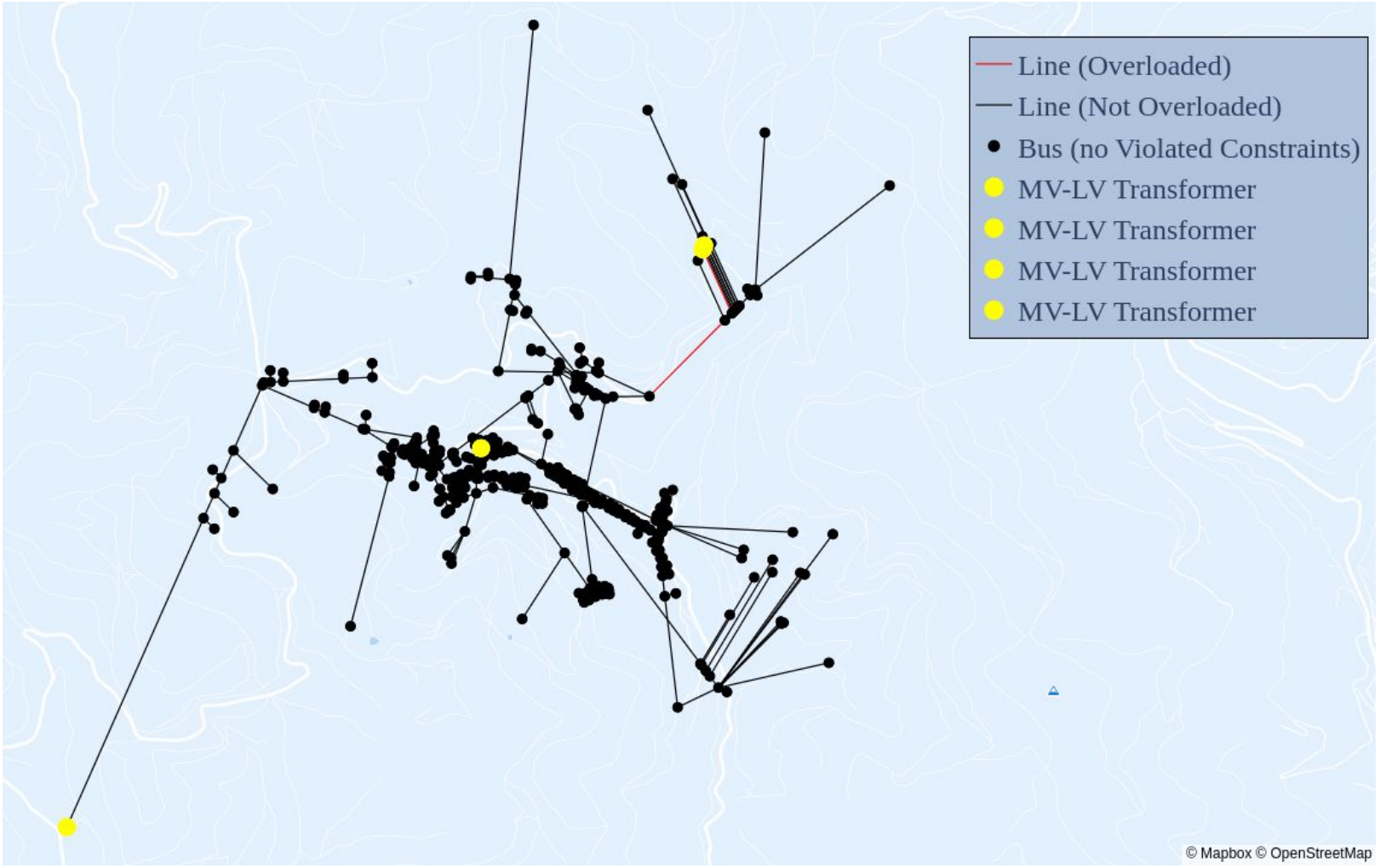
Generating Low-Voltage Grid Benchmarks with OpenStreetMap

Metty Kapgen

University of Freiburg
Faculty of Engineering
Department of Computer Science
Chair of Algorithms and Data Structures

UNI FREIBURG

Introduction



Introduction

Problem

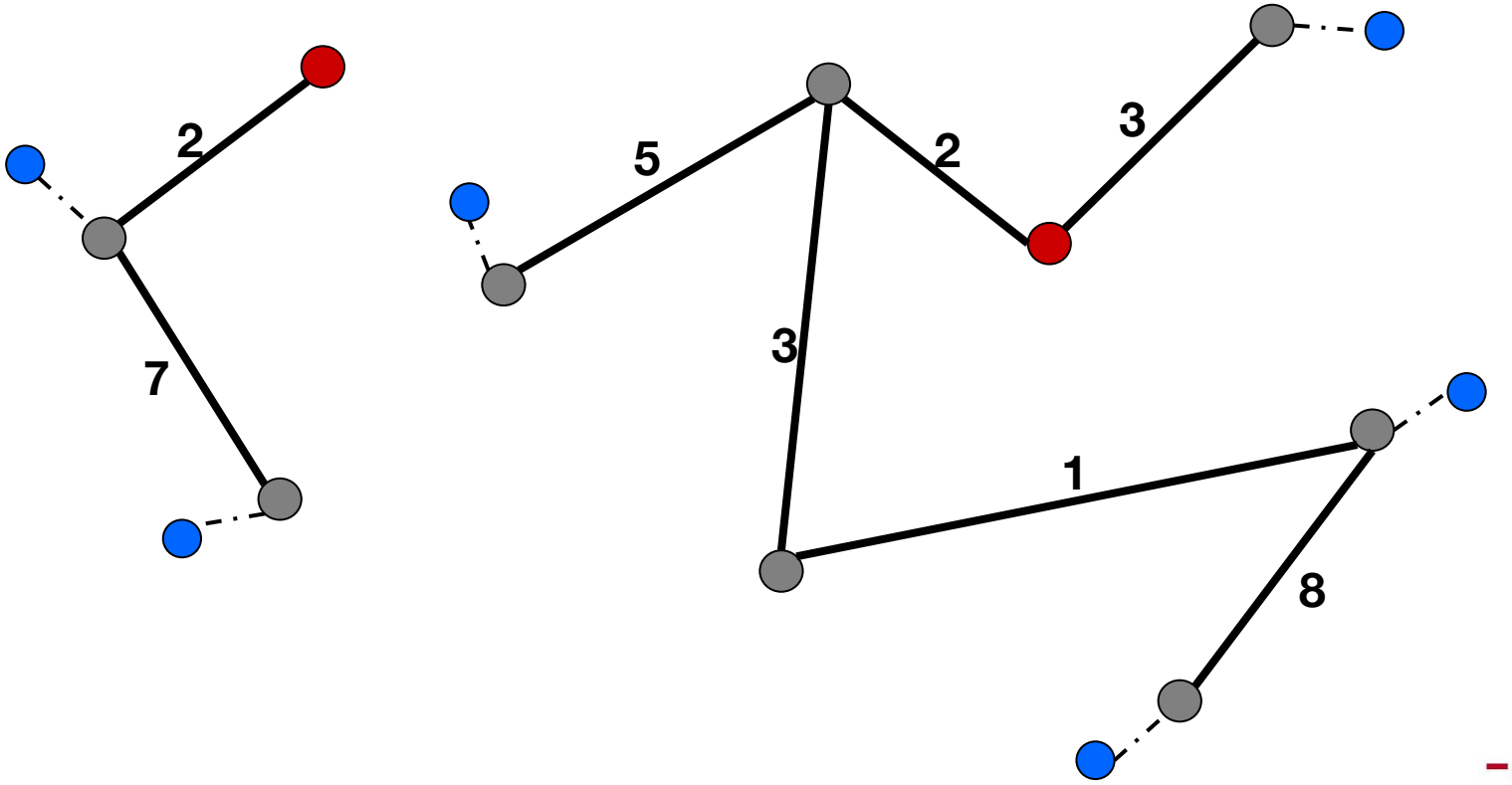
- Constant rise in demand for electrical energy
- Thus, we need to expand the electrical supply of villages
- More and more algorithms are proposed for grid expansion
- Realistic grid plans are hard to find

