Urban transit network design using spanning tree: A case study of Canberra transit network

Satoshi Sugiura[1)*], Kam-Fung Cheung[2)], Michael Bell[3)], Hitomi Nakanishi[4)], Fumitaka Kurauchi[5)], Supun Perera[6)] and Yogi Vidyattama[4)]

[1]Graduate School of Engineering, Hokkaido University, Hokkaido, Japan

[2]School of Information Systems and Technology Management, University of New South Wales Business School, Sydney, Australia

[3]Institute of Transport and Logistics Studies, University of Sydney Business School, Sydney, Australia

[4]School of Design and Built Environment, Faculty of Arts and Design, University of Canberra, Canberra, Australia

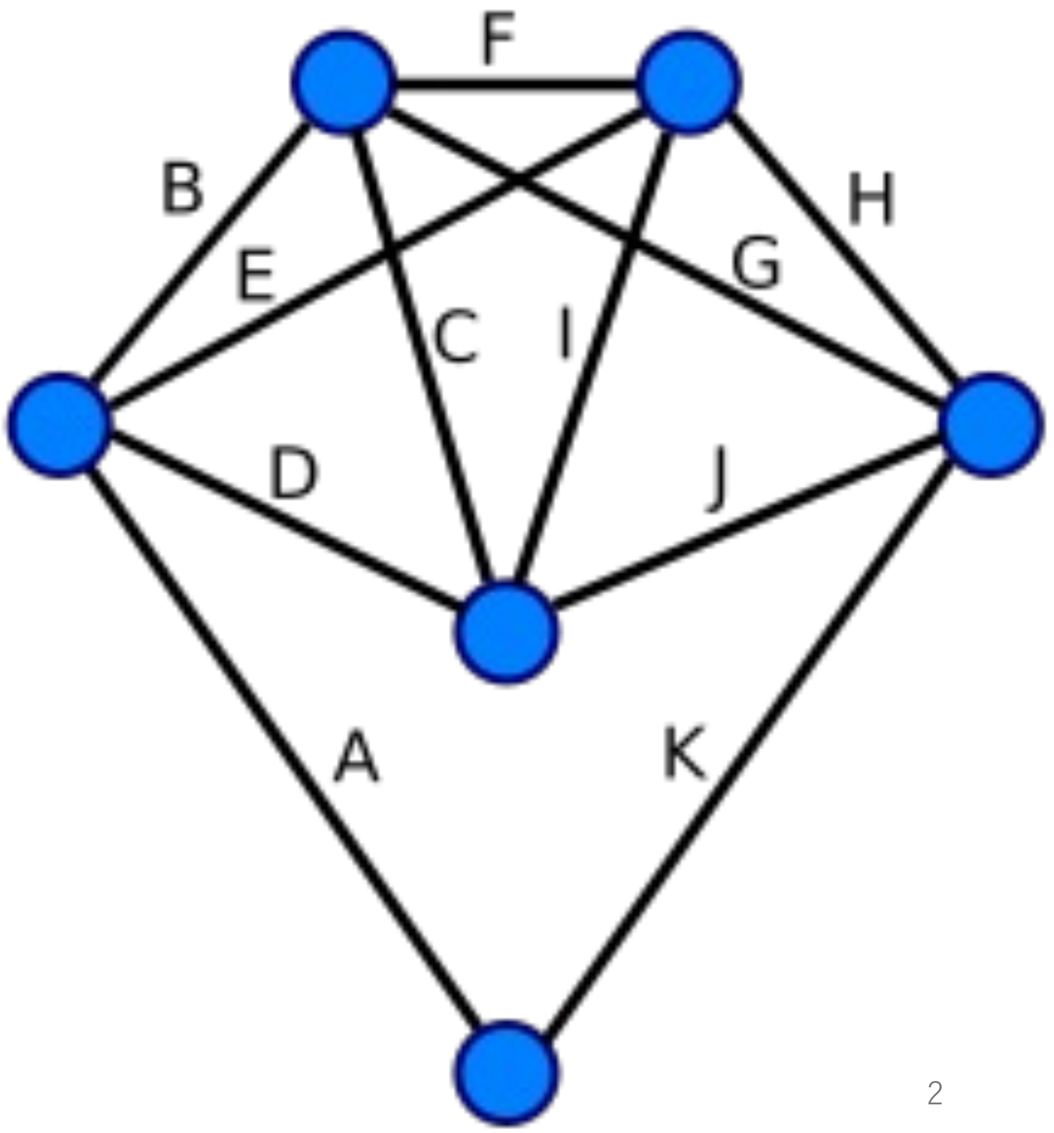
[5]Faculty of Engineering, Gifu University, Gifu, Japan

[6]Canberra School of Politics, Economics and Society, University of Canberra, Canberra, Australia

*Corresponding author: sugiura@eng.hokudai.ac.jp

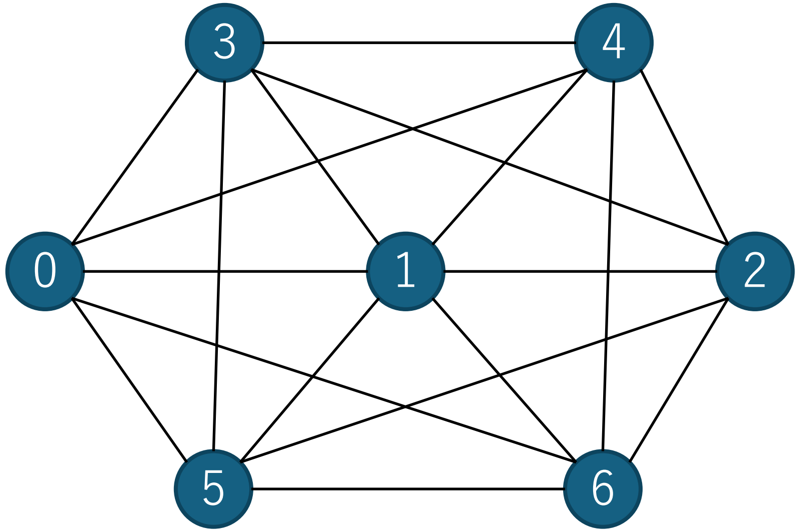
Graph theory (Eulerian circuit)

Can you determine whether there is a path that traverses all edges exactly once and returns to the starting node?



Background

- Passenger costs are minimized if all the shortest paths between OD pairs are planned to be traversed.
  - The operator will have a relatively inefficient operation plan.
  - e.g., Number of vehicles, driver, no transit, etc.,

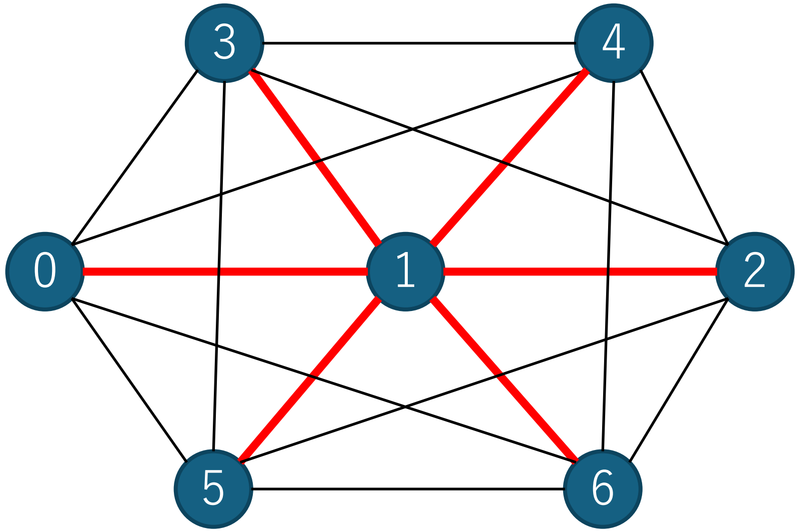


Background (cont.)

- Bundling travel demand into a few routes will make operators more cost-efficient.
  - How much should we bundle?
  - Spanning tree illustrates the route with the most tightly bundled trip demands.

Spanning Tree of original graph

$|\dot{E}| = |\dot{V}| - 1$



Bell et al. (2020)

- Bell et al. (2020) provide the formulation and heuristic solution algorithm for the ferry network design problem with ST constraints.
  - Their model maximize passenger utility by maximizing entropy.
  - Our model minimize total passenger kilometers (TPK)
- They propose two heuristic algorithm to solve the model.
  - Heuristic 1: link swapping
  - Heuristic 2: link deletion
- The computational load of their method is high
  - We aim to develop an efficient solution algorithm.

Objective

- We aim to find a spanning tree with minimized passenger kilometers.
  - The optimized tree is expected to show the essential routes and hub location.
- Cayley's formula tells us there are $n^{(n-2)}$ spanning trees in a network of n nodes.
  - The full search method can not be employed for a large network.
  - Also, the combinatorial optimization method seems to be unsuitable.
- We derive a meta-heuristic method to solve this problem.

Contribution

- Regarding transit network design, the proposed model aims to locate hubs and trunk routes in the preliminary planning stage.
- We developed an efficient tabu search heuristic to solve the proposed model for a promising spanning tree.
- We developed a greedy algorithm by adding additional links to the spanning tree obtained from the proposed tabu search.
- Canberra bus network data and compare the performance with the state-of-the-art heuristics in Bell et al. (2020) in the context of transit network design.

