Recent Algorithmic Developments in NetworKit

Alexander van der Grinten

Department of Computer Science, Humboldt-Universität zu Berlin, Germany

Agenda

Since last NetworKit Day:

- numerous algorithmic additions to NetworKit
- and lots of refactoring (see previous talk)

Agenda

Since last NETWORKIT Day:

- numerous algorithmic additions to NETWORKIT
- and lots of refactoring (see previous talk)

In this talk: new algorithms and features since ND'17

(Brief tour through various modules . . .)

New Module: Randomization

Task: Given a graph G, construct a new randomized graph G' that preserves some properties (e.g., the degree sequence).

New Module: Randomization

Task: Given a graph G, construct a new randomized graph G' that preserves some properties (e.g., the degree sequence).

Use cases:

- Null-model for network analytics (e.g., modularity)
- Benchmarking graph algorithms

Randomization: Curveball algorithm

[SNBFS14], [CHMPTW18], code contributed by Manuel Penschuck

Curveball: Randomize by applying sequence of “trades”.

Randomization: Curveball algorithm

[SNBFS14], [CHMPTW18], code contributed by Manuel Penschuck

Curveball: Randomize by applying sequence of “trades”.

Each trade:

1. Pick random vertices u and v

Randomization: Curveball algorithm

[SNBFS14], [CHMPTW18], code contributed by Manuel Penschuck

Curveball: Randomize by applying sequence of “trades”.

Each trade:

1. Pick random vertices u and v
2. Keep common neighbors of u, v

Randomization: Curveball algorithm

[SNBFS14], [CHMPTW18], code contributed by Manuel Penschuck

Curveball: Randomize by applying sequence of “trades”.

Each trade:

1. Pick random vertices u and v
2. Keep common neighbors of u, v
3. Pick random permutation of disjoint neighbors of u and v

Randomization: Curveball algorithm

[SNBFS14], [CHMPTW18], code contributed by Manuel Penschuck

Curveball: Randomize by applying sequence of “trades”.

Each trade:

1. Pick random vertices u and v
2. Keep common neighbors of u, v
3. Pick random permutation of disjoint neighbors of u and v
4. Replace i-th neighbor by $\sigma(i)$

Randomization: Curveball algorithm
[SNBFS14], [CHMPTW18], code contributed by Manuel Penschuck

Curveball: Randomize by applying sequence of “trades”.

Each trade:

1. Pick random vertices u and v
2. Keep common neighbors of u, v
3. Pick random permutation of disjoint neighbors of u and v
4. Replace i-th neighbor by $\sigma(i)$

Quickly converges to null model.

Randomization: Curveball algorithm

[SNBFS14], [CHMPTW18], code contributed by Manuel Penschuck

Curveball: Randomize by applying sequence of “trades”.

Each trade:

1. Pick random vertices u and v
2. Keep common neighbors of u, v
3. Pick random permutation of disjoint neighbors of u and v
4. Replace i-th neighbor by $\sigma(i)$

Quickly converges to null model.

In NetworKit: Curveball and GlobalCurveball (pick many trades at a time)

Network Centrality Module

Centrality measure quantify the importance of vertices within a graph.

Network Centrality Module

Centrality measure quantify the importance of vertices within a graph.

Major additions to NetworKit:

- KADABRA betweenness approximation [BN16], [vdGAM19]

Network Centrality Module

Centrality measure quantify the importance of vertices within a graph.

Major additions to NETWORKIT:

- KADABRA betweenness approximation [BN16], [vdGAM19]

Major additions, not in detail here:

- Katz centrality [vdGBGBM18]
  Counts walks that start/end at a vertex, weighted by their length

Network Centrality Module

Centrality measure quantify the importance of vertices within a graph.

Major additions to NETWORKIT:

- KADABRA betweenness approximation [BN16], [vdGAM19]

Major additions, not in detail here:

- Katz centrality [vdGBGBM18]
  Counts walks that start/end at a vertex, weighted by their length
- Spanning edge centrality [HAY16]
  Considers graph as electrical network, measures resistance of edges
  (More related electrical centralities in the pipeline [APvdGM20])

Network Centrality Module

Centrality measure quantify the importance of vertices within a graph.

