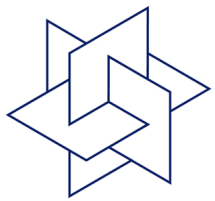
Australia Day

Industry problems

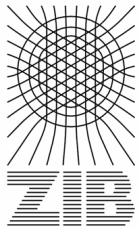
Christina Burt and Yao-ban Chan
with Ambros Gleixner

MASCOS, University of Melbourne

28 September, 2009



DFG Research Center MATHEON
Mathematics for key technologies



Outline of the day...

9.00 - 11.00 Telecommunication transmission radius in wireless ad-hoc networks

11.30 - 12.30 Equipment selection for surface mines

2.00 - 3.00 Equipment selection for surface mines exercise

3.00 - 4.00 Open pit mining production planning

4.30 - 5.30 Open pit mining production planning exercise

About MASCOS

The ARC Centre of Excellence for Mathematics and Statistics of Complex Systems is a government funded collaborative research centre involving:

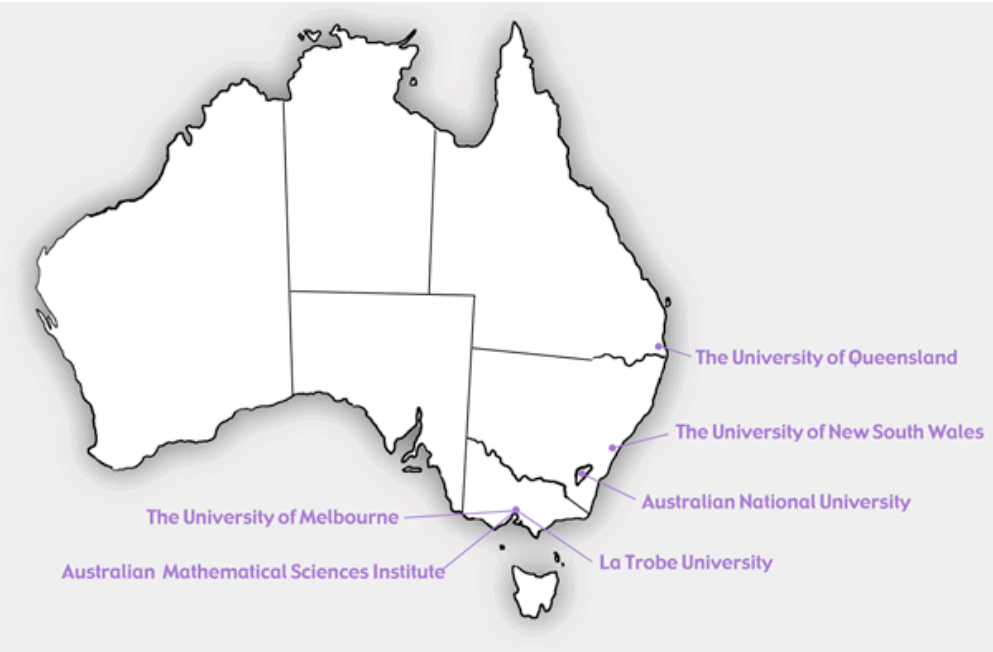
- A Director (Tony Guttmann), Deputy Director (Ian Sloan),12 Chief Investigators and 13 Associate Investigators;
- 23 Research Fellows and 27 PhD Scholars;
- 5 nodes within Australia;
- 6 international collaborating institutions.

About MASCOS

Nodes in Australia

MASCOS

AUSTRALIAN RESEARCH COUNCIL
Centre of Excellence for Mathematics and Statistics of Complex Systems



About MASCOS

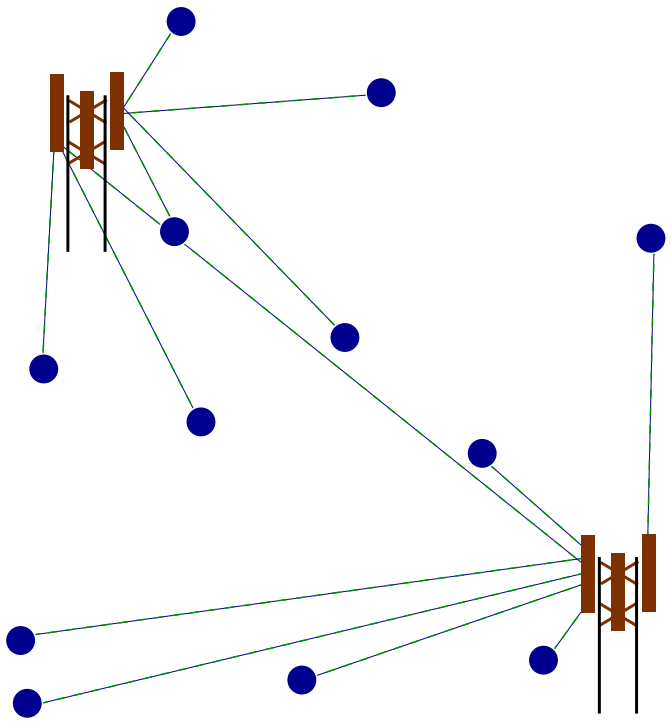
Some current projects

- Optimal control in population networks;
- Integrated optimisation of airline schedules with stochastic disruption;
- Optimisation of freight trains on single track networks;
- Equipment selection for surface mines; and
- Design of k-connected mobile ad hoc networks.

Telecommunication transmission radius in wireless ad-hoc networks

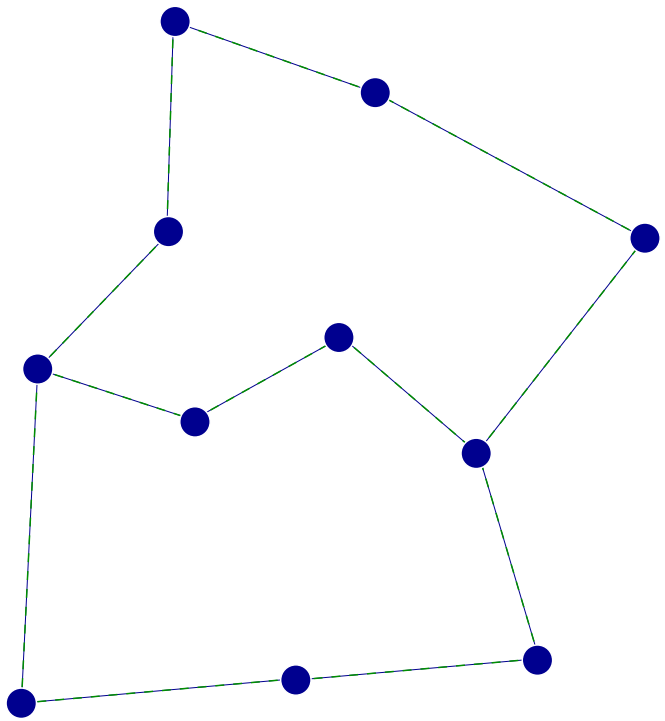
Christina Burt, Yao-ban Chan and Nikki Sonenberg

Current mobile telephony



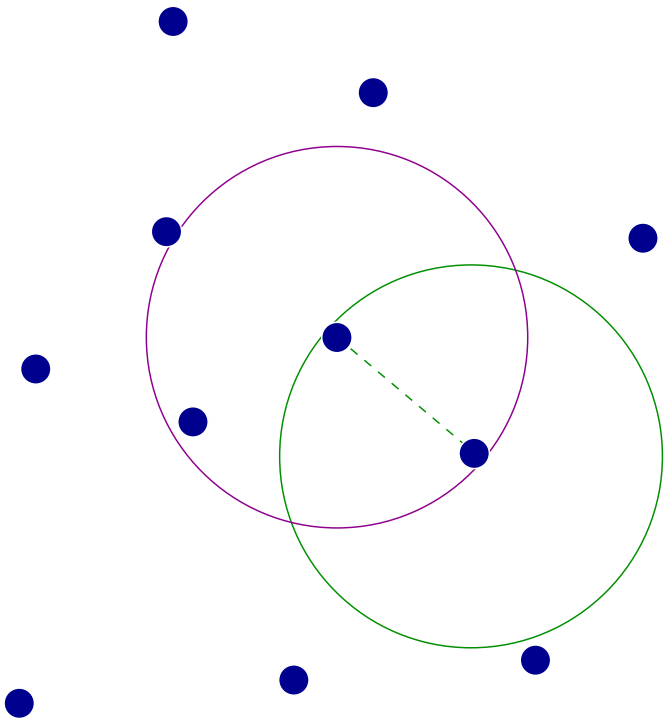
Very reliable. . . but expensive.

Ad hoc network mobile telephony



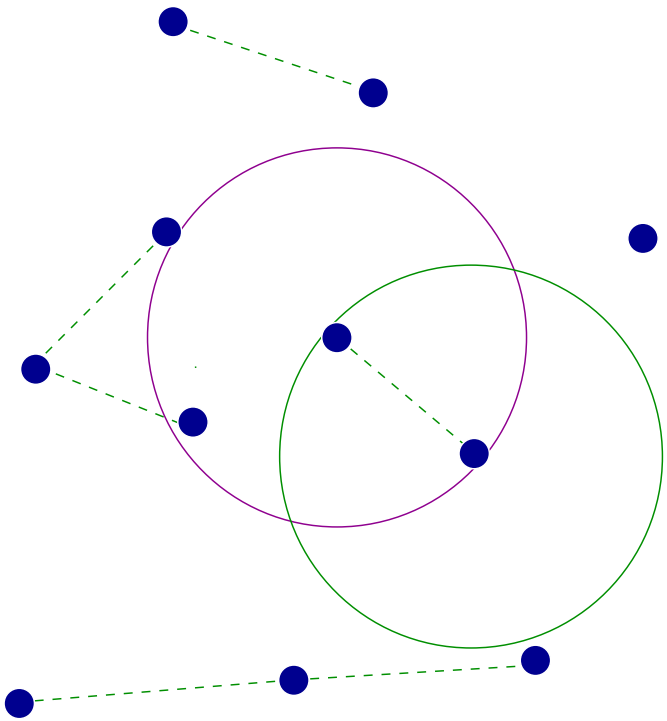
Need to connect to far-away phones via multi-hop routes.