Problem definition

- Minimize TPK
- ST topology constraint
  - $\left|\dot{E}\right| = |V| - 1$
  - $\forall(u, v) \in V$ are connected

$$\min_{x,y} \sum_{w \in W} d_w c_w \tag{1}$$

Subject to

$$d_w = \sum_{p \in \mathcal{P}_w} b_{wp} h_p \,;\forall w \in W \tag{2}$$

$$c_w = \sum_{(ij) \in E} \sum_{p \in \mathcal{P}_w} b_{wp} q_{(ij)p} t_{(ij)} \;;\forall w \in W \tag{3}$$

$$x_{(ij)} = \sum_{w \in W} \sum_{p \in \mathcal{P}_w} b_{wp} q_{(ij)p} h_p \,;\forall i, j \in N \tag{4}$$

$$\tau \geq \sum_{w \in W} \sum_{p \in \mathcal{P}_w} c_w b_{wp} h_p = \sum_{(ij) \in E} t_{(ij)} x_{(ij)} \tag{5}$$

$$\sum_{i,j \in N} y_{(ij)} \leq |N| - 1 \tag{6}$$

$$\sum_{i,j \in V} y_{(ij)} \leq |V| - 1; \forall V \subset N, V \neq N, V \neq \emptyset \tag{7}$$

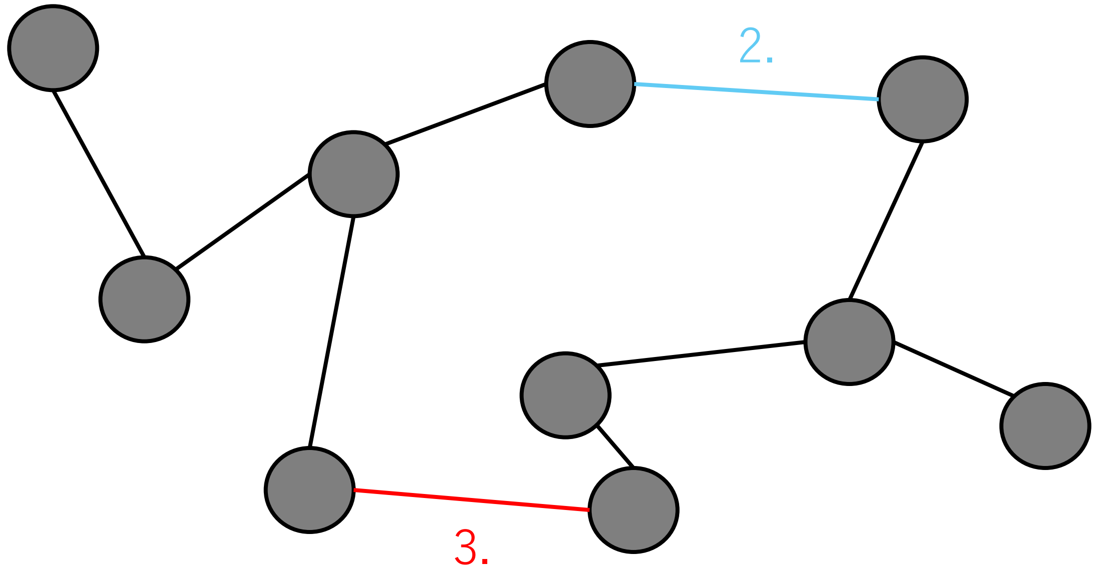
$$h_p > 0; \forall p \in \mathcal{P} \tag{8}$$

$$x_{(ij)} \leq M y_{(ij)}; \forall i, j \in N \tag{9}$$

$$y_{(ij)} \in \{0,1\}; \forall i, j \in N \tag{10}$$

Solution algorithm in Bell et al. (2020) Heuristic 1: link swapping

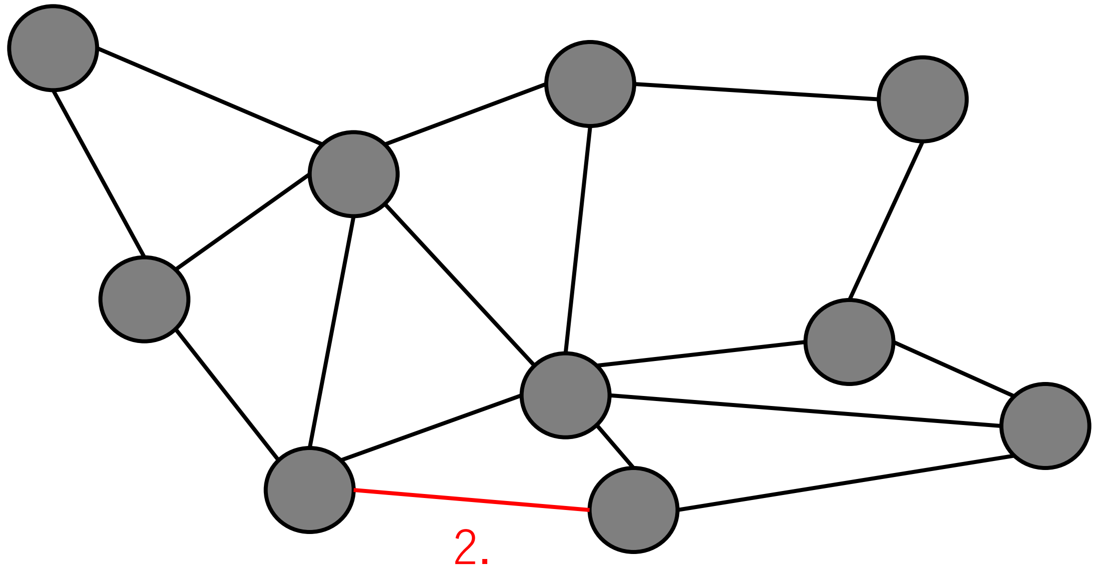
1. Find the spanning tree that minimises sailing time (proxied by distance) using Kruskal's algorithm, and store the corresponding links in *E*.
2. Find the link that when inserted decreases *TPK* most, add this link to *E*.
3. Find the link that when deleted increases *TPK* least while ensuring that the resulting network remains fully connected. Remove this link from *E*.
4. Return to Step 2 until the link deleted is the link just inserted, at which point a local optimum is reached.
5. When a local optimum is reached, taboo the link just inserted and return to Step 2 until there are no further links to taboo.



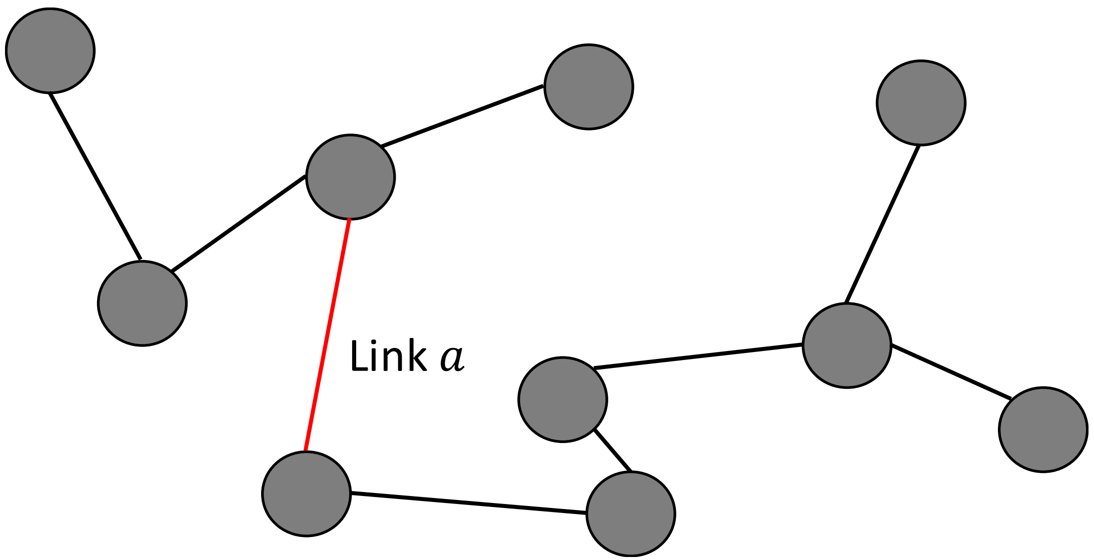
A huge number of shortest path search is required.