Solution

- Use OpenStreetMap and a set of forest algorithms
- Generate realistic grids to validate and benchmark such expansion planning algorithms

What is a “low-voltage grid”

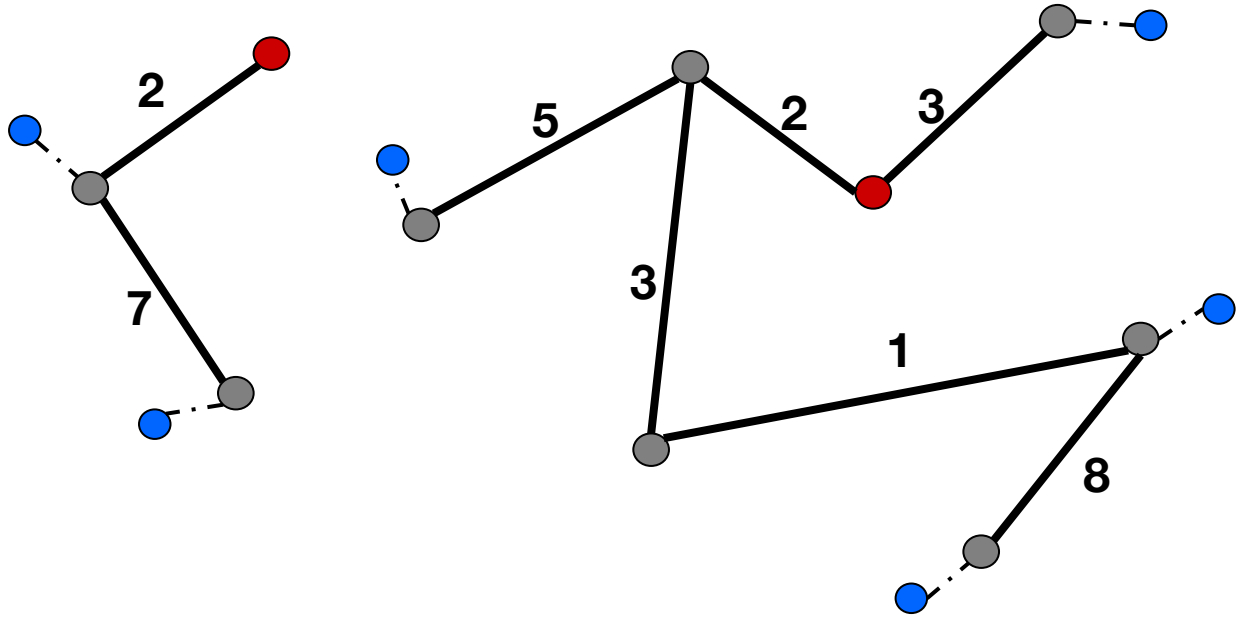
- Simple graph representing the electrical supply of a village



What is a “low-voltage grid”

Nodes can be:

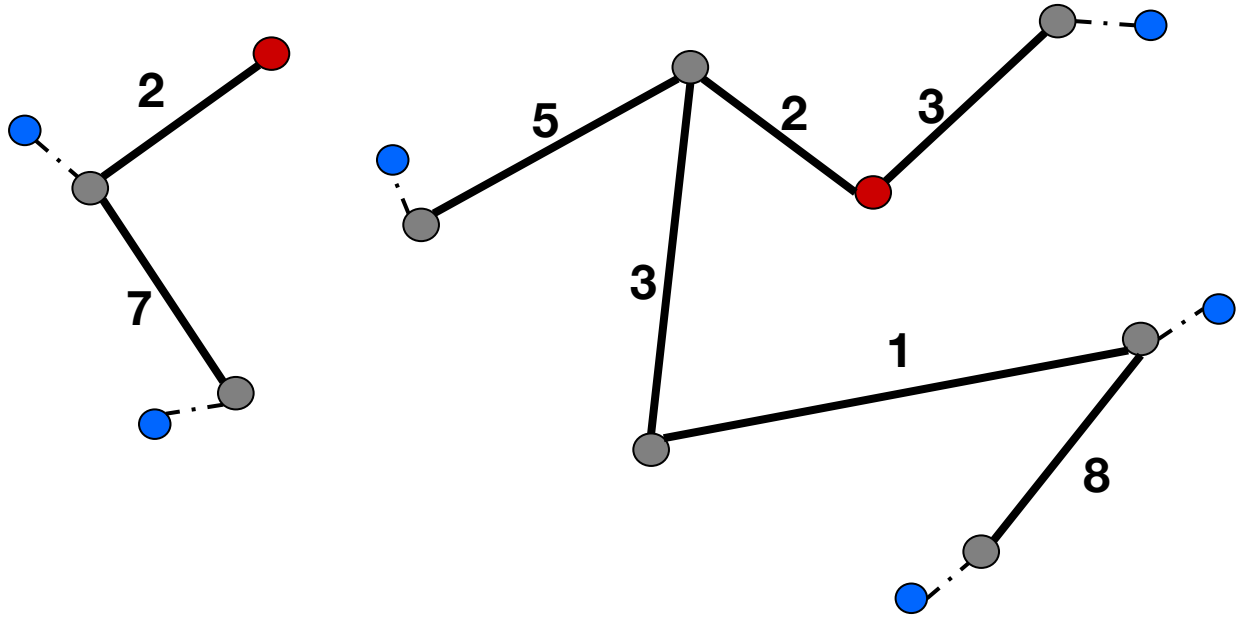
- Substations (supply energy)
- Buildings (consume energy)
- Generators (supply energy)
- Cable intersections