Ad hoc network mobile telephony



We want to reserve our own battery power as much as possible,

Ad hoc network mobile telephony



. . . but this does not result in a useful network.

k-connectedness

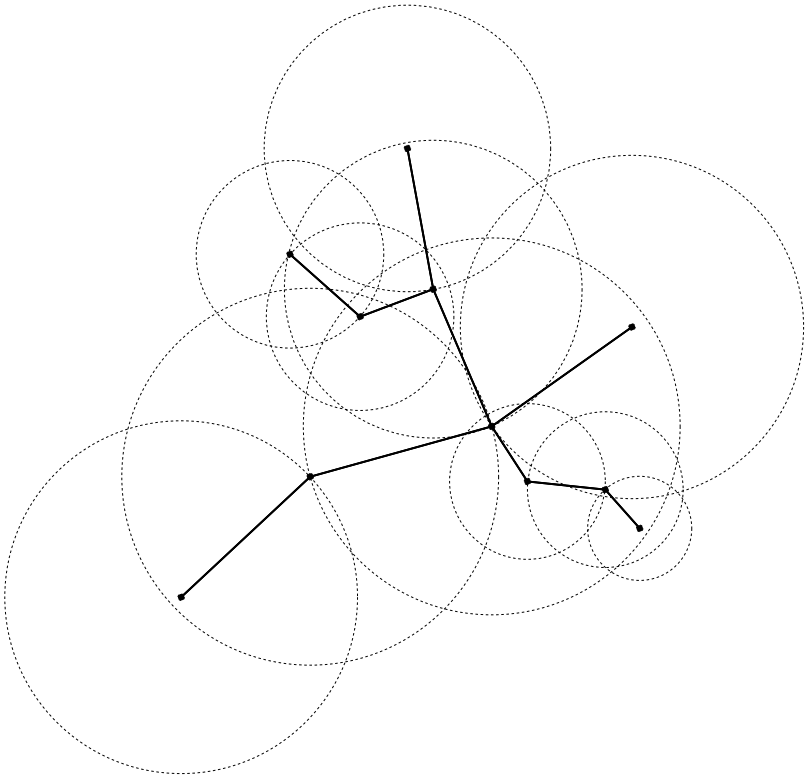
We are not only interested in attaining a connected network, but a measurable level of robustness too.

Definition

A graph is k-connected if removing any set of $k-1$ nodes results in a connected graph.

k-connectedness

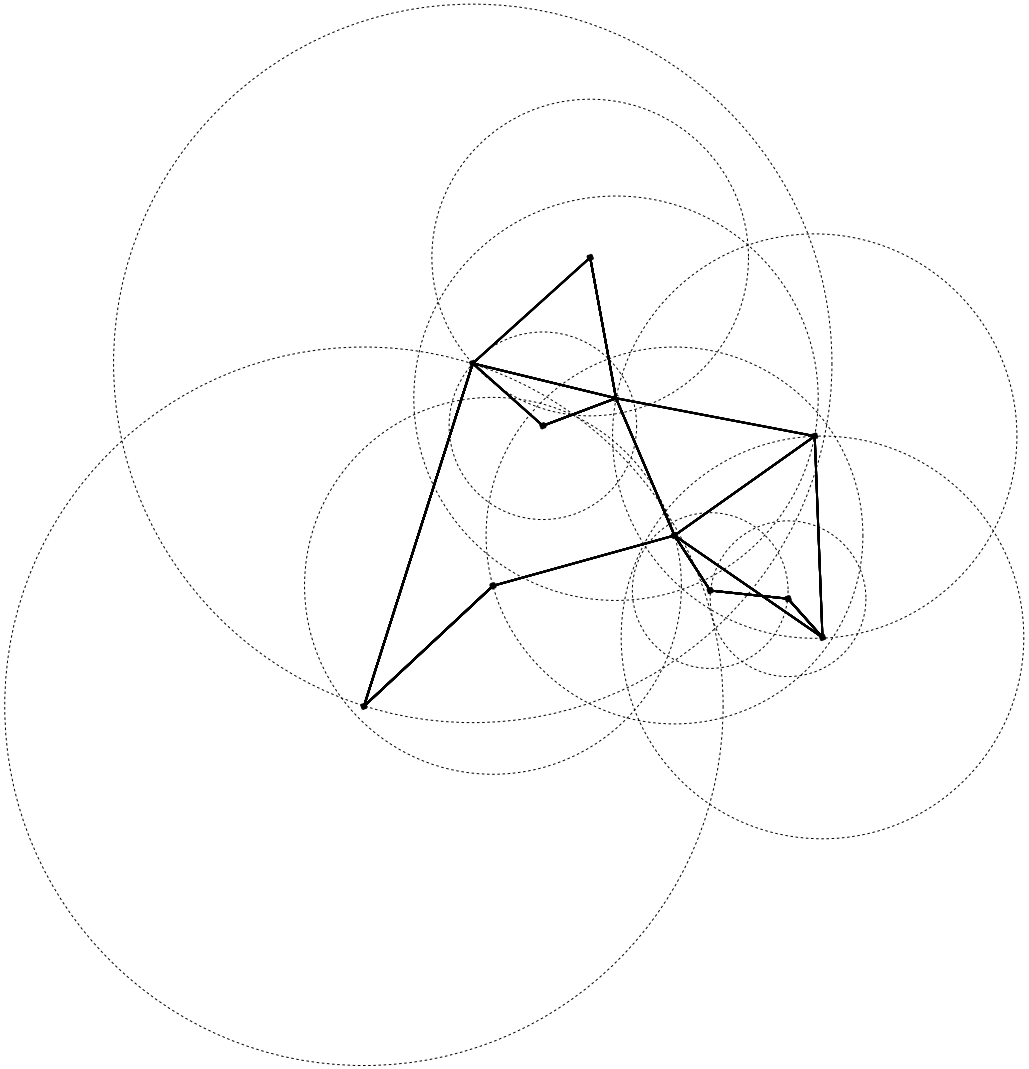
A 1-connected network



We can only guarantee connectivity if no nodes have dropped out.

k-connectedness

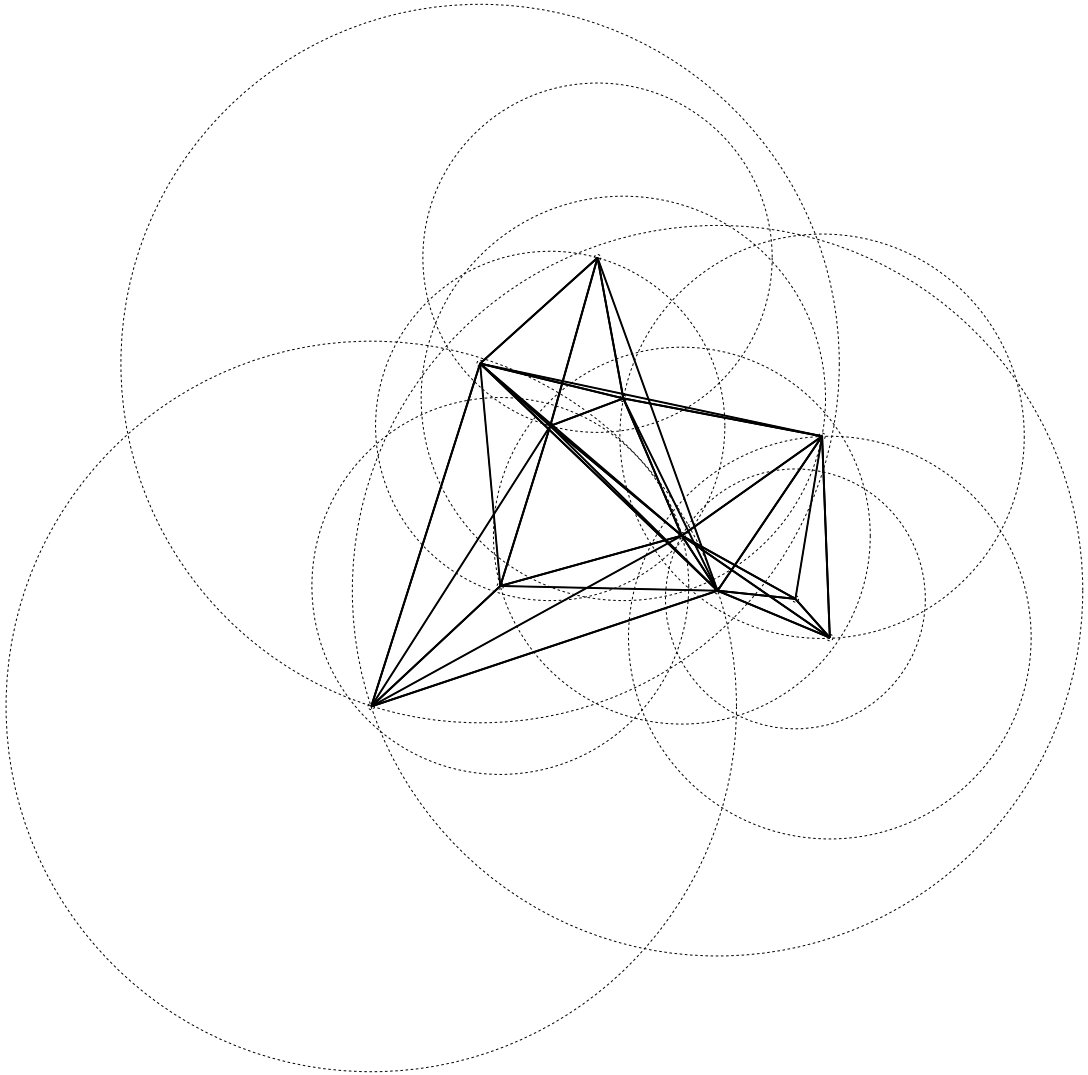
A 2-connected network



We can guarantee connectivity if only 1 node has dropped out.

k-connectedness

A 3-connected network



We can guarantee connectivity if only 2 nodes have dropped out.

