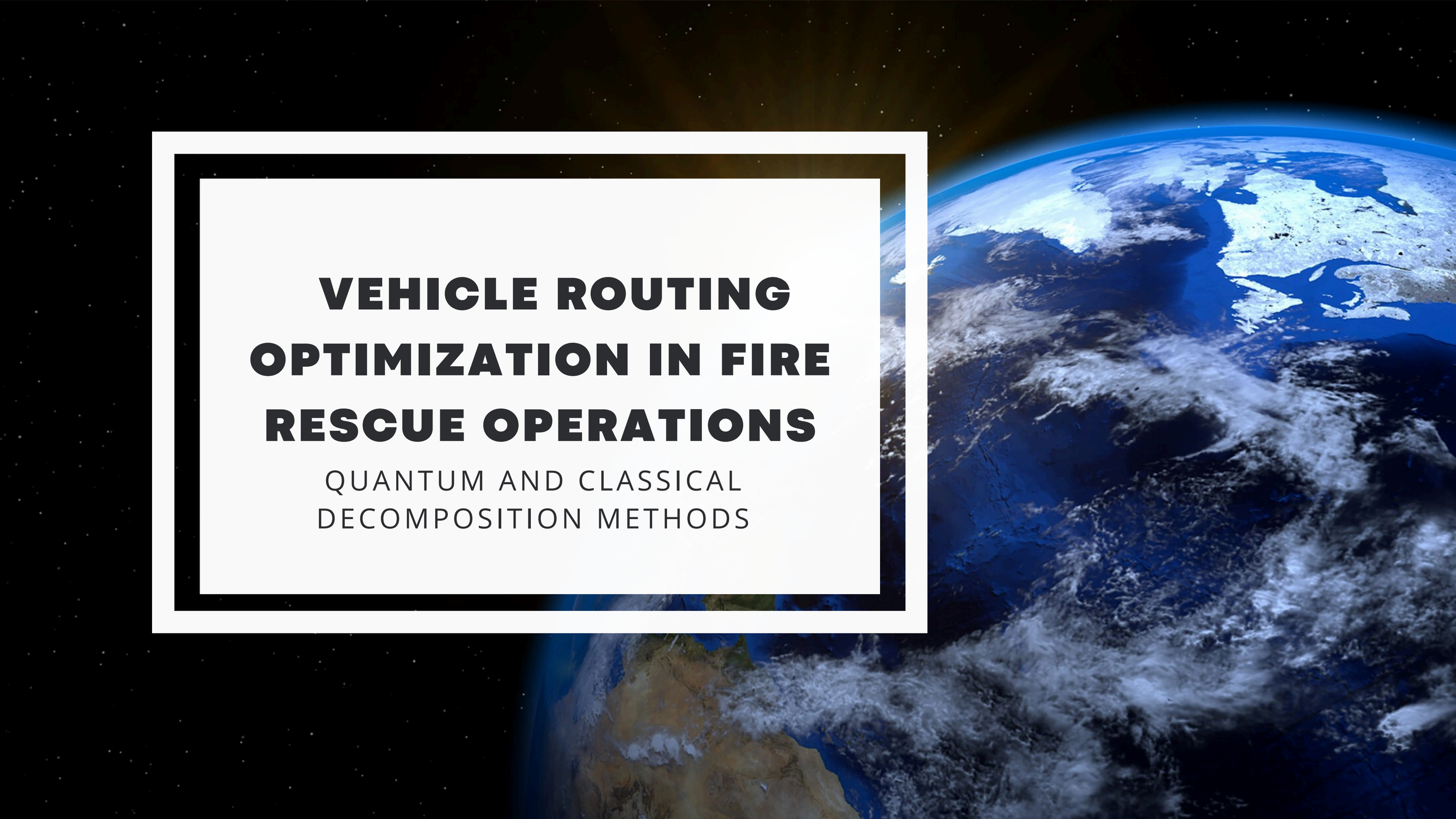


VEHICLE ROUTING OPTIMIZATION IN FIRE RESCUE OPERATIONS

QUANTUM AND CLASSICAL DECOMPOSITION METHODS

Problems

Fire rescue vehicle routing optimization encounters complexities stemming from dynamic emergencies, spatial considerations, resource allocation, and the demand for real-time adjustments.