What is a “low-voltage grid”

Edges are:

- Lines of a certain cable type
- connecting different Nodes
- Weights represent the physical distance of a line



What is a “low-voltage grid”

Private Line: 

- Connects a “private” entity (building) to the grid

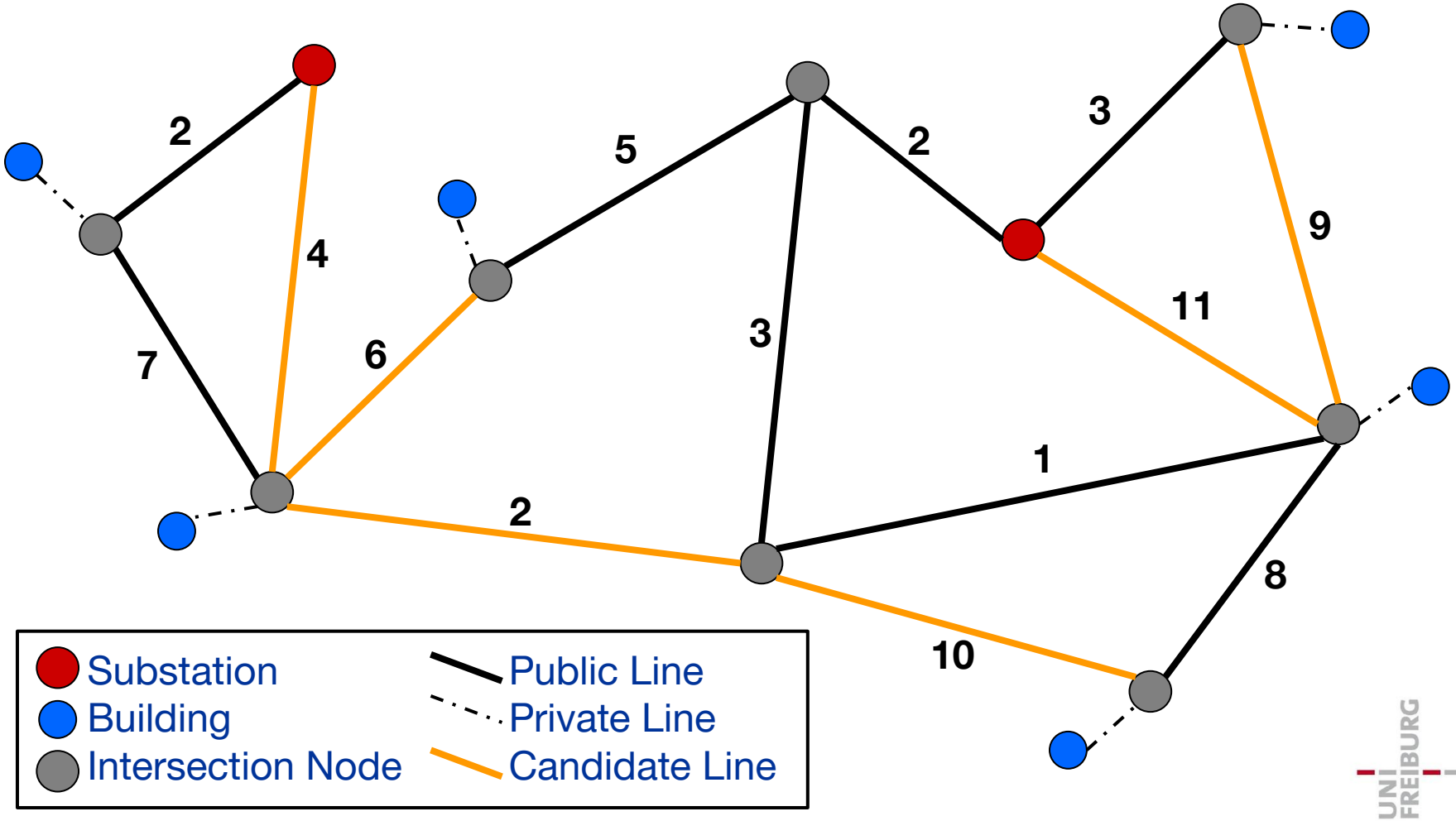
Public Line: 

- All lines that are owned and maintained by the grid operator and are currently in use

New candidate Lines: 

- Lines that are not part of the currently installed grid, but may be considered in future upgrades

Currently installed grid with candidate lines



Constraints for a “valid” currently installed grid

Topological Constraints:

- Grid can not have circles
- Grid can not have a path between different substations

Electrical Constraints:

- Voltages and line loadings must not exceed given limits -> PyPSA’s Powerflow

What is "expansion planning"