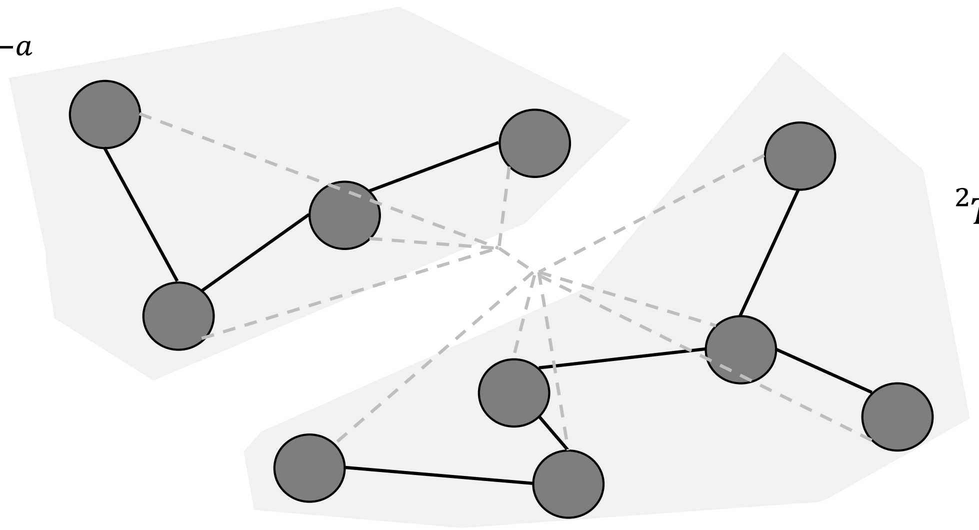
Solution algorithm in Bell et al. (2020) Heuristic 2: link deletion

1. Connect all pairs of ferry stations to each other and store the $n(n-1)/2$ links, where $n = |F|$, in E.
2. Find the link that when deleted decreases TPK least while ensuring that the resulting network remains fully connected. Remove this link from E.
3. Repeat Step 2 until a spanning tree is reached (until $|E| = n - 1$).



An idea to improve link swapping algorithm



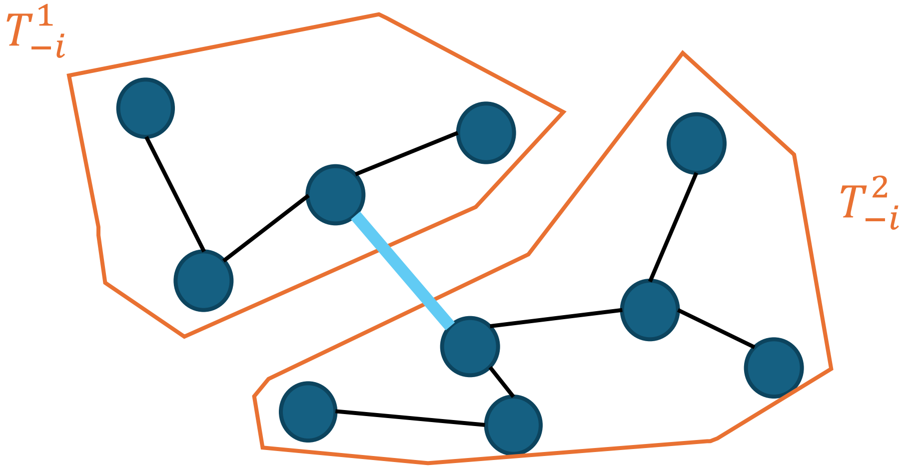


A graph that reconnects two subgraphs is always a spanning tree.

To calculate TPK on temporary tree

$$
\begin{aligned}
&Z\left(a, b \middle| {}_nT\right) \\
&= \sum_{i \in N\left({}^1_nT_{-a}\right), j \in N\left({}^2_nT_{-a}\right)} \left[\left(c_{(ib(1))} + c_{b(2)j}\right)\left(d_{(ij)} + d_{(ji)}\right) + t_{(b(1)b(2))}\left(d_{(ij)} + d_{(ji)}\right)\right] \\
&+ \sum_{i,j \in N\left({}^1_nT_{-a}\right), i \neq j} c_{(ij)}\left(d_{(ij)} + d_{(ji)}\right) \\
&+ \sum_{i,j \in N\left({}^2_nT_{-a}\right), i \neq j} c_{(ij)}\left(d_{(ij)} + d_{(ji)}\right)
\end{aligned}
$$

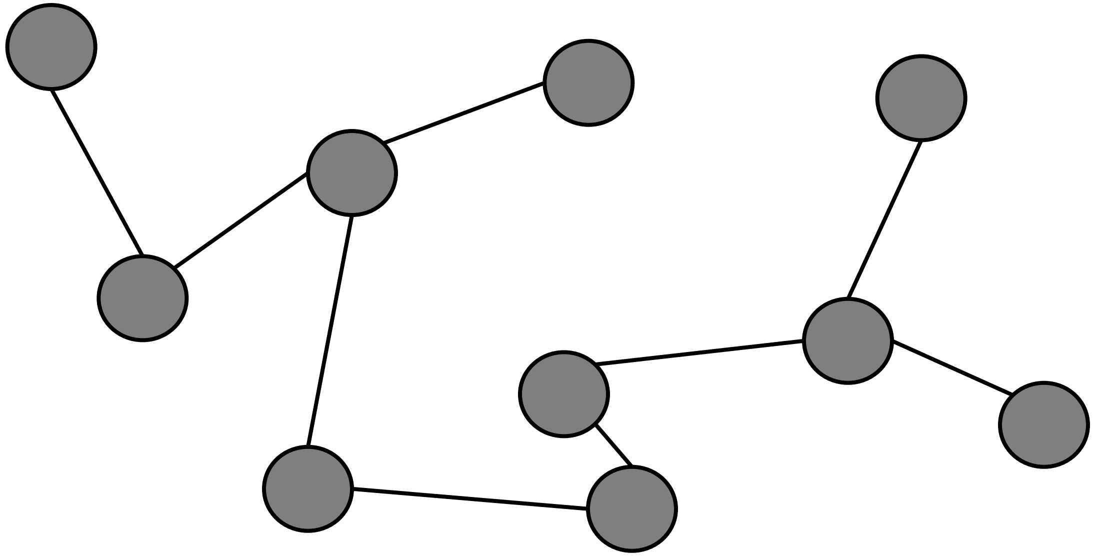
- No shortest path is needed if the distance matrix of ${}_nT$ is obtained.



$k_i = n(T^1_{-i}) \times n(T^2_{-i})$

When reconnecting, no shortest-path search is needed to compute TPK.

Solution algorithm
1. initialization



$$n \leftarrow 1,$$
$$_nT \leftarrow Kruskal\big(G(V,E)\big)$$
$$T^* \leftarrow \ _nT,$$
$$Z^* \leftarrow Z(T^*)$$

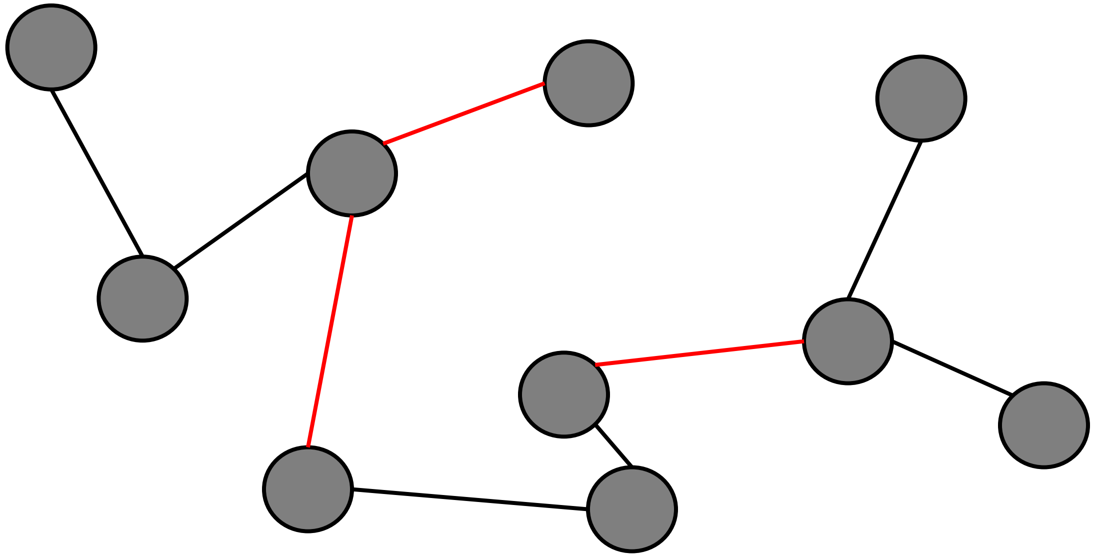
Distance matrix ($\boldsymbol{c}$)

OD	1	2	…	
1	0	△		
2	△	0		
…				

Tabu list ($\mathcal{L}$)

	Removed	Added
1		
2		
…		

Solution algorithm
2. finding local optima



Randomly select ψ links in ${}_nT$ and store them in $\mathcal{A}$

$$\boldsymbol{\gamma} \leftarrow \emptyset$$

FOR each link a in $\mathcal{A}$

$$\{{}^1_nT_{-a}, {}^2_nT_{-a}\} \leftarrow {}_nT \setminus a$$

$$\mathcal{B}_a \leftarrow \{(i,j) \in E - \dot{E} \mid i \in {}^1_nT_{-a}, j \in {}^2_nT_{-a}\}$$

FOR each link b in $\boldsymbol{\mathcal{B}_a}$

$${}_nT_{-a}^{+b} \leftarrow \{{}^1_nT_{-a}, {}^2_nT_{-a}\} \cup b$$

$$\gamma_{(ab)} \leftarrow Z\left({}_nT_{-a}^{+b}\right)$$

$$\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} \cup \{\gamma_{(ab)}\}$$

