more closed triangle than random graphs
- “Communities” have more closed triangles



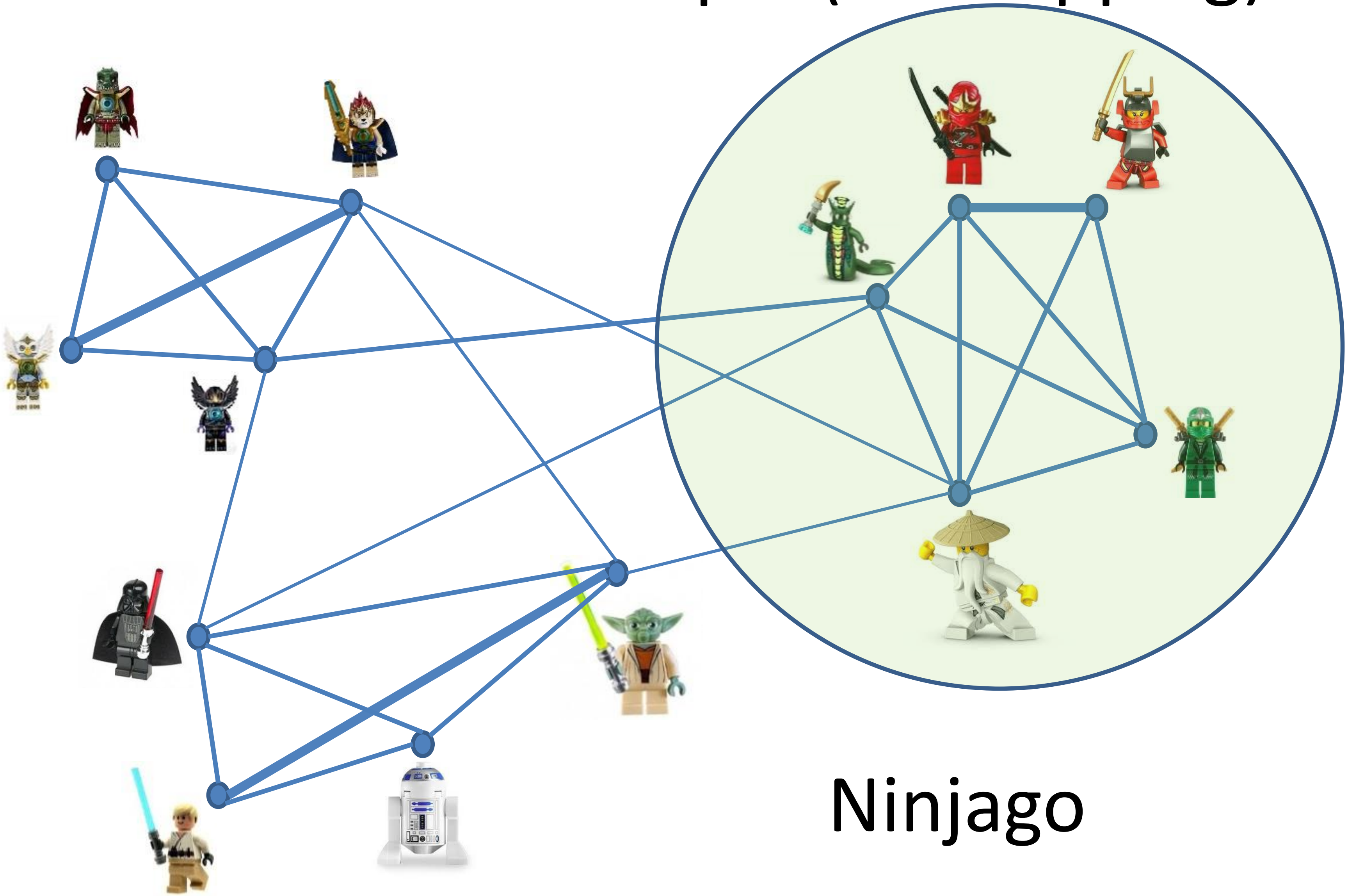
…Mining the link structure

- **Centrality** (who are the most important nodes?)
- **Similarity** of nodes (link prediction, targeted ads, friend/product recommendations, Meta-Data completion)
- **Communities**: set of nodes that are more tightly related to each other than to others
- **"cover:"** set of nodes with good coverage (facility location, influence maximization)

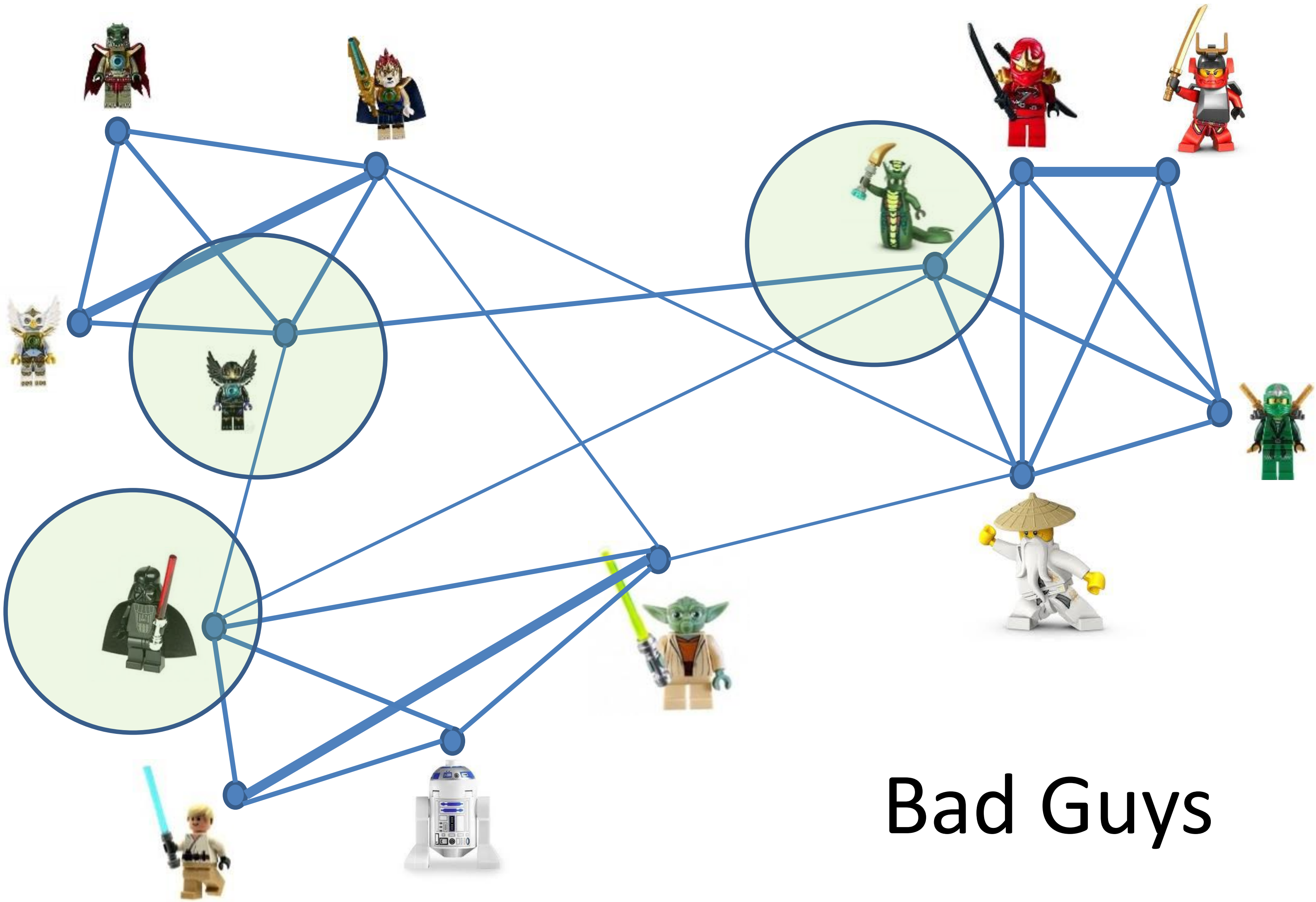
Communities Example (overlapping)



Communities Example (overlapping)

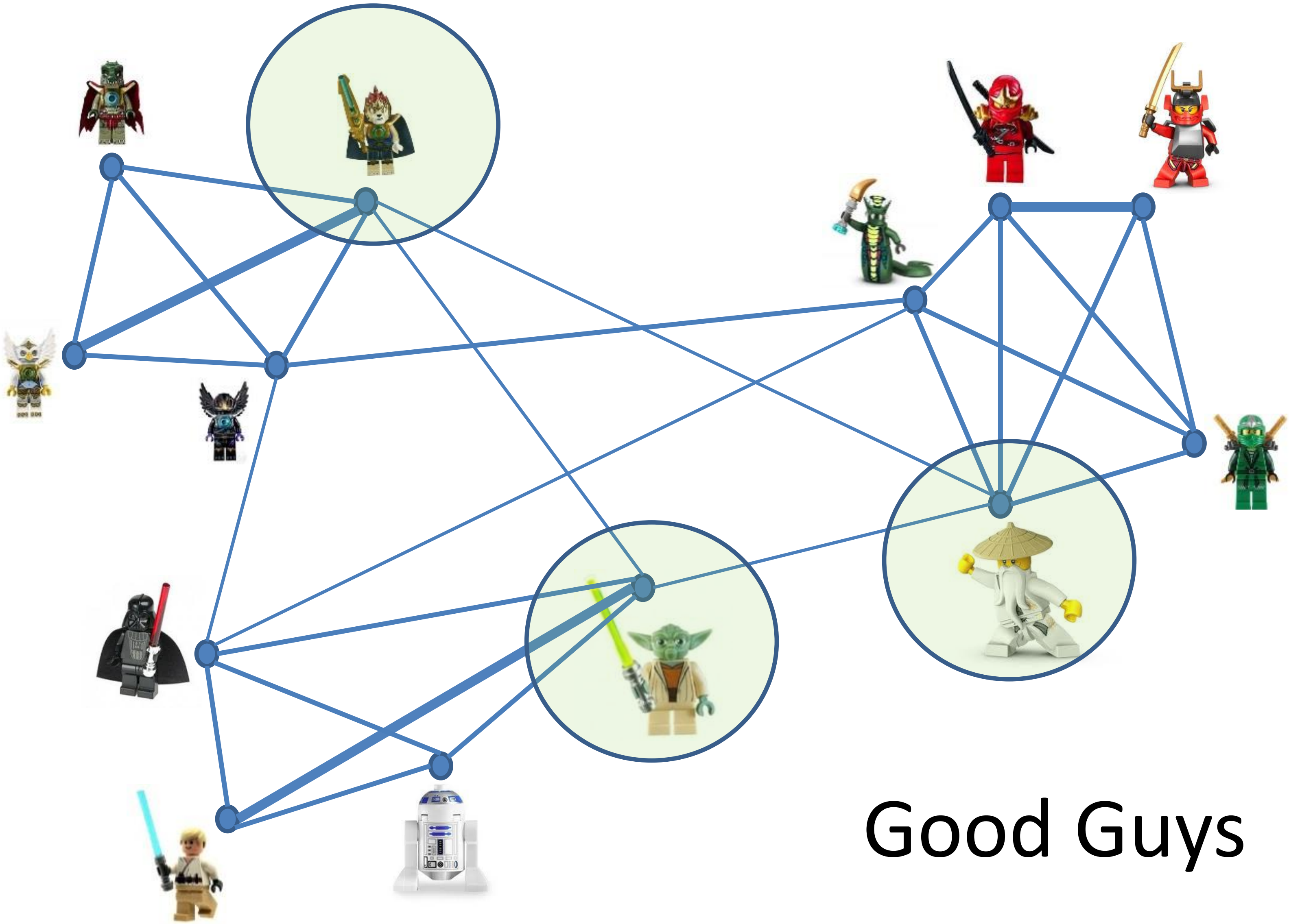


Communities Example (overlapping)