The problem we have described is:

the minimum energy k-connected network problem.

The industry problem

Description: We are interested in finding the best transmission range for each mobile phone user in a dynamic network such that the network is k-connected.

Desirable outcomes: Given the rate of change in the topology of phone users, it would be desirable to develop a fast algorithm for this problem that can run in real time on regular time intervals of, say, around 300 – 500 ms. Ideally, we would like this to be a heuristic that runs independently in each peer, given that each peer has incomplete information about the topology.

Our proposed approach

There are two approaches to this problem:

- exact;
- heuristic.

It would be nice to have heuristics that are fast enough for real-time deployment.

Exact solutions may allude to such heuristics.

Our proposed approach

1. Model the global problem (where full topology is known) as a mixed integer program;
2. Analyse the results from (1);
3. Create fast heuristics based on local information, and compare against our exact solutions from (1).

In this talk we will focus on what we have done for step (1).

What has been done before?

Exact approaches have focussed on mixed-integer programming models.

We found a handful of models that solved the minimum energy 1-connected problem.

What has been done before?

Some notation:

$y_{i,j}$ indicator variable for edge i to j;

$x_{i,j}$ flow variable from node i to j;

$z_{i,j}$ indicator variable of bi-connectivity from i to j;

$c_{i,j}$ the cost of edge i to j;

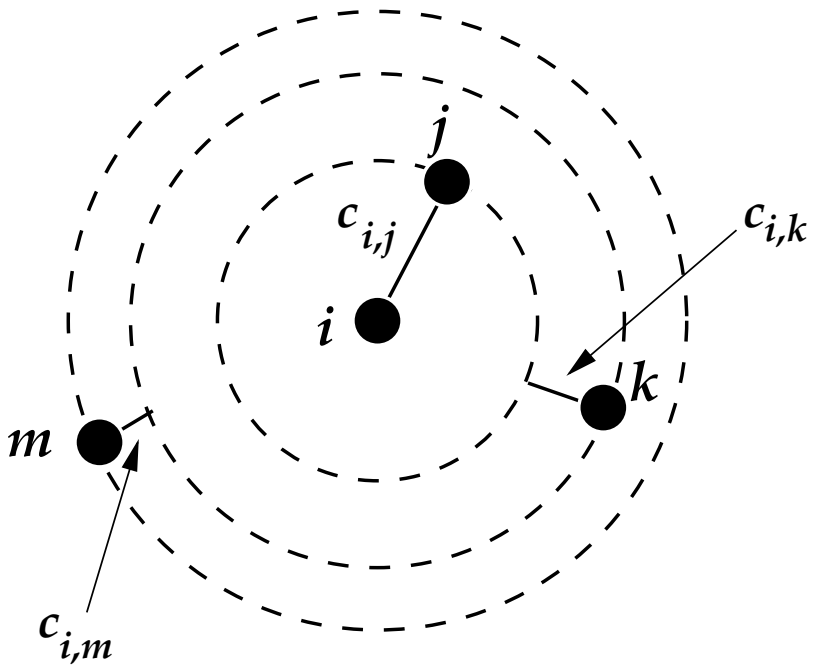
s source node;

k the level of connectivity.

What has been done before?

Montemanni and Gambardella (2005)

- network flow model
- achieves a 1-connected network
- uses 'ancestor nodes'



What has been done before?

Montemanni and Gambardella (2005)

$$\text{Minimise} \quad \sum_{(i,j)\in A} c_{i,j} y_{i,j}$$

$$\text{subject to} \quad y_{i,j} \leq y_{i,a_j^i} \qquad \forall\ (i,j) \in A, a_j^i \neq i, \qquad (1)$$

$$x_{i,j} \leq (|V|-1) y_{i,j} \qquad \forall\ (i,j) \in A, \qquad (2)$$

$$x_{i,j} \leq (|V|-1) y_{j,i} \qquad \forall\ (i,j) \in A, \qquad (3)$$

$$\sum_{j:(i,j)\in A} x_{i,j} - \sum_{j:(j,i)\in A} x_{j,i} = \begin{cases} |V|-1 & \text{if } i = s, \\ -1 & \text{otherwise,} \end{cases} \qquad (4)$$

$$x_{i,j} \in \mathbb{R}, \qquad (5)$$

$$y_{i,j} \in \{0,1\}. \qquad (6)$$

where a_j^i is calculated by a formula given in the paper.

What has been done before?

Montemanni and Gambardella (2005)

$$\text{Minimise} \quad \sum_{(i,j)\in A} c_{i,j} y_{i,j}$$

$$\text{subject to} \quad y_{i,j} \leq y_{i,a_j^i} \qquad \forall\ (i,j) \in A, a_j^i \neq i, \tag{7}$$

$$z_{i,j} \leq y_{i,j} \qquad \forall\ (i,j) \in E, \tag{8}$$

$$z_{i,j} \leq y_{j,i} \qquad \forall\ (i,j) \in E, \tag{9}$$

$$\sum_{\{i,j\}\in E, i\in S, j\in V\setminus S} z_{i,j} \geq 1 \qquad \forall\ S \subset V, \tag{10}$$

$$z_{i,j} \in \{0,1\}, \tag{11}$$

$$y_{i,j} \in \{0,1\}. \tag{12}$$

where constraint (10) is added using a separation algorithm.

What has been done before?

Yuan, Bauer and Haugland (2008)

- multi-commodity flow model;
- achieves a 1-connected network.

What has been done before?

Yuan, Bauer and Haugland (2008)

$$\text{Minimise} \quad \sum_{(i,j)\in A} c_{i,j} z_{i,j}$$

$$\text{subject to} \quad \sum_{j:(i,j)\in A} z_{i,j} \leq 1 \qquad \forall\ i \in V, \tag{13}$$

$$\sum_{j:(i,j)\in A} x^d_{i,j} - \sum_{j:(j,i)\in A} x^d_{j,i} = \begin{cases} 1 & \text{if } i = s, \\ -1 & \text{if } i = d, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall\ i \in V,\ \forall\ d \in V\backslash\{s\}, \tag{14}$$

$$y_{i,j} \leq \sum_{p_{i,k} \geq p_{i,j}} z_{i,k} \qquad \forall (i,j) \in A, \tag{15}$$

$$y_{j,i} \leq \sum_{p_{i,k} \geq p_{i,j}} z_{i,k} \qquad \forall (i,j) \in A, \tag{16}$$

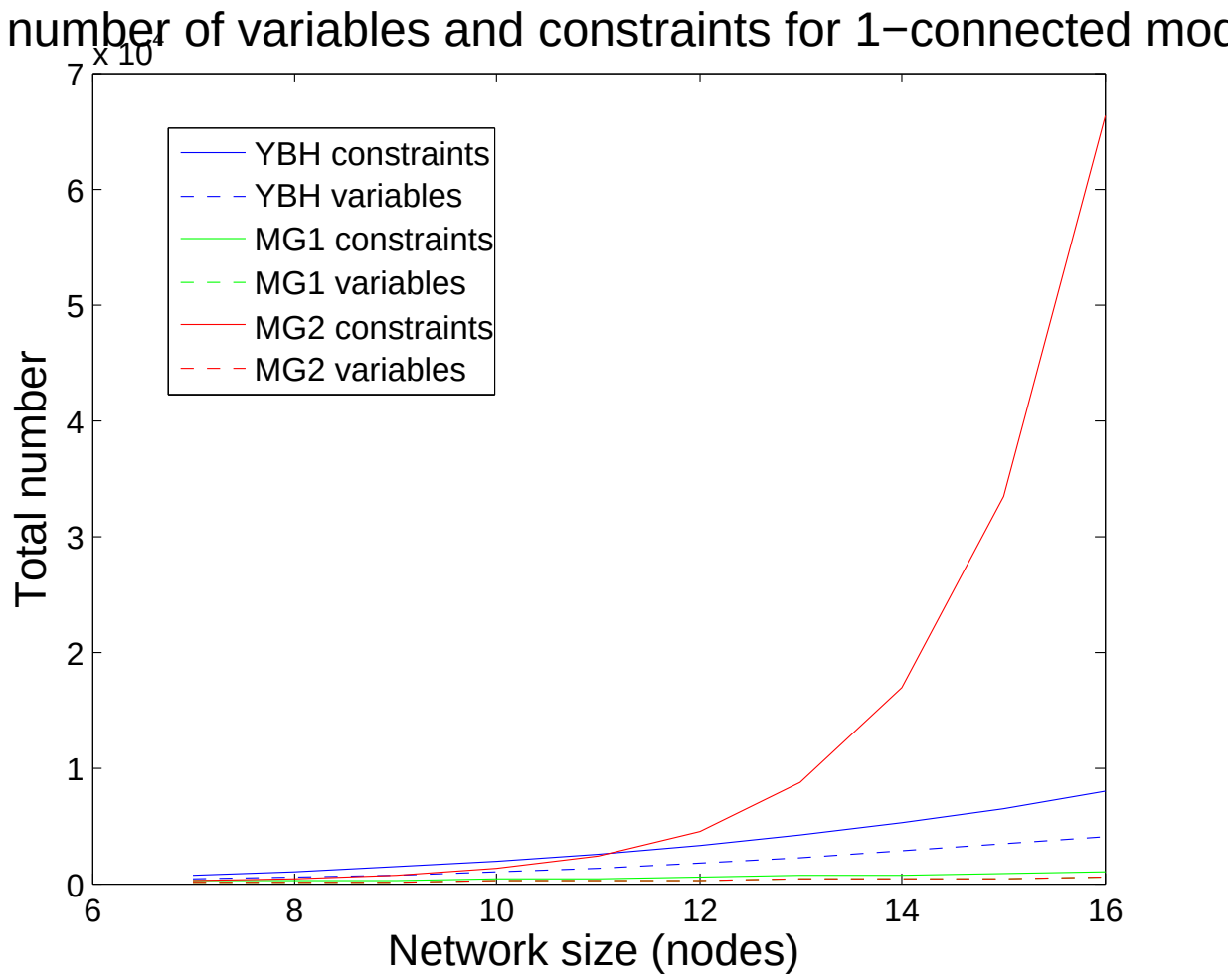
$$x^d_{i,j} \leq y_{i,j} \qquad \forall (i,j) \in A, d \in V\backslash\{s\}, \tag{17}$$

$$x^d_{i,j} \geq 0, \tag{18}$$

$$y_{i,j} \in \{0,1\}, \tag{19}$$

$$z_{i,j} \in \{0,1\}. \tag{20}$$

Constraints and variables



What has been done before?

