

Série dos Seminários de Acompanhamento à Pesquisa

DEI
DEPARTAMENTO
DE ENGENHARIA
INDUSTRIAL

Número 09 | 05 2021

Adjacent Quadratic Shortest Path Problem

An Approach for Electric Transmission Line Route Optimization

Autor(es):

João Marcos Dusi Vilela



Série dos Seminários de Acompanhamento à Pesquisa

Número 09 | 05 2021

Adjacent Quadratic Shortest Path Problem

An Approach for Electric Transmission Line Route Optimization

Autor(es):

João Marcos Dusi Vilela

Orientador: Bruno Fanzeres

Coorientador: Rafael Martinelli

CRÉDITOS:

SISTEMA MAXWELL / LAMBDA

<https://www.maxwell.vrac.puc-rio.br/>

Organizadores: Fernanda Baião / Soraida Aguilar

Layout da Capa: Aline Magalhães dos Santos

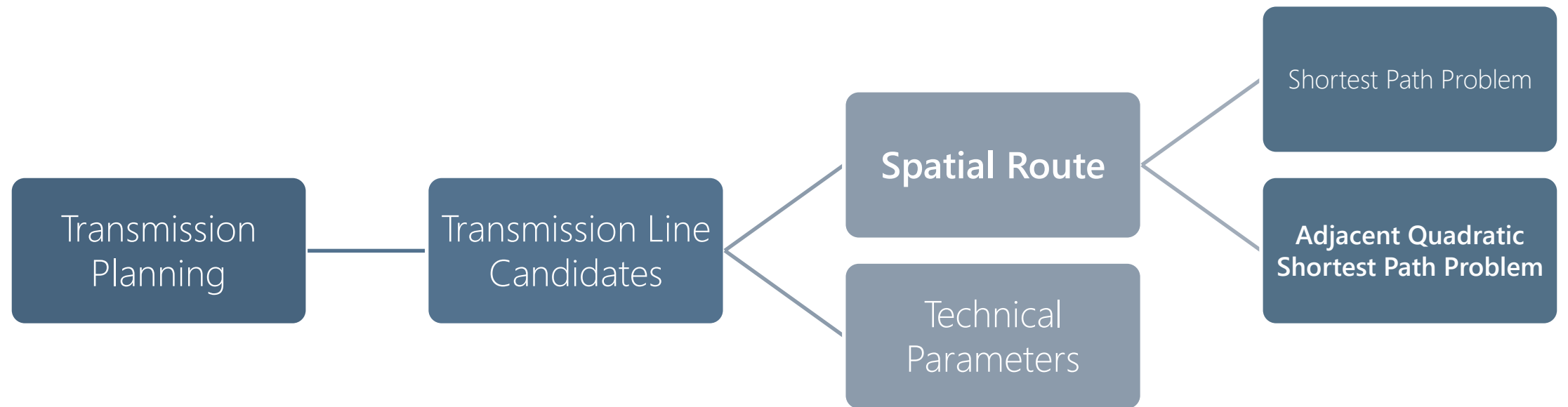


1

Proposal Overview

Context & Theoretical Background

Proposal Overview | Context

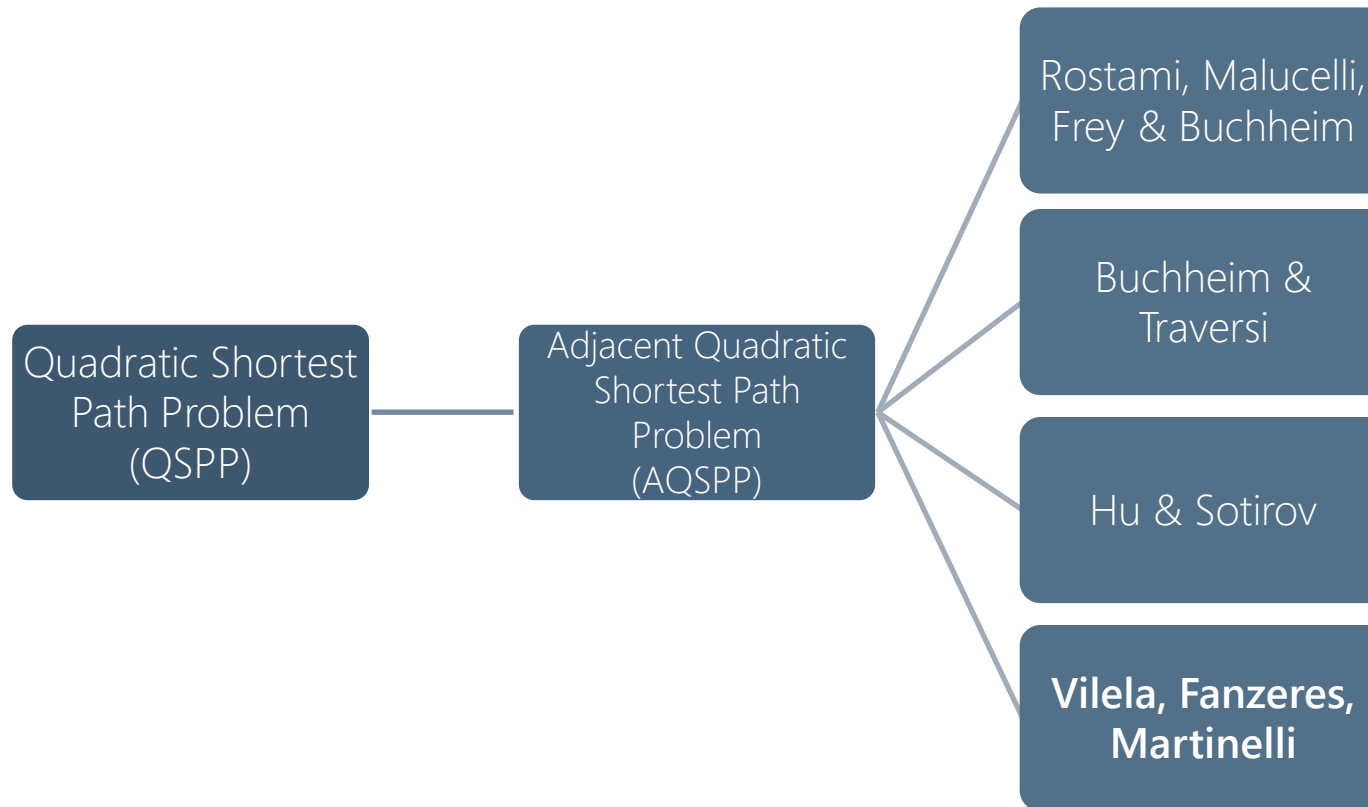


Proposal Overview | Context

- The transmission line routing can be considered as an optimization problem that aims to find the **spatial path** that minimize the **construction costs**, subjected to **technical**, **social**, **economical** and **environmental** constraints.
- Most methodologies presented in the literature can model relatively well social-economic-environmental variables and use SPP algorithms to solve the routing problem. However, technical aspects have been neglected by the community and are still in need of deeper research,
- In this work we continue to fill this gap by developing a novel path search algorithm that avoid sharp angles between consecutive towers. These angle points are associated with higher mechanical efforts, and consequently with more expensive structures. The most common towers and their respective angle range and relative costs are present on the table below.

| Tower | Angle Range | Relative Tower Cost |
|------------|-----------------------------|---------------------|
| Suspension | $0^{\circ} \sim 3^{\circ}$ | 1.00 |
| Angle | $3^{\circ} \sim 30^{\circ}$ | 1.75 |
| Dead-end | $<30^{\circ}$ | 3.00 |

Proposal Overview | Theoretical Background





2

Theory

Quadratic Formulation, Proposed Algorithm & Graph Instance

Theory | Quadratic Formulation

- The AQSP can be formulated as a mixed integer-linear problem (MILP) problem by adapting the traditional minimum cost flow problem.
- For this problem, we define:
 - x_{ij} as the variable for the binary decision of using arc (i, j) on the path