To efficiently find an expansion of the grid, an algorithm should

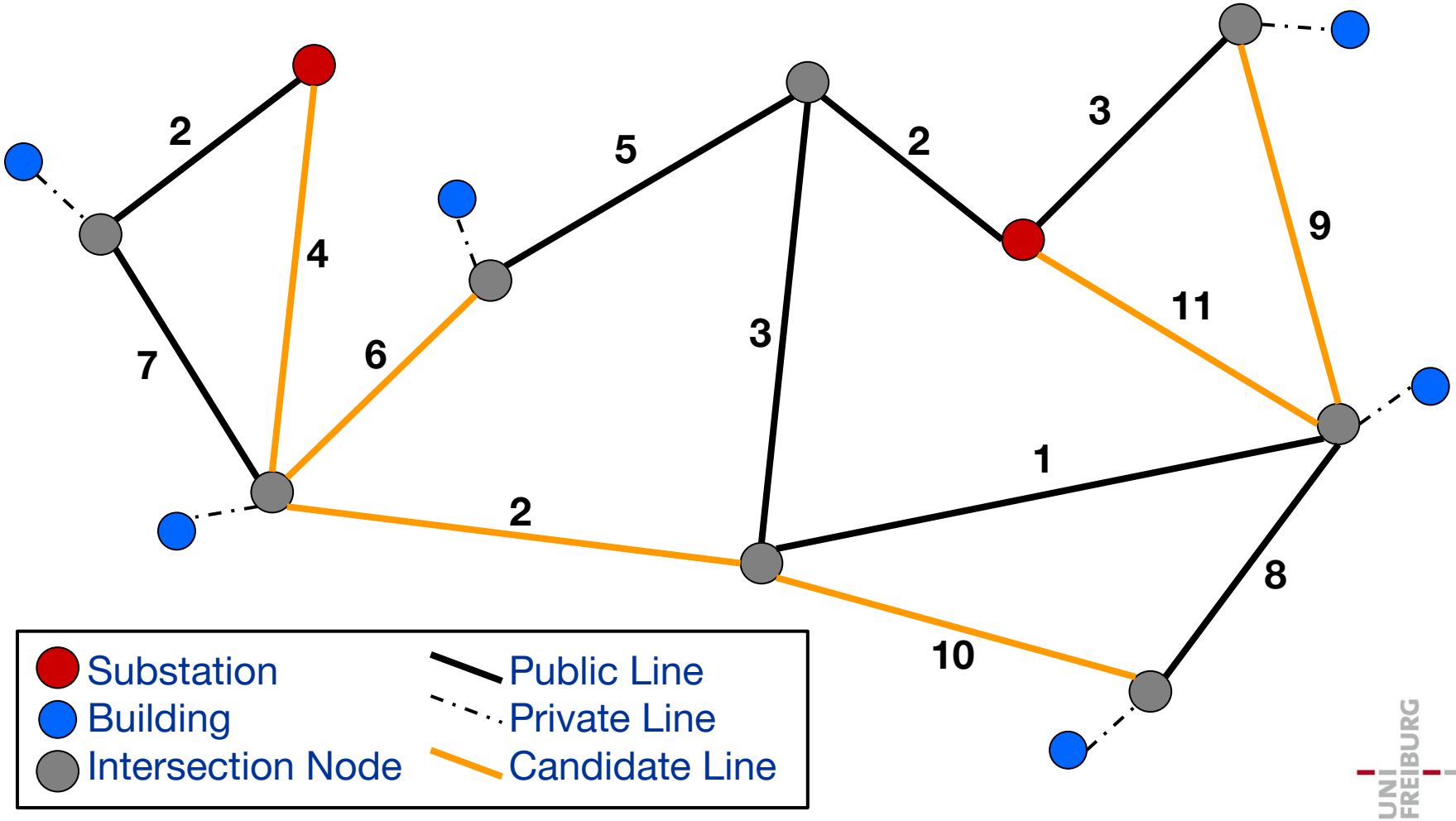
- upgrade different lines with a better cable type
- adding new candidate lines to the grid

A valid expansion should:

- not violate the CONSTRAINTS
- minimize the overall cost of the upgrades

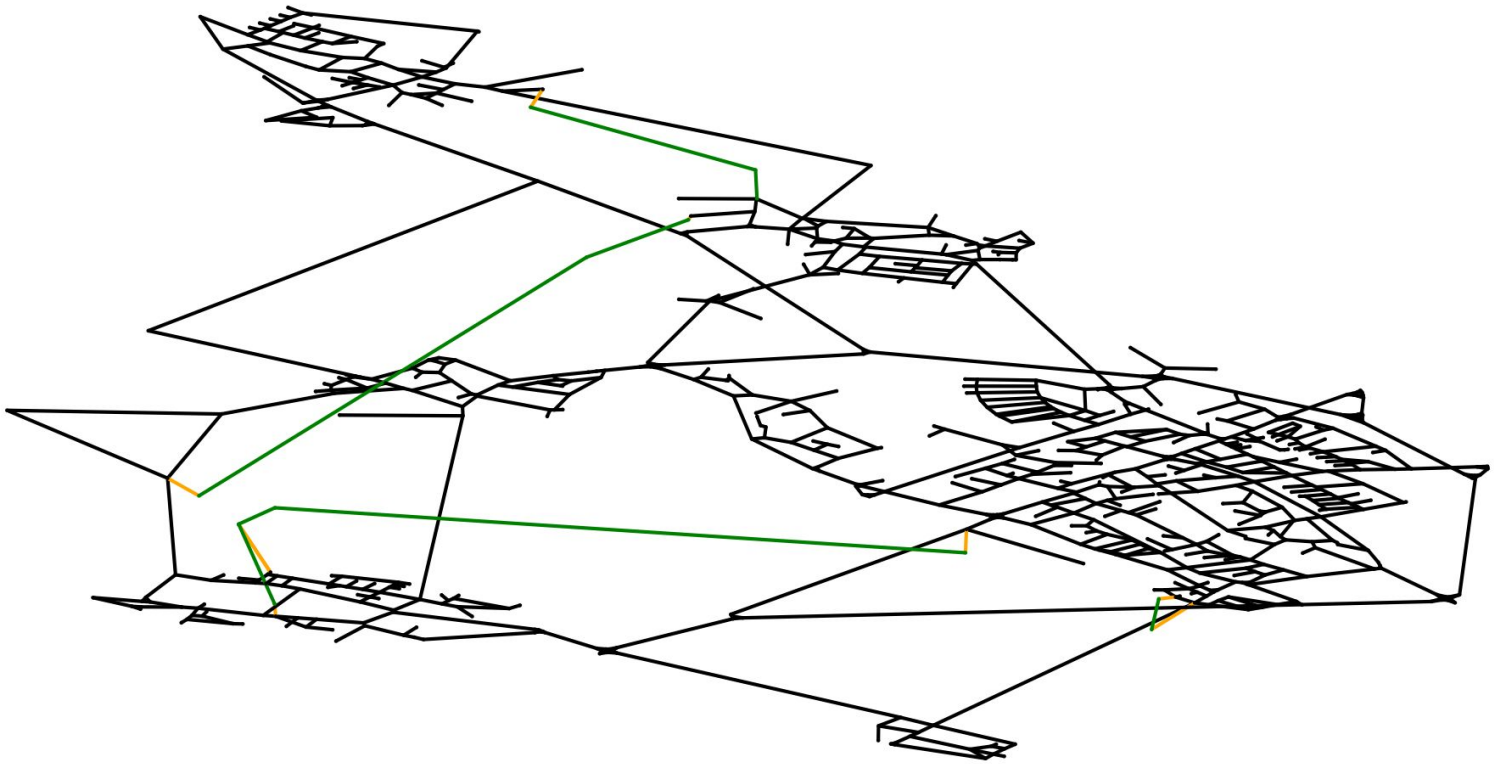
Note: We want to provide such algorithms with grids that have electrical constraint violations