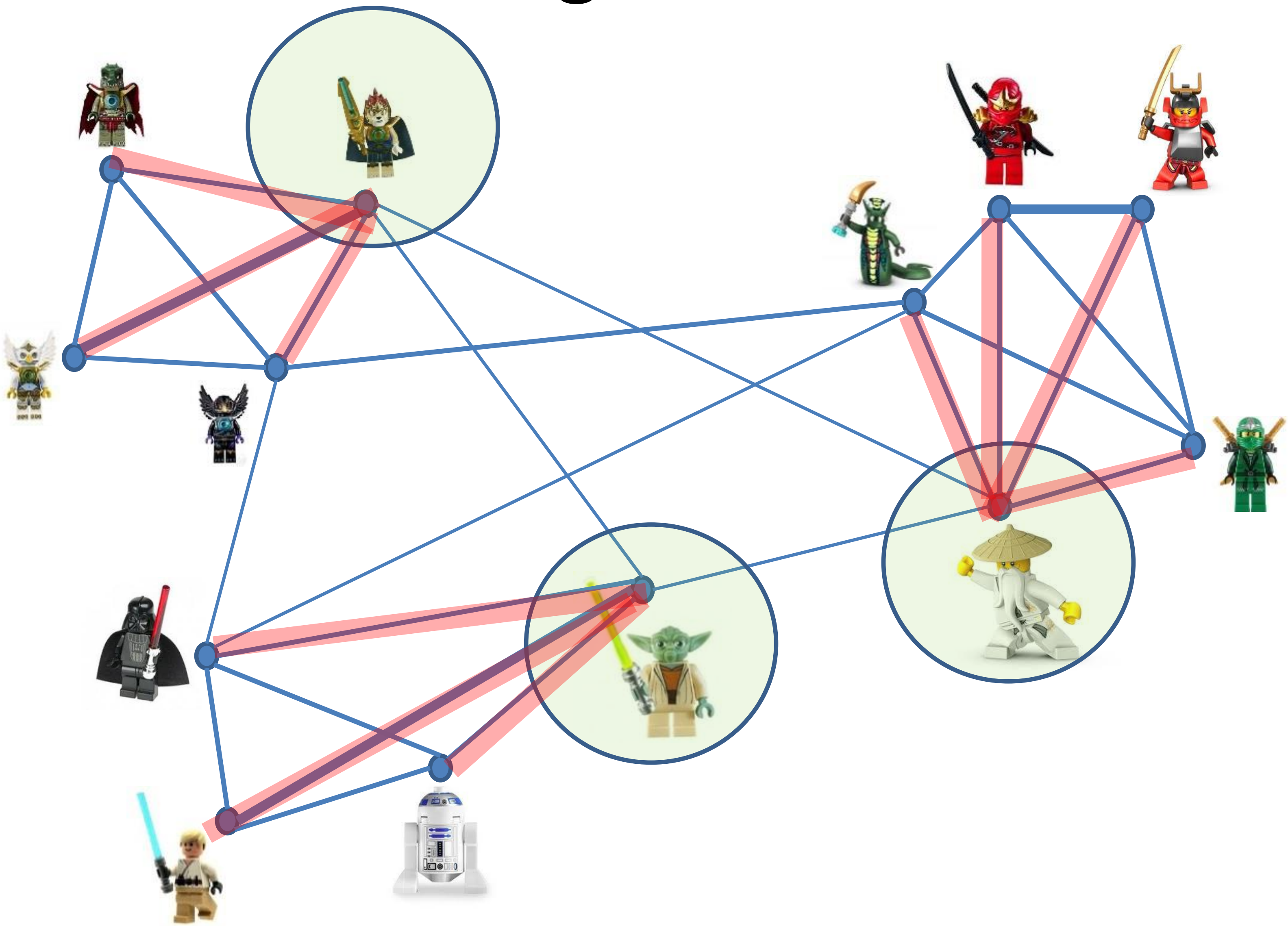
Bad Guys

Communities Example (overlapping)

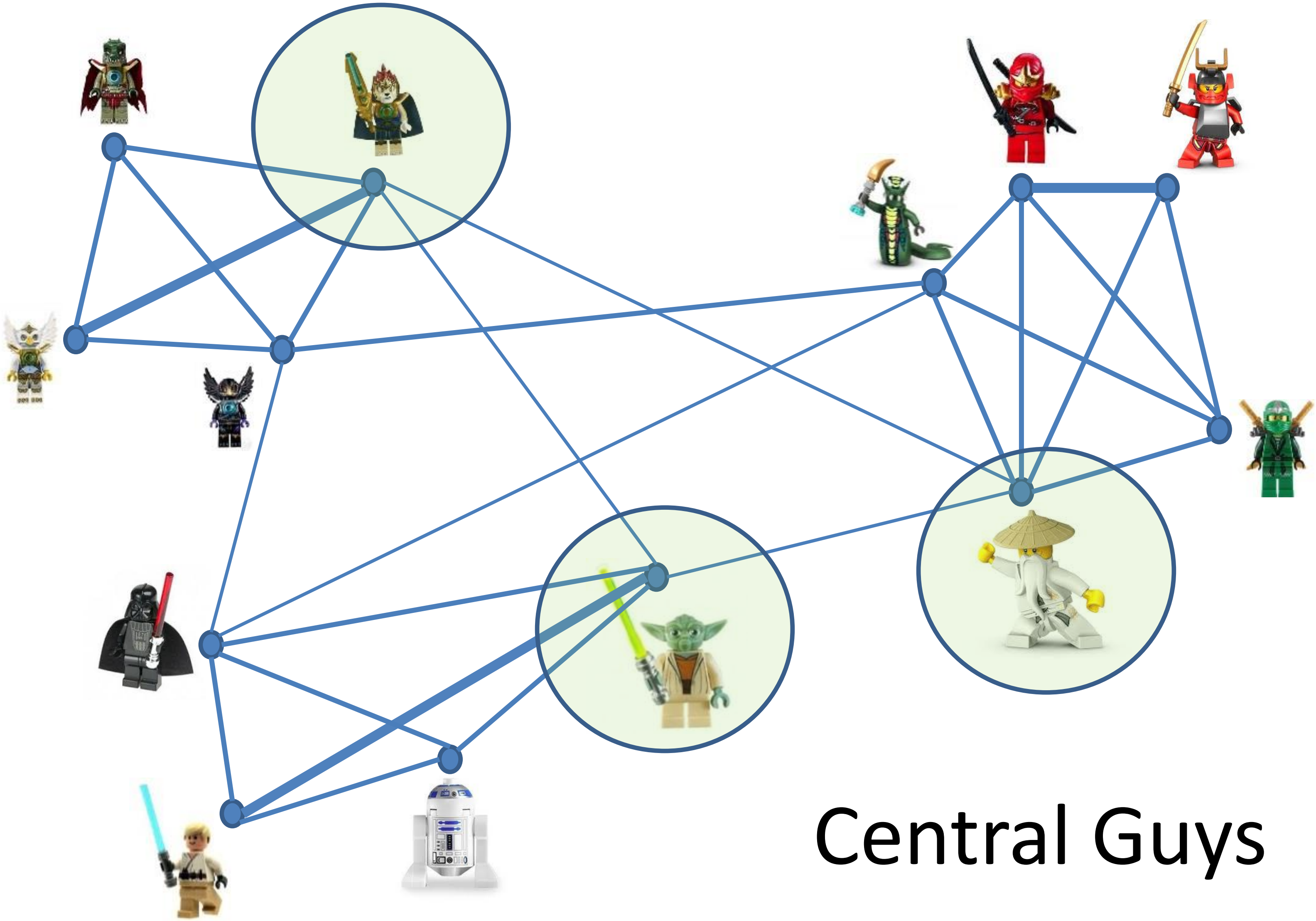


Good Guys

Also a good “cover”



Centrality



Central Guys

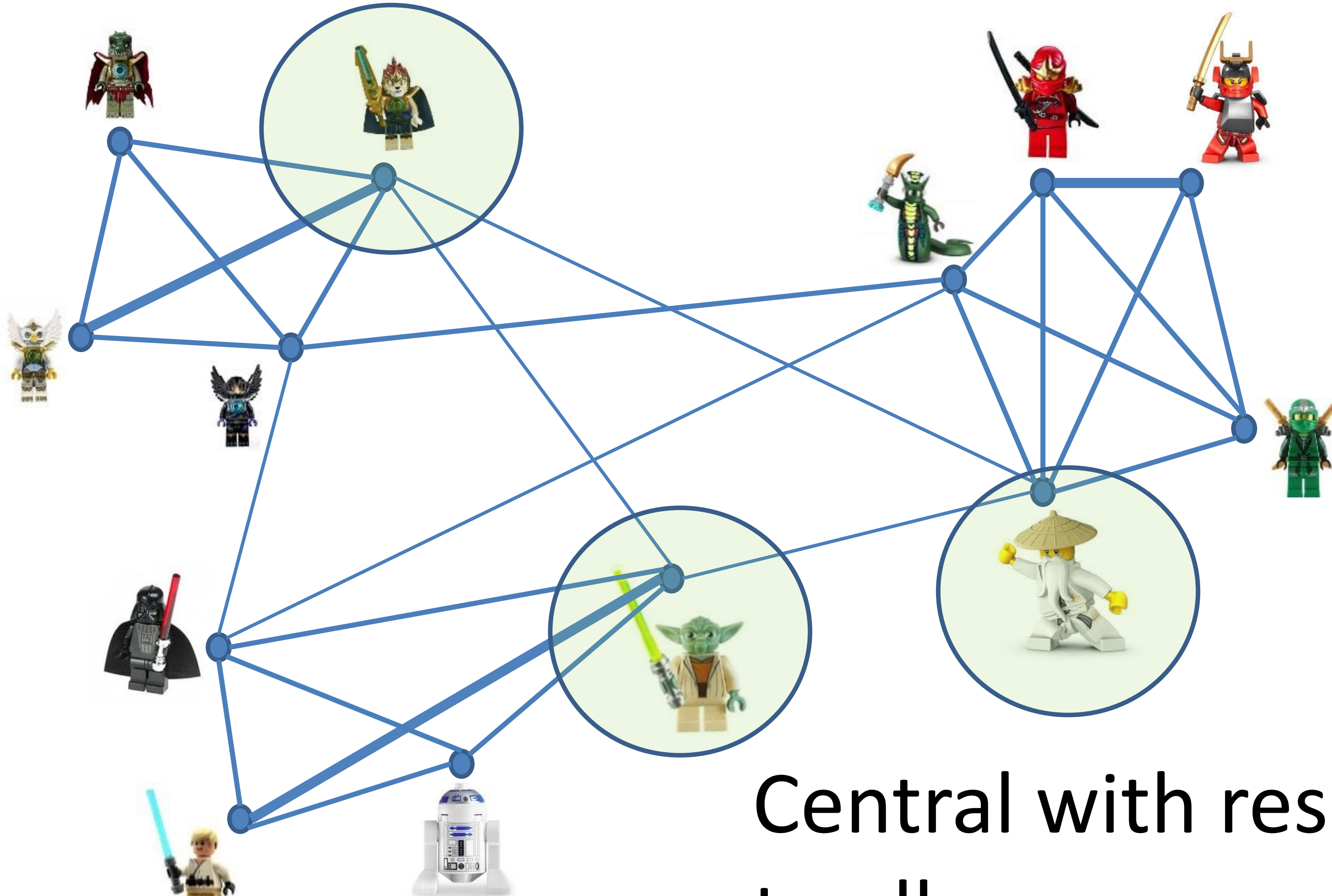
Centrality

Which are the most important nodes ?