 - c_{ij} as the cost associated with using arc (i, j) .
 - x_{jk} as the variable representing the decision of adjacent arc (j, k)
 - q_{ijk} as the adjacent quadratic cost for adjacent vertices (i, j, k) .
 - \tilde{A} and A as the sets of nodes from adjacent arcs and neighbor nodes, respectively

$$\text{Min}_{x_{ij}, x_{jk}} \sum_{(i,j,k) \in \tilde{A}} q_{ijk} x_{ij} x_{jk} + \sum_{i,j \in A} c_{ij} x_{ij}$$

s.t.

$$\sum_{j \in \delta_i^+} x_{ij} - \sum_{j \in \delta_i^-} x_{ji} = b_i \quad \forall i \in V$$

$$b_i = \begin{cases} 1, & \text{for } i = s \\ -1, & \text{for } i = d \\ 0, & \text{otherwise} \end{cases}$$

Theory | Adjacent Quadratic Dijkstra

- We propose an extension of classical Dijkstra algorithm, which we call Adjacent Quadratic Dijkstra (or aqDijkstra).
- The aqDijkstra algorithm follows a similar structure then classic Dijkstra's algorithm, but modifies how labels and accumulated costs are defined.
- On a quadratic perspective, a label should be defined as the edge $l = (i, j)$ that arrives at vertex j .



- This would allow us to evaluate the shortest path to j that passes through (i, j) .
- Therefore, when evaluating a connection between j and k , we can map adjacent quadratic cost $q_{L(j)k} = q_{ijk}$.

Algorithm 1: Adjacent Quadratic Dijkstra (pseudo-code)

```
Function aqDijkstra( $G, s, t$ ):  
    initialize  $D_{ij}$   
    initialize  $Q_{il}$   
     $D_{sl} \leftarrow 0$   
    for each vertex  $v$  in  $G$  do  
        for each label  $l$  in  $L(v)$  do  
            if  $v \neq s$  then  
                 $D_{vl} \leftarrow Inf$   
                 $P_{vl} \leftarrow$  origin vertex from label  $l$   
            end  
        end  
    while  $Q_{il} \neq \emptyset$  do  
         $(j, l_1) \leftarrow \text{argmin}_{il} \{Q_i\}$   
        remove  $(j, l_1)$  from  $Q_{il}$   
        for each  $k$  in neighbors of  $j$  do  
             $l_2 \leftarrow$  label index of edge  $(j, k)$   
             $C_{jl_2} \leftarrow D_{jl_1} + c_{jk} + q_{ijk}$   
            if  $C_{jl_2} < D_{jl_2}$  then  
                 $D_{jl_2} \leftarrow C_{jl_2}$   
                 $P_{jl_2} \leftarrow j$   
                 $Q_{jl_2} \leftarrow C_{jl_2}$   
            end  
        end  
    end  
    return  $D_{il}, P_{il}$   
end
```

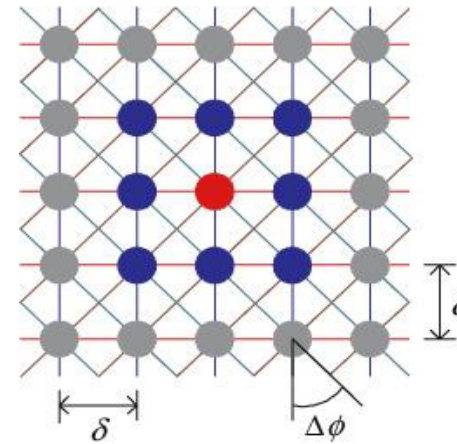
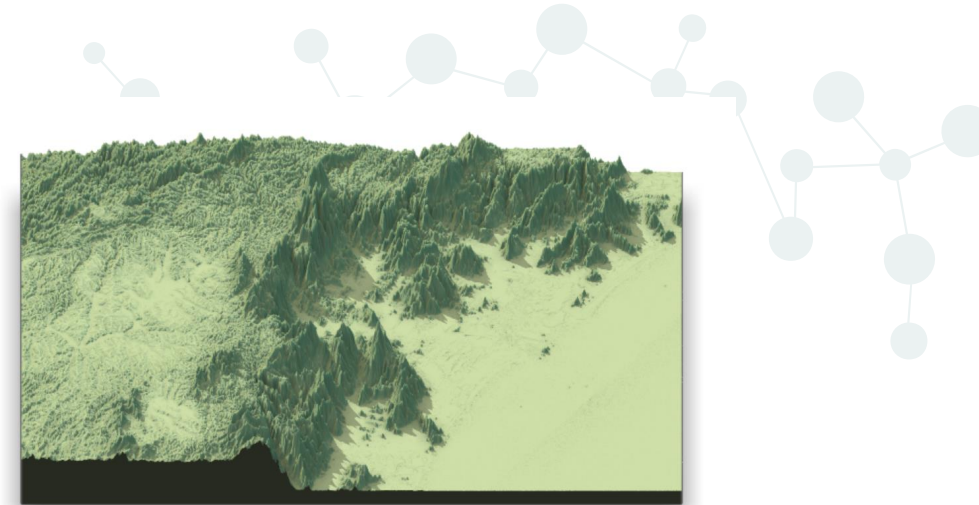
Theory | Graph Instance