Q&A


2
4
7
6
5
3
2
3
9
11
1
2
8
10
Substation
Building
Intersection Node
Public Line
Private Line
Candidate Line

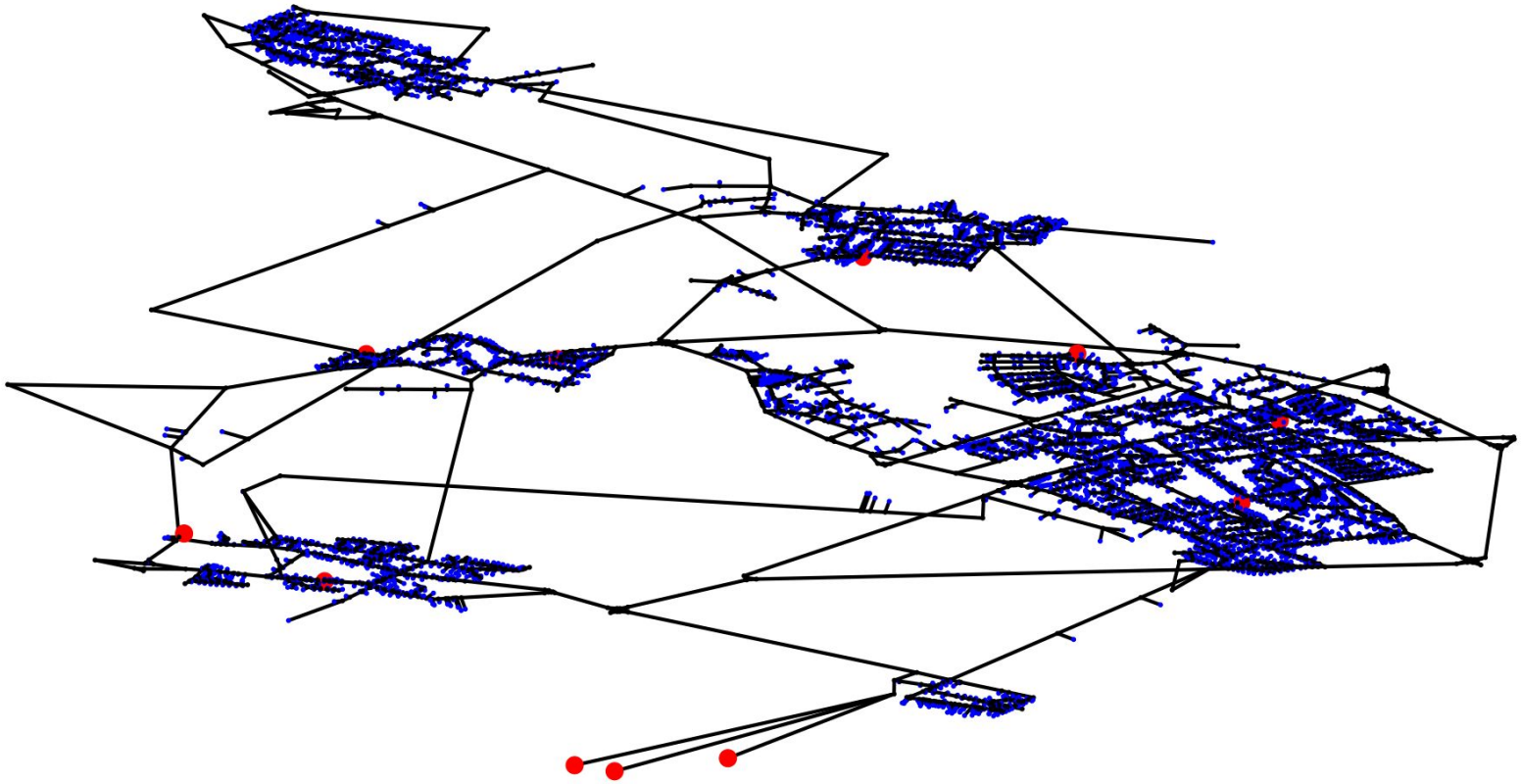
Data extraction (for Bad Krozingen)

Using OSMnx[1] to combine the street and powerline graphs



Data extraction (for Bad Krozingen)

Using PyOsmium[2] to add substations and buildings



Grid generation

Define the current state of the grid

- Which lines are part of the currently installed grid?
- Which lines are part of the new candidate lines?
- Which nodes are connected to which subtree?

