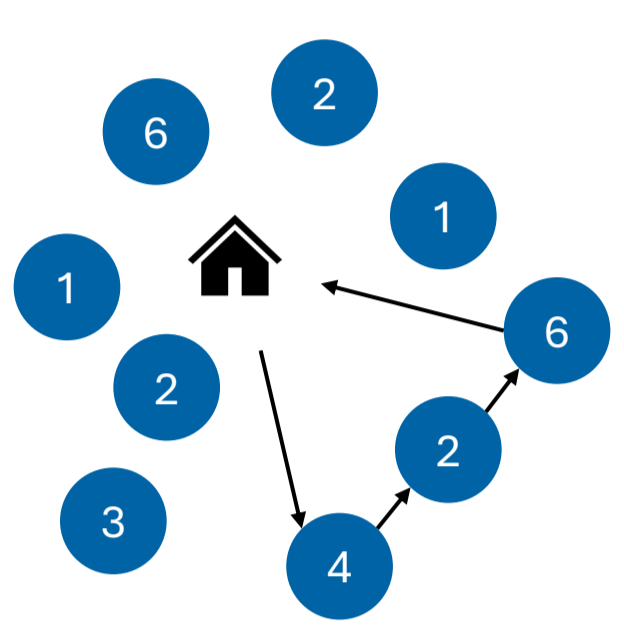


Problem

Vienna offers a wealth of attractions and activities, more than you can see in one day. The aim of this project is to create the **optimal day plan** based on the **user's preferences**. To do this, our algorithm creates a route so that users can do as much as possible in one day.

The Orienteering Problem

The underlying problem is the orienteering problem. This is a routing problem in which the goal is to **collect as many points as possible** while limiting the duration or distance of the journey.



- Gathering points by **visiting nodes**
- Constraints** make it infeasible to visit all nodes
- > Find route that **maximizes points** under constraints

Constraints

In our problem we have the following constraints:

- Time limit:** We want to finish our tour withing a timeframe specified by the user
- Opening hours:** locations are not available constantly
- Visit duration:** we need to spend time at the locations to gather the points
- Scheduled stops:** we might have mandatory stops that we have to visit

Data

OpenStreetMaps

- Information about locations of interest like position, category, opening hours and name
- Based on open data and community contribution
- Data often not complete -> Imputation necessary

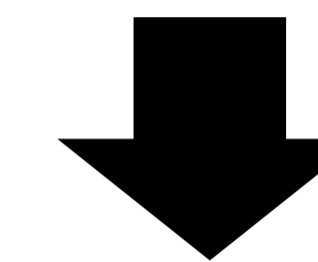


Wiener Linien GTFS

- GTFS is a common data format for public transportation schedules and associated geographic information
- We use parts of the GTFS set to calculate travel times between metro stations



Processing



Travel Time Matrix

- Distance matrix that contains the walking time distance between all nodes and travel distance between metro stations

	0	1	2	3	4	5	6	7	8	9
0	0.00	65.21	102.71	88.59	107.31	2.00	3.00	22.00	2.00	4.00
1	65.21	0.00	70.58	36.96	76.26	79.95	81.01	194.00	28.32	83.23
2	102.71	70.58	0.00	87.15	5.72	127.66	134.69	133.64	80.92	148.04
3	88.59	36.96	87.15	0.00	82.63	110.92	108.80	219.42	63.77	119.33
4	107.31	76.26	5.72	82.63	0.00	132.01	139.34	128.74	86.05	152.68
5	2.00	79.95	127.66	110.92	132.01	0.00	101.66	101.66	101.66	101.66
6	3.00	81.01	134.69	108.80	139.34	101.66	0.00	101.66	101.66	101.66
7	22.00	194.00	133.64	219.42	128.74	101.66	101.66	0.00	101.66	101.66
8	2.00	28.32	80.92	63.77	86.05	101.66	101.66	101.66	0.00	101.66
9	4.00	83.23	148.04	119.33	152.68	101.66	101.66	101.66	101.66	0.00