- To create a graph instance, we convert a matrix Y (map) into a weighted graph $G = (V, A)$

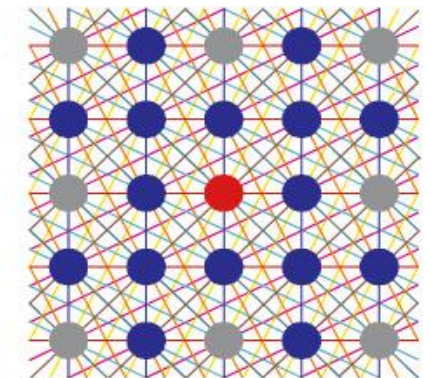
$$Y = \begin{bmatrix} y_{11} & \cdots & y_{1M} \\ \vdots & \ddots & \vdots \\ y_{N1} & \cdots & y_{NM} \end{bmatrix}$$

- Each vertex i is equivalent to element y_{nm} and arc (i, j) represents a connection between neighbor element.
- The number of arcs $|A|$ can vary according to the number of neighbors n considered. The figure aside presents two illustrative examples, considering $n = 8$ and $n = 16$.
- Furthermore, weights are defined as:

$$c_{ij} = \begin{cases} \frac{y_{nm} + y_{op}}{2}, & \text{if ortogonal movemnt} \\ \frac{y_{nm} + y_{op}}{2} \sqrt{2}, & \text{if diagonal movemnet} \end{cases}$$



8-neighborhood



16-neighborhood



3

Improvements

Bidirectional Search
& Embedded Quadratic Cost Calculation

Improvements | Bidirectional Search

- To improve performance, we can add a Bidirectional Search to the proposed aqDijkstra.
- The scheme works similarly to A*, where a distance from evaluated node to target node can be estimated and used to speed up search.
- First, we use the classical Dijkstra algorithm to calculate the distance between target node and all the other nodes. Then, we add this estimation to the distance cost being evaluated on each iteration.
- It is important to stand out that, differently from A*, this bidirectional search does guarantee global optimality (B. Thomas 2019).

Algorithm 2: Adjacent Quadratic Dijkstra with Bidirectional Heuristic Search (pseudo-code)

Function *aqDijkstra*(G, s, t):

$B_i \leftarrow \text{Dijkstra}(G, t)$

initialize D_{ij}

initialize Q_{il}

$D_{sl} \leftarrow 0$

for each vertex v in G **do**

for each label l in $L(v)$ **do**

if $v \neq s$ **then**

$D_{vl} \leftarrow \text{Inf}$

$P_{vl} \leftarrow \text{origin vertex from label } l$

end

end

while $Q_{il} \neq \emptyset$ **do**

$(j, l_1) \leftarrow \text{argmin}_{il} \{Q_i\}$

 remove (j, l_1) from Q_{il}

for each k in neighbors of j **do**

$l_2 \leftarrow \text{label index of edge } (j, k)$

$C_{jl_2} \leftarrow D_{jl_1} + c_{jk} + q_{ijk}$

if $C_{jl_2} < D_{jl_2}$ **then**

$D_{jl_2} \leftarrow C_{jl_2}$

$P_{jl_2} \leftarrow j$

$Q_{jl_2} \leftarrow C_{jl_2} + B_k$

end

end

return D_{il}, P_{il}

end

Improvements | Embedded Quadratic Cost Calculation

- To handle memory issues on pre-calculating and pre-allocating adjacent quadratic costs on very large instances, the algorithm was modified to calculate the quadratic cost during the execution.
- Although some negative impact on performance is expected, embedding the quadratic cost calculation allows solving even larger instances

Algorithm 3: Adjacent Quadratic Dijkstra with Embedded Adjacent Cost Calculation (pseudo-code)

```
Function aqDijkstra( $G, s, t$ ):  
    initialize  $D_{ij}$   
    initialize  $Q_{il}$   
     $D_{sl} \leftarrow 0$   
    for each vertex  $v$  in  $G$  do  
        for each label  $l$  in  $L(v)$  do  
            if  $v \neq s$  then  
                 $D_{vl} \leftarrow \text{Inf}$   
                 $P_{vl} \leftarrow \text{origin vertex from label } l$   
            end  
        end  
    while  $Q_{il} \neq \emptyset$  do  
         $(j, l_1) \leftarrow \text{argmin}_{il} \{Q_i\}$   
        remove  $(j, l_1)$  from  $Q_{il}$   
        for each  $k$  in neighbors of  $j$  do  
             $l_2 \leftarrow \text{label index of edge } (j, k)$   
             $q_{ijk} \leftarrow \text{AdjacentQuadraticCost}(G, i, j, k)$   
             $C_{jl_2} \leftarrow D_{jl_1} + c_{jk} + q_{ijk}$   
            if  $C_{jl_2} < D_{jl_2}$  then  
                 $D_{jl_2} \leftarrow C_{jl_2}$   
                 $P_{jl_2} \leftarrow j$   
                 $Q_{jl_2} \leftarrow C_{jl_2}$   
            end  
        end  
    end  
    return  $D_{il}, P_{il}$   
end
```