… answer depends on what we want to capture:

- **Degree (in/out):** largest number of followers, friends. Easy to compute locally. Spammable.
- **Eigenvalue (PageRank):** Your importance/ reputation recursively depend on that of your friends
- **Betweenness:** Your value as a "hub" -- being on a shortest path between many pairs.
- **Closeness:** Centrally located, able to quickly reach/infect many nodes.

Centrality



Central with respect to all measures

Computing on Very Large Graphs

- Many applications, platforms, algorithms
- **Clusters** (Map Reduce, Hadoop) when applicable
- **iGraph/Pregel** better with edge traversals
- (Semi-)**streaming** : pass(es), keep small info (per-node)
  - **General algorithm design principles** :
    - settle for approximations
    - keep total computation/ communication/ storage "linear" in the size of the data
    - Parallelize (minimize chains of dependencies)
    - Localize dependencies

Next: Node sketches
(this lecture and the next one)

- Min-Hash sketches of reachability sets
- All-distances sketches (ADS)
- Connectivity sketches (Guha/McGregor)

Sketching:

- Compute a sketch for each node, efficiently
- From sketch(es) can estimate properties that are "harder" to compute exactly

Review (lecture 2): Min Hash Sketches

- "Items" V
- Random hash function $h: V \to [0,1]$
- For a subset $N \subset V$ we get a sketch $s(N)$
- From $s(N)$ we can:
  - Estimate cardinality $|N|$,
  - Obtain a random sample of N,
  - Estimate similarity, union, sketch merged sets
- Basic sketch ($k = 1$) : maintain the minimum $h(N)$

Review: Min-Hash Sketches

k values $\boldsymbol{y_1, y_2, \ldots, y_k}$ from the range of the hash function (distribution).

k-mins sketch: Use k "independent" hash functions: $h_1, h_2, \ldots, h_k$

Track the respective minimum $y_1, y_2, \ldots, y_k$ for each function.

Bottom-k sketch: Use a single hash function: h

Track the k smallest values $y_1, y_2, \ldots, y_k$

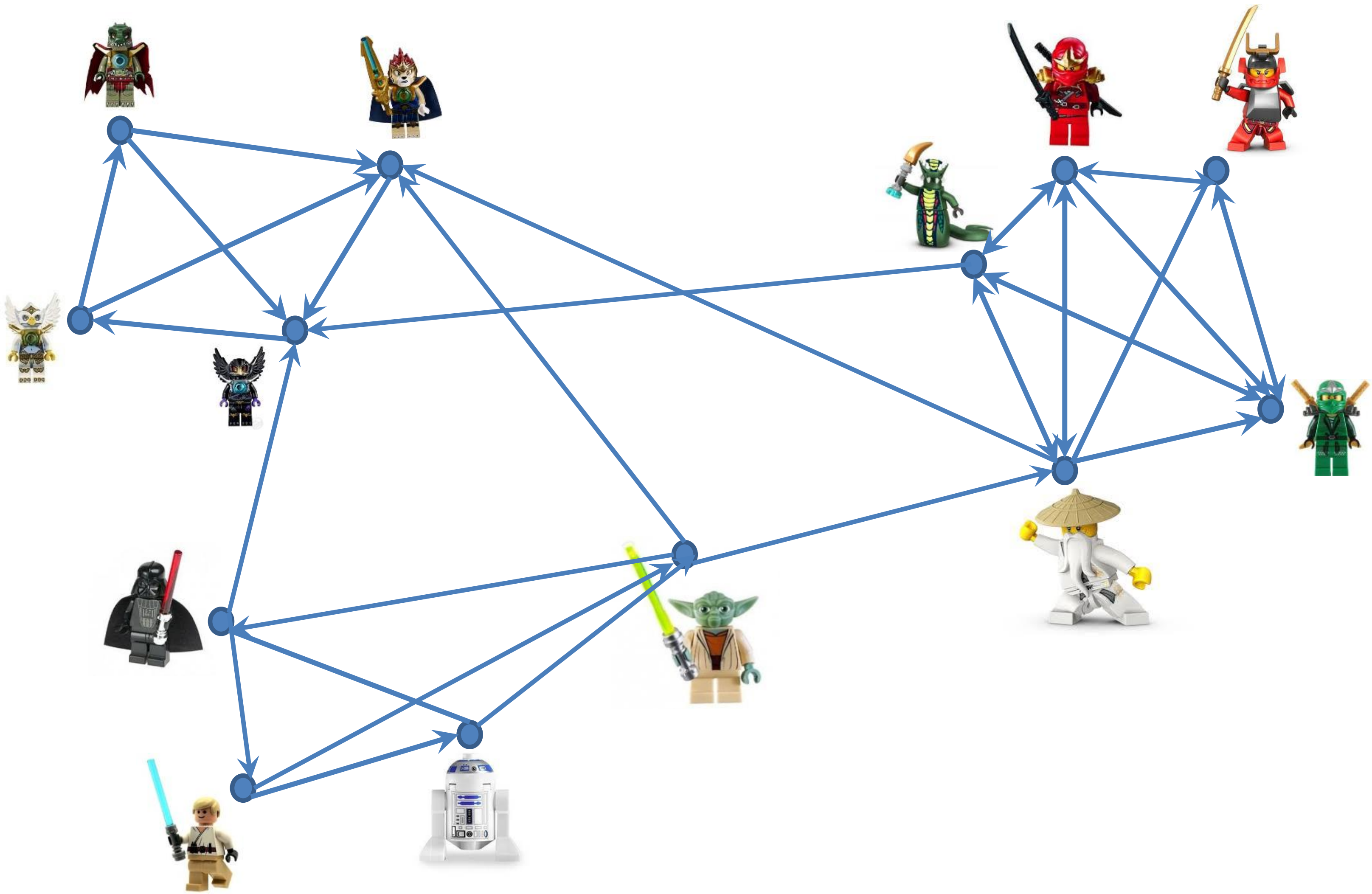
k-partition sketch: Use a single hash function: h'

Use the first $\log_2 k$ bits of $h'(x)$ to map x uniformly to one of k parts. Call the remaining bits $h(\mathrm{x})$.

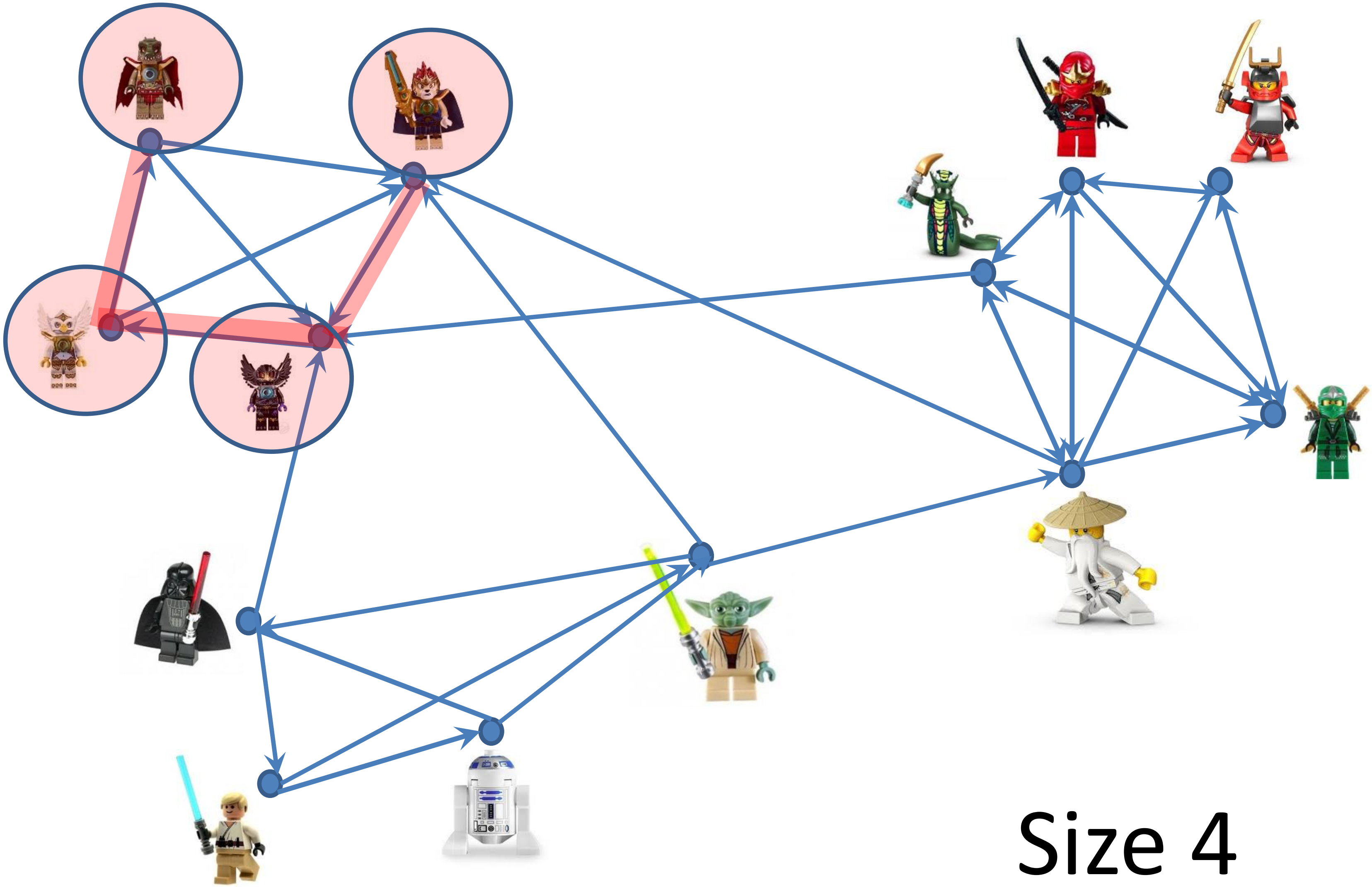
For $i = 1, \ldots, k$: Track the minimum hash value y_i of the elements in part i.