Major additions to NetworKit:

- KADABRA betweenness approximation [BN16], [vdGAM19]

Major additions, not in detail here:

- Katz centrality [vdGBGBM18]
  Counts walks that start/end at a vertex, weighted by their length
- Spanning edge centrality [HAY16]
  Considers graph as electrical network, measures resistance of edges
  (More related electrical centralities in the pipeline [APvdGM20])
- Top-k (harmonic) closeness [BBCMM16]
  Computes k vertices with highest closeness w/o computing all scores

Network Centrality: KADABRA

Let $G = (V, E)$ be a graph. $s, t \in V$.

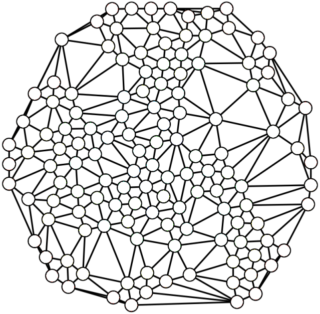


Image by Claudio Rocchini (CC-BY). Taken from `wikipedia.org/wiki/Betweenness_centrality`.

Alexander van der Grinten, HU Berlin

Network Centrality: KADABRA

Let $G = (V, E)$ be a graph. $s, t \in V$.

σ_{st}: number of shortest s-t paths
$\sigma_{st}(x)$: number of shortest s-t paths over vertex $x \in V$

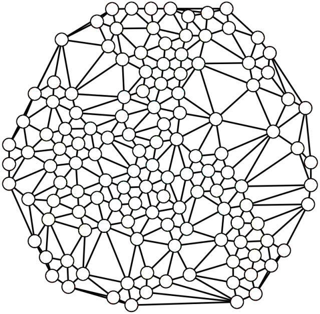


Image by Claudio Rocchini (CC-BY). Taken from `wikipedia.org/wiki/Betweenness_centrality`.

Alexander van der Grinten, HU Berlin

Network Centrality: KADABRA

Let $G = (V, E)$ be a graph. $s, t \in V$.

σ_{st}: number of shortest s-t paths
$\sigma_{st}(x)$: number of shortest s-t paths over vertex $x \in V$

Betweenness centrality
of a vertex $x \in V$:

$$\mathrm{BC}(x) = \sum_{s,t \in V \setminus \{x\}} \frac{\sigma_{st}(x)}{\sigma_{st}}$$

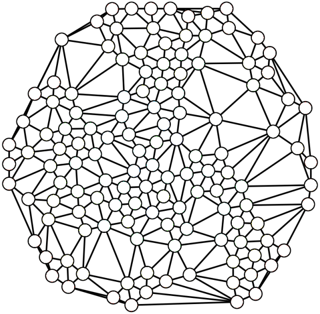


Image by Claudio Rocchini (CC-BY). Taken from `wikipedia.org/wiki/Betweenness_centrality`.

Network Centrality: KADABRA

Let $G = (V, E)$ be a graph. $s, t \in V$.

σ_{st}: number of shortest s-t paths
$\sigma_{st}(x)$: number of shortest s-t paths over vertex $x \in V$

Betweenness centrality
of a vertex $x \in V$:

$$\text{BC}(x) = \sum_{s,t \in V \setminus \{x\}} \frac{\sigma_{st}(x)}{\sigma_{st}}$$

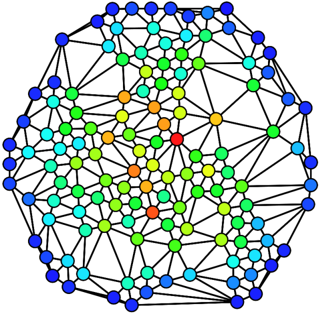


Image by Claudio Rocchini (CC-BY). Taken from `wikipedia.org/wiki/Betweenness_centrality`.