4

Preliminary Results

Computational Experiments & Application

Results | Computational Experiments



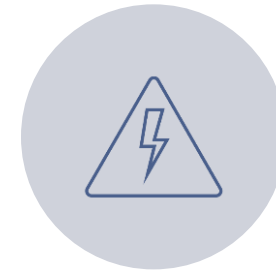
PERFORMANCE



COST
VARIATION



RANDOM
COSTS



STRESS

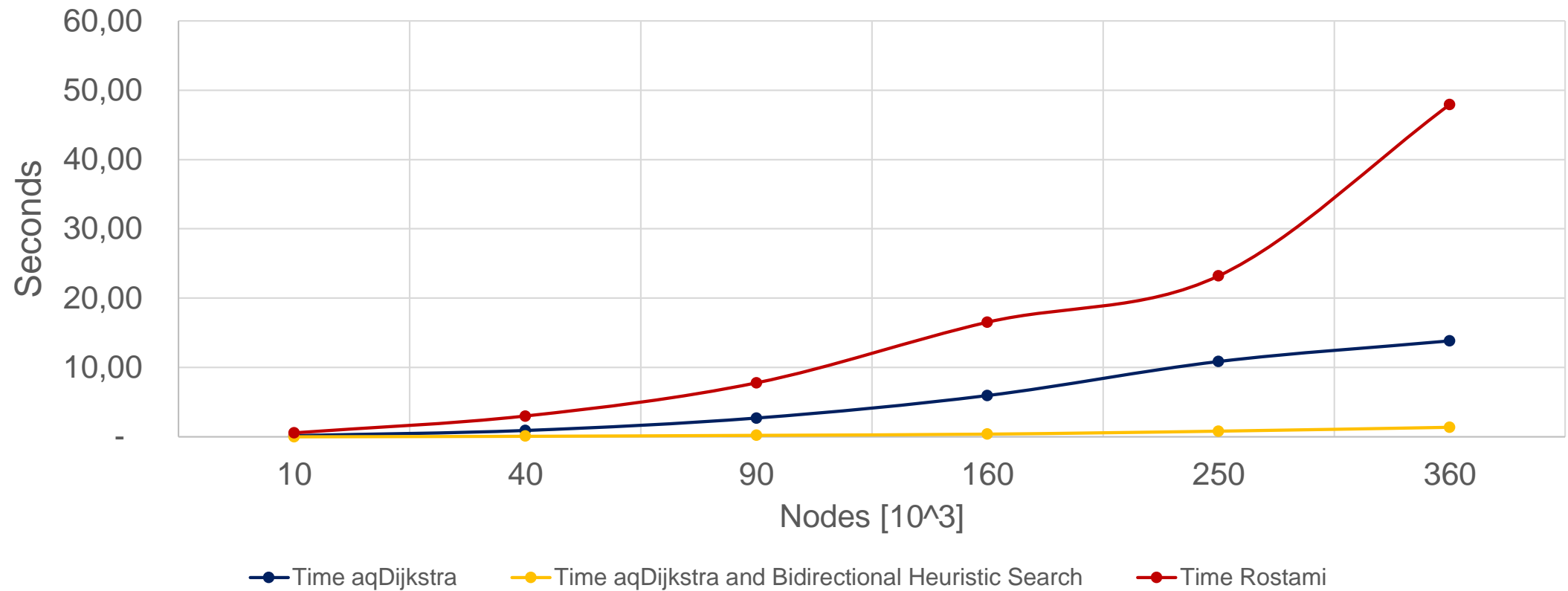
Results | Computational Experiments: Performance

- To evaluate performance, we considered a 6 different graph instances ranging from 10,000 to 360,000 nodes.
- Solution time of classical Dijkstra, proposed aqDijkstra and both improvement suggested were recorded for comparison.
- In addition, a similar algorithm proposed by Rostami et. al, was also included on the computational experiments for benchmark purposes.
- An attempt with a MILP formulation was also made, but it couldn't solve even the first instance due memory issues.

| Nodes (10 ³) | Dijkstra (s) | aqDijkstra (s) | Embedded Quadratic Calculation (s) | Bidirectional Search (s) | Rostami's Algorithm (s) |
|-----------------------------|-----------------|-------------------|---|--------------------------------|-------------------------------|
| 10 | 0.00 | 0.18 | 0.22 | 0.02 | 0.55 |
| 40 | 0.02 | 0.92 | 1.15 | 0.09 | 2.98 |
| 90 | 0.05 | 2.71 | 3.23 | 0.22 | 7.77 |
| 160 | 0.11 | 5.95 | 6.54 | 0.39 | 16.53 |
| 250 | 0.22 | 10.85 | 12.61 | 0.81 | 23.21 |
| 360 | 0.24 | 13.82 | 15.65 | 1.35 | 47.95 |

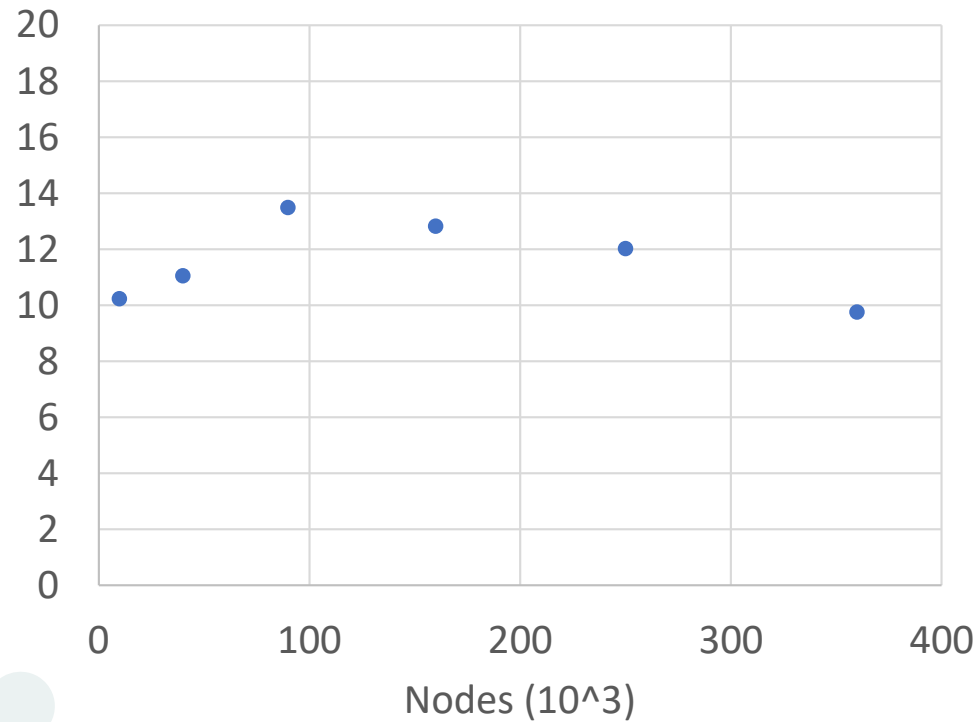
Results | Computational Experiments: Performance