➡ All sketches are the same for $k = 1$

Sketching Reachability Sets

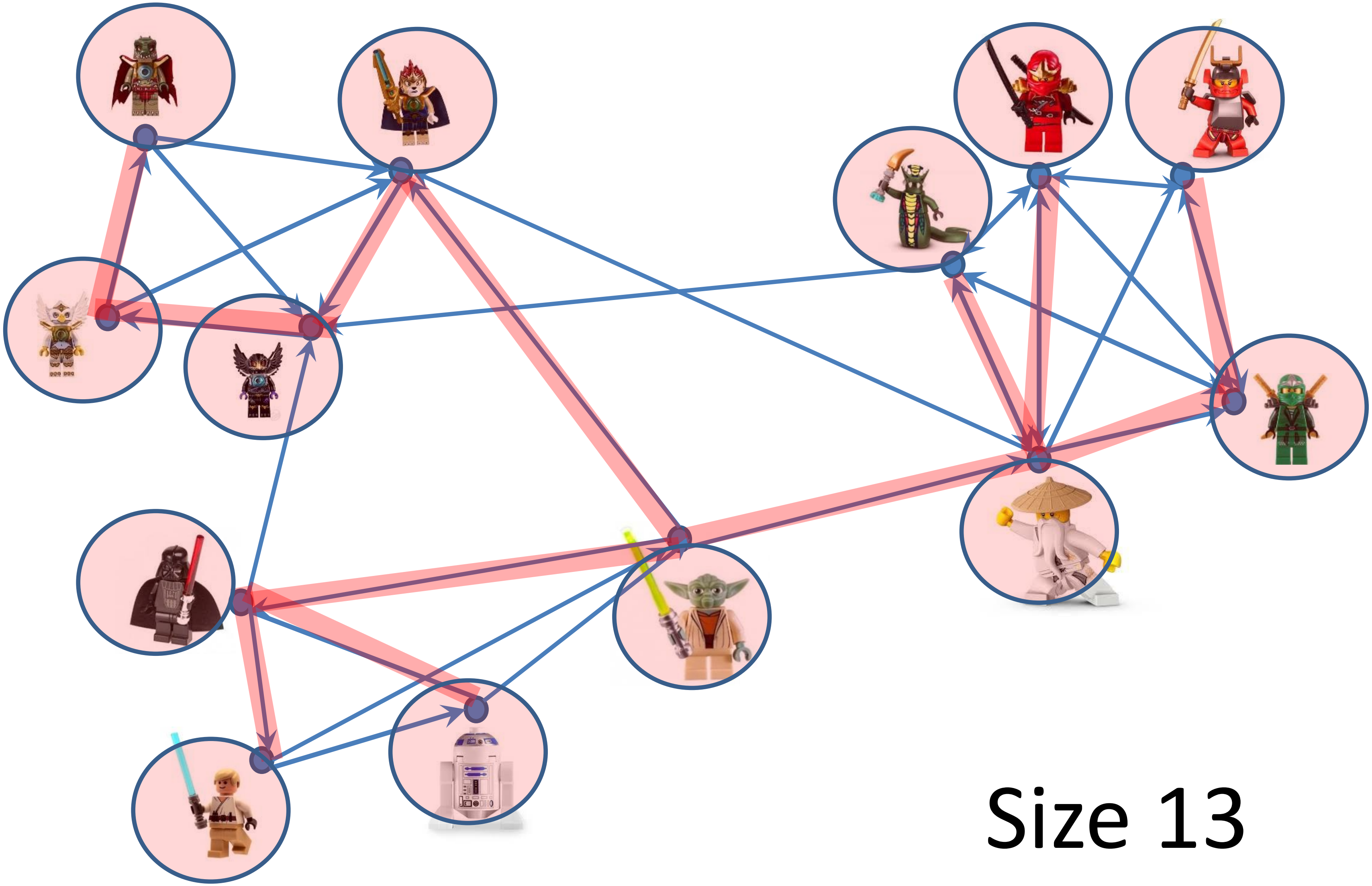


Reachability Set of 



Size 4

Reachability Set of 



Size 13

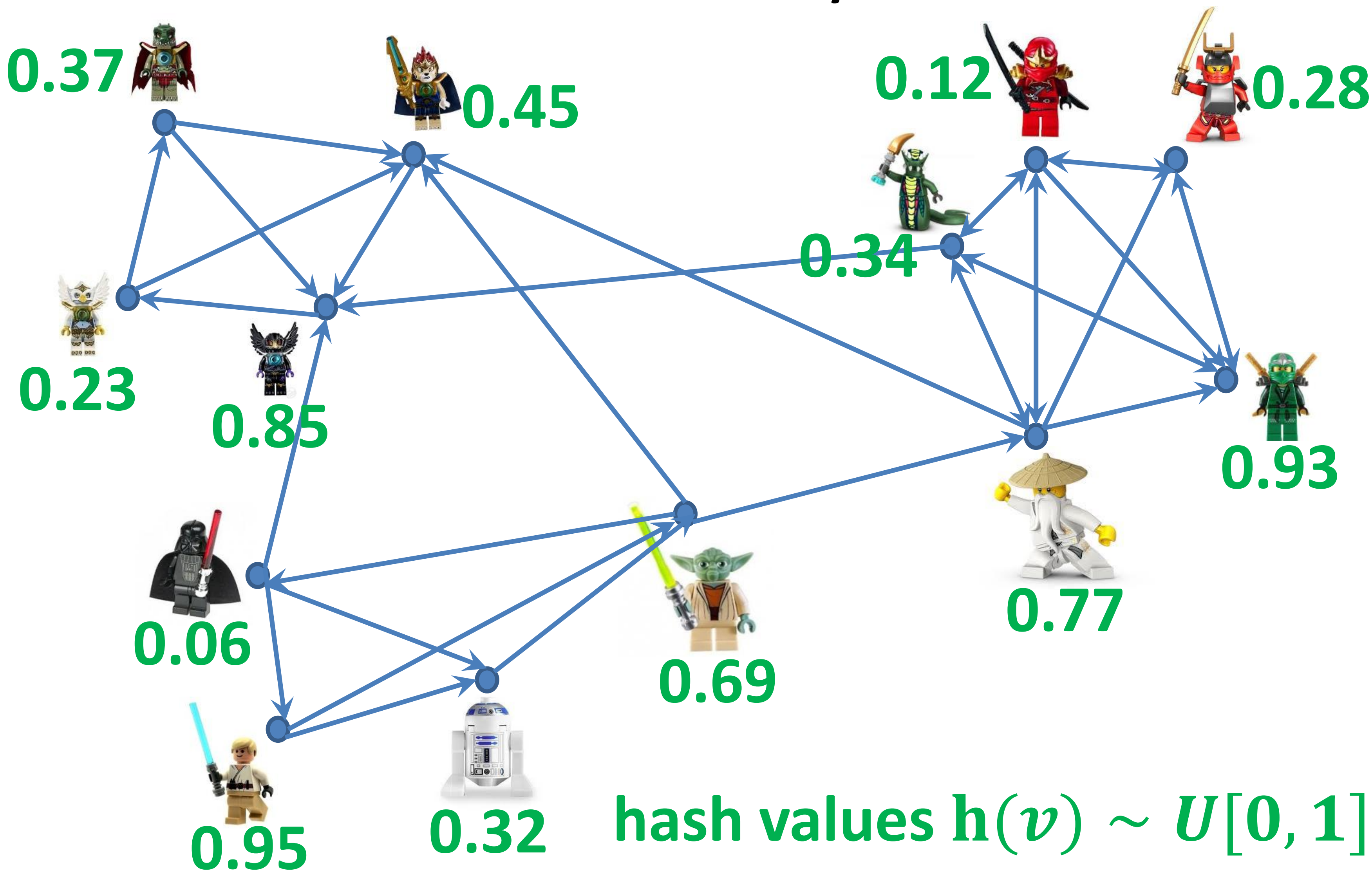
Why sketch reachability sets ?

From reachability sketch(es) we can:

- Estimate cardinality of reachability set
- Get a sample of the reachable nodes
- Estimate relations between reachability sets (e.g., Jaccard similarity)

> Exact **computation** is *costly*: $O(mn)$ with n nodes and m edges, **representation size** is massive: does not scale to large networks!

Min-Hash sketches of all Reachability sets



hash values $\mathbf{h}(v) \sim U[0,1]$

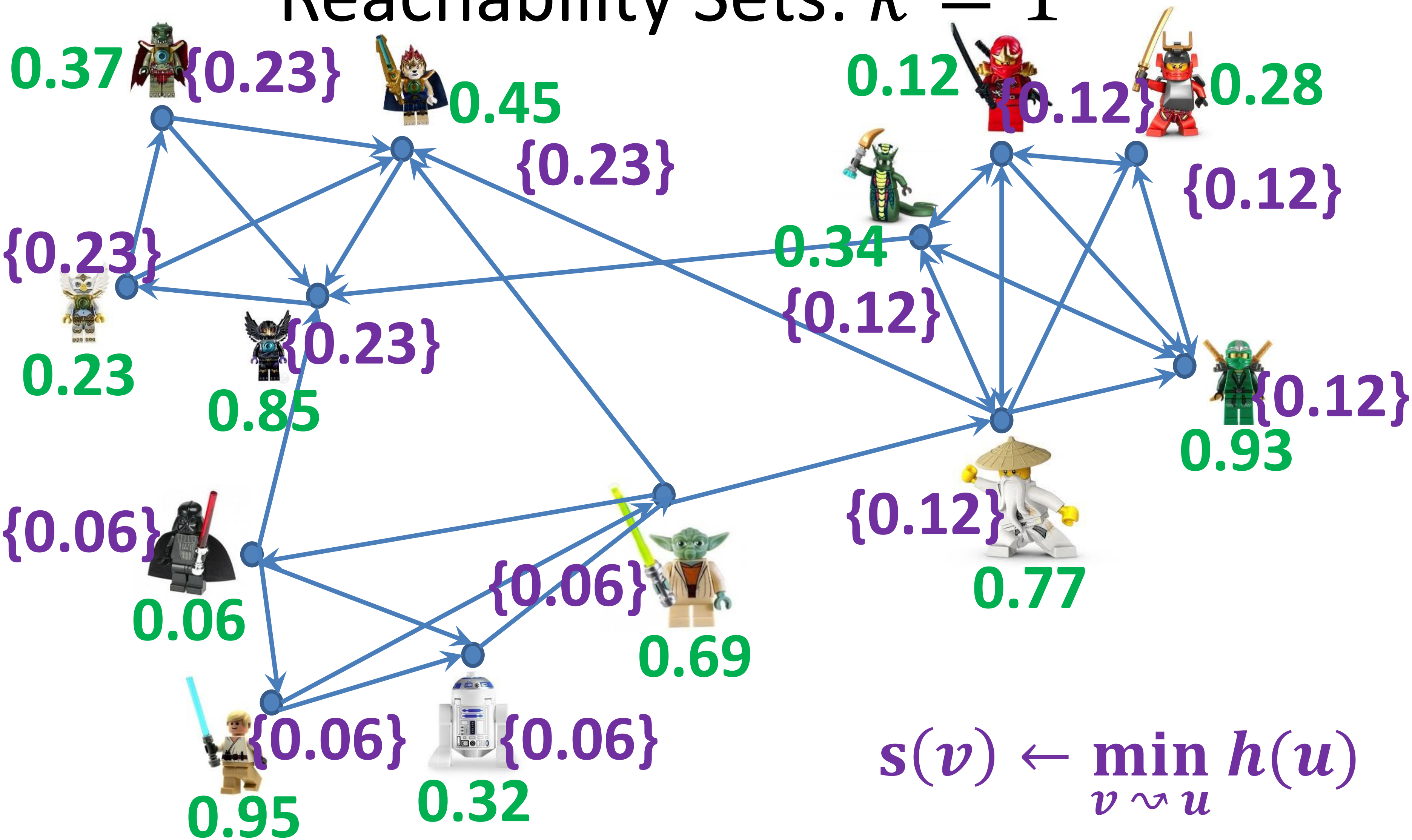
Min-Hash sketches of all Reachability Sets: $k = 1$