Unpredictable Nature

Real-Time Decision Making

Resource Allocation

Complex Spatial Constrains

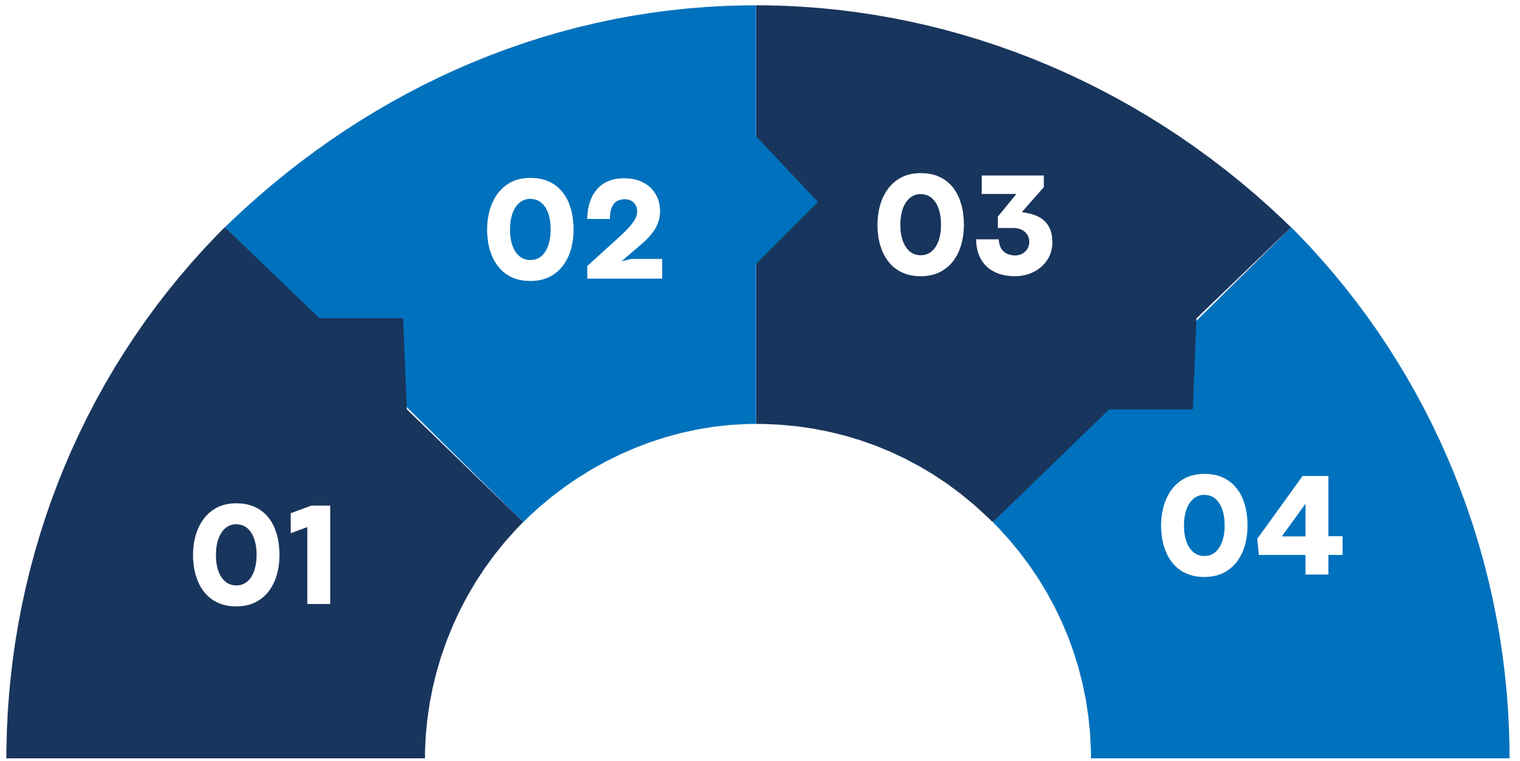
Objectives

01 Illustrate the urgency of rapid response to fire emergencies.

02 Integrate real-time data effectively, adjusting routes based on evolving conditions like shifting priorities or resource availability.

03 Optimizing routes while ensuring equitable distribution of resources across different areas when multiple incidents occur simultaneously.

04 Flying around lakes and coordinating water collection points.



Solutions

01

Solution 01
Transferring a present issue into a traditional computational problem.

02

Solution 02
Using real geographic and logistic data for a more reliable interpretation of the results.

03

Solution 03
Reducing th problem sizes to make quantum computations more feasible.

04

Solution 04
Implementing the CVR problem using Quantum Computing.

Problem formulation

$$water\ needed\ [a.u.] = d_i,\ canadair\ capacity = C$$

$$TSP: \sum d_i \leq C$$

$$CVRP: \sum d_i > C$$

Graph considerations:

1. Node weight = water needed
2. Edge weight = real distance
3. Fully connected (it’s a canadair)

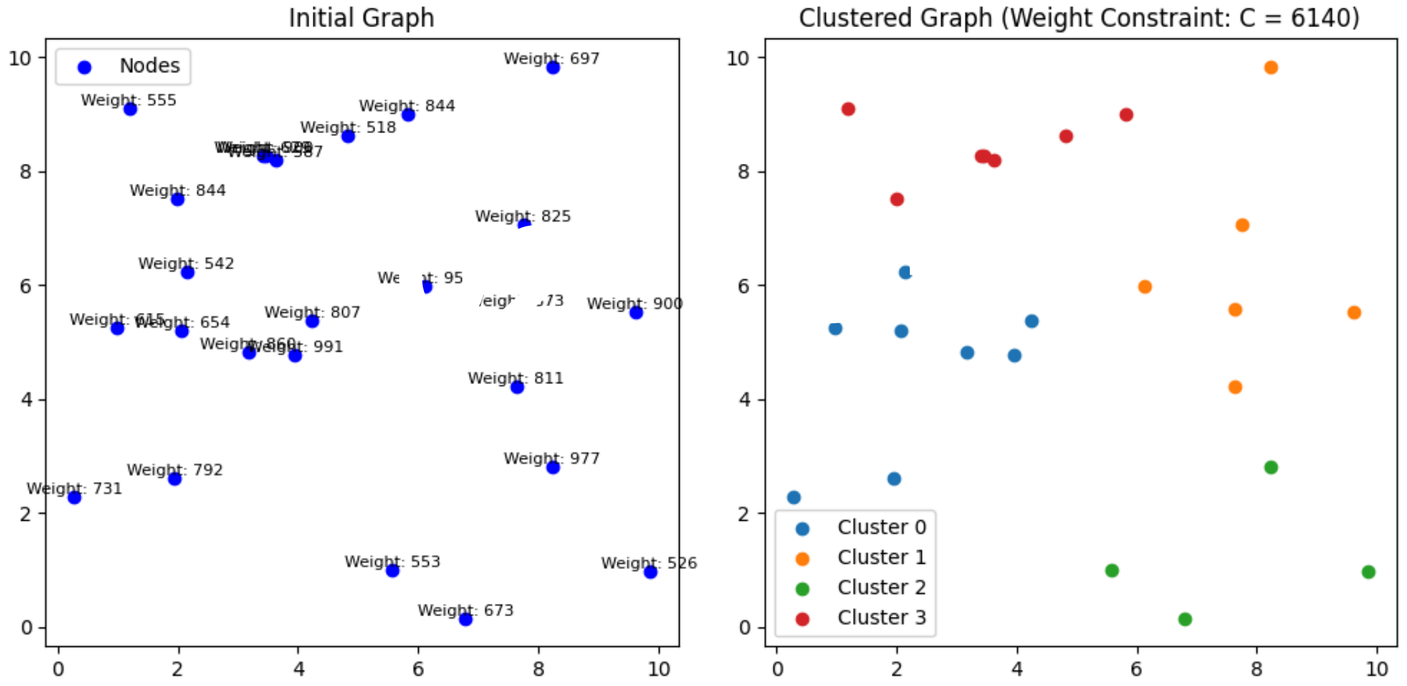
Clustering

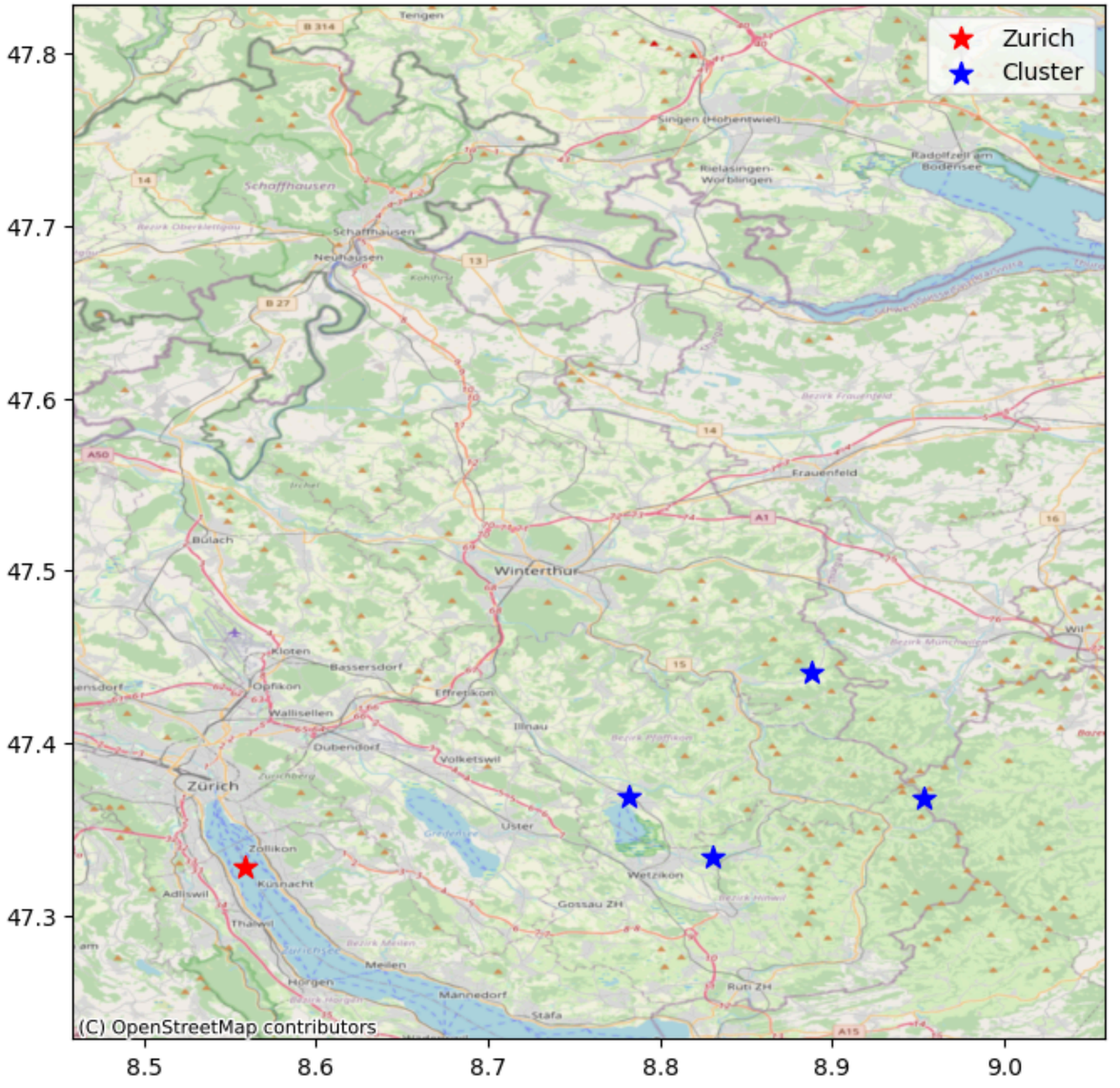
01 Use KMeans algorithm to cluster input nodes based on spatial distribution.