Heuristic approach

The most common approach is using graph augmentation.

This has been used to achieve 2-connected networks to sub-optimality.

A recent paper suggested approximating a Hamiltonian cycle to approximate the 2-connected solution.

Another technique, analogous to finding the cut-vertex, has been used to solve 3-connected and 4-connected solutions to sub-optimality.

What has been done before?

Heuristic approach

These approaches may be faster than exact approaches, but they are sub-optimal.

All these approaches require information about the full topology of the network, which is unrealistic for real-time deployment.

Ideally we would like to develop an understanding of the exact solutions, and use this information to develop a local heuristic that can be deployed in 'real-time'.

What are we doing?

We are interested in obtaining exact k-connected solutions.

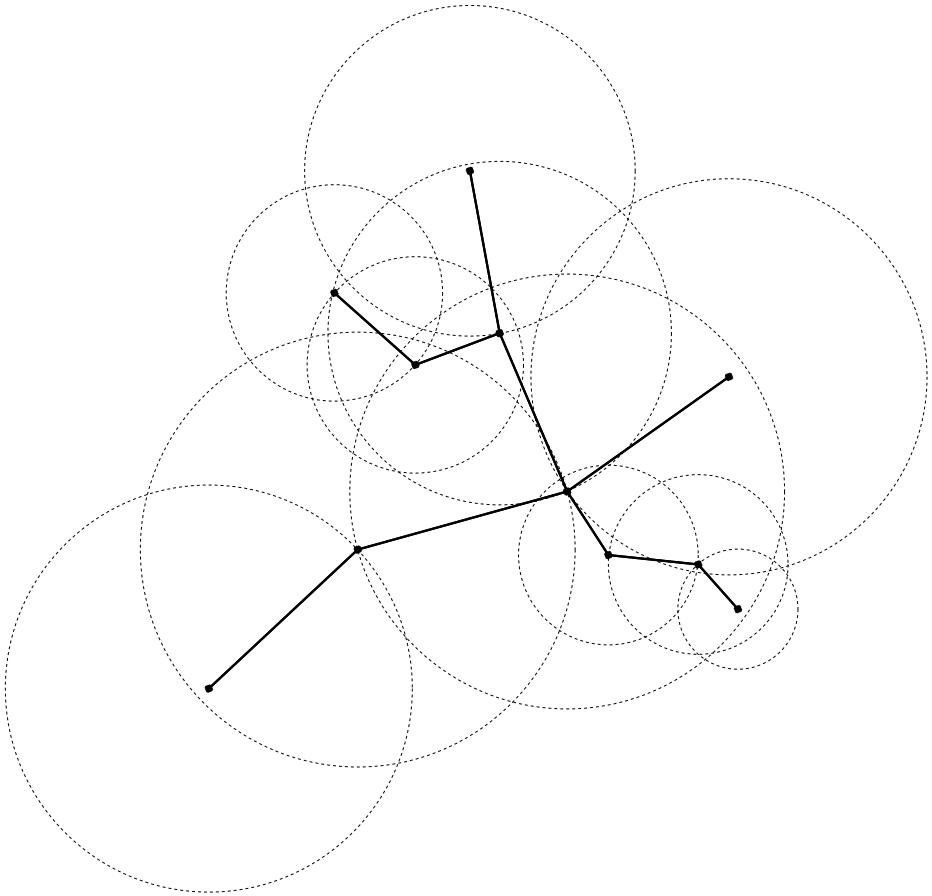
To do this, we have two approaches:

- 'naive' approach;
- 'smart' approach.

We are also studying ways to 'speed-up' our solutions in the hope of solving for much greater instances.

Spanning tree based method

Note that if a network is connected, then it contains a spanning tree.



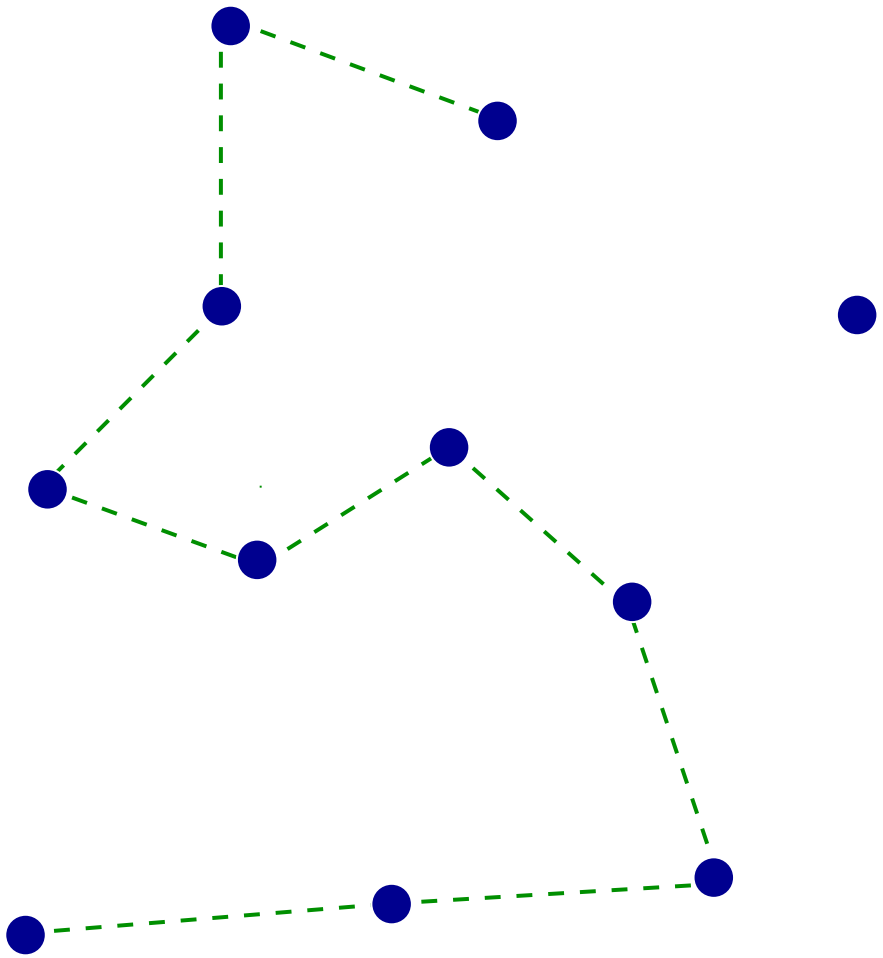
Spanning tree based method

Combining this observation with the definition of k-connectedness we note the following:

A network is k-connected if and only if we can remove any combination of $k-1$ nodes and find a spanning tree for the remaining nodes. There are $\binom{|V|}{k-1}$ such combinations.