Category Dataframe

- Dataframe that contains the filtered information of the OpenStreetMaps data

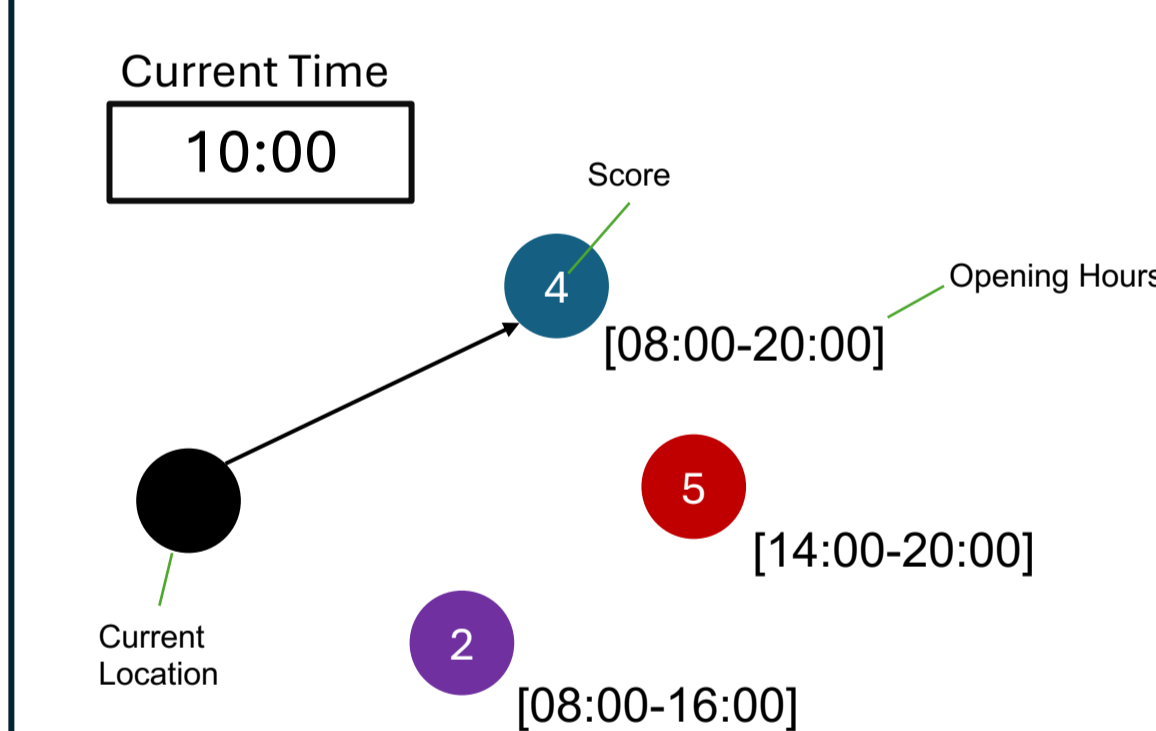
id	category	opening hours	name	lat	lon	score
0	station		Schottentor	48.204138	16.373127	
1	pharmacy	Mo-Fr 08:00-18:00, Sa 09:00-12:00	Apothek Zur Universität	48.204138	16.373127	
2	pharmacy	Mo-Fr 08:00-18:00, Sa 09:00-12:00	Apothek Apotheke	48.204138	16.373127	
3	pharmacy	Mo-Fr 08:00-18:00, Sa 09:00-12:00	Waldhof Apotheke	48.204138	16.373127	
4	park		Urbanize	48.204138	16.373127	
5	library		Dietmar Steiner Bibliothek	48.204138	16.373127	
6	station		Volkstheater	48.204138	16.373127	
7	station		Schottentor	48.204138	16.373127	
8	park		University of Vienna	48.204138	16.373127	

Algorithm

Users can specify which **category of locations** they want to visit and decide the **relevance of each category**, by assigning points to it. Furthermore users can specify start and stop location, timeframes when they want to visit categories and how much time they want to spend at a category. The algorithm will construct a route with the goal to gather as many points as possible given the constraints.