ENDFOR

ENDFOR

$$(a', b') \leftarrow \arg\min_{a \in \mathcal{A}, b \in \mathcal{B}_a} \{\gamma_{(ab)} \mid \gamma_{(ab)} \in \boldsymbol{\gamma}\}$$

$$\gamma_{(a'b')} \leftarrow Z\left({}_nT_{-a'}^{+b'}\right),$$

Solution algorithm
3. update the solution

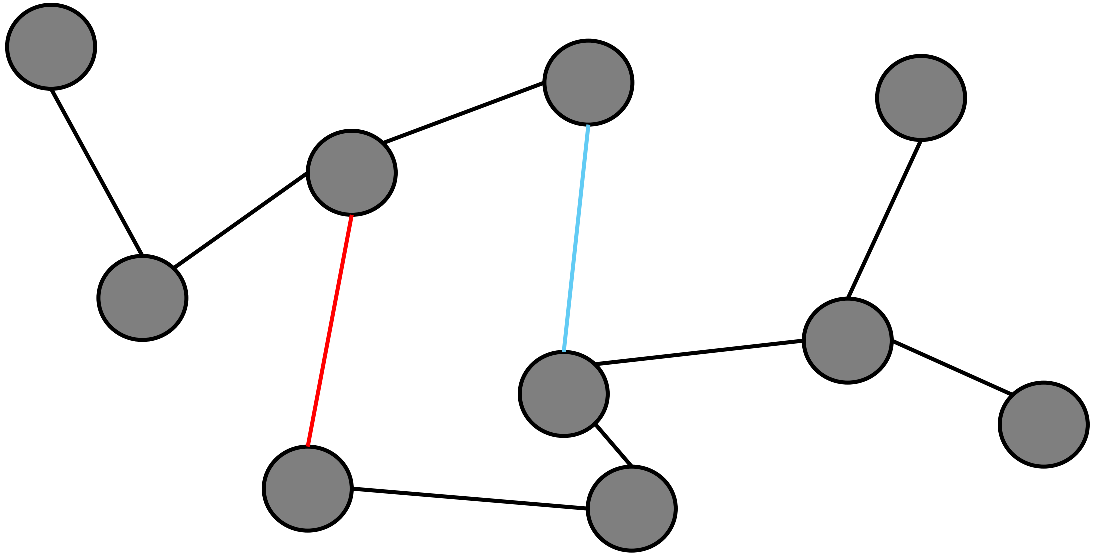
IF $\gamma_{(a'b')} < Z^*$

$$Z^* \leftarrow \gamma_{(a'b')}$$

$$T^* \leftarrow {}_{n}T_{-a'}^{+b'}$$

$${}_{n+1}T \leftarrow {}_{n}T_{-a'}^{+b'}$$

$$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\acute{a}, \acute{b})\}$$



Tabu list ($\mathcal{L}$)

	Removed	Added
1	$\acute{a}$	$\acute{b}$
2		
…		

(Note: If $\mathcal{L}$ is full, remove the oldest link swap, then update $\mathcal{L}$ by including $(\acute{a}, \acute{b})$.)

Solution algorithm
3. update solution

ELSE

FOR each $\boldsymbol{\gamma}$ in $\boldsymbol{\Upsilon}$

IF $(\mathbf{a}', \mathbf{b}')$ not in $\boldsymbol{\mathcal{L}}$

$$ {}_{n+1}T \leftarrow {}_{n}T_{-a'}^{+b'} $$

EXITFOR

ELSE

$$ \boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} \setminus \left\{ \gamma_{(a'b')} \right\} $$

ENDIF

$$ (a', b') = \arg \min_{a \in \mathcal{A}, b \in \mathcal{B}_a} \left\{ \gamma_{(ab)} \middle| \gamma_{(ab)} \in \boldsymbol{\gamma} \right\} $$

$$ \gamma_{(a'b')} = Z\left({}_{n}T_{-a'}^{+b'} \right) $$

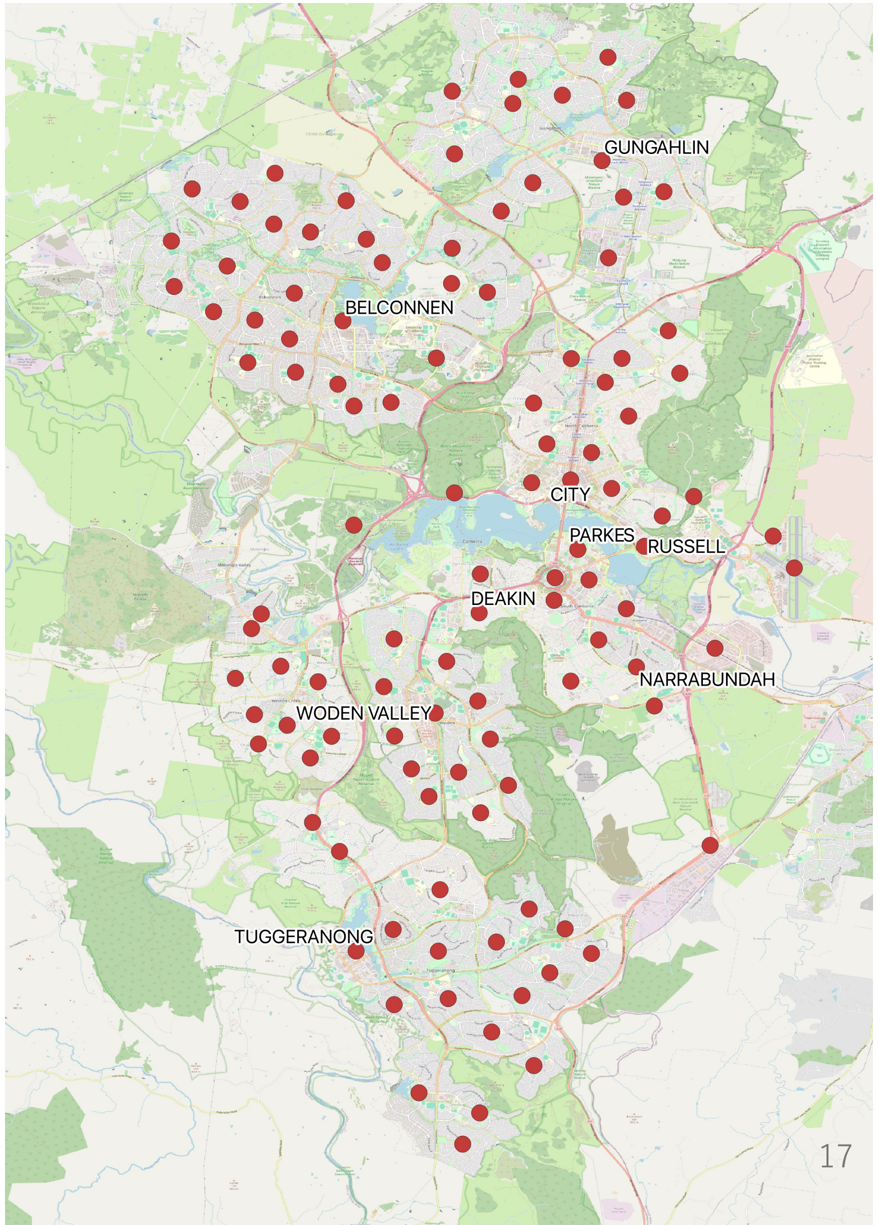
ENDFOR

$$ \mathcal{L} \leftarrow \mathcal{L} \cup \left\{ \left(\acute{a}, \acute{b} \right) \right\} $$

- If the swapping operation as local optima is in tabu, seek the next best operation.
- The next best operation is allowed and carries the ST that gave that operation to the next iteration.

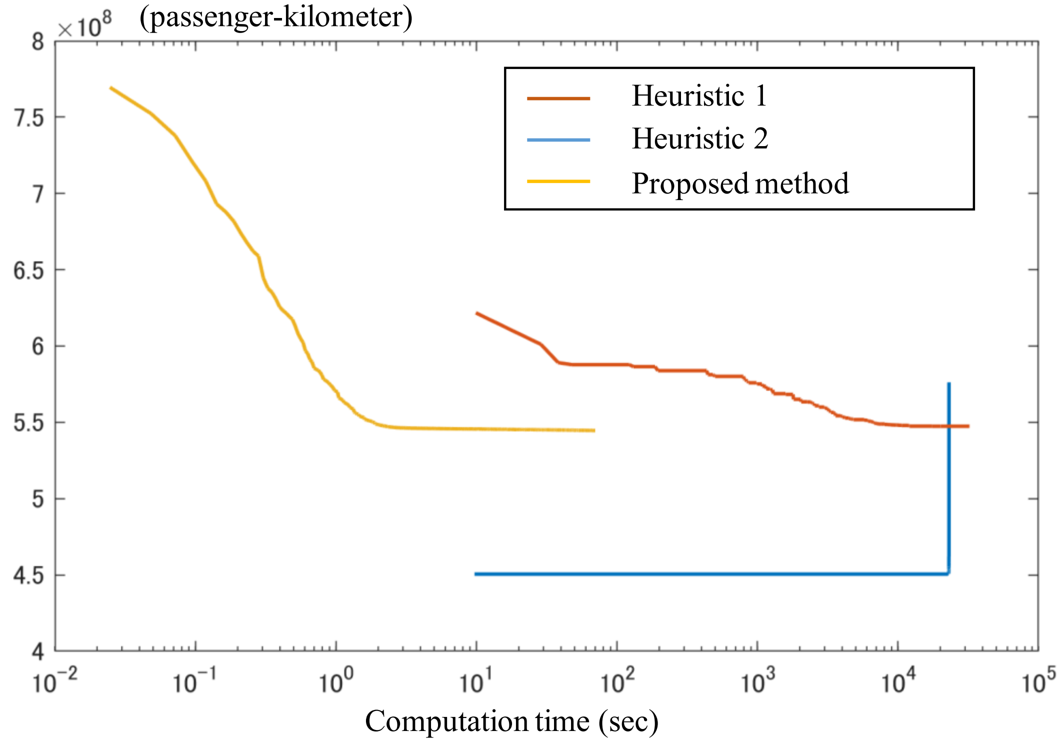
Application to Canberra bus smart card data