02 Incorporate node weights for balanced clusters while meeting weight constraint C.

03 Dynamic Adjustment: Optimize cluster composition by adjusting cluster numbers and redistributing points to meet weight constraint.

04 Plot Zurich's coordinates and a cluster's points on a map to show their locations in Switzerland.






```python
# Assuming x, y are the coordinates of the smallest cluster
x = [coordinate[0] for coordinate in smallest_cluster]
y = [coordinate[1] for coordinate in smallest_cluster]
latitude, longitude = coordinate_to_latlon(x, y)
latZurigo = 47.328152
lonZurigo = 8.558752
# Create a GeoDataFrame for Zurich
zurich = gpd.GeoDataFrame(geometry=gpd.points_from_xy([lonZurigo], [latZurigo]))
# Create a GeoDataFrame for the smallest cluster
cluster = gpd.GeoDataFrame(geometry=gpd.points_from_xy(longitude, latitude))
# Plotting
fig, ax = plt.subplots(figsize=(10, 8))
# Plot Zurich
zurich.plot(ax=ax, color='red', marker='*', markersize=100, label='Zurich')
# Plot the smallest cluster
cluster.plot(ax=ax, color='blue', marker='*', markersize=100, label='Cluster')
# Set the geographical limits
ax.set_xlim(lonZurigo - 0.1, lonZurigo + 0.5)
ax.set_ylim(latZurigo - 0.1, latZurigo + 0.5)
# Add topographic basemap
ctx.add_basemap(ax=ax, crs='EPSG:4326', source=ctx.providers.OpenStreetMap.Mapnik)
plt.legend()
plt.show()
```

QUBO & MaxCut formulation of STP

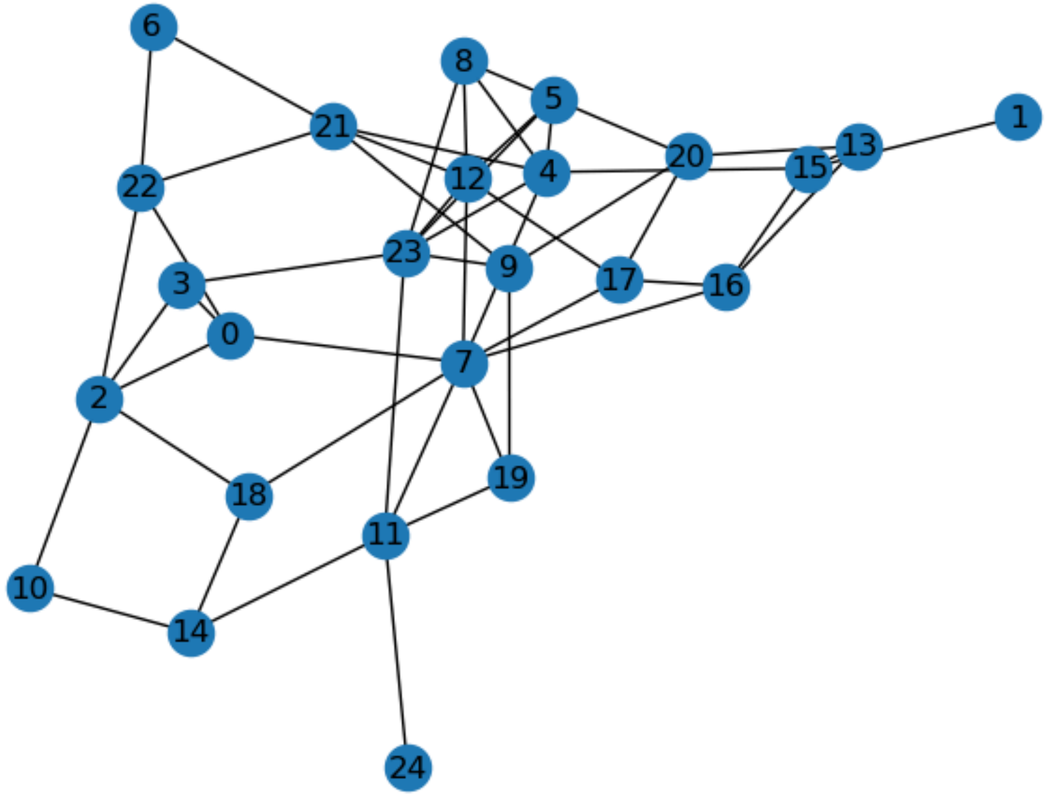
$$\underbrace{A\sum_{v}\left(1-\sum_{t}x_{vt}\right)^{2}}_{\text{visit each fire once}}+\underbrace{A\sum_{t}\left(1-\sum_{v}x_{vt}\right)^{2}}_{\text{no bilocation}}+\underbrace{B\sum d_{uv}\sum_{t}x_{ut}x_{u,t+1}}_{\text{minimize route distance}}$$

$$q_{ijkl}=A(\delta_{ij}+\delta_{kl})+Bd_{ij}\delta_{k+1,l}$$

$$w_{ij}=q_{ij}+q_{ji}\ (i,j>0)$$

$$w_{0j}=\sum q_{kj}+q_{jk}$$

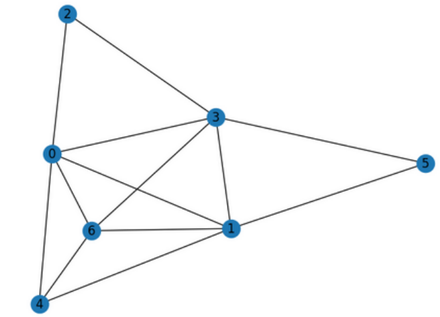
Computing max cut with quantum computers is computationally expensive.