We need to generate a (**s**ubstation-**b**uilding)
sb-forest

What is an sb-forest

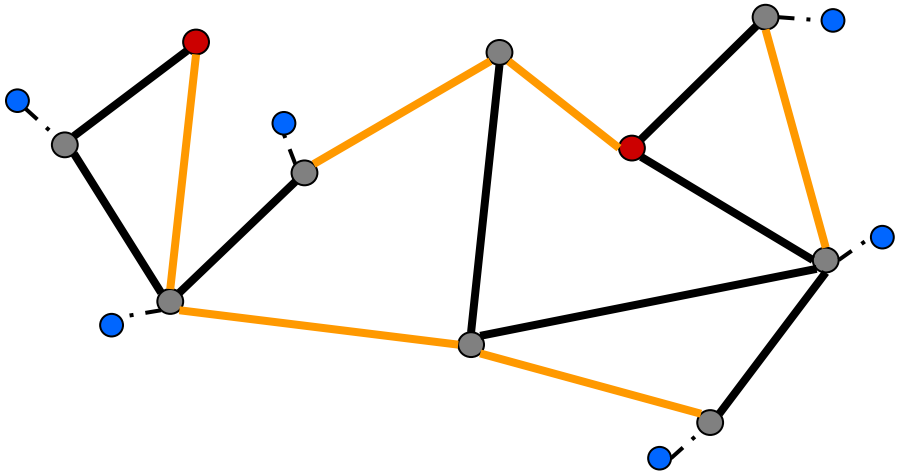
Is **not** a spanning forest

- Not every cable intersection must be part of the currently installed grid

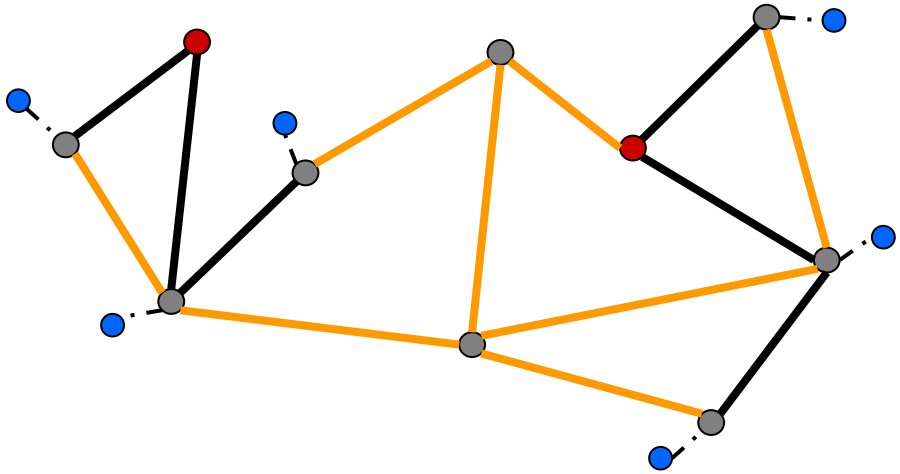
SB-Forest guarantees that

- **Every** building is connected to **exactly** one substation
- The forest is cycle-free
- Nodes and edges that are not part of sb-forest will become candidate lines

Spanning Forest vs SB-Forest



Spanning Forest



SB-Forest

Grid generation

Algorithms to generate different SB-Forests

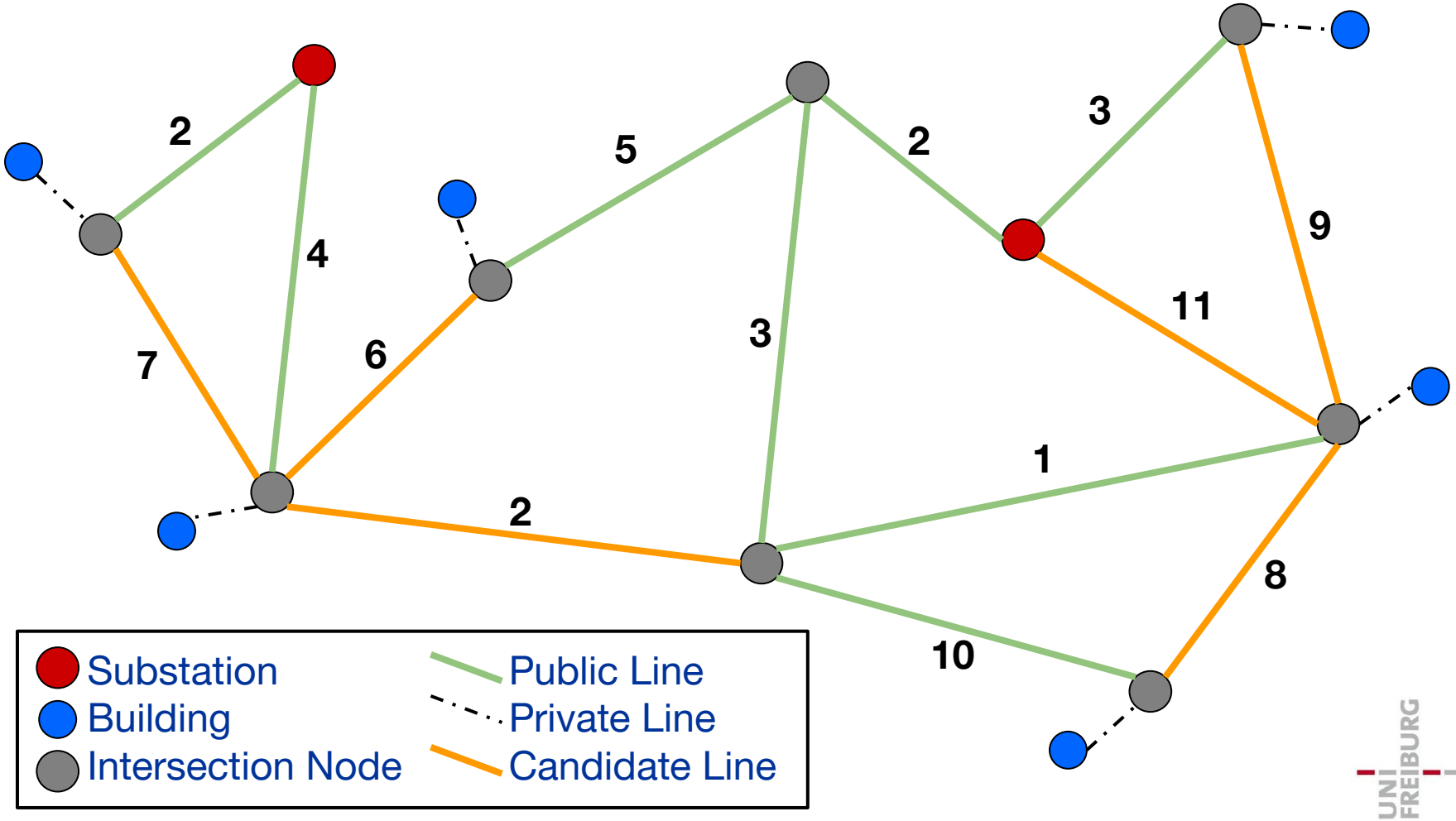
- Shortest-Path SB-Forest
- Random SB-Forest
- Minimum SB-Forest
- Maximum SB-Forest