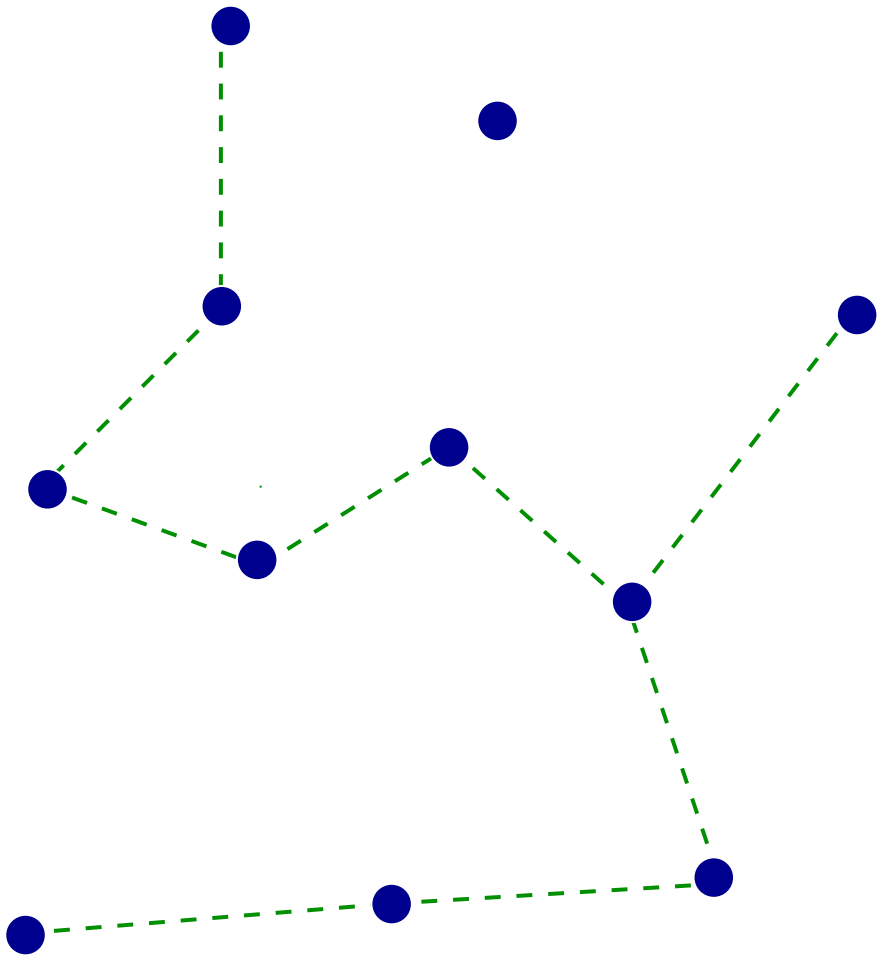
Spanning tree based method

An example



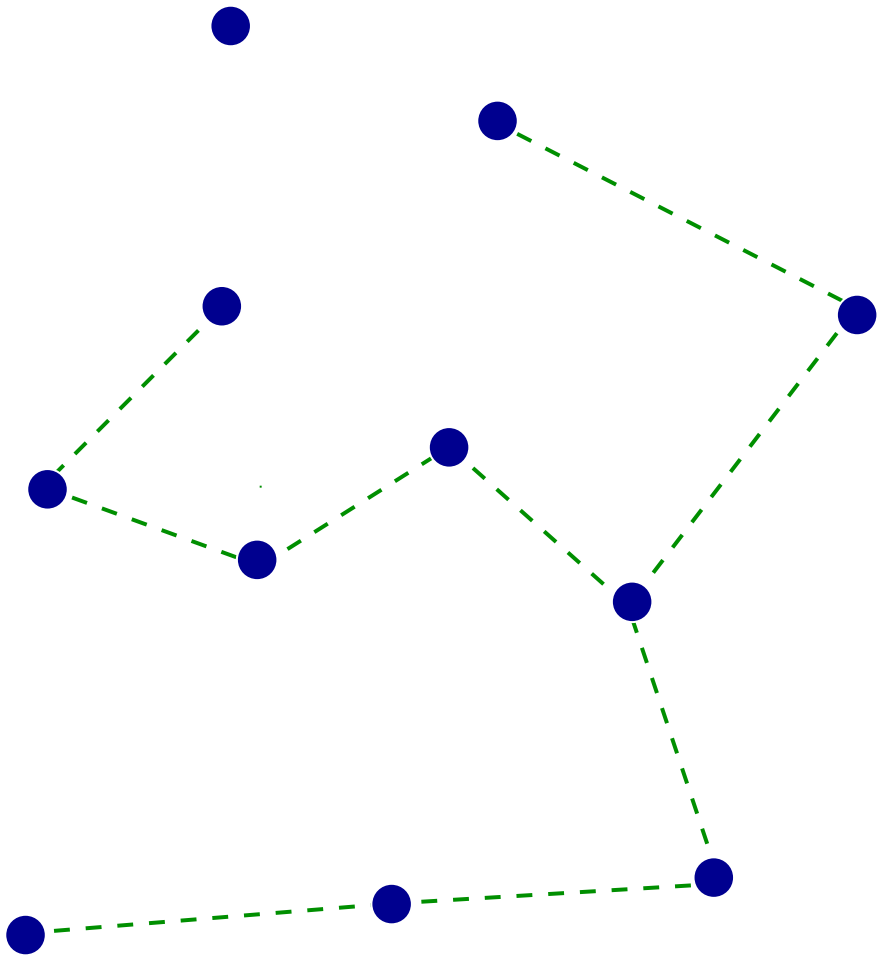
Spanning tree based method

An example



Spanning tree based method

An example



Spanning tree based method

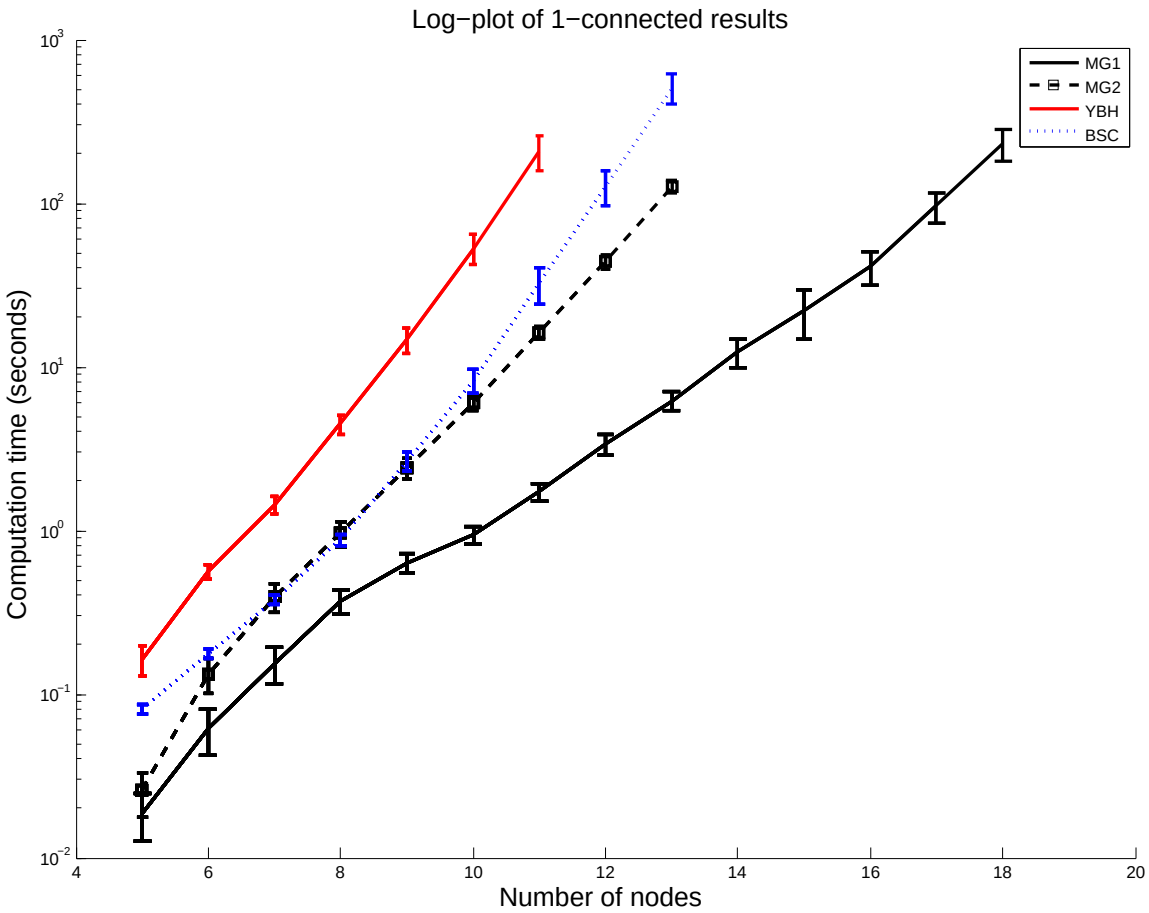
We optimise over all sub-problems, solving numerous 1-connected problems to obtain the 2-connected solution.

We need to consider every combination of $v - 1$ nodes and remove them iteratively (and then solve the 1-connected problem).

We expect from the outset that this will obtain poor computational results!

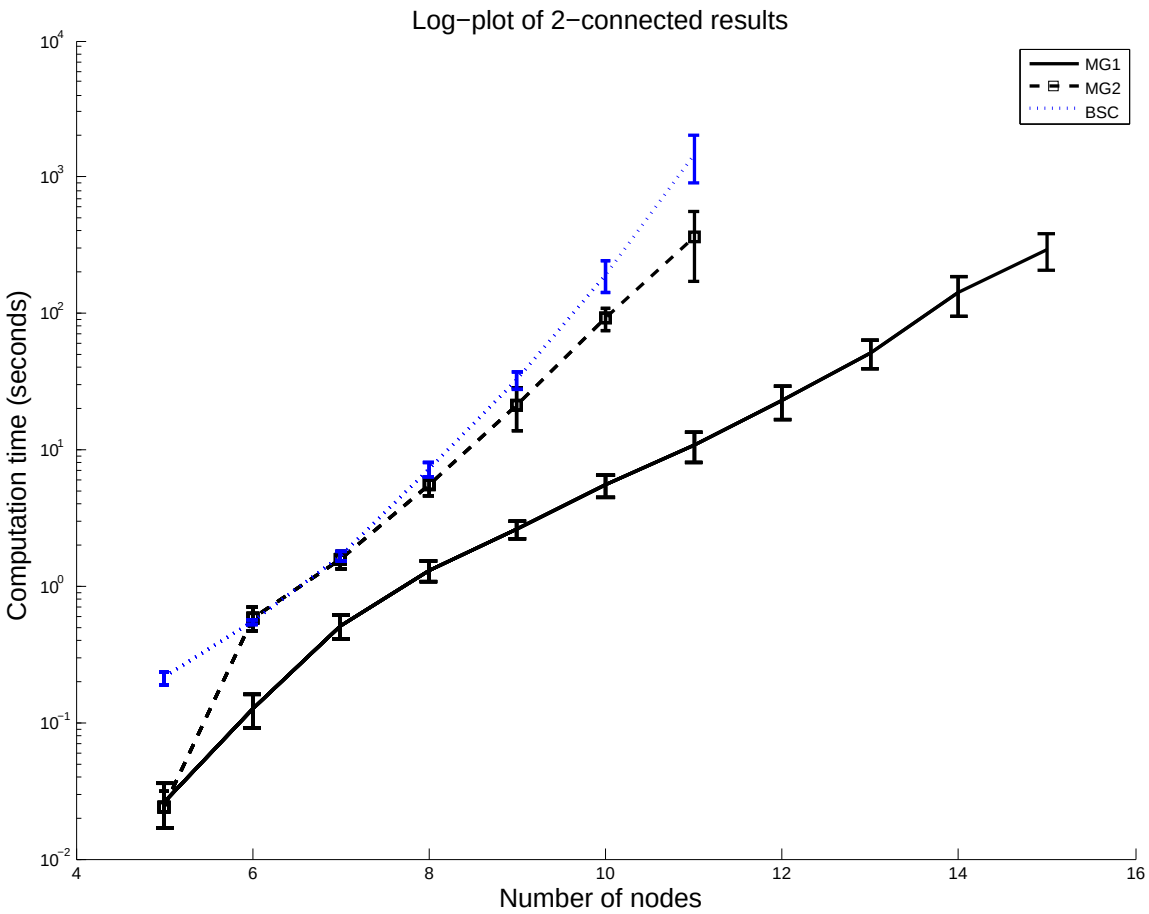
Spanning tree based method

A log-plot of the 1-connected solution time vs. network size.