Performance Comparison

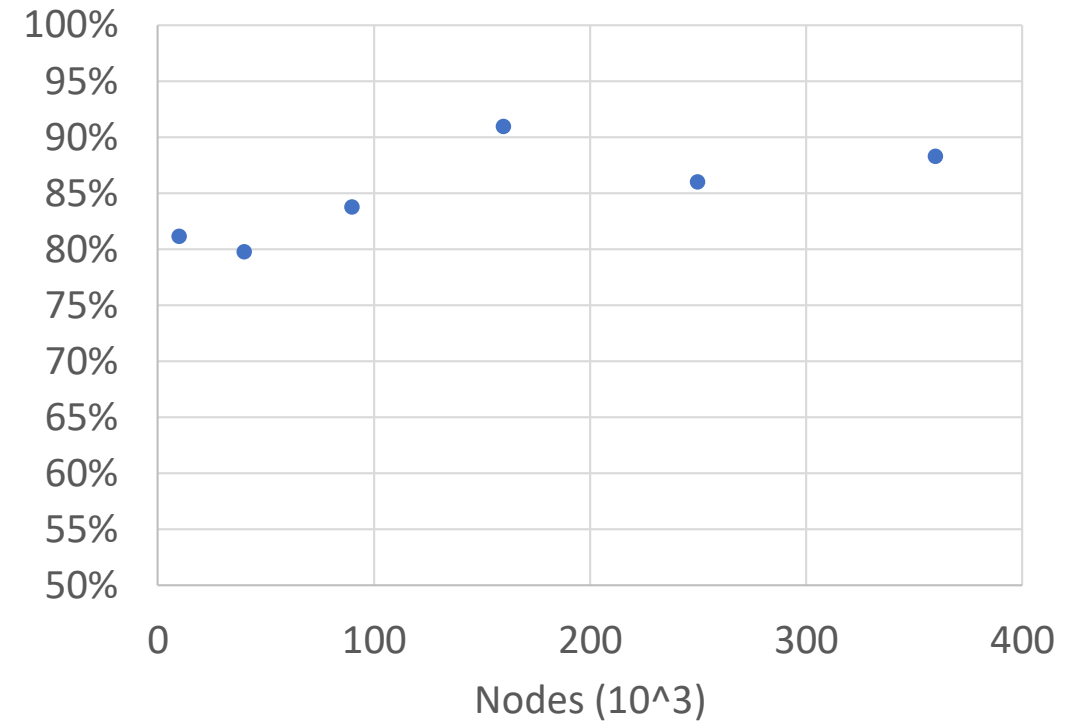


Results | Computational Experiments: Performance

Ratio Between Bidirectional Search and aqDijkstra



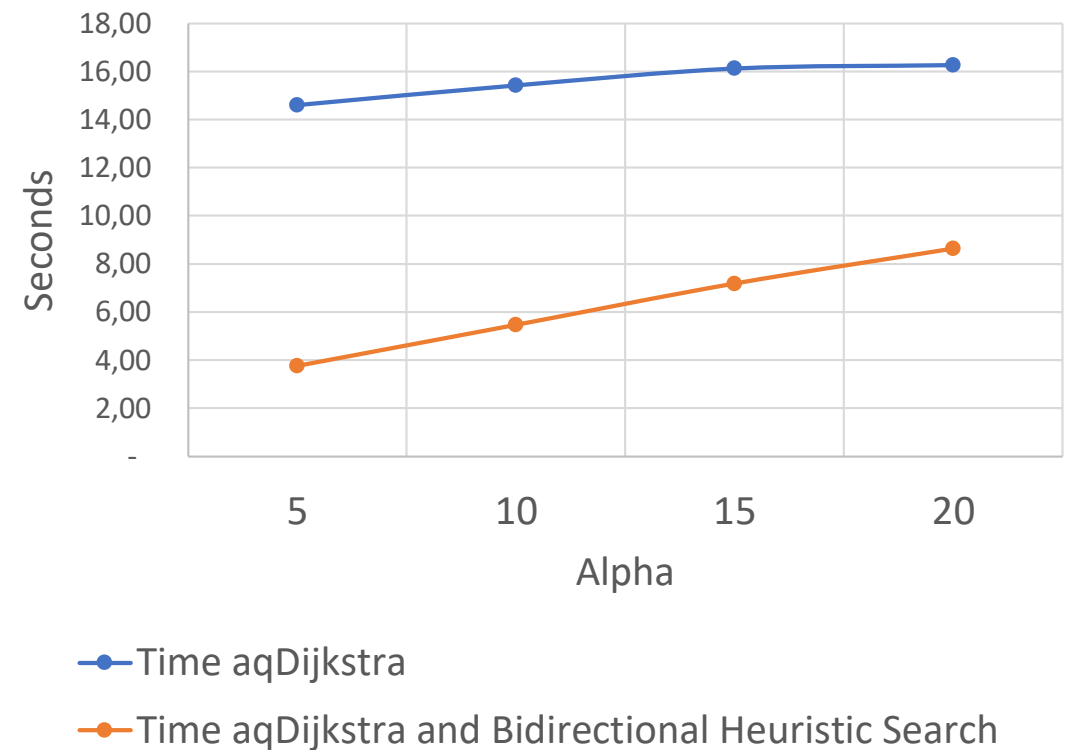
Ratio Between aqDijkstra and Embedded Quadratic Calculation



Results | Computational Experiments: Cost Variation

- We also wanted to verify the sensibility of the proposed algorithms the magnitude of adjacent quadratic costs.
- A variable $\alpha \in \{5,10,15,20\}$ was set to scale quadratic costs. For each alpha, 6 graph instances were evaluated, ranging from 10,000 to 360,000 nodes.
- Only aqDijkstra with Bidirectional Search presented significant variation in performance.
- The probable explanation comes from the fact that the estimators do not contemplate quadratic costs. Therefore, as quadratic costs become greater, the estimation becomes insignificant.