For each v: $\mathbf{s}(\boldsymbol{v}) \leftarrow \min_{\boldsymbol{v} \rightsquigarrow \boldsymbol{u}} \boldsymbol{h}(\boldsymbol{u})$

Depending on application, may also want to include node ID in sketch:

$$\operatorname*{argmin}_{\boldsymbol{v} \rightsquigarrow \boldsymbol{u}} \boldsymbol{h}(\boldsymbol{u})$$

Min-Hash sketches of all Reachability Sets: $k = 1$

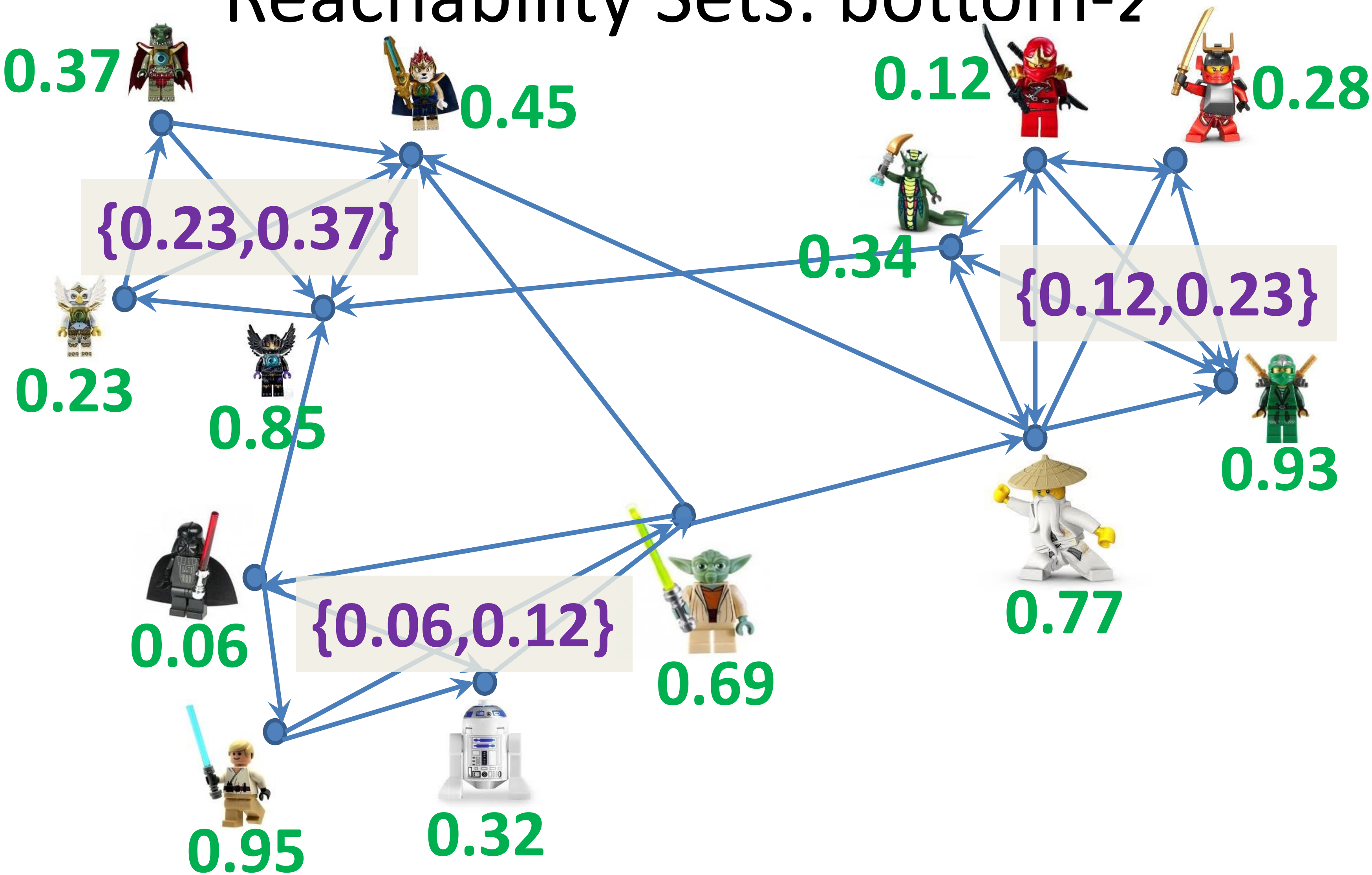


$$\mathbf{s}(v) \leftarrow \min_{v \rightsquigarrow u} h(u)$$

Min-Hash sketches of all Reachability Sets: bottom-2 ($k = 2$)

For each v: $\mathbf{s}(v) \leftarrow \underset{v \rightsquigarrow u}{\mathbf{bottom}\text{–}\mathbf{2}}\, \boldsymbol{h}(\boldsymbol{u})$

Min-Hash sketches of all Reachability Sets: bottom-2



Next: Computing Min-Hash sketches of all reachability sets efficiently

Sketch size for a node: $O(k)$

Total computation $\approx O(km)$

Algorithms/methods:

- Graphs searches (say BFS)
- Dynamic programming/ Distributed

Computing Min-Hash Sketches of all Reachability Sets: $k = 1$ BFS method

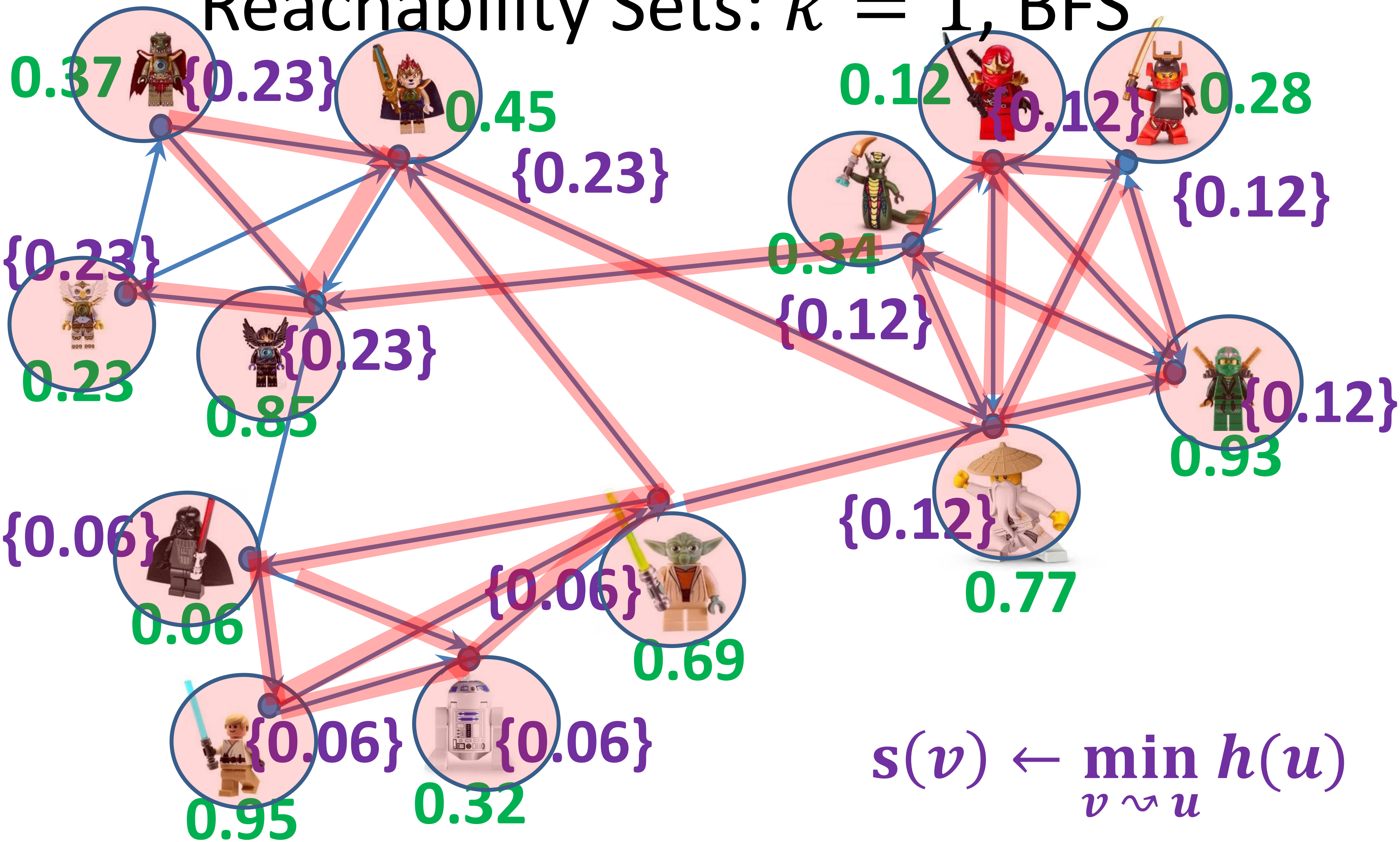
$$\mathbf{s}(\boldsymbol{v}) \leftarrow \min_{\boldsymbol{v} \rightsquigarrow \boldsymbol{u}} \boldsymbol{h}(\boldsymbol{u})$$

Iterate over nodes u by increasing $h(u)$:

Visit nodes v through a reverse search from u:

- **IF** $\mathrm{s}(v) = \emptyset$,
  - $s(v) \leftarrow h(u)$
  - Continue search on $\mathrm{inNeighbors}(v)$
- **ELSE**, truncate search at v

Compute Min-Hash sketches of all Reachability Sets: $k = 1$, BFS



$$\mathbf{s}(v) \leftarrow \min_{v \rightsquigarrow u} h(u)$$

Computing Min-Hash sketches of all reachability sets: $k = 1$ BFS method

Analysis:

Each arc is used exactly once $O(m)$

Parallelizing BFS-based Min-Hash computation

Each graph search depends on all previous ones: seems like we need to perform n searches sequentially.

How can we reduce dependencies ?

Parallelizing BFS-based Min-Hash computation

Idea ($k = 1$):

- Create a super-node of the $n/2$ lowest hash nodes.
- Perform a (reverse) search from super-node and mark all nodes that are accessed.
- Concurrently perform searches:
  - From the lowest-hash $n/2$ nodes (sequentially)
  - From the highest-hash $n/2$ (sequentially). Prune searches **also** at marked nodes

Parallelizing BFS-based Min-Hash computation

Correctness:

- **For the lower $n/2$ hash values**: computation is the same.
- **For the higher $n/2$:**

  We do not know the minimum reachable hash from higher-hash nodes, but we do know it is one of the lower $n/2$ hash values. This is all we need to know for correct pruning.