- Canberra
  - Population: 460,900
  - Area: 807.6 km^2
- Smart card data
  - 1,207,494 trip (2016)
  - 111 Suburbs
    - 111×111 OD trip table
    - Aggregated bus stops to suburbs



Computational result

- Heuristic 1
  - Iteration: 6105 $\left(= 111 \times \frac{110}{2}\right)$
- Heuristic 2
  - Iteration: 5995 $(= 6105 - 110)$
- Proposed method
  - Iteration: 3000
  - $\psi = 5$ (candidate of deleted links)
  - $|\mathcal{L}| = 80$ (length of tabu)



	Heuristic 1 (Link Swapping)	Heuristic 2 (Link Deletion)	Proposed method
Computation time (s)	32,237	23,047	69
Objective value (TPK) (in 10^3 km)	547,418	576,220	544,633

CPU: AMD Ryzen3800x, RAM: DDR4-3200 64GB, OS: Windows 10, Coded on Matlab 2019b.
(*) the computation time and the objective value are obtained by averaging 100 simulations using the algorithm

Comparison for other ST

- Minimum distance spanning tree (MST)

$$\min_{\boldsymbol{y}} \sum_{i,j \in N, i \neq j} t_{(ij)} y_{(ij)}$$

- Minimizing total length of tree
- Solved by Kruskal's algorithm

- Maximum demand spanning tree (MDST)