Note: Edge weights are defined through the distances of their nodes

Shortest-Path SB-Forest

- Uses Dijkstras' algorithm with multiple sources
- Sources are the substations
- Initial Queue of algorithm is filled with substation nodes having weight 0
- Returns shortest Path for each node to closest substation

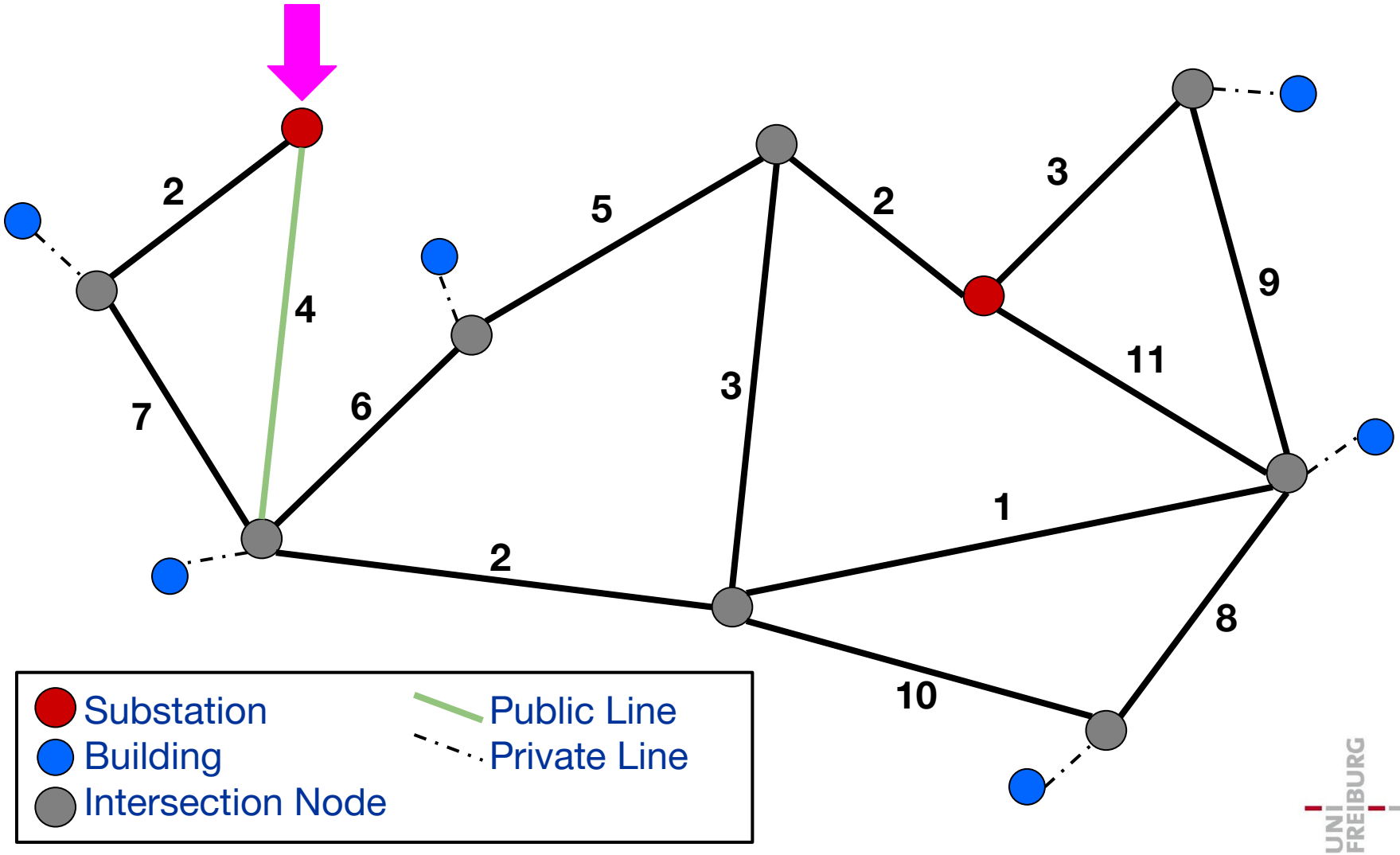
Shortest-Path SB-Forest