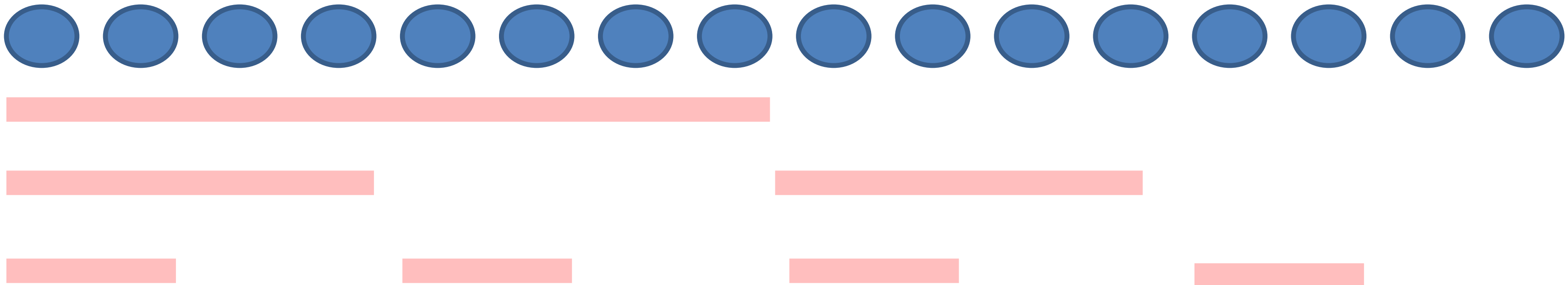
Parallelizing BFS-based Min-Hash computation

- This only gives us $n/2$ instead of n sequential searches.

How can we obtain more parallelism ?

- We **recursively** apply this to each of the lower/higher sets:

Parallelizing BFS-based Min-Hash computation



Nodes ordered by $h(u)$

Super-nodes created in recursion

- The depth of dependencies is at most $\log_2 n$
- The total number of edge traversals can increase by a factor of $\log_2 n$

Computing Min-Hash Sketches of all Reachability Sets: bottom-k, BFS method

Next: Computing sketches using the BFS method for k>1

$$\mathbf{s}(v) \leftarrow \underset{v \leadsto u}{\mathbf{bottom\text{-}}k}\ h(u)$$

Computing Min-Hash Sketches of all Reachability Sets: bottom-k, BFS method

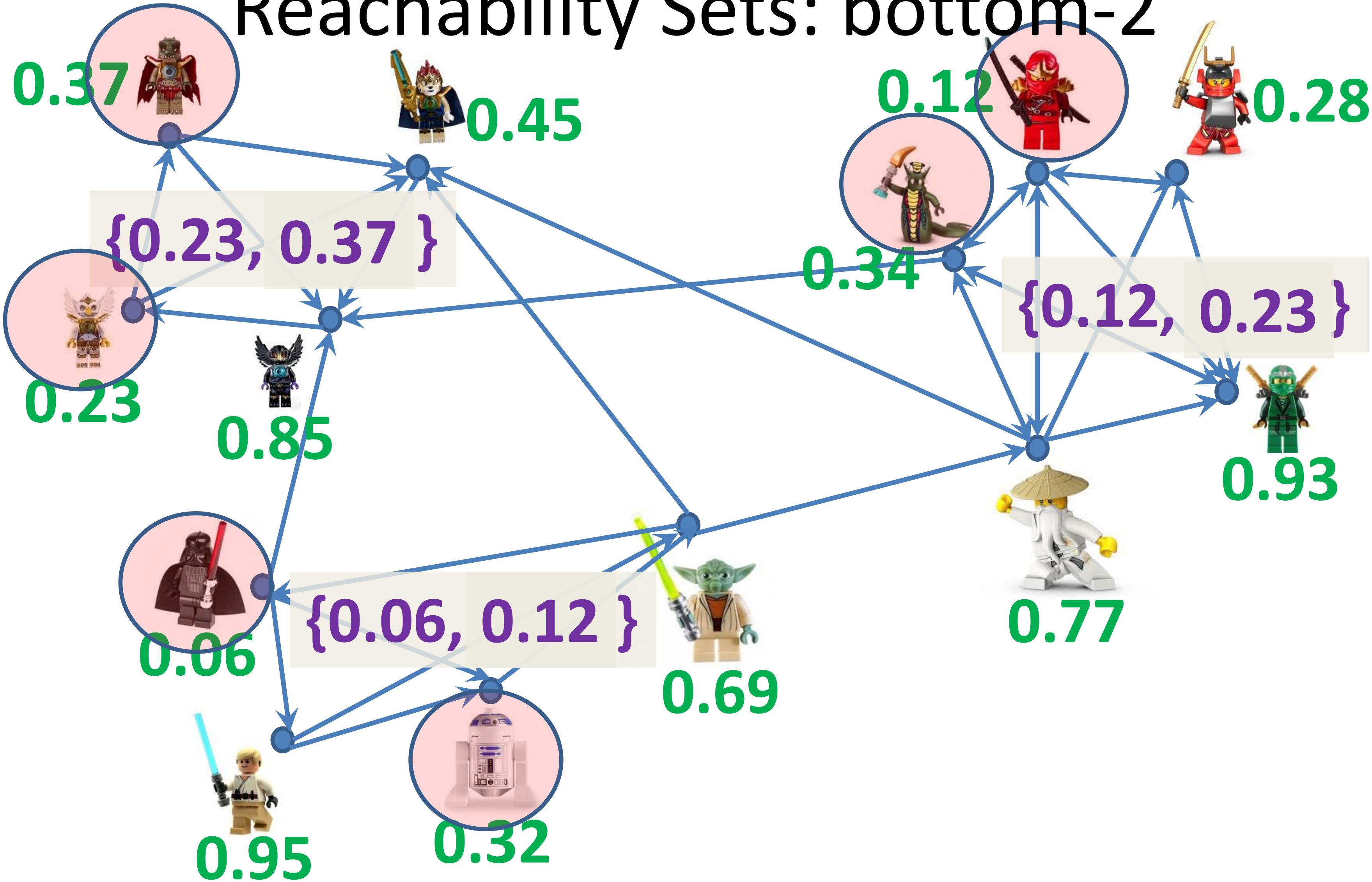
$$\mathbf{s}(v) \leftarrow \underset{v \rightsquigarrow u}{\mathbf{bottom}\text{-}k}\ h(u)$$

Iterate over nodes u by increasing $h(u)$:

Visit nodes v through a reverse search from u:

- **IF** $|\mathrm{s}(v)| < k$,
  - $s(v) \leftarrow s(v) \cup \{h(u)\}$
  - Continue search on $\mathrm{inNeighbors}(v)$
- **ELSE**, truncate search at v

Min-Hash sketches of all Reachability Sets: bottom-2



Computing Min-Hash Sketches of all Reachability Sets: $k = 1$ Distributed (DP)

Next: back to $k = 1$.

We present **another method** to compute the sketches. The algorithm has fewer dependencies. It is specified for each node. It is suitable for computation that is:

- Distributed, Asynchronous
- Dynamic Programming (DP)
- Multiple passes on the set of arcs

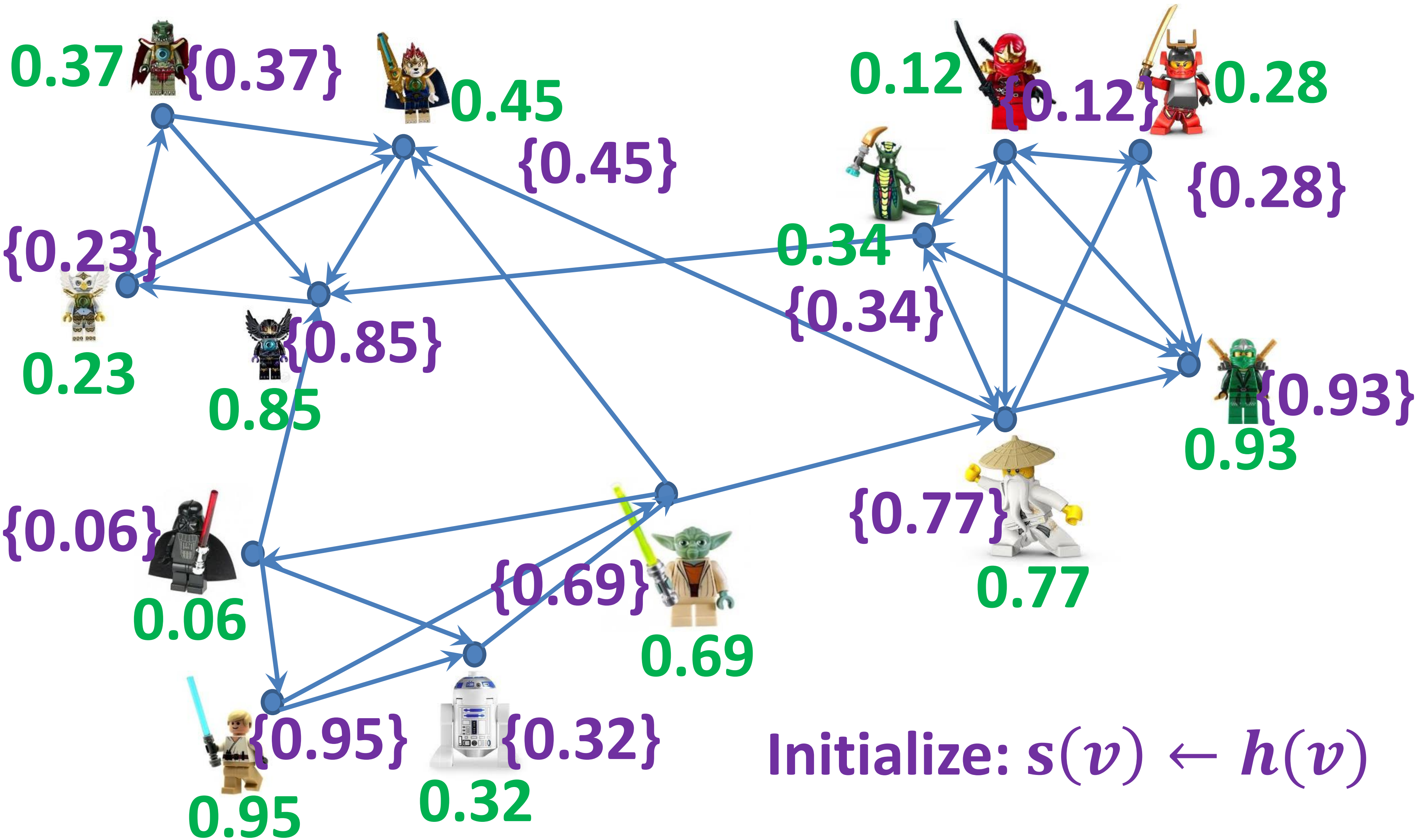
Computing Min-Hash Sketches of all Reachability Sets: $k = 1$ Distributed (DP)

$$\mathbf{s}(\boldsymbol{v}) \leftarrow \min_{\boldsymbol{v} \rightsquigarrow \boldsymbol{u}} \boldsymbol{h}(\boldsymbol{u})$$