$$\min_{\boldsymbol{y}} \sum_{i,j \in N, i \neq j} -d_{(ij)} y_{(ij)}$$

- Maximizing direct connected OD demand
- Solved by Kruskal's algorithm

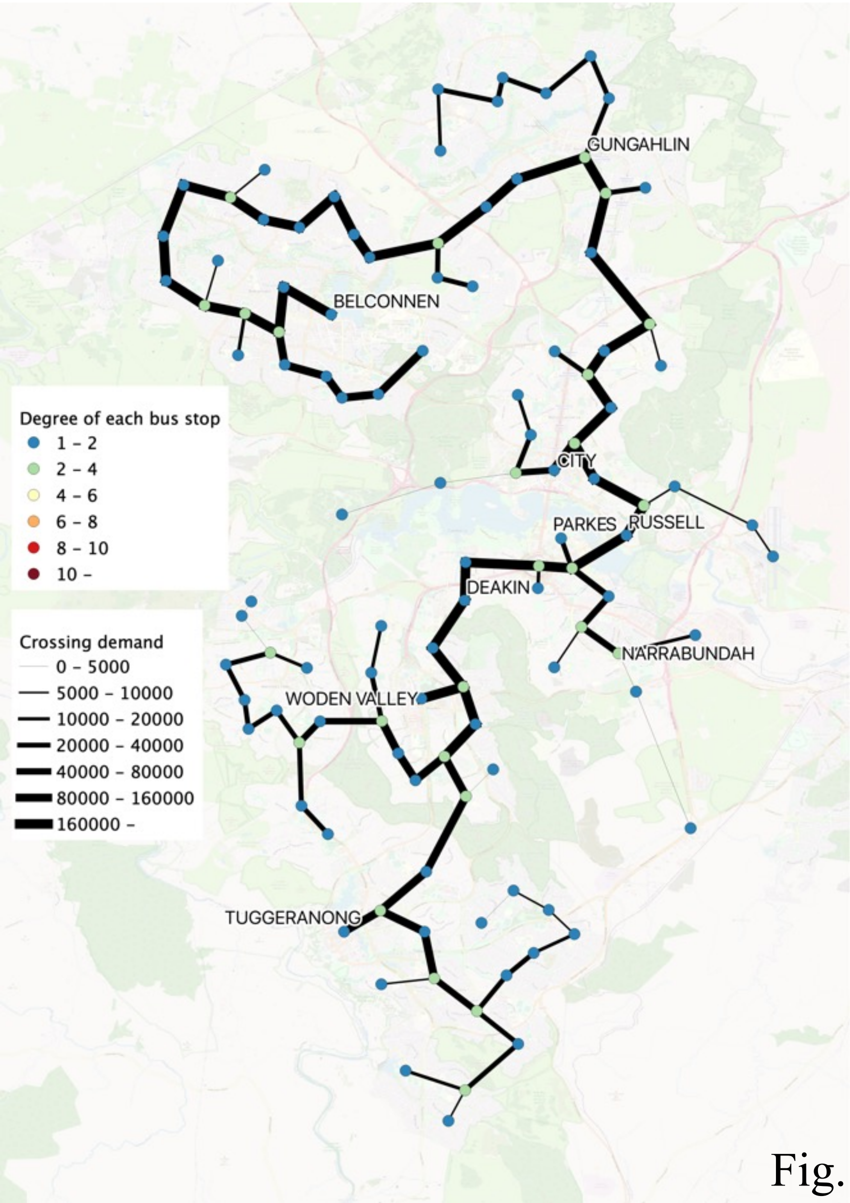


Fig. Minimum distance spanning tree

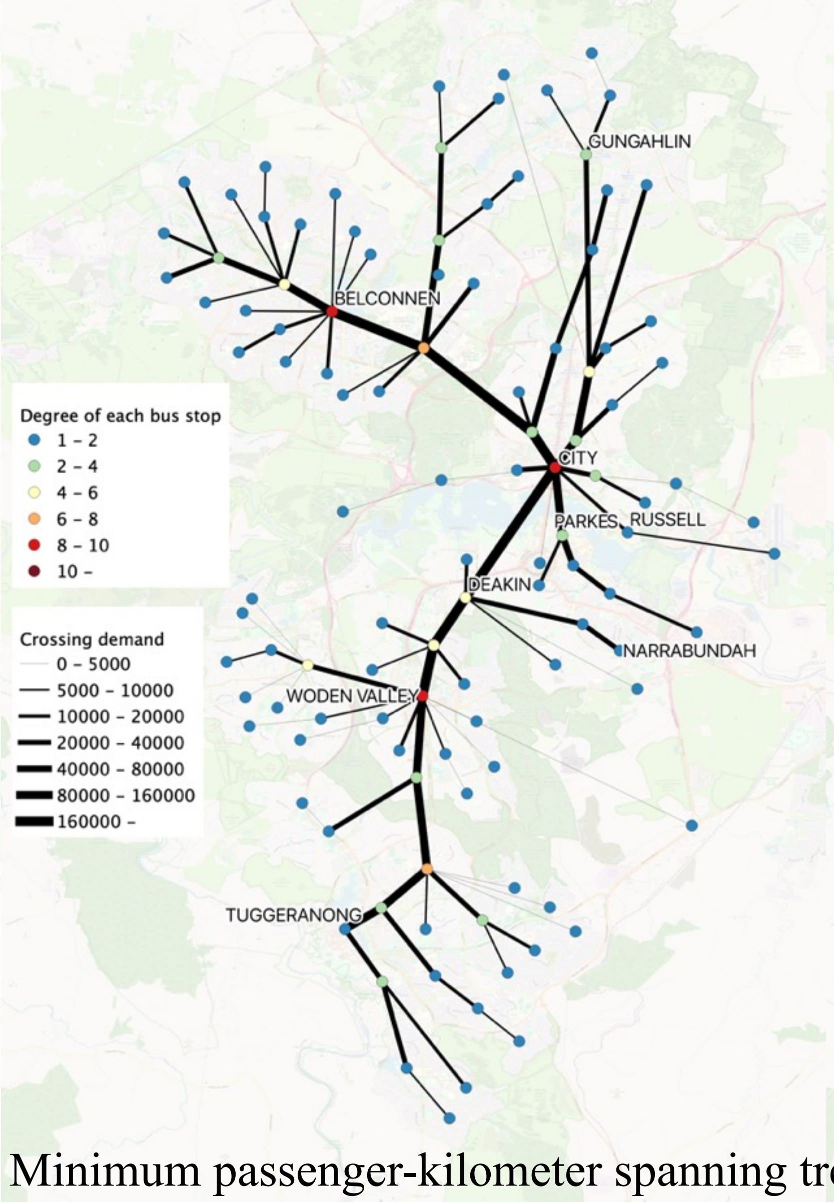


Fig. Minimum passenger-kilometer spanning tree

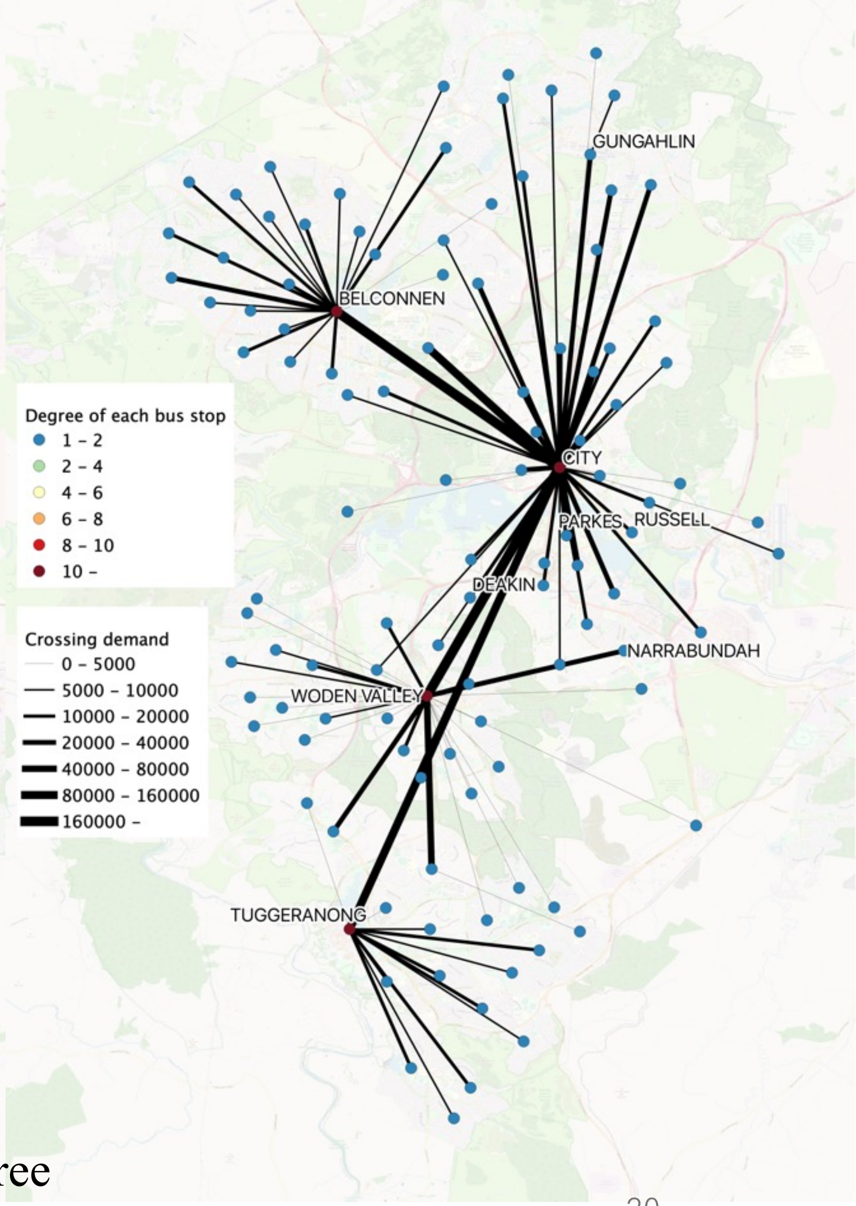


Fig. Maximum demand spanning tree

Comparison for other ST

- MPKST connects high-demand nodes and several branch lines.
  - MPKST indicates that the red nodes with higher degrees are shown at the major traffic points such as Belconnen, City and Woden Valley.

	Total passenger-kilometers	Percentage change with respect to MST	Percentage change with respect to MDST
MST	2,375,188,779	-	+49.31%
MDST	1,590,809,637	-33.02%	-
MPKST	544,329,352	-77.08%	-65.78%

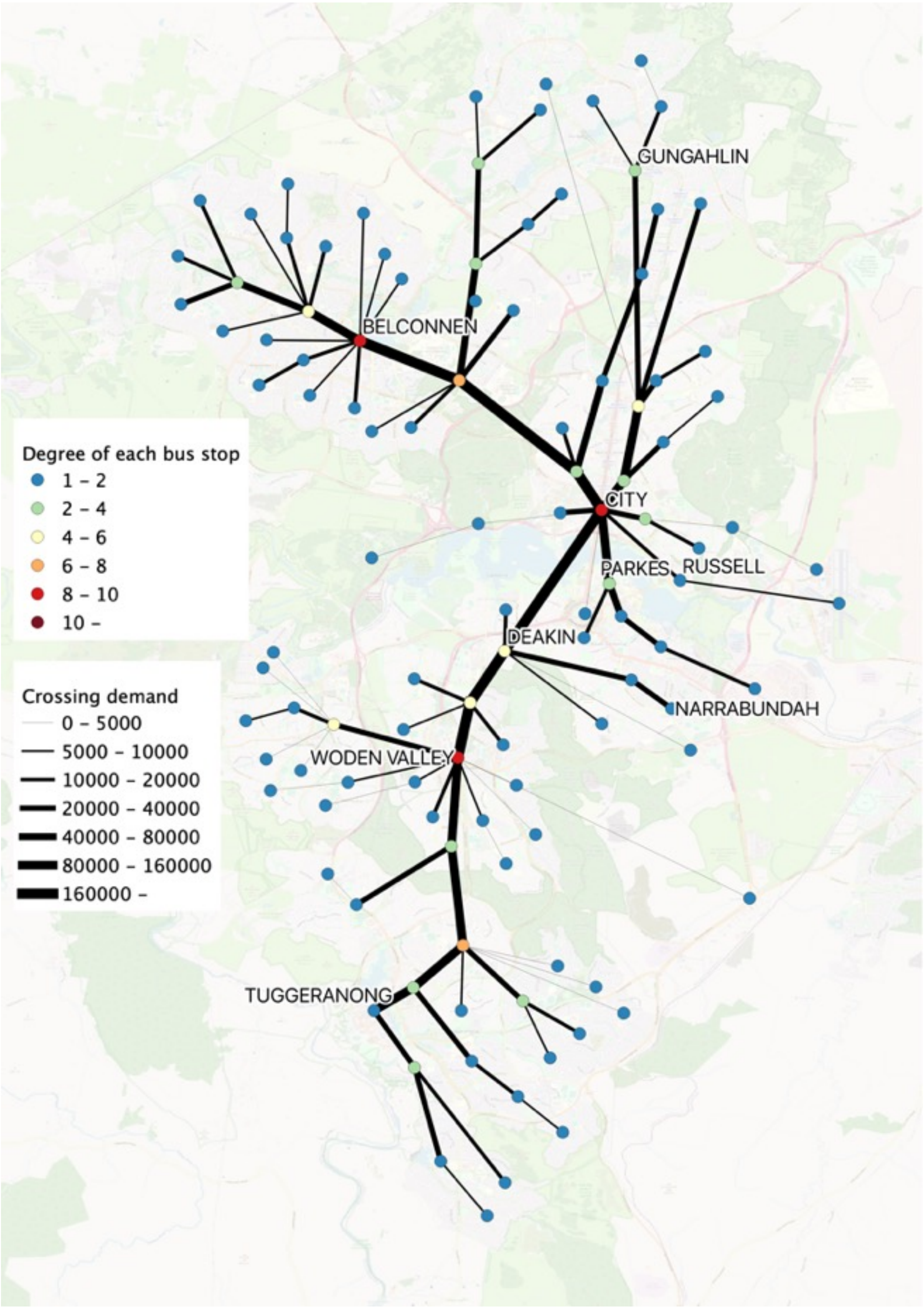


Fig. MPKST(proposed method)