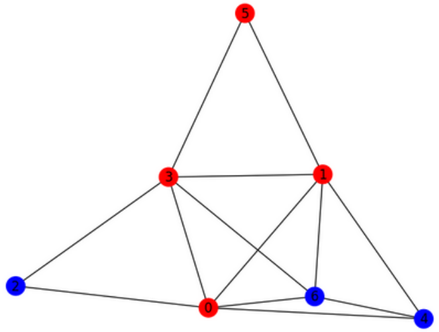


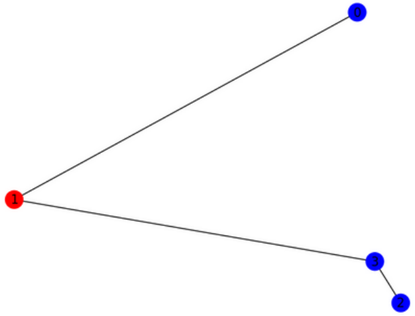
Trying to solve the max cut directly on this graph with a quantun computer wouldn’t be feasible, we can do better.

Graph transformation

- We can use a fun property of graphs, which is that some particular transformations, if reversible, preserve the max cut solution.
- We aim to find a simpler graph

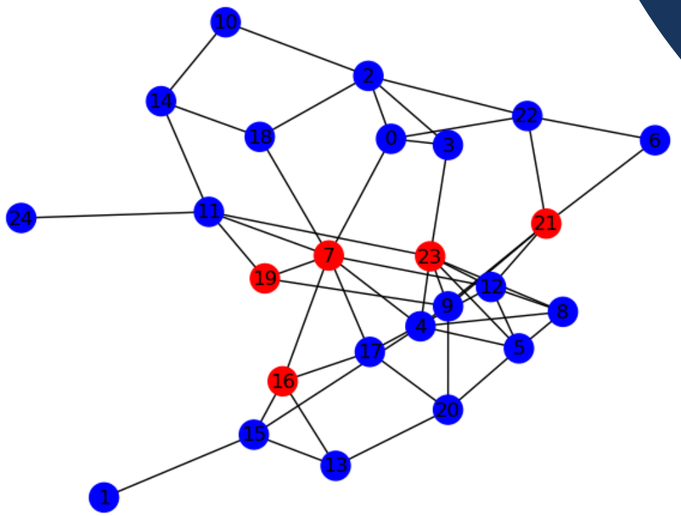


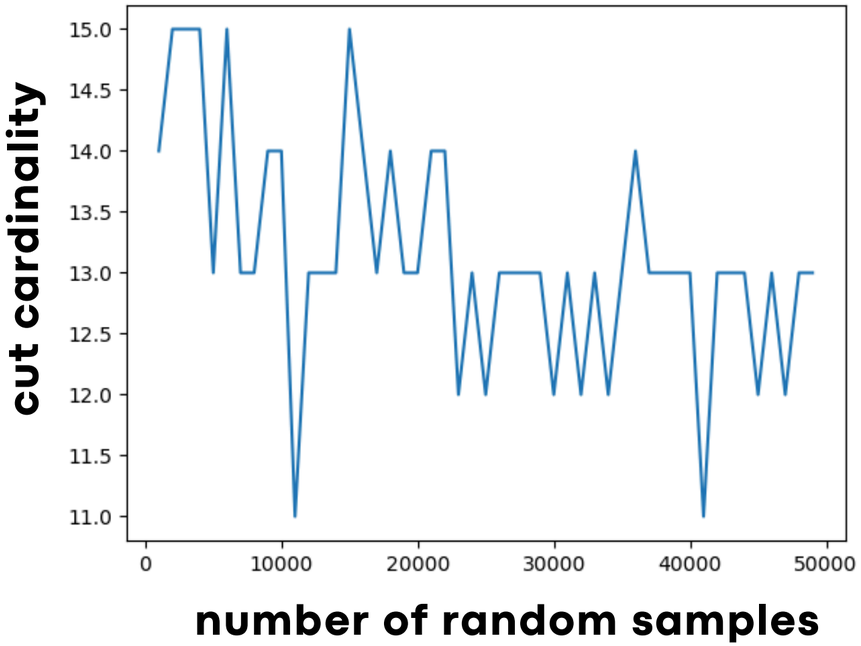