Alexander van der Grinten, HU Berlin

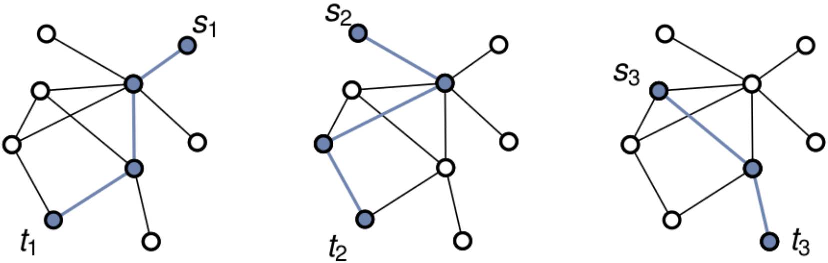
Network Centrality: KADABRA

KADABRA: Sampling-based approximation for betweenness

Alexander van der Grinten, HU Berlin

Network Centrality: KADABRA

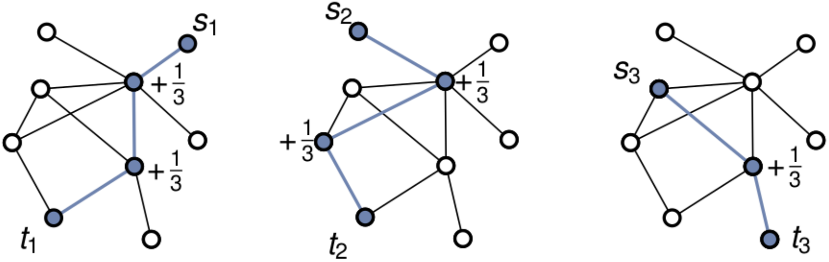
KADABRA: Sampling-based approximation for betweenness



Alexander van der Grinten, HU Berlin

Network Centrality: KADABRA

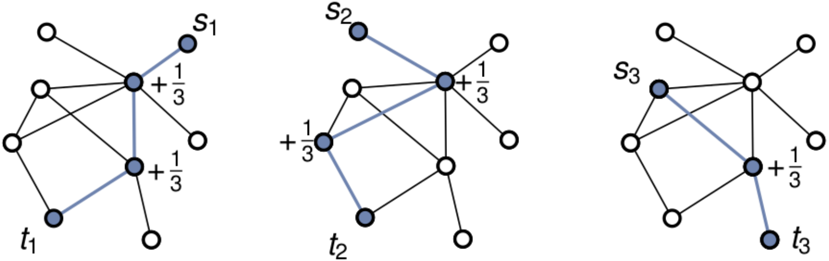
KADABRA: Sampling-based approximation for betweenness



Not shown here:
(complicated) adaptive stopping condition to determine # of samples.

Network Centrality: KADABRA

KADABRA: Sampling-based approximation for betweenness



Not shown here:
(complicated) adaptive stopping condition to determine # of samples.

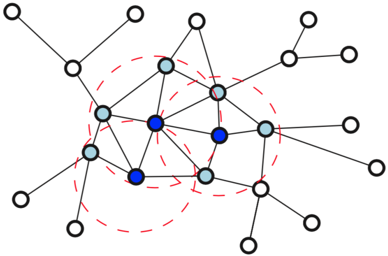
In NETWORKIT: fastest available betweenness approximation

Group Centrality Module

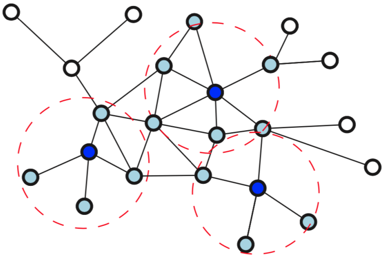
Group centrality: measures importance of sets of vertices