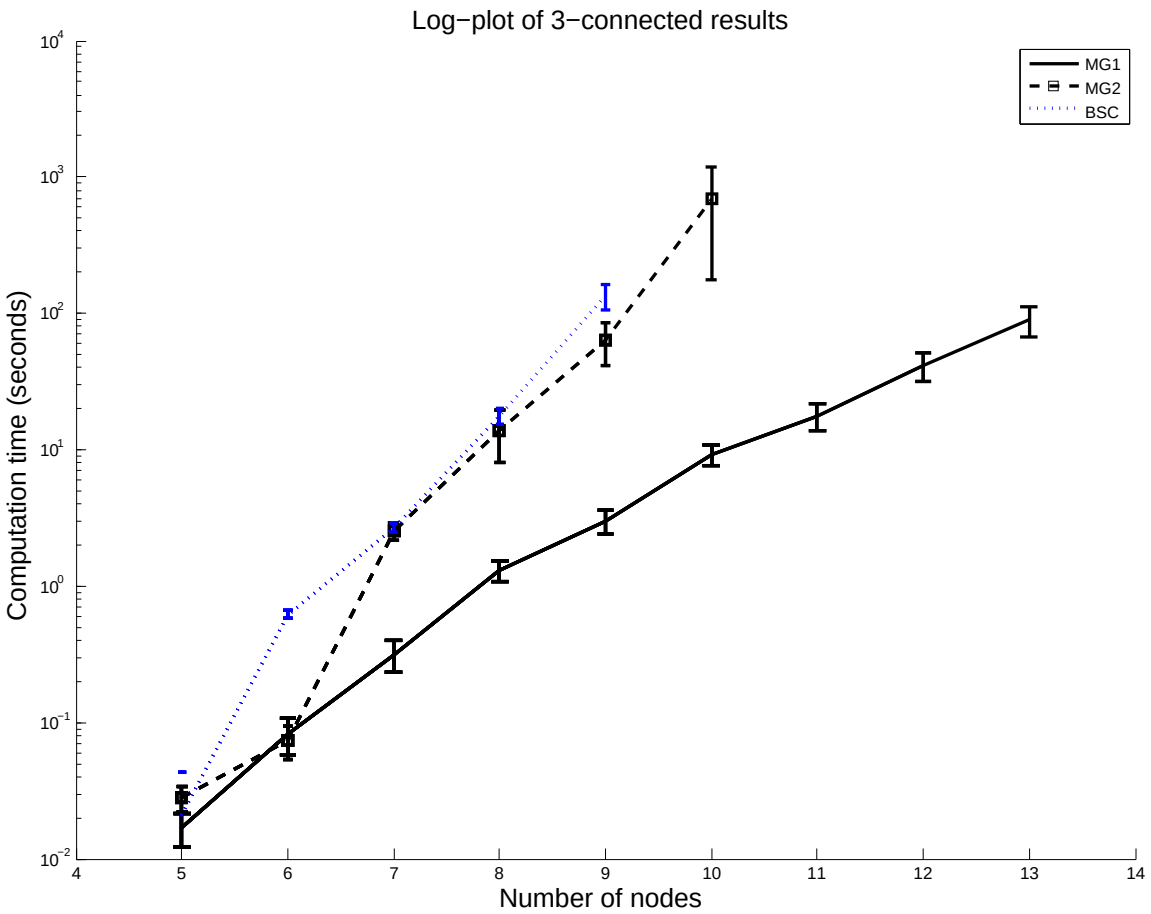
Spanning tree based method

A log-plot of the 2-connected solution time vs. network size.



Spanning tree based method

A log-plot of the 3-connected solution time vs. network size.



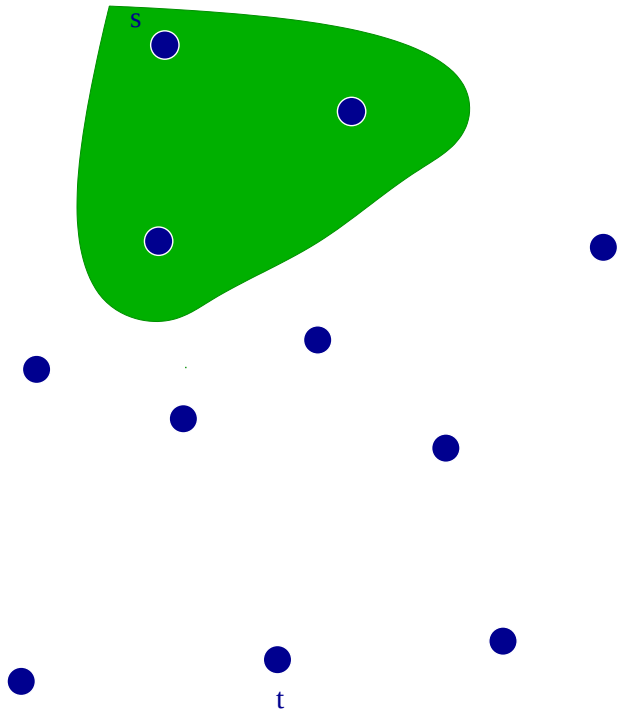
Spanning tree based method

We can obtain optimal solutions for up to 13 nodes in a 24 hour experiment of 50 random networks.

This method increases the number of constraints by a factor of $\binom{|V|}{k-1}$.

We think we can do better . . .

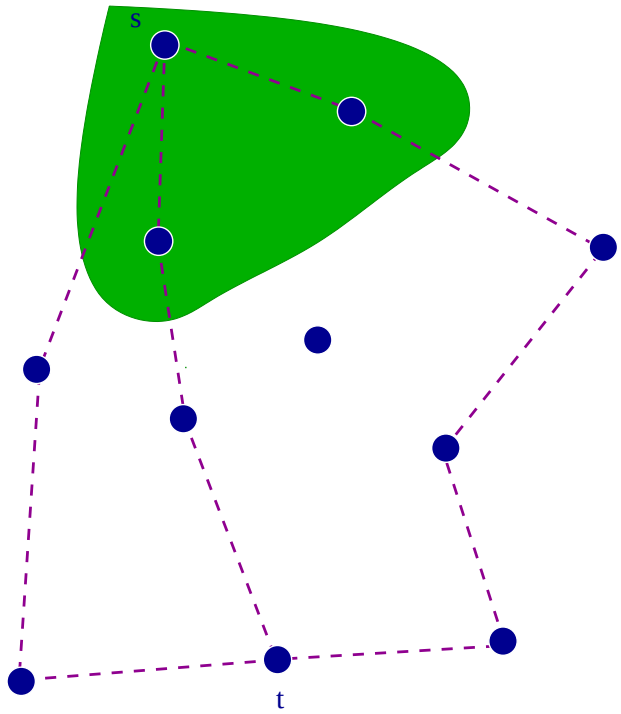
k node-disjoint paths concept



Theorem

Let G be an undirected graph containing at least $2k$ nodes and let $S = \{s_1, s_2, \ldots, s_k\}$ be a set of k nodes in G. Then G is k-connected if and only if, for each node $v \in G \backslash S$, there exist k node-disjoint paths from s_i to v for all i.

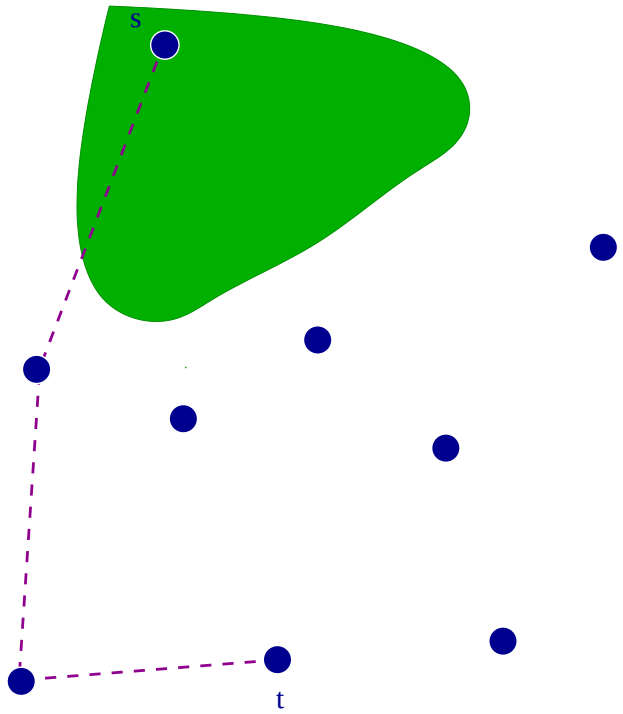
k node-disjoint paths concept



Theorem

Let G be an undirected graph containing at least $2k$ nodes and let $S = \{s_1, s_2, \ldots, s_k\}$ be a set of k nodes in G. Then G is k-connected if and only if, for each node $v \in G \backslash S$, there exist k node-disjoint paths from s_i to v for all i.