Finding the optimal cut

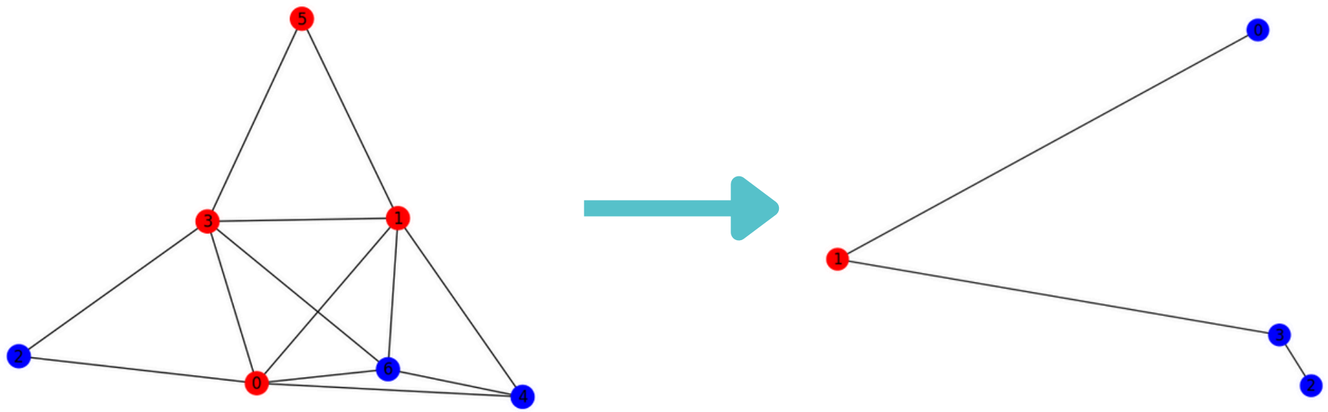
- The first step constists in finding the **Balanced Low Cardinality Vertex Separator**
- We used **random sampling** to drastically improve the time complexity of the algorithm



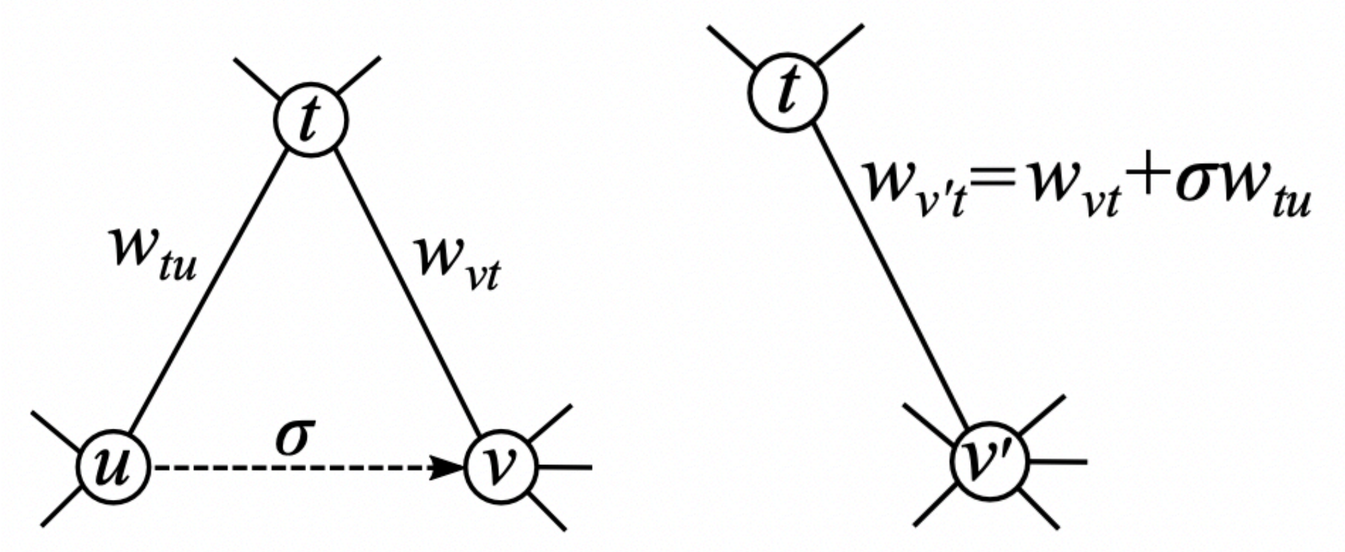


Graph Shrinking

- We want to shrink the cut to a single node
- We do that by applying consecutive vertex merging transformation
- Weights are modified in order to preserve the max cut solution