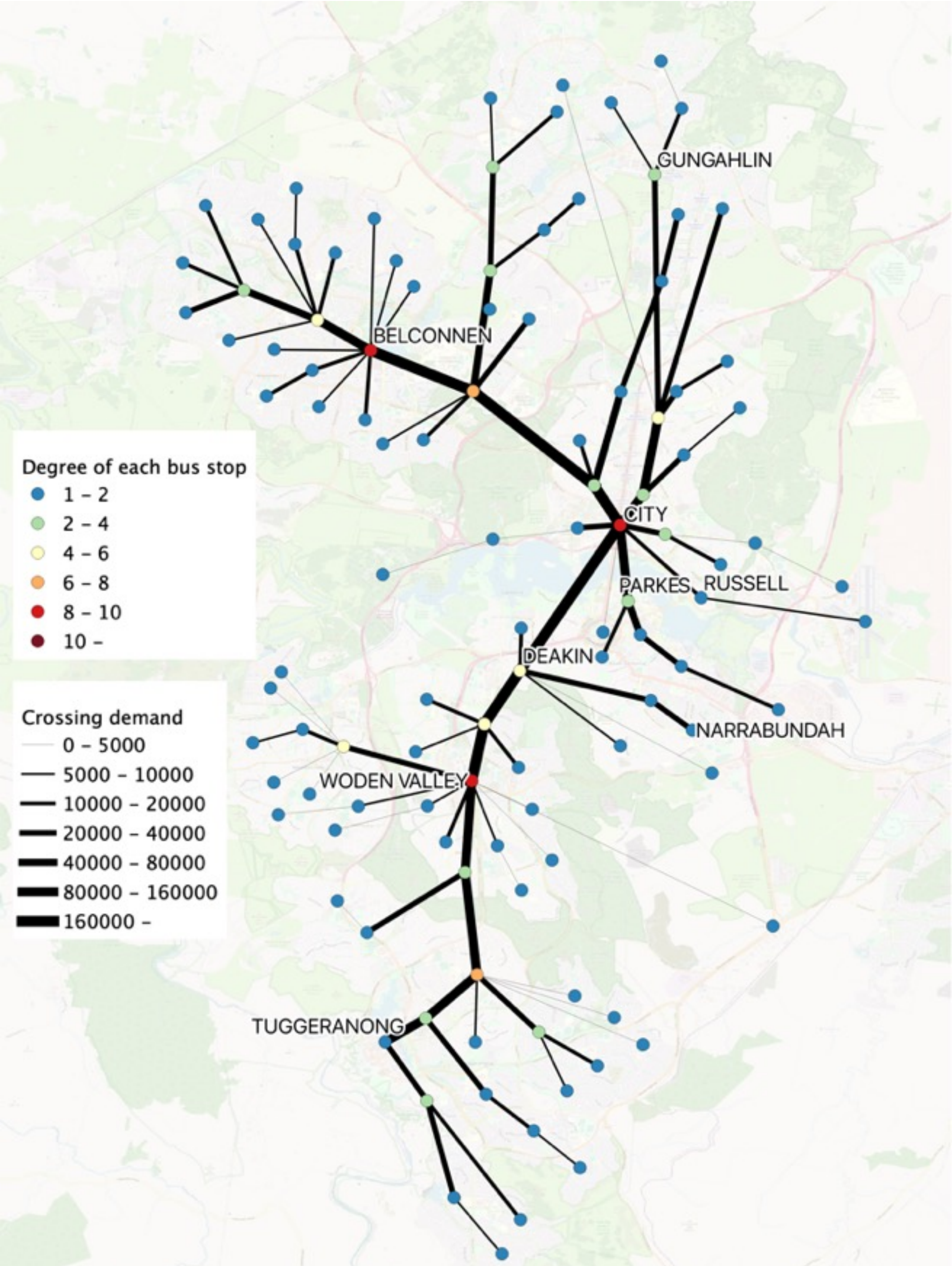
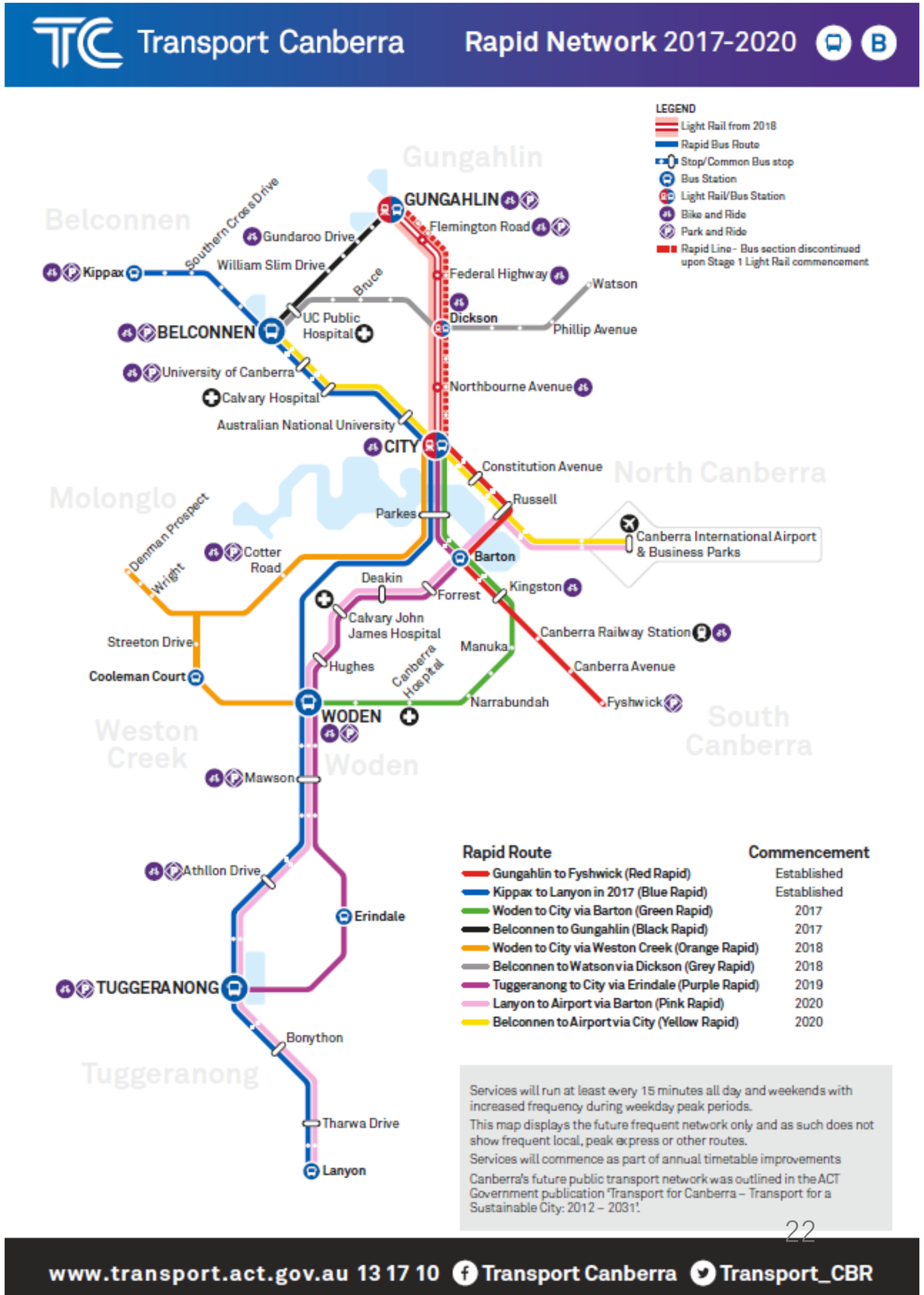


Fig. Minimum passenger-kilometer spanning tree



Relationship between cumulative frequency distribution and ratio of link distance to path travel distance