k node-disjoint paths concept



Theorem

Let G be an undirected graph containing at least $2k$ nodes and let $S = \{s_1, s_2, \ldots, s_k\}$ be a set of k nodes in G. Then G is k-connected if and only if, for each node $v \in G \backslash S$, there exist k node-disjoint paths from s_i to v for all i.

k node-disjoint paths concept

$$\text{Minimise} \sum_{(i,j)\in A} c_{i,j}\, y_{i,j}$$

$$\text{subject to } y_{i,j} \leq y_{i,a^i_j} \qquad \forall\ (i,j) \in A, a^i_j \neq i, \tag{21}$$

$$x^{l,m}_{i,j} \leq y_{i,j} \qquad \forall\ (i,j) \in A, l \in \{1,2,\ldots,k\}, m \in V\backslash S, \tag{22}$$

$$x^{l,m}_{i,j} \leq y_{j,i} \qquad \forall\ (i,j) \in A, l \in \{1,2,\ldots,k\}, m \in V\backslash S, \tag{23}$$

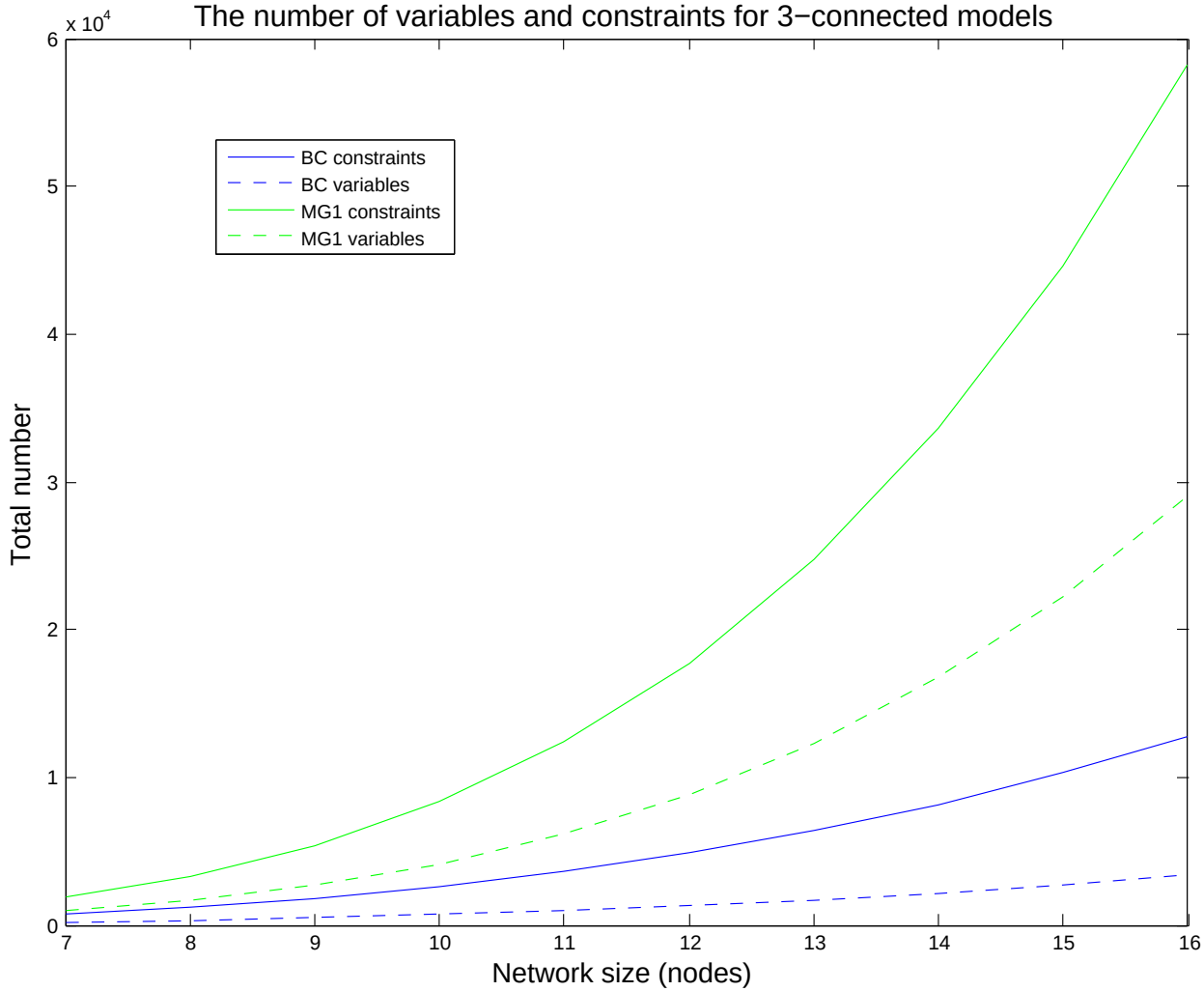
$$\sum_{j:(i,j)\in A} x^{l,m}_{i,j} - \sum_{j:(j,i)\in A} x^{l,m}_{j,i} = \begin{cases} k & \text{if } i = s_l, \\ -k & \text{if } i = m, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall\ i \in V, l \in \{1,2,\ldots,k\}, m \in V\backslash S, \tag{24}$$

$$\sum_{j:(i,j)\in A} x^{l,m}_{i,j} \leq 1 \qquad \forall\ i \in V\backslash\{s_l, m\}, l \in \{1,2,\ldots,k\}, m \in V\backslash S, \tag{25}$$

$$x^{l,m}_{i,j}, y_{i,j} \in \{0,1\}. \tag{26}$$

k node-disjoint paths concept

Constraints and variables



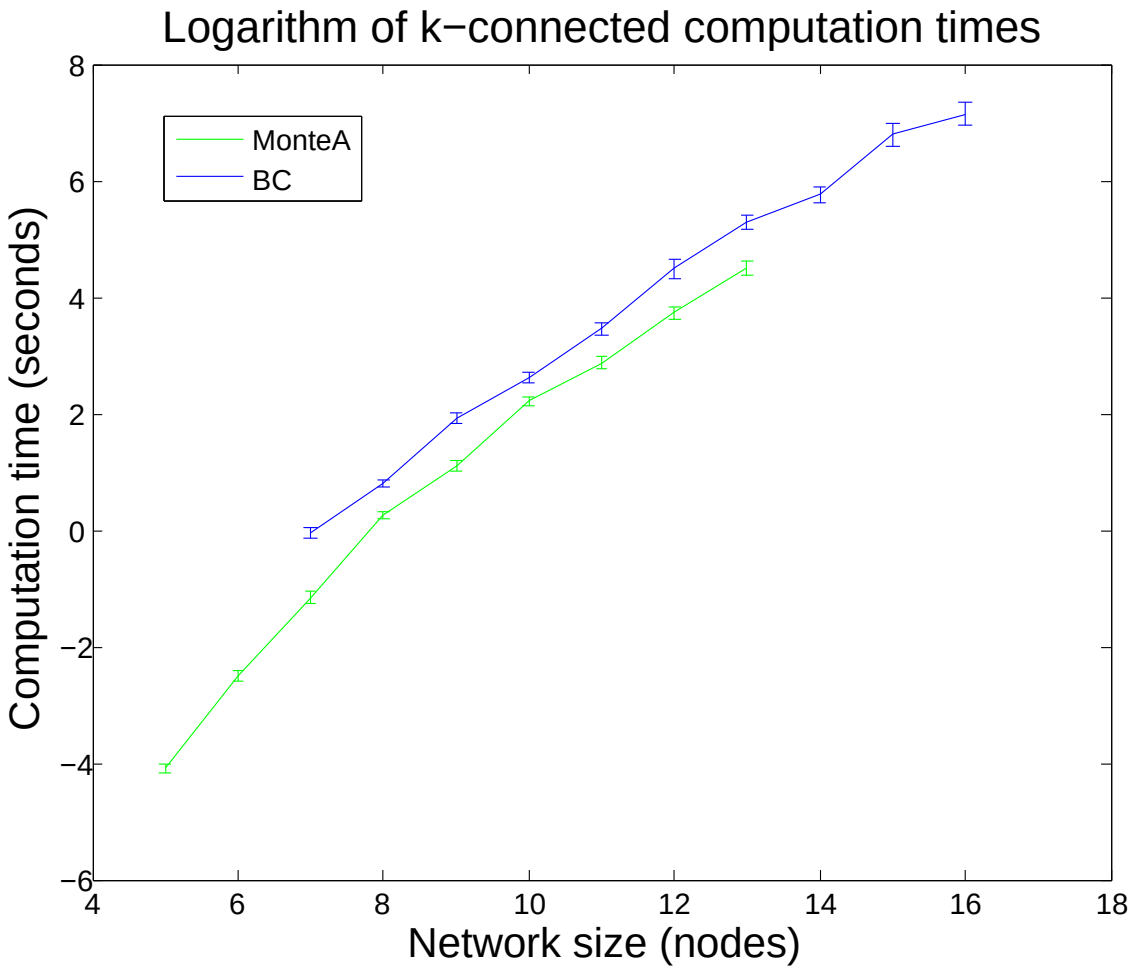
k node-disjoint paths concept

Constraints and variables for k-connected networks

	MG1	MG2	YBH	BC
variables	$\frac{1}{(k-1)!}n^{(k+1)}$	$\frac{1}{(k-1)!}n^{(k+1)}$	$\frac{1}{(k-1)!}n^{(k+2)}$	kn^3
constraints	$\frac{2}{(k-1)!}n^{(k+1)}$	$\frac{1}{(k-1)!}2^n n^{(k-1)}$	$\frac{1}{(k-1)!}n^{(k+1)} + n^3$	$2kn^3$

k node-disjoint paths concept

Computational results



Where do we go from here?