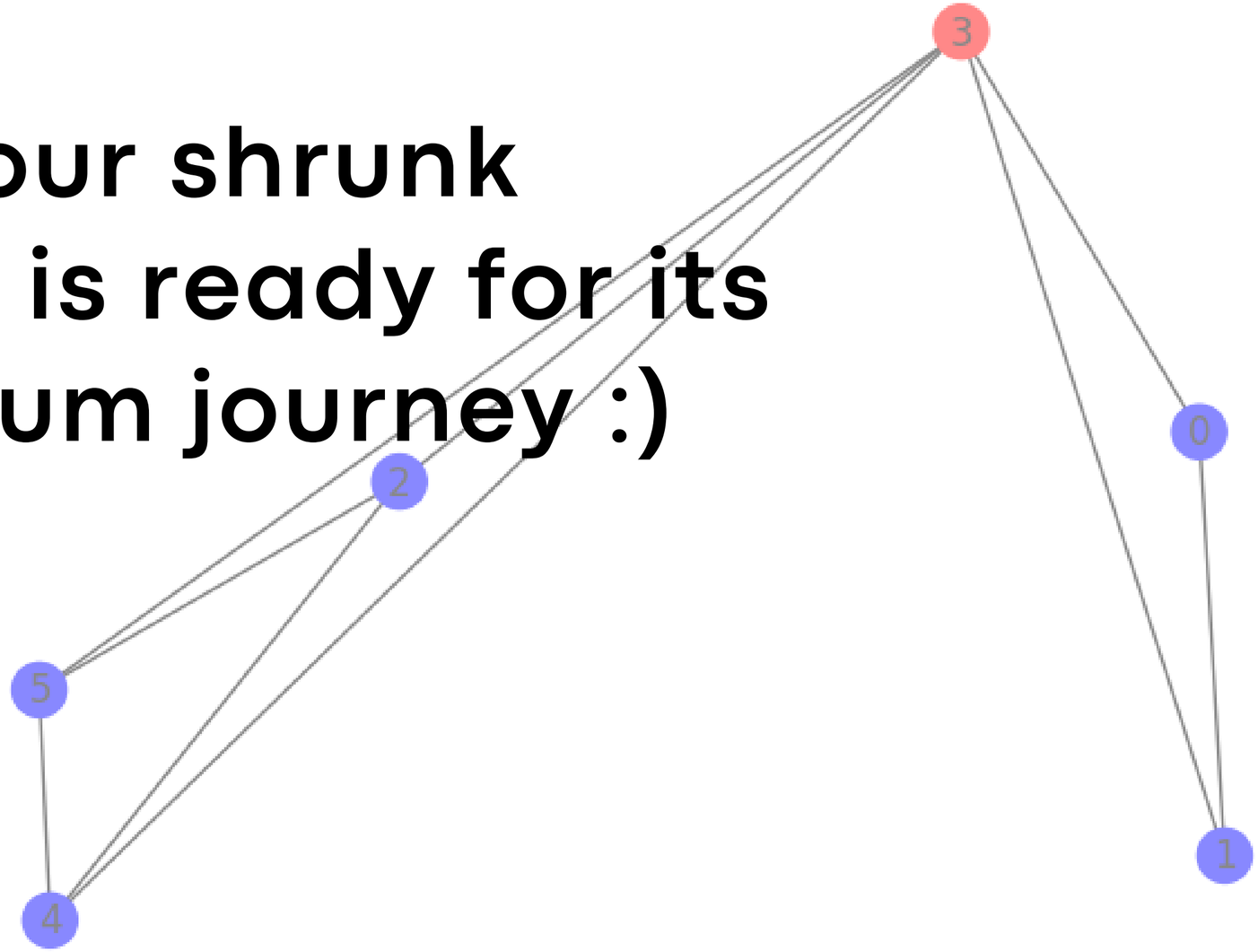


Semi-Definite Relaxation



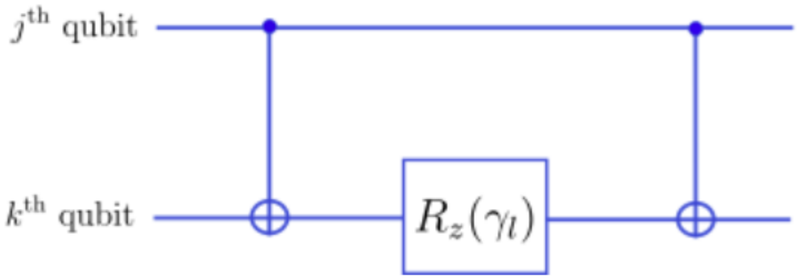
- The Goemans-Williamson algorithm can be used to update the weights
- A probabilistic approach is used to calculate the likelyhood of an edge linking two vertices of the same or opposite sets of the optimal max cut partition.

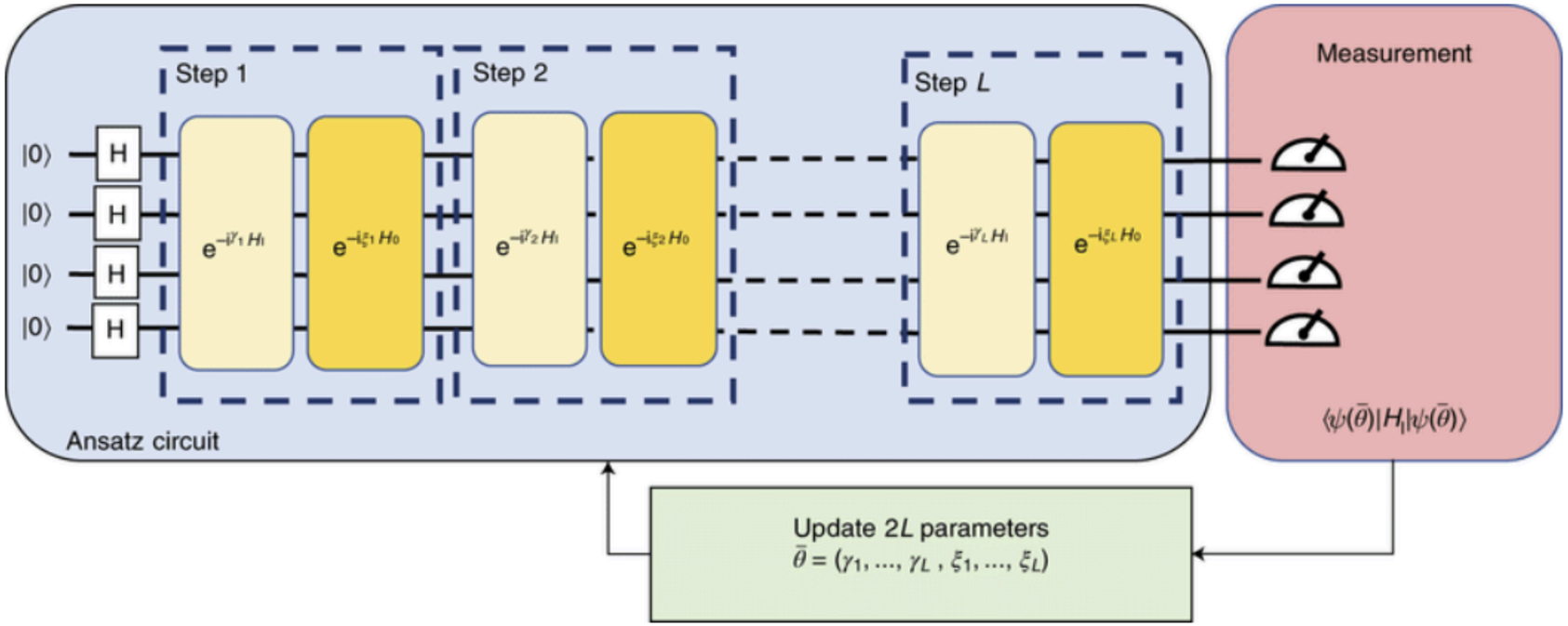
Now, our shrunk graph is ready for its quantum journey :)