Group Centrality Module

Group centrality: measures importance of sets of vertices



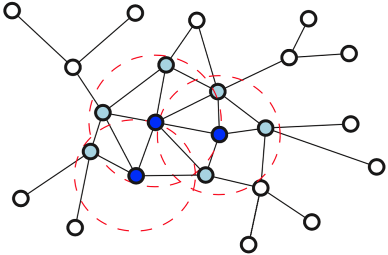
Top-k centrality



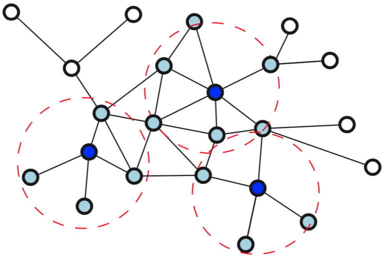
Group centrality

Group Centrality Module

Group centrality: measures importance of sets of vertices



Top-k centrality



Group centrality

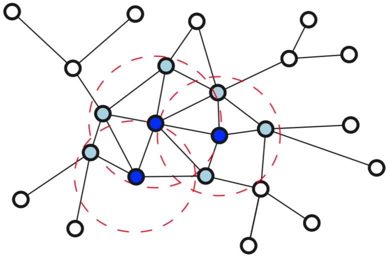
Support for group centralities recently added to NetworKit.

- Computation of group centrality scores

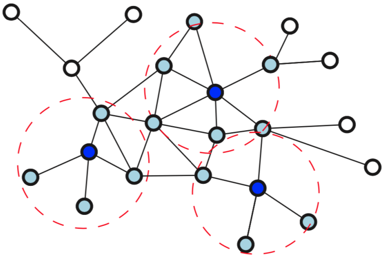
Alexander van der Grinten, HU Berlin

Group Centrality Module

Group centrality: measures importance of sets of vertices



Top-k centrality



Group centrality

Support for group centralities recently added to NetworKit.

- Computation of group centrality scores
- Finding groups with maximal centrality (usually a hard problem)

Group Centrality Module

New (approximation) algorithms:

- Group degree [Folklore]

Group Centrality Module

New (approximation) algorithms:

- Group degree [Folklore]
- Group betweenness [MTU16]
  Approximation via sampling on hypergraphs

Group Centrality Module

New (approximation) algorithms:

- Group degree [Folklore]
- Group betweenness [MTU16]
  Approximation via sampling on hypergraphs
- GED-Walk [AvdGBZGM19]
  Similar to Katz centrality:
  counts all walks that cross the vertex group, weighted by their length.

Group Centrality Module

New (approximation) algorithms:

- Group degree [Folklore]
- Group betweenness [MTU16]
  Approximation via sampling on hypergraphs
- GED-Walk [AvdGBZGM19]
  Similar to Katz centrality:
  counts all walks that cross the vertex group, weighted by their length.
- Upcoming: Group closeness [AvdGM19]
  Fast local-search algorithm

Graph I/O Module

Addition of various new formats to NetworKit, including:

Graph I/O Module

Addition of various new formats to NetworKit, including:

- Formats for graph partitions / communities

Graph I/O Module

Addition of various new formats to NetworKit, including:

- Formats for graph partitions / communities
- Thrill-compatible binary format

Graph I/O Module

Addition of various new formats to NETWORKIT, including:

- Formats for graph partitions / communities
- Thrill-compatible binary format
- NETWORKIT-native binary format

Graph I/O Module

Addition of various new formats to NetworKit, including:

- Formats for graph partitions / communities
- Thrill-compatible binary format
- NetworKit-native binary format

NetworKit's binary format:

- Substantially faster to read than other formats

Graph I/O Module

Addition of various new formats to NetworKit, including:

- Formats for graph partitions / communities
- Thrill-compatible binary format
- NetworKit-native binary format

NetworKit's binary format:

- Substantially faster to read than other formats
- Smaller than most other (text/binary) formats

Graph I/O Module

Addition of various new formats to NetworKit, including:

- Formats for graph partitions / communities
- Thrill-compatible binary format
- NetworKit-native binary format

NetworKit's binary format:

- Substantially faster to read than other formats
- Smaller than most other (text/binary) formats
- Varint encoding: bit-length of IDs adapted to # nodes of the graph

Graph I/O Module

Addition of various new formats to NetworKit, including:

- Formats for graph partitions / communities
- Thrill-compatible binary format
- NetworKit-native binary format

NetworKit's binary format:

- Substantially faster to read than other formats
- Smaller than most other (text/binary) formats
- Varint encoding: bit-length of IDs adapted to # nodes of the graph
- Goal: represent all data available in NetworKit(weights, IDs, . . .) in a compact format

Alexander van der Grinten, HU Berlin

Upcoming: Graph Embeddings

a.k.a. Representation Learning

Problem: Given graph G, map each vertex $v \in V(G)$ to $f(v) \in \mathbb{R}^d$ for some d.

Upcoming: Graph Embeddings

a.k.a. Representation Learning

Problem: Given graph G, map each vertex $v \in V(G)$ to $f(v) \in \mathbb{R}^d$ for some d.

Applications: Enables use of downstream machine learning algorithms (which work on feature vectors, not graphs).

Upcoming: Graph Embeddings

a.k.a. Representation Learning

Problem: Given graph G, map each vertex $v \in V(G)$ to $f(v) \in \mathbb{R}^d$ for some d.

Applications: Enables use of downstream machine learning algorithms (which work on feature vectors, not graphs).

Start with well-known node2vec algorithm [GL16]:

- Idea: if u, v appear together in many random walks, $f(u)$ should be close to $f(v)$

Upcoming: Graph Embeddings

a.k.a. Representation Learning

Problem: Given graph G, map each vertex $v \in V(G)$ to $f(v) \in \mathbb{R}^d$ for some d.

Applications: Enables use of downstream machine learning algorithms (which work on feature vectors, not graphs).

Start with well-known node2vec algorithm [GL16]:

- Idea: if u, v appear together in many random walks, $f(u)$ should be close to $f(v)$

Other embedding algorithms in the future?

Other features

Miscellaneous additions to NETWORKIT:

- Bidirectional shortest path algorithms

  Faster than SSSP if source + target is known

Alexander van der Grinten, HU Berlin

Other features

Miscellaneous additions to NetworKit:

- Bidirectional shortest path algorithms
  Faster than SSSP if source + target is known
- Biconnected Components
  Classical graph problem; useful building block for other algorithms

Other features

Miscellaneous additions to NetworKit:

- Bidirectional shortest path algorithms
  Faster than SSSP if source + target is known
- Biconnected Components
  Classical graph problem; useful building block for other algorithms
- Graph generator by F.-B. Mocnik [M18]
  Models spacial graphs

simexpal

simexpal: tool to manage algorithmic experiments

external project (not part of NetworKit)

simexpal

simexpal: tool to manage algorithmic experiments

external project (not part of NETWORKIT)

- Replace hand-written scripts to run algorithms on large numbers of instances

simexpal

simexpal: tool to manage algorithmic experiments

external project (not part of NetworKit)

- Replace hand-written scripts to run algorithms on large numbers of instances
- Useful for algorithm engineering, but also for analysis pipelines

simexpal

simexpal: tool to manage algorithmic experiments
external project (not part of NetworKit)

- Replace hand-written scripts to run algorithms on large numbers of instances
- Useful for algorithm engineering, but also for analysis pipelines
- Support for multiple configurations of algorithms out of the box

simexpal

simexpal: tool to manage algorithmic experiments
external project (not part of NetworKit)

- Replace hand-written scripts to run algorithms on large numbers of instances
- Useful for algorithm engineering, but also for analysis pipelines
- Support for multiple configurations of algorithms out of the box
- Support for reproducible builds (e.g., of C++ code)

Alexander van der Grinten, HU Berlin

simexpal

simexpal: tool to manage algorithmic experiments
external project (not part of NETWORKIT)

- Replace hand-written scripts to run algorithms on large numbers of instances
- Useful for algorithm engineering, but also for analysis pipelines
- Support for multiple configurations of algorithms out of the box
- Support for reproducible builds (e.g., of C++ code)

Available from: `https://github.com/hu-macsy/simexpal`

Thank You!