Initialize $\mathbf{s}(\boldsymbol{v}) \leftarrow \boldsymbol{h}(\boldsymbol{v})$

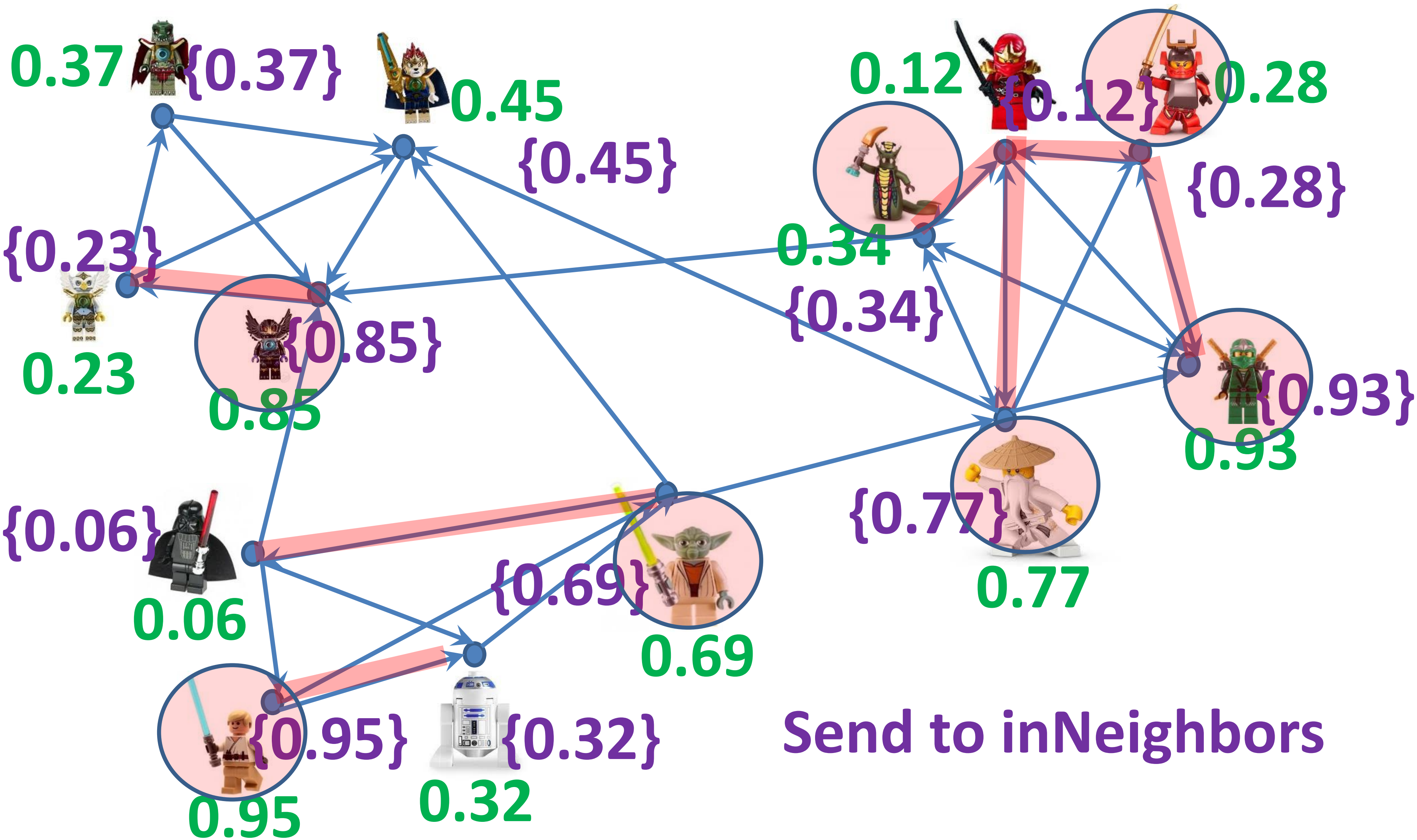
- **IF** $\mathrm{s}(v)$ is initialized/updated, send $s(v)$ to $\mathrm{inNeighbors}(v)$
- **IF** value x is received from neighbor:
  - $s(v) \leftarrow \min\{s(v), x\}$