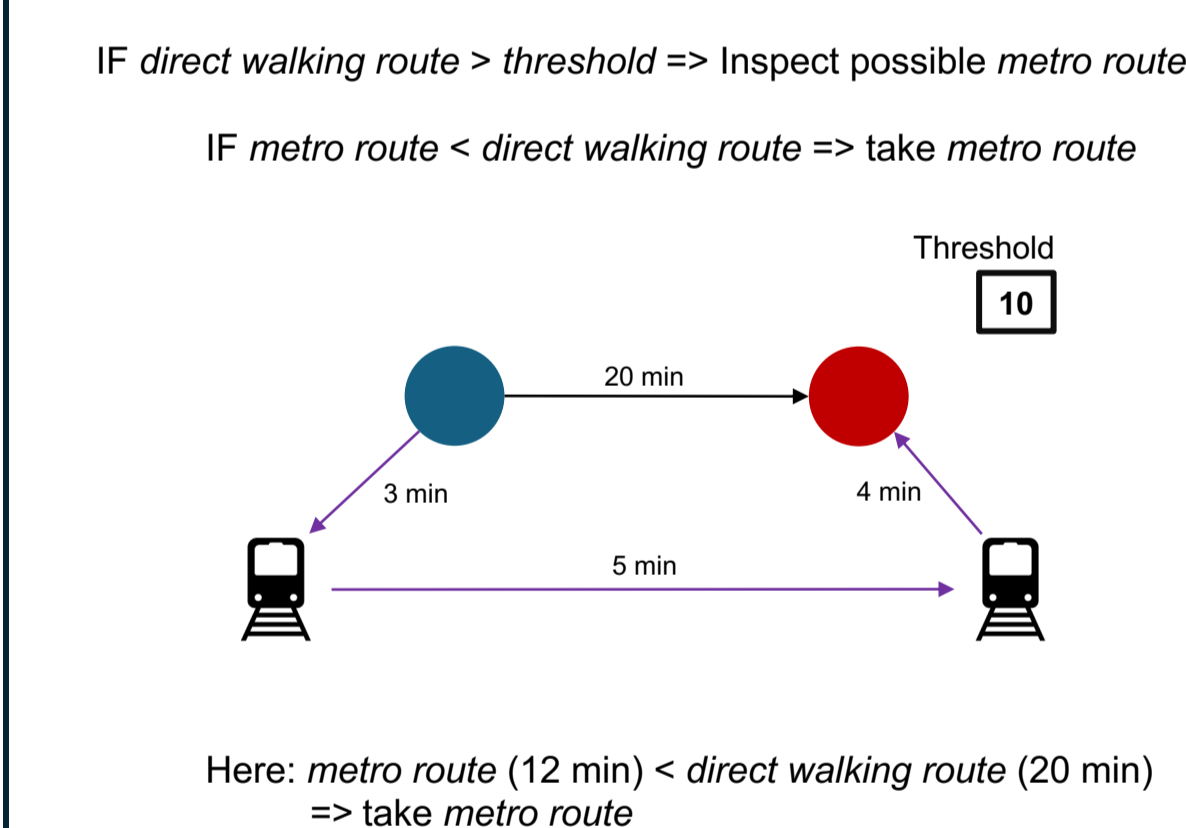
Greedy Approach

The next location to visit is chosen based on a score calculated by points/distance. The best location which is currently available will be visited.



Metro Traveling

The algorithm speeds up travel time by using the metro system of Vienna. If a threshold is passed the walking distance will be compared to an alternative route that includes travelling by metro. The faster option will be chosen.



Mandatory Visits

Users have the option of specifying whether they want to visit certain locations at certain times. The route is created according to these mandatory stops.

Route Generation

The algorithm is strongly path-dependent. To explore the solution space further, the algorithm is executed several times. At each iteration, locations that were visited in previous iterations are excluded. The result is a selection of several routes from which the user can choose.

Pseudo Code

```

BEGIN
  IF categories isNotEmpty
    FOR EACH location IN categories
      calculate score

  IF location with highest score open
    ADD location to route

  IF walking dist > threshold
    IF walking dist > metro dist
      take metro

  categories REMOVE location.category
  time = time + travel time + visit duration

  Check if end point is reachable in time

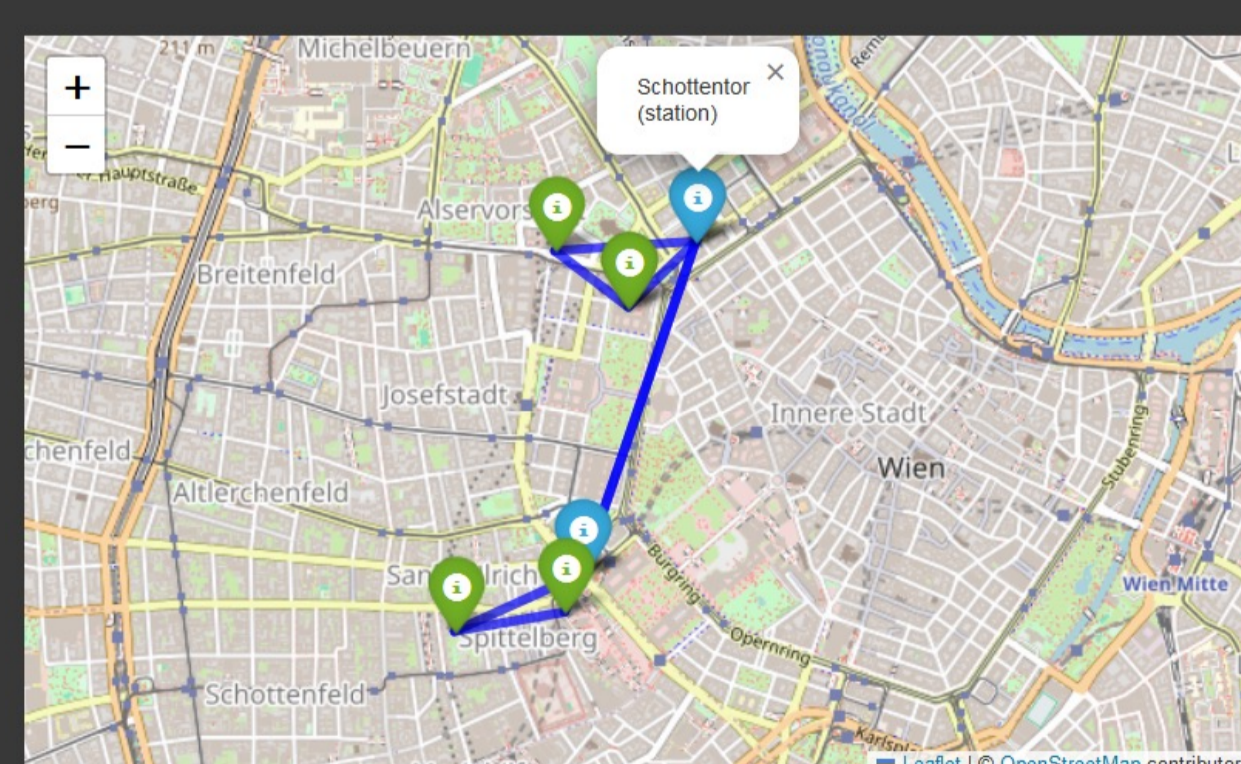
REPEAT
END
    
```