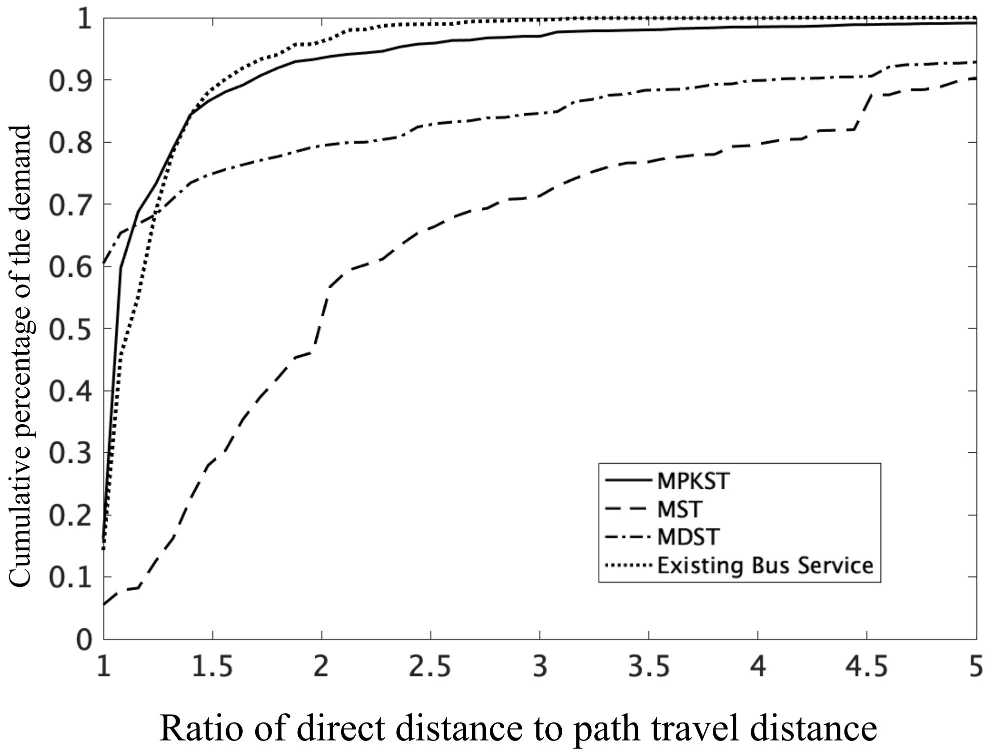


Fig. (a) The volume of demand

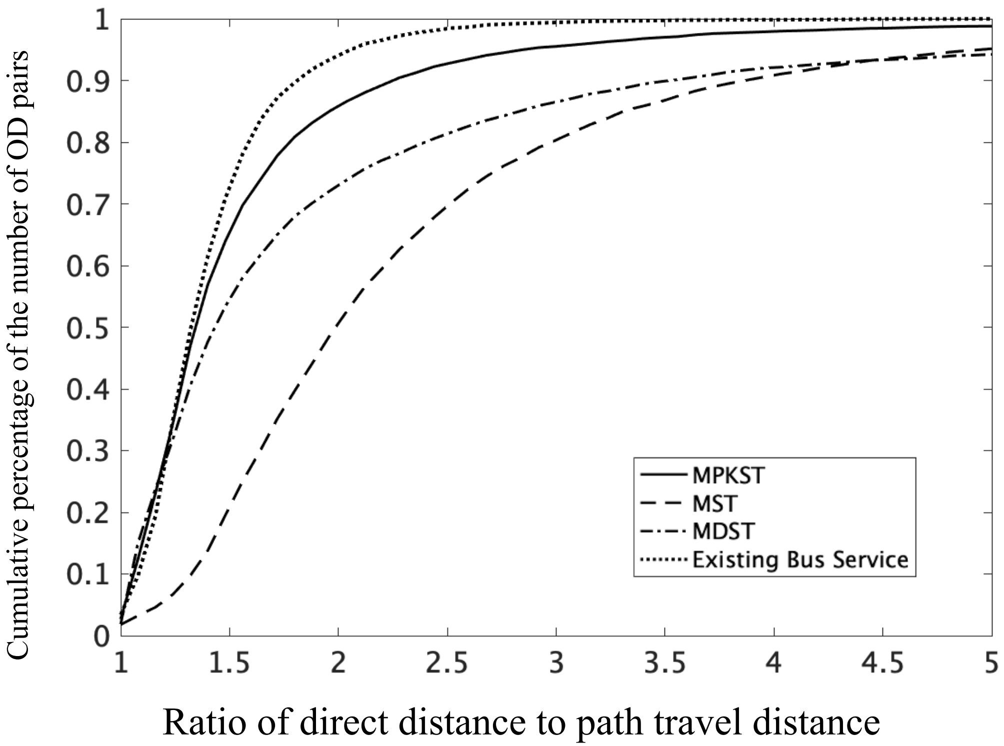


Fig. (b) The number of OD pairs

Further improvements on transit network design

$$T^*_{+\alpha} = T^*_{+\alpha-1} \cup \underset{y_{(ij)}}{\arg\min}\, Z\left(T^*_{+\alpha-1} \cup y_{(ij)}\right)$$

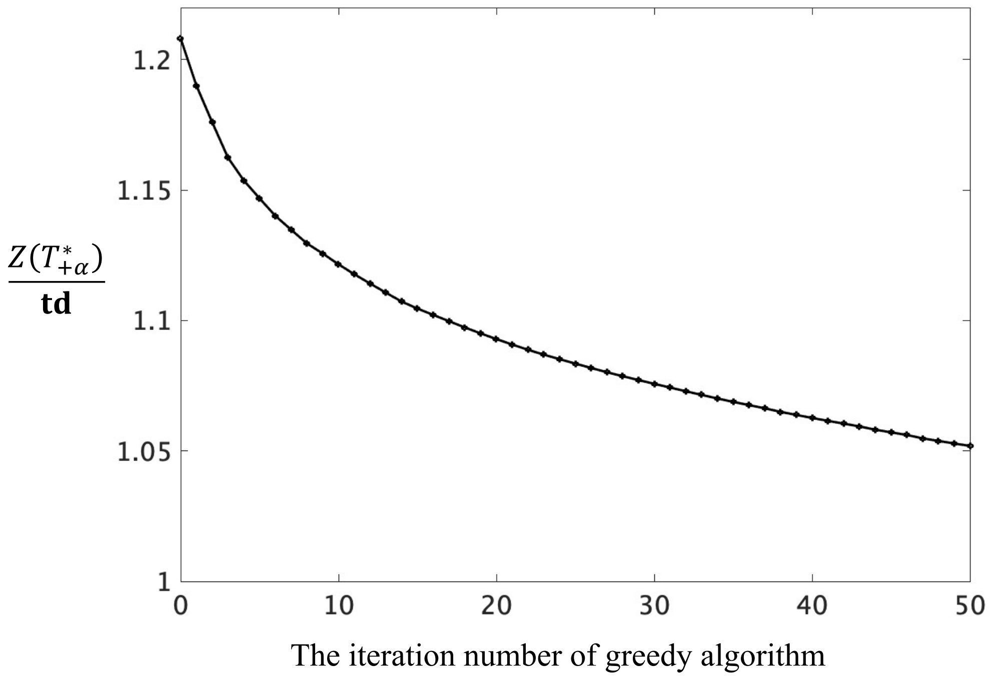


Fig. Transition of the ratio of the lower bound to the passenger-kilometers obtained using the greedy algorithm

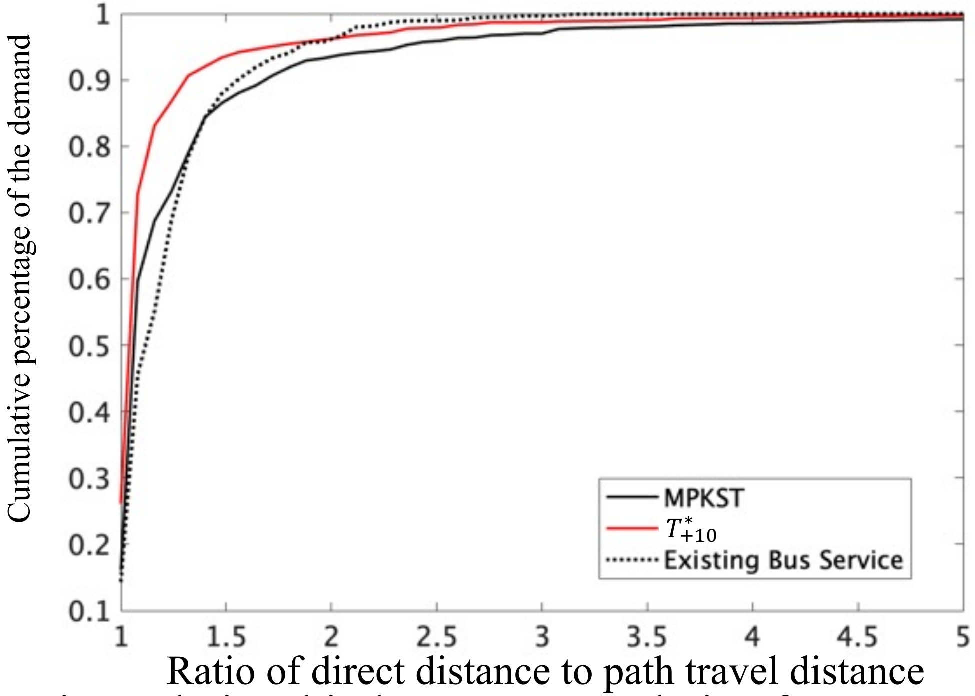


Fig. Relationship between cumulative frequency distribution of the demand and ratio of link distance to path travel distance.

Conclusion

- We proposed an optimization model to study a TNDP using the concept of spanning tree, which minimizes the TPK in the network
- Also, we developed a solution algorithm: Link Swapping with Tabu Search to quickly solve the model
  - Our method makes a decline in the number of shortest path searches from the algorithm
- We apply our method to Canberra smart card data
  - Our method illustrates decreasing computational time dramatically and finding a better approximate solution.
- We proposed further improvements on TNDP from ST

Thank you for listening

Ref.

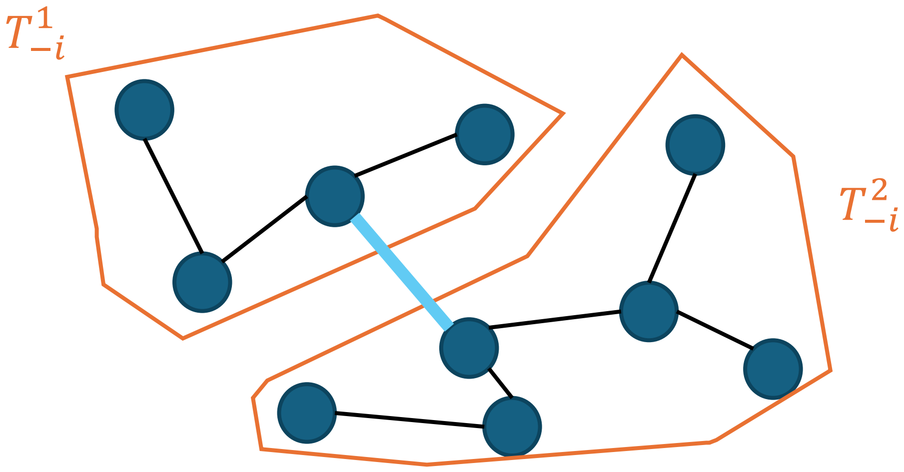
Bell, M. G. H., Pan, J. J., Teye, C., Cheung, K. F., & Perera, S. (2020). An entropy maximizing approach to the ferry network design problem. *Transportation Research Part B: Methodological, 132*, 15-28. https://doi.org/10.1016/j.trb.2019.02.006

Cayley, A. (1889). A theorem on trees. *Quarterly Journal of Mathematics, 23*, 376-378.

Öncan, T., Cordeau, J. F., & Laporte, G. (2008). A tabu search heuristic for the generalized minimum spanning tree problem. *European Journal of Operational Research, 191*(2), 306-319.

Rothlauf, F. (2009). On optimal solutions for the optimal communication spanning tree problem. *Operations Research, 57*(2), 413-425. https://doi.org/10.1287/opre.1080.0592

Tsubakitani, S., & Evans, J. R. (1998). Optimizing tabu list size for the traveling salesman problem. *Computers & Operations Research, 25*(2), 91-97. https://doi.org/10.1016/S0305-0548(97)00030-0



$$k_i = n(T^1_{-i}) \times n(T^2_{-i})$$