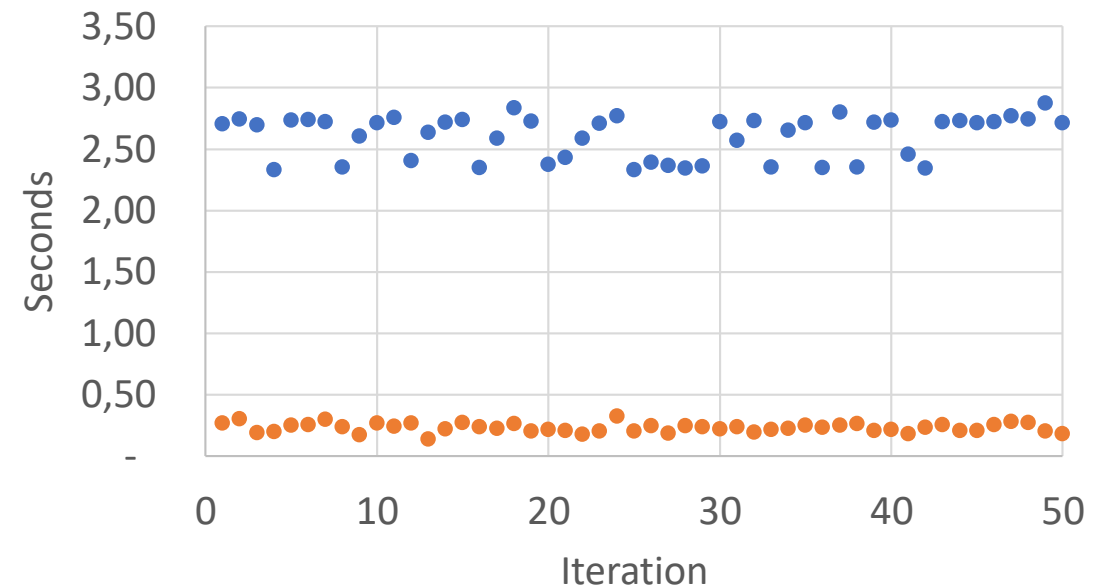
Solution Time for Cost Variation
360,000 nodes



Results | Computational Experiments: Random Costs

- The third experiment concerned the algorithm's performance considering randomly generated instances.
- A random variable $u \sim \text{Unif}(0,1)$ was chosen to represent graph weights.
- 6 graph reference instances ranging from 10,000 to 360,000 nodes were selected and, for each instance, 50 random graphs were constructed.
- No significant change in performance was identified due to the randomness of costs.

Solution Time for Random Costs
90,000 nodes




• Time aqDijkstra

• Time aqDijkstra and Bidirectional Heuristic Search

Results | Computational Experiments: Stress



- The stress test was performed with instances ranging **1,000,000** to **9,000,000** nodes with the following approaches:
 - aqDijkstra
 - aqDijkstra with the embedded quadratic calculation
 - aqDijkstra with the embedded quadratic calculation and bidirectional search
 - Attempt though original **aqDijkstra** presented **memory problems** for instances with more than 4,840,000 nodes.
 - The other two approaches were **able to solve** the largest instance, since quadratic cost calculations were **embedded**.
- 