DP computation of Min-Hash sketches $k = 1$



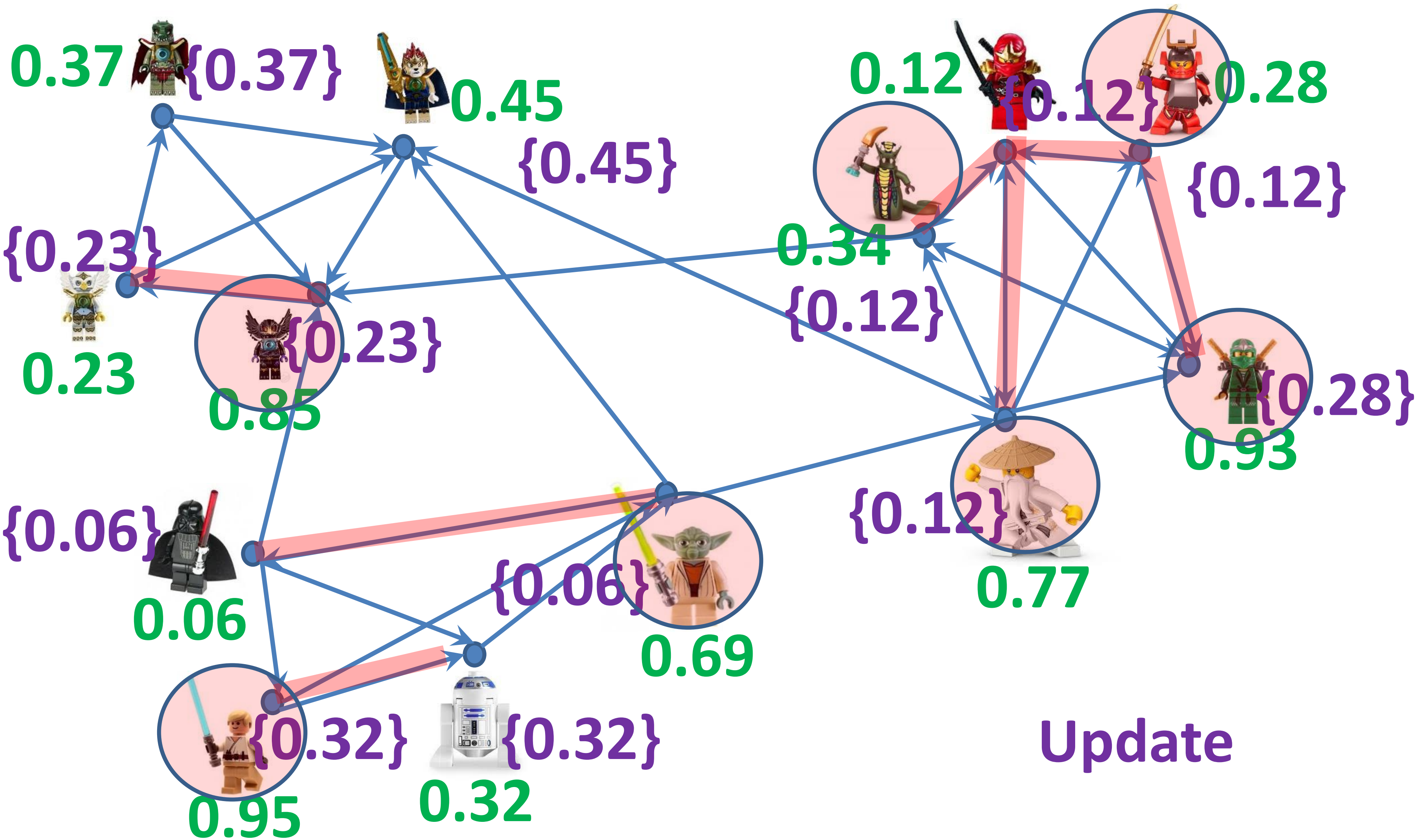
Initialize: $s(v) \leftarrow h(v)$

DP computation of Min-Hash sketches $k = 1$

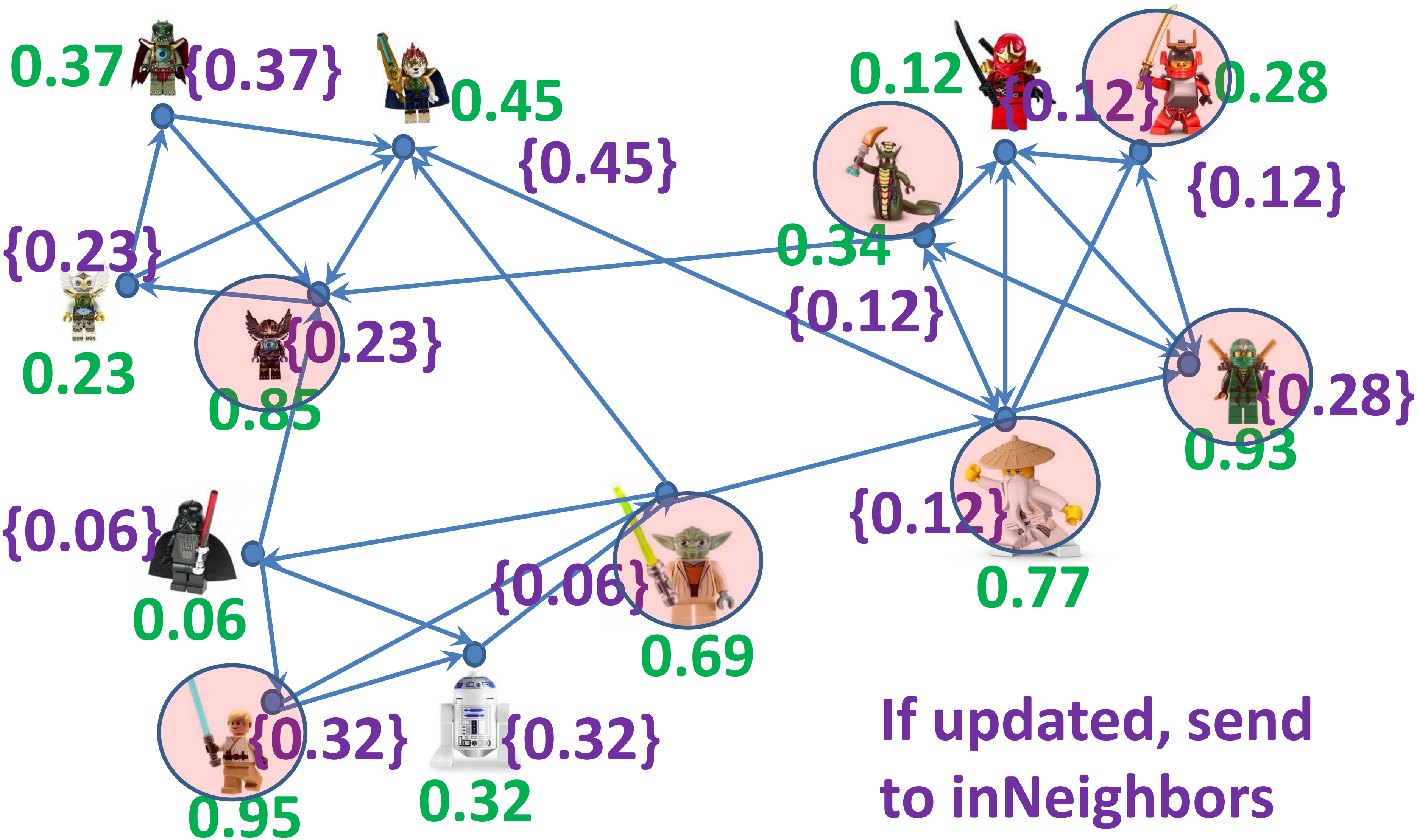


Send to inNeighbors

DP computation of Min-Hash sketches $k = 1$

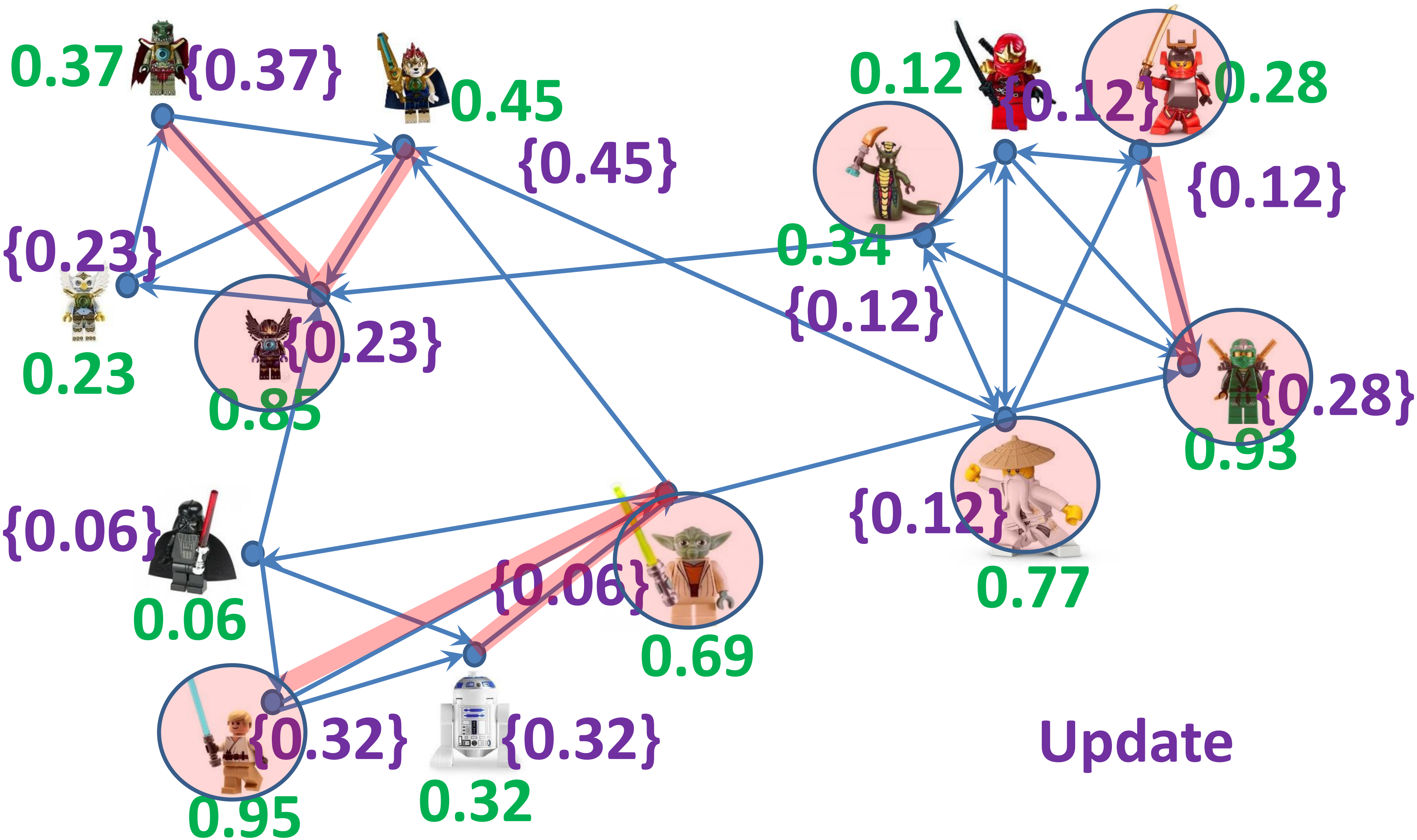


DP computation of Min-Hash sketches $k = 1$

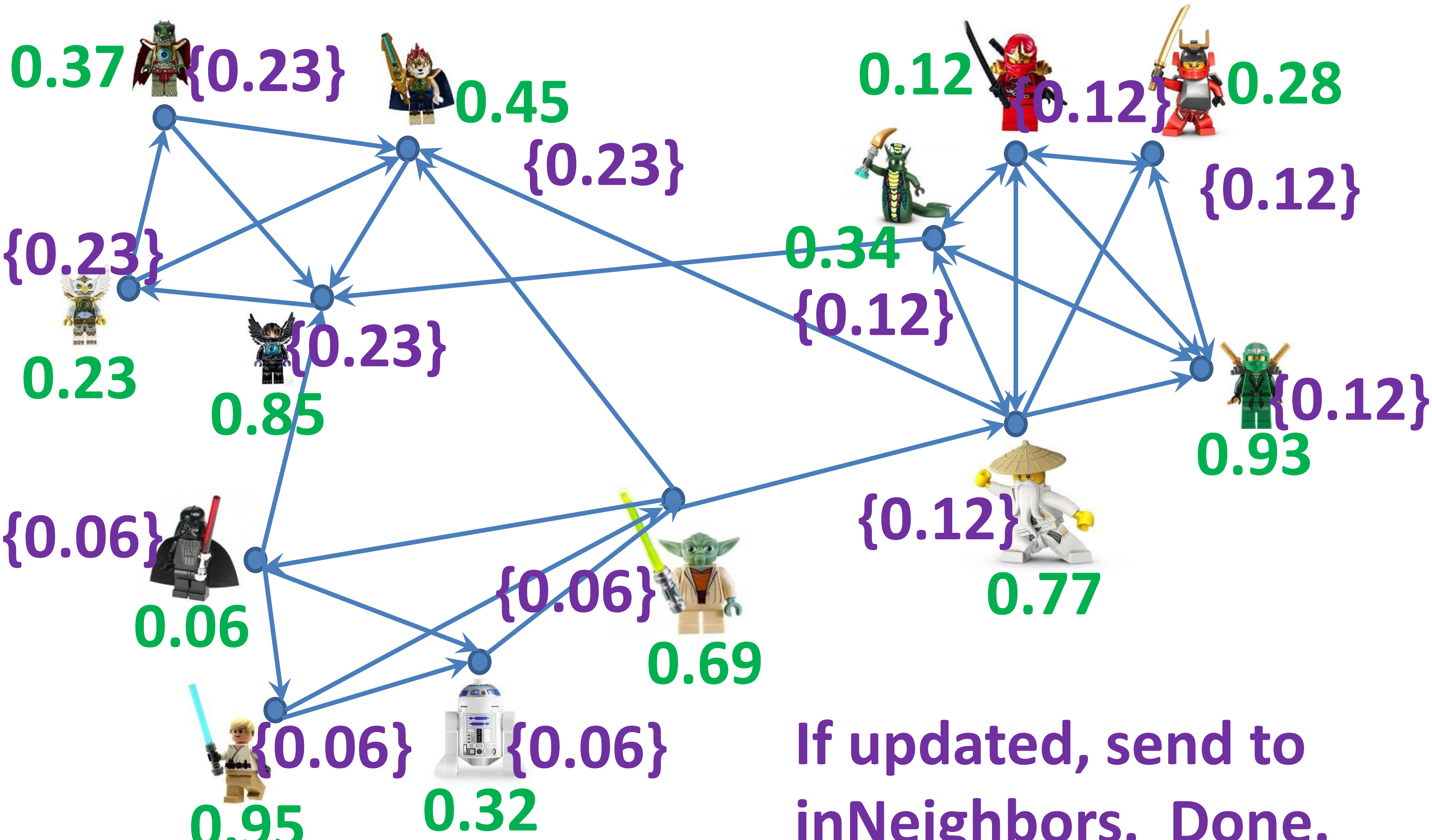


If updated, send to inNeighbors

DP computation of Min-Hash sketches $k = 1$



DP computation of Min-Hash sketches $k = 1$



If updated, send to inNeighbors. Done.

Analysis of DP: Edge traversals

Lemma: Each arc is used in expectation $< \ln \boldsymbol{n}$ times.

Proof: We bound the expected number of updates of $\boldsymbol{s(v)}$**.** (similar to lecture2)

- Consider nodes $v = u_1, u_2, \ldots$ in order that $h(u_i)$ is propagated to (can reach) v.
- The probability that $\mathrm{h}(u_i)$ updates $\mathrm{s}(v)$:
  $\mathbf{Pr}[\boldsymbol{h(u_i)} < \min_{j<i} \boldsymbol{h(u_j)}] = \frac{1}{i}$
- Summing over nodes (linearity of expectation):
  $\sum_{i=1}^{n} \frac{1}{i} = \boldsymbol{H_n} < \ln \boldsymbol{n}$

Analysis of DP: dependencies

The longest chain of dependencies is at most the longest shortest path (the **diameter** of the graph)

Next: All-Distances Sketches (ADS)

Often we care about distance, not only reachability:

- Nodes that are closer to you, in distance or in Dijkstra (Nearest-Neighbor) rank, are more meaningful.
- We want a sketch that supports distance-based queries.

Applications of ADSs

Estimate node/subset/network level properties that are expensive to compute exactly:

Applications of ADSs

- Distance distribution, effective diameter
- Closeness centrality
- Similarity (e.g., Jaccard similarity of d-hop neighborhoods or x nearest neighbors, closeness)
- Distance oracles
- Tightness of $F \subset V$ as a community
- Coverage of $F \subset V$

Bibliography

Recommended further reading on social networks analysis:

- Book: “Networks, Crowds, and Markets: Reasoning About a Highly Connected World” By David Easley and Jon Kleinberg.

http://www.cs.cornell.edu/home/kleinber/networks-book/

- Course/Lectures by Lada Adamic: https://www.coursera.org/course/sna
- http://open.umich.edu/education/si/si508/fall2008

Bibliography

Reachability Min-Hash sketches, All-Distances sketches (lectures 11,12):

- E. Cohen “Size-Estimation Framework with Applications to Transitive Closure and Reachability” JCSS 1997
- E. Cohen H. Kaplan “Spatially-decaying aggregation over a network” JCSS 2007
- E. Cohen H. Kaplan “Summarizing data using bottom-k sketches” PODC 2007
- E. Cohen: “All-Distances Sketches, Revisited: HIP Estimators for Massive Graphs Analysis” arXiv 2013