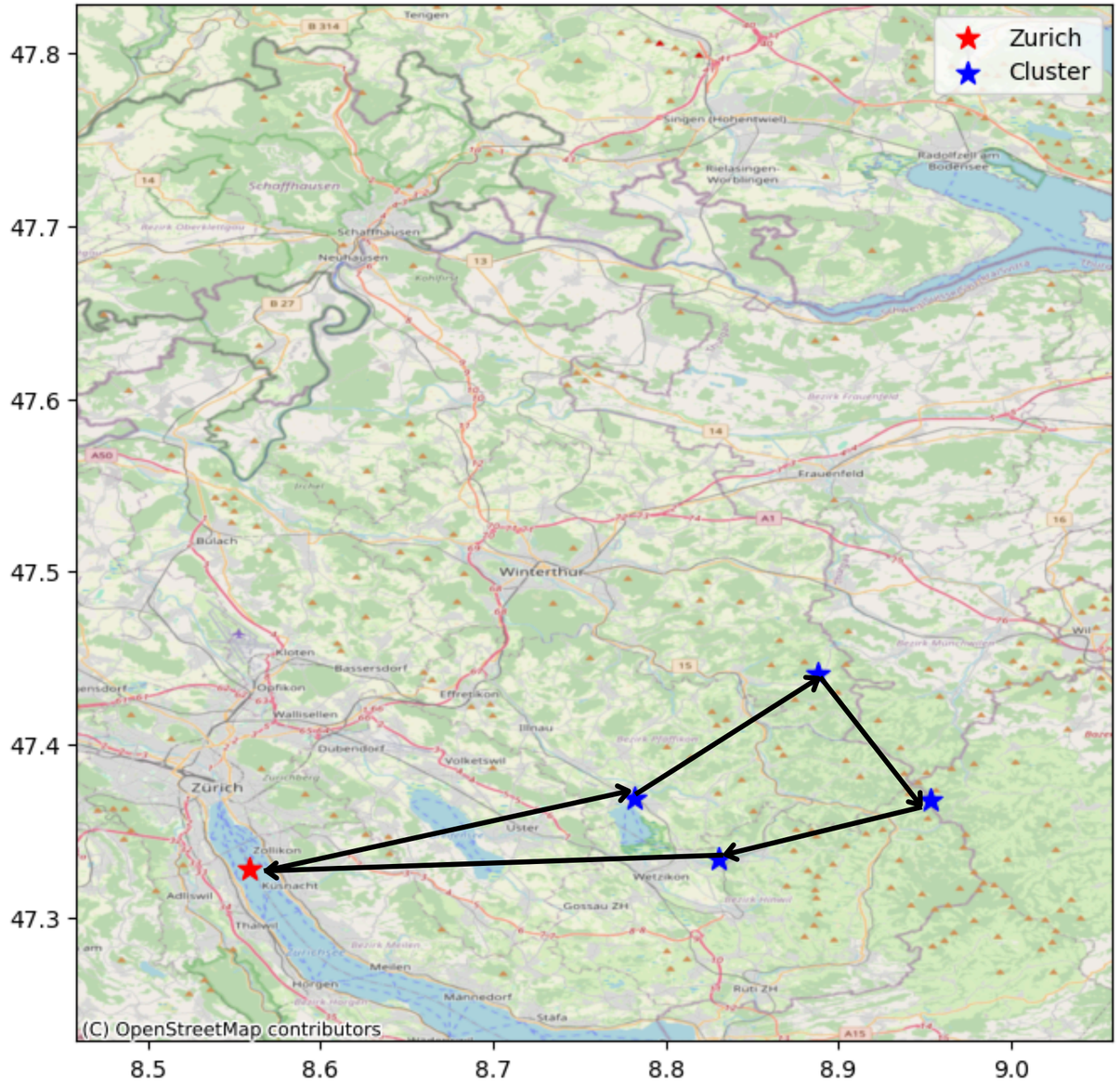


QAOA implementation

$$H = \sum_{(i,j)\in E} w_{ij}\sigma_z^i\sigma_z^j$$






Zurich
Cluster
47.8
47.7
47.6
47.5
47.4
47.3
8.5
8.6
8.7
8.8
8.9
9.0
(C) OpenStreetMap contributors
Winterthur
Zürich
Schaffhausen
Frauenfeld
Bülach
Kloten
Bassersdorf
Wallisellen
Dübendorf
Uster
Wetzikon
Gossau ZH
Zollikon
Küsnacht
Meilen
Männedorf
Rüti ZH
Stäfa

Our Team

Luis Fernández

Lorenzo De Lotto

Tommaso Guarniera

Stefano Carra

Roger Serrat


THANK
YOU