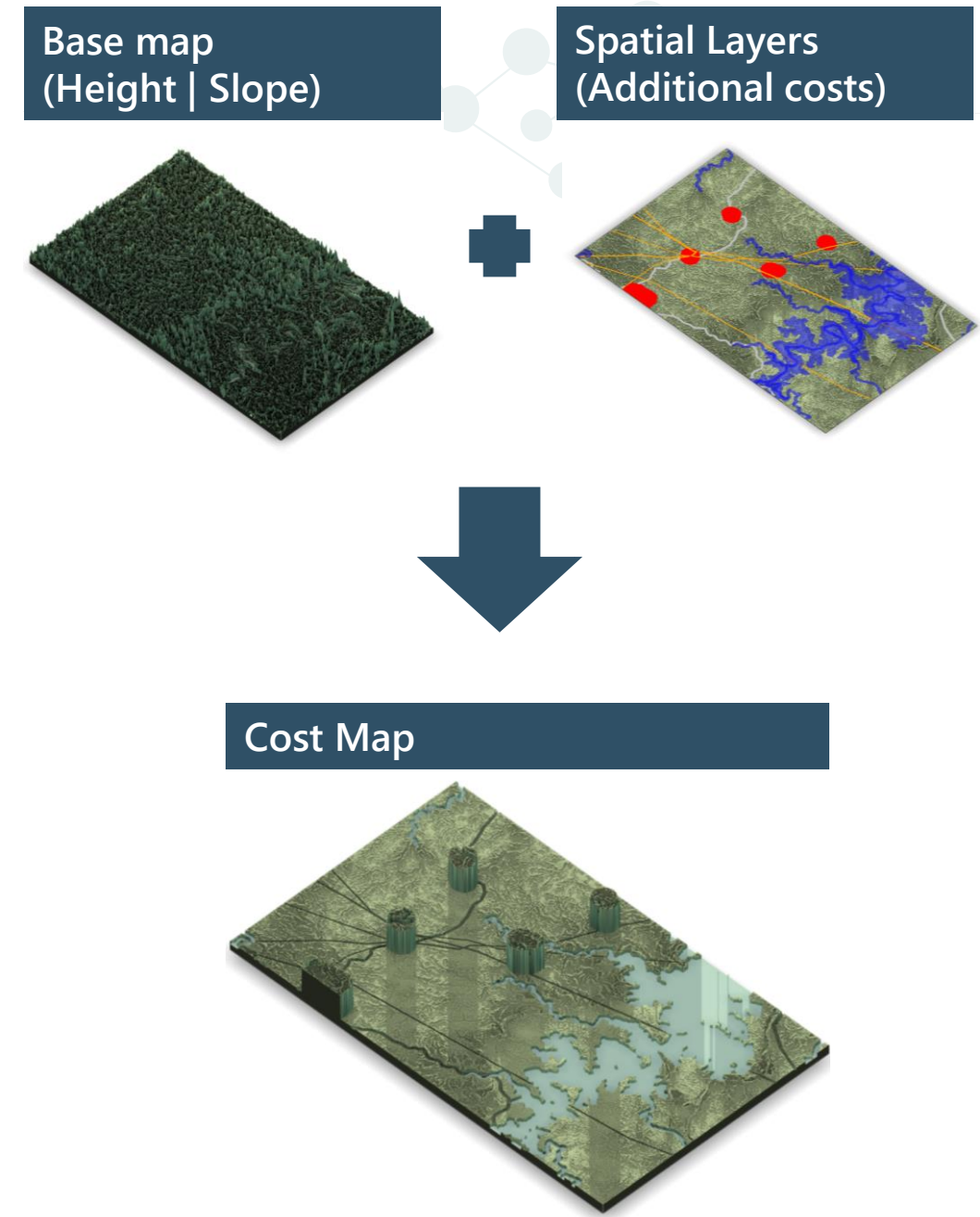
Results | Application

- We study the connection between two substations on the southern region of Brazil as a transmission line candidate for expansion.
- For a 230kV transmission line, we assume a reference 1.2 MR\$/km for construction costs.
- To generate the graph instance, two types of input data are required: a base map and spatial layers.
- The base map has spatial resolution of 30x30m, describes terrain slope and part of NASA's SRTM public database.
- The spatial layers were obtained from public Brazilian agencies, listed on the table aside. Each layer represent a spatial constraint and is associated with additional construction costs.

| Layer | Source | Cost [MR\$/km] |
|---------------------------------|--------|----------------|
| Urban Areas | IBGE | 1.5 |
| Water Bodies | IBGE | 4.5 |
| Indigenous Lands | INCRA | - |
| Environmentally Protected Areas | INCRA | - |
| Traditional Communities | INCRA | - |
| Rural Settlements | INCRA | 2.0 |
| Railways | DNIT | -0.3 |
| Highways | DNIT | -0.5 |
| Existing Transmission Lines | EPE | -0.2 |

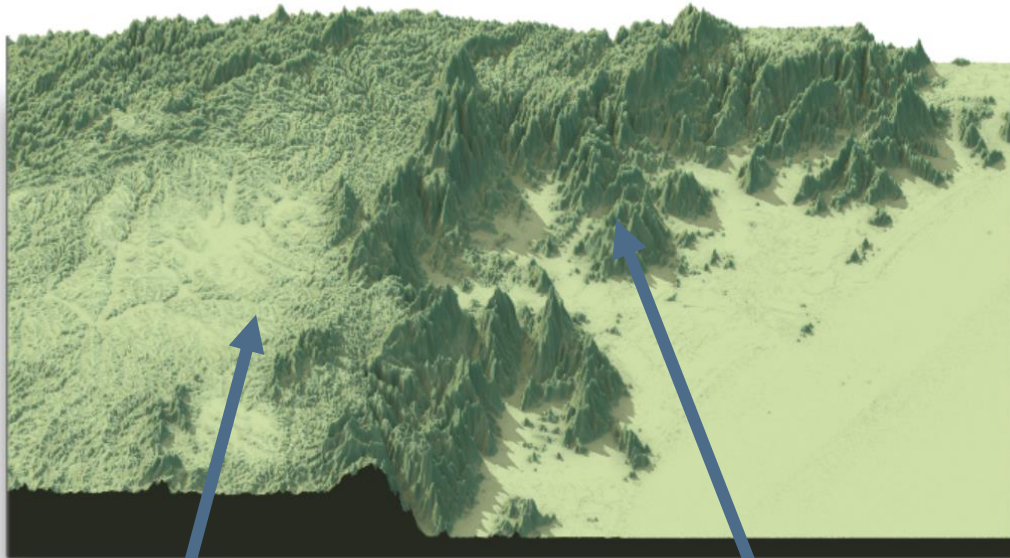
Results | Application

- Before building the graph instance, we must convert the base map and spatial layers into a construction cost map.
- First, we transform the base map into a linear distance map, by using terrain slope and map resolution to approximate a linear distance (km).
- The product between this linear distance and construction reference cost will provide an estimation of building the line on each region of the map.
- Furthermore, construction costs can vary according to areas defined by spatial layers.



Results | Application

Base map



Origin
Substation

Destination
Substation



Final Cost Map



Urban Area
(Curitiba City)

Conservation Area
(Legally protected)

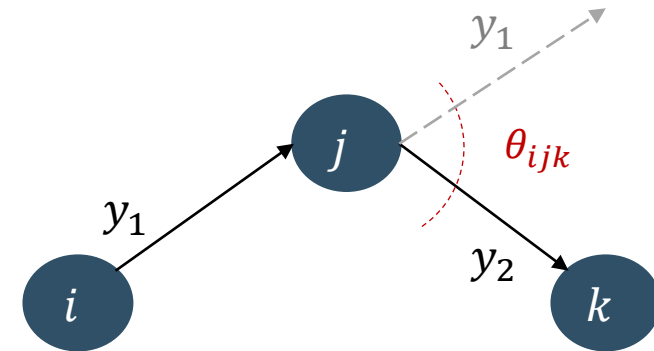
Results | Application

- The use of AQSPP brings innovation to LT routing, as it allows us to model penalties for sharp angles.
- Depending on how the towers are positioned spatially, sharp inflection angles can occur, which requires more expansive towers.
- We suggest an adjacent cost $q(c_{ij}, c_{jk}, \theta_{ijk})$ where θ is the inflection angle between two adjacent arcs (i, j) and (j, k) .

$$q_{ijk} = \left(\frac{c_{ij} + c_{jk}}{2} \right) \left(\frac{\theta_{ijk}}{\pi} \right)$$