Results

Users are given a **selection of routes** that are optimized to collect as many points as possible by visiting the desired location categories during the specified period.

Example Output

Type	Name	Arrival Time	Departure Time
0	University of Vienna	09:00	09:00
1	pharmacy Apotheke "Zur Universität"	09:05	09:26
2	station Schottentor	09:35	09:35
3	station Volkstheater	09:43	09:43
4	park Urbanize	09:52	11:32
5	library Dietmar Steiner Bibliothek	11:39	12:39
6	station Volkstheater	12:42	12:42
7	station Schottentor	12:50	12:50
8	park University of Vienna	12:56	13:56



Alternative Routes

Mixed Integer Programming

Objective Function

$$\text{Maximize } \sum_{c \in C} R_c \cdot z_c$$

Constraints

1. Start and End Points

$$\sum_{j \in V \setminus \{0\}} x_{0j} = 1, \quad \sum_{i \in V \setminus \{n\}} x_{in} = 1$$

2. Flow Conservation

$$\sum_{j \in V \setminus \{i\}} x_{ij} = y_i, \quad \sum_{j \in V \setminus \{i\}} x_{ji} = y_i \quad \forall i \in V$$

3. Time Windows at Points

$$e_i \cdot y_i \leq T_i \leq l_i \cdot y_i \quad \forall i \in V$$

4. Travel Time with Stay Time

$$T_j \geq T_i + t_{ij} + \tau_i - M \cdot (1 - x_{ij}) \quad \forall i, j \in V$$

5. Category Visits

$$z_c \leq \sum_{i \in V: \text{cat}(i)=c} y_i \quad \forall c \in C$$

$$\sum_{i \in V: \text{cat}(i)=c} y_i \leq 1 \quad \forall c \in C$$

6. Global Time Frame for the Process

$$T_0 \geq E, \quad T_n \leq L$$

7. Binary and Non-Negativity Constraints

$$x_{ij} \in \{0, 1\}, \quad y_i \in \{0, 1\}, \quad z_c \in \{0, 1\}, \quad T_i \geq 0 \quad \forall i \in V, \quad c \in C$$

- Notation**
- $V = \{0, 1, \dots, n\}$: Set of points (0 = start, n = endpoint).
 - C : Set of categories.
 - $S \subseteq V$: Set of stations (can be visited multiple times, no reward).
 - R_c : Reward for visiting category $c \in C$.
 - $\text{cat}(i)$: Category assigned to point i , if i belongs to a category.
 - $P_i = 0$: Reward points for a point depending on category.
 - $[e_i, l_i]$: Time window for point i (earliest and latest arrival time).
 - t_{ij} : Travel time from point i to point j .
 - τ_i : Stay time at point i .
 - $[E, L]$: Global time frame:
 - E : Earliest possible start time.
 - L : Latest possible end time.
 - M : A large constant (Big-M) used to deactivate constraints.
- Decision Variables**
- $x_{ij} \in \{0, 1\}$: Indicates whether the route from point i to point j is used.
 - $y_i \in \{0, 1\}$: Indicates whether point i is visited.
 - $z_c \in \{0, 1\}$: Indicates whether category $c \in C$ has been visited.
 - $T_i \in \mathbb{R}^+$: Arrival time at point i .