Academic questions:

- can we implement strengthening techniques for the multi-commodity flow problem to improve the instance size we can solve?
- are there any other IP tricks we can use to help solve larger instances?

Practical questions:

- we can start to think about local heuristics using our exact solutions.

Tutorial: Implementing a warm-start heuristic

A very brief introduction to warm start solutions

A warm start solution is an initial feasible solution which we hope will help reduce the size of the branch and bound search tree and therefore speed the time to obtain the optimal solution.

For it to work, we need to find a feasible value for every variable in our model.

To do this, you need to think about what you know of the application/problem itself.

A very brief introduction to warm start solutions

Recall that we have two types of variables in our model: transmission radii variables and flow variables.

We use the idea of finding the minimum spanning tree to find a feasible transmission radius and a version of the Ford-Fulkerson max-flow algorithm to find a feasible flow.

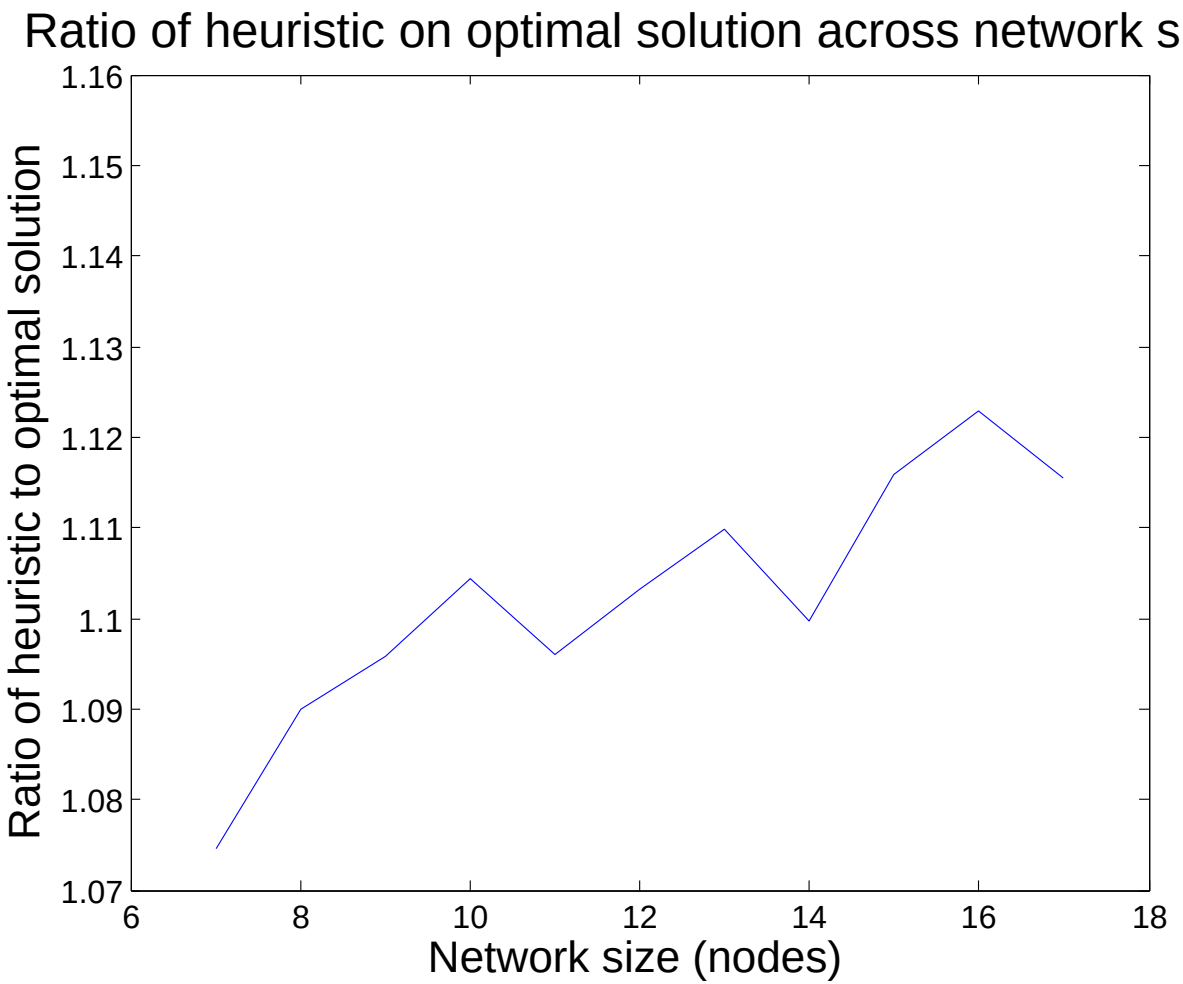
Warm start solution for transmission radius

Input: Set of all nodes, V; level of connectivity, k.
Output: Transmission radius vector, R.

Let L be the set of all node sets of size $k-1$.
for l in L **do**
  Let $V_l = V \backslash l$.
  Find minimum spanning tree for V_l.
  for i in V_l **do**
    $r_{l,i} = \max\{\text{edges from node } i\}$.
  end for
end for
for i in V **do**
  $R_i = \max_{l \in L} r_{l,i}$.
end for

This algorithm finds the transmission radius for each node. The range can then be used to calculate the connectivity between each pair of nodes. We suggest using Kruskal's algorithm to find the minimum spanning tree.

Warm start solution for transmission radius



Warm start solution for transmission radius

This heuristic gives us a reasonable estimate of the transmission radius when the size of the network is small.

Next we need to obtain a feasible solution for the flow variables. There are many ways to do this. A simple way that is guaranteed to work all the time is to use an adaptation of the max flow algorithm.

In order to get it to work, we need to alter the network a little so that we can guarantee that the paths we find remain node disjoint.

Warm start solution for flow variables

Adapted Max Flow algorithm with depth first search

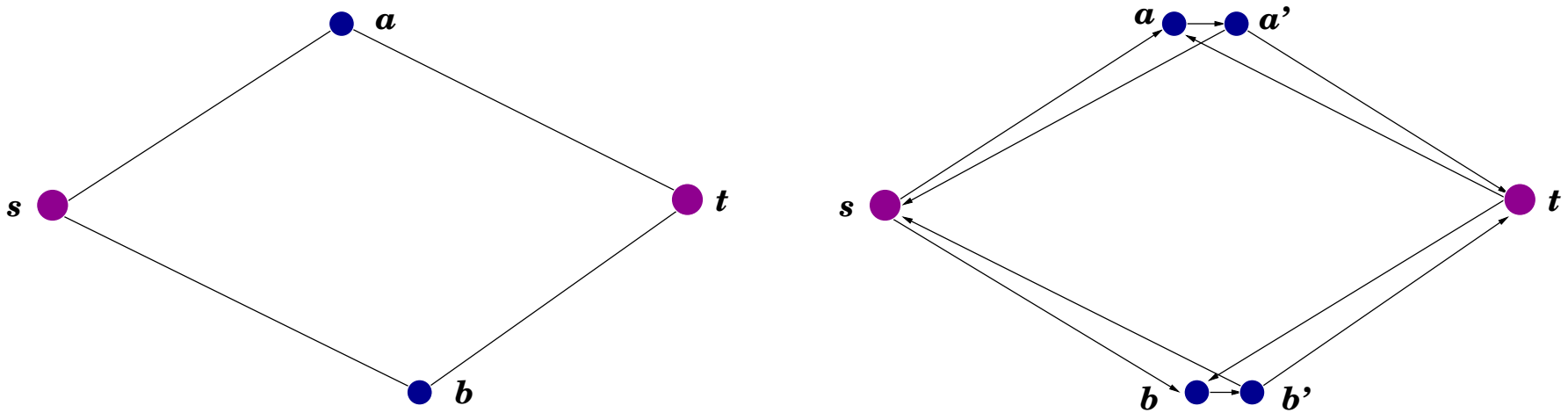
Input: C : the capacity matrix, E : the incidence matrix, s : the source, t : the destination, k : level of connectivity.

Output: Flow matrix F.

Let $f = 0$, $F = zeros(n, n)$.
while $f < k$ **do**
    Find a $path$ from s to t (e.g. using depth first search).
    Let $l = length(path)$.
    $f = f + 1$.
    for i in $(1{:}l - 1)$ **do**
        Reduce capacity on each edge in the path.
        Increase capacity on reverse edges.
        Increase flow on each edge in the path.
    end for
end while

Warm start solution for flow variables

However, to ensure that the solution you obtain is node-disjoint, we must transform the network to a directed network and split the interior nodes as shown below:



Warm start solution for flow variables

Transforming the network for each s and t

1: $C =$ zeros(2*$|V|$, 2*$|V|$).
2: **for** $i \in V$ **do**
3: **if** $(i \neq s$ and $t)$ **then**
4: $C(i, |V| + i) = 1.$
5: **end if**
6: **end for**
7: **for** $i \in V$ **do**
8: **for** $j \in (i + 1 : V)$ **do**
9: **if** $E(i, j) = 1$ **then**
10: $C(i + |V|, j) = 1.$
11: $C(j + |V|, i) = 1.$
12: **end if**
13: **end for**
14: **end for**

The overall heuristic

The complete heuristic is in the following order:

1. Find the transmission radii using the minimum spanning tree heuristic.
2. For each possible source and destination pair:
   - 2.1 Transform the network using the algorithm on the previous slide;
   - 2.2 Apply the adapted Max Flow algorithm provided above;
   - 2.3 Transform the network flow results back to the original network.

What to do?

Implement step 1 of the warm start algorithm in the programming language of your choice. You may steal bits of code from the internet to aid your cause.

The first to finish the heuristic for finding transmission radius variable values and report the correct upper bound for the test problem may choose one prize.

Adventurous students may attempt to implement the entire heuristic to gain a prize.

We encourage you to collaborate.

Acknowledgements

The authors would like to thank MASCOS for supporting our research.

We welcome comments and collaborations.