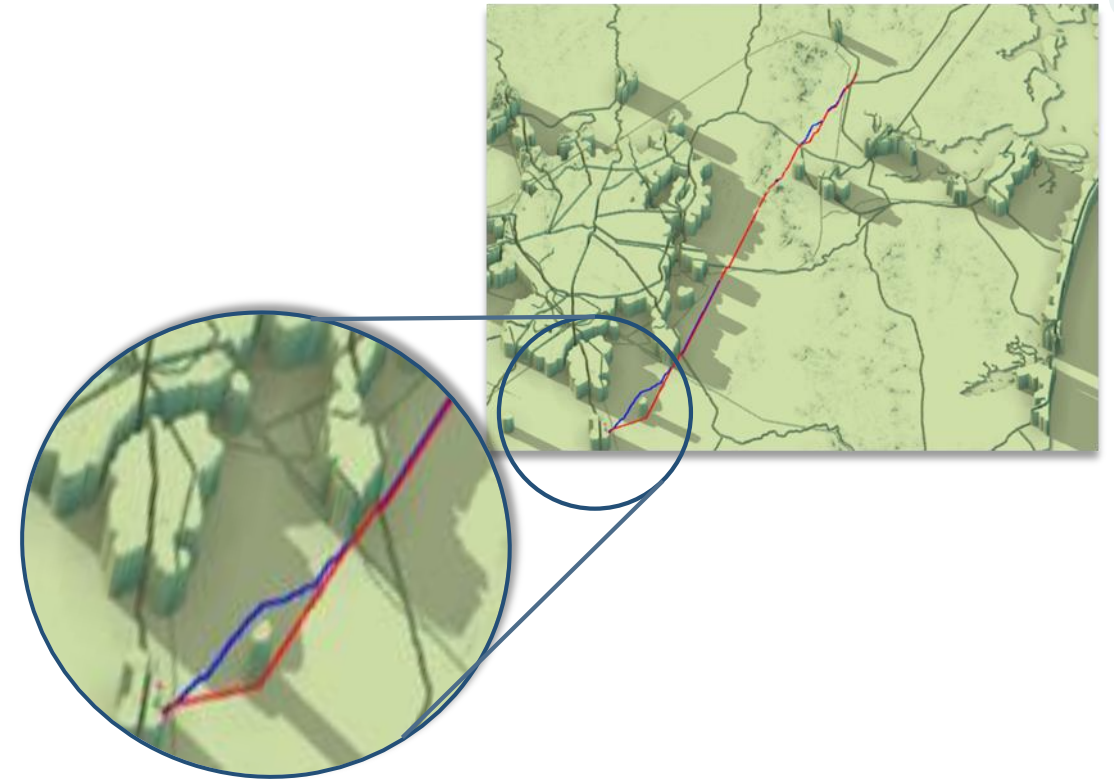
- Where the angle can be found through:

$$\theta_{ijk} = \arccos \left(\frac{y_1 y_2}{\|y_1\| \|y_2\|} \right)$$



Results | Application

- The figure aside presents the routes resulted from Dijkstra (blue) and aqDijkstra (red) approaches.
- As expected, both avoided expensive areas (high areas on the map), specially towards the end, where the route contoured the urban area.
- Although routes can be considered very similar, we should highlight that aqDijkstra avoided sharp angles, due to the addition of quadratic costs.
- The aqDijkstra route behavior is presented on the figure zoom aside.





5

Conclusions

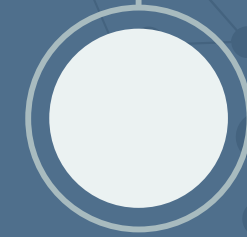
Conclusions reached so far

Conclusions*

- This work proposes an Adjacent Quadratic version of Dijkstra's algorithm to solve specific AQSP problems, for graphs with no Improving Cycle.
- Further on, two improvements on the algorithm are developed, one to improve performance and another to handle memory issues. Performance gains from using a bidirectional search were in the order of 10 times. The embedded quadratic cost calculation allowed solving the largest instance evaluated.
- We provided empirical proof that the proposed aqDijkstra algorithm and its improved version can be solved at polynomial times. The approaches present a constant solution time ratio compared to classical Dijkstra.
- A benchmark with MILP and Rostami's Algorithm are provided, with aqDijkstra outperforming both approaches.
- For graphs with more than 10,000 vertices, MILP formulation presented memory issues.
- We were able to verify visually the impact of quadratic costs on routing aqDijkstra's routes, confirming its tendency to avoid sharp angles.

Ongoing research

- Proof that graph instance results in graphs with no improving cycle
- Further analysis on chosen application
- Impact of different quadratic cost functions on route decision
- Study of other applications



Thank you

References



1. C. Monteiro, I.J. Ramirez-Rosado, V. Miranda, P.J. Zorzano-Santamaria, E. Garcia- Garrido, L.A. Fernandez-Jimenez, GIS spatial analysis applied to electric line routing optimization, IEEE Trans. Power Deliv. 20 (2005) 934–942
2. E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1(1959) 269–271.
3. H. Eroglu, M. Aydin, Solving power transmission line routing problem using improved genetic and artificial bee colony algorithms, Springer-Verlag GmbH Germany, part of Springer Nature 2018.
4. S. G. Shandiz, G. Doluweera, W. D. Rosehart, L. Behjat, J. A. Bergerson, Investigation of different methods to generate Power Transmission Line routes, Electric Power Systems Research 165 (2018), 110-119.
5. B. Rostami, A. Chassein, M. Hopf, D. Frey, C. Buchheim, F. Malucelli, M. Goerigk. The quadratic shortest path problem: complexity, approximability, and solution methods, Optimization Online, 2016.
6. B. Rostami, F. Malucelli, D. Frey, C. Buchheim. On the quadratic shortest path problem. In: E. Bampis (ed.) Experimental Algorithms, Lecture Notes in Computer Science, Springer International Publishing, 9125, 379–390, 2015.
7. H. Hu, A polynomial time algorithm for the linearization problem of the QSPP and its applications arXiv:1802.02426v1 [math.OC]
8. H. Hu, R. Sotirov. Special cases of the quadratic shortest path problem, Journal of Combinatorial Optimization (accepted), 2017.
9. C. Buchheim, E. Traversi. Quadratic 0-1 optimization using separable underestimators, Technical Report, Optimization Online, 2015
10. Thomas, Barrett W., Calogiuri, Tobia, Hewitt, Mike. An exact bidirectional A * approach for solving resource-constrained shortest path problems. Networks 2019