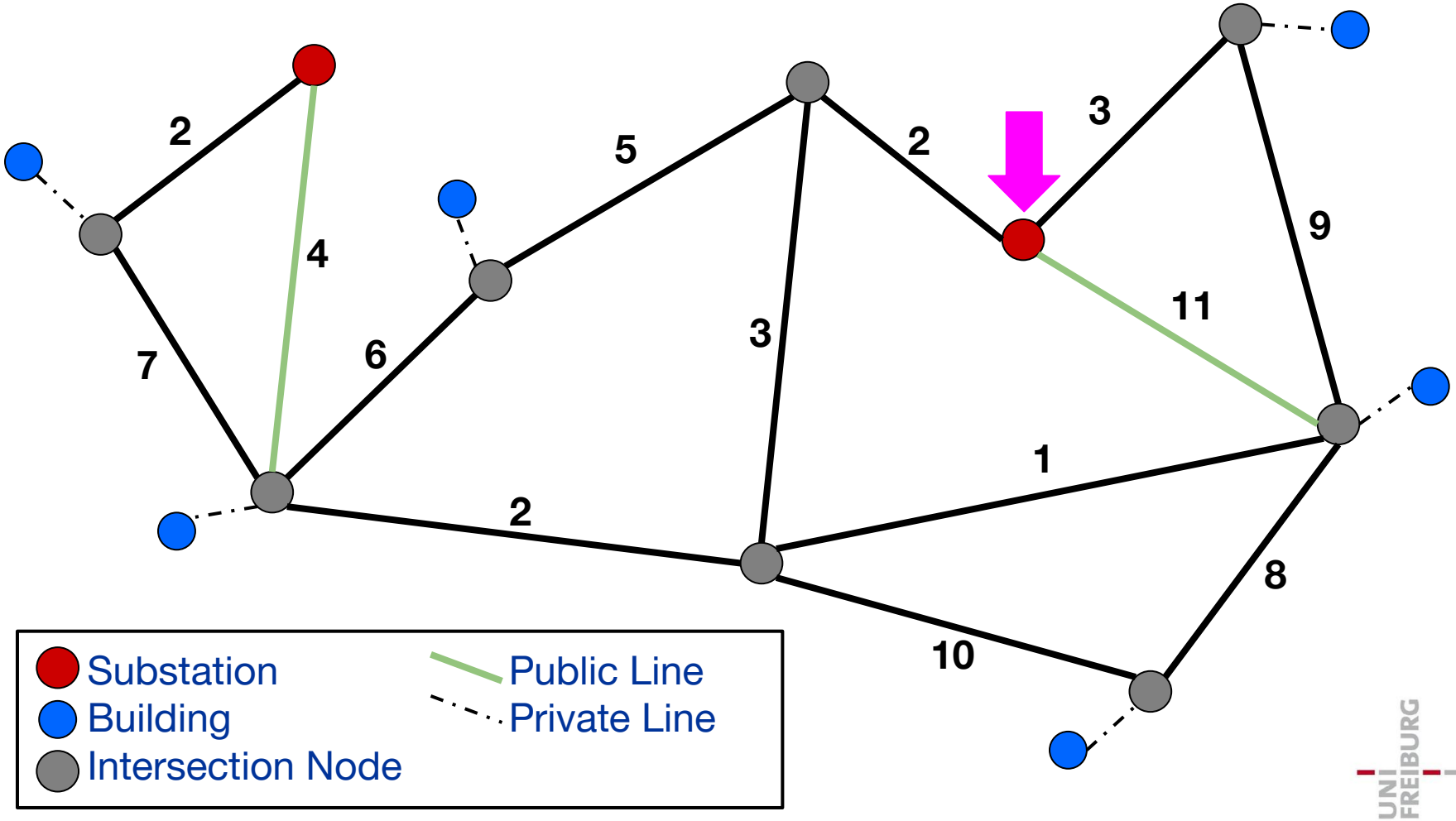
Random SB-Forest

- Iteratively let substations select neighbor node until we have a spanning forest
- Then, trims the spanning forest back until it is an sb-forest by removing all nodes of degree one that are neither a substation nor have a connection to a building

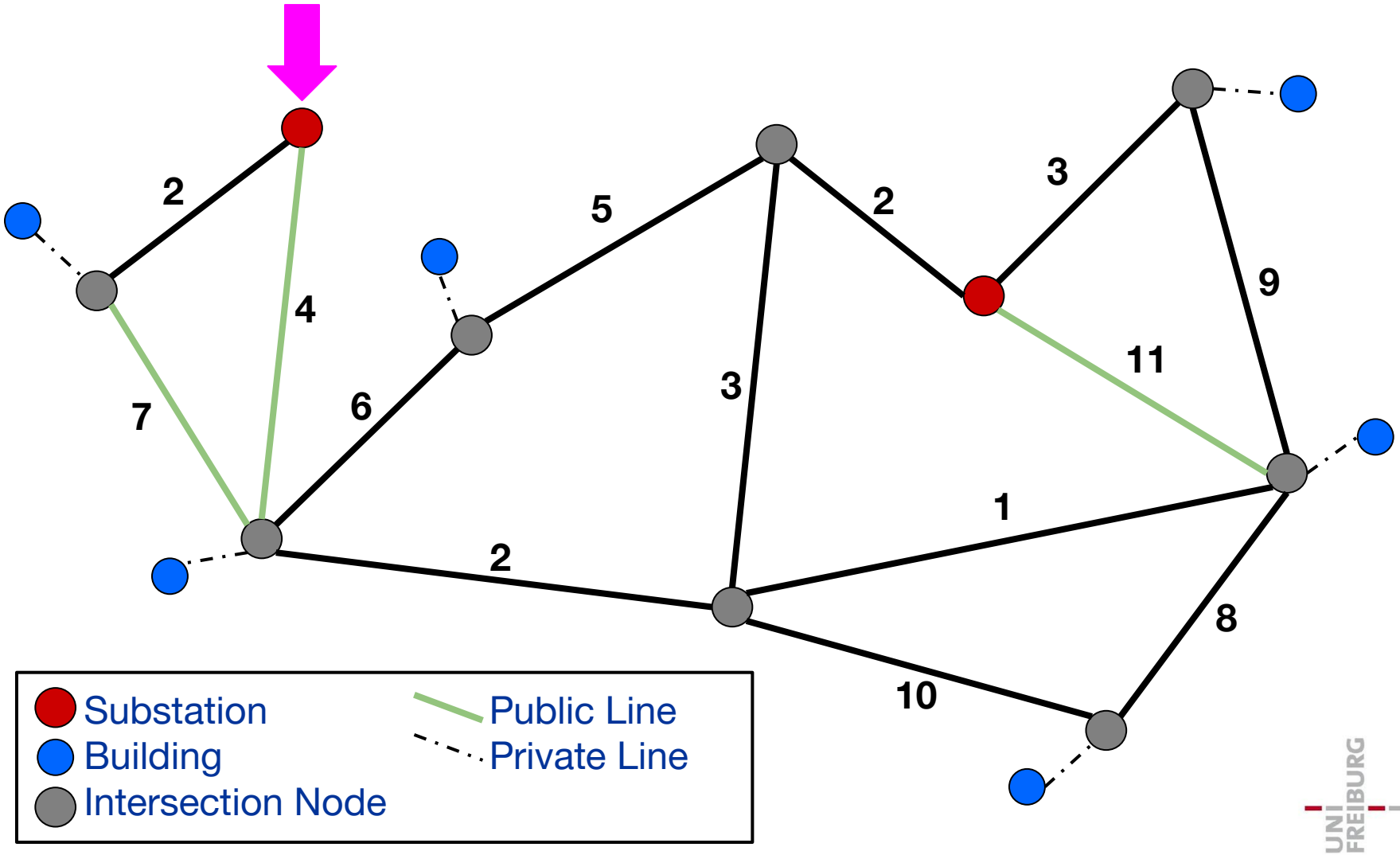
Random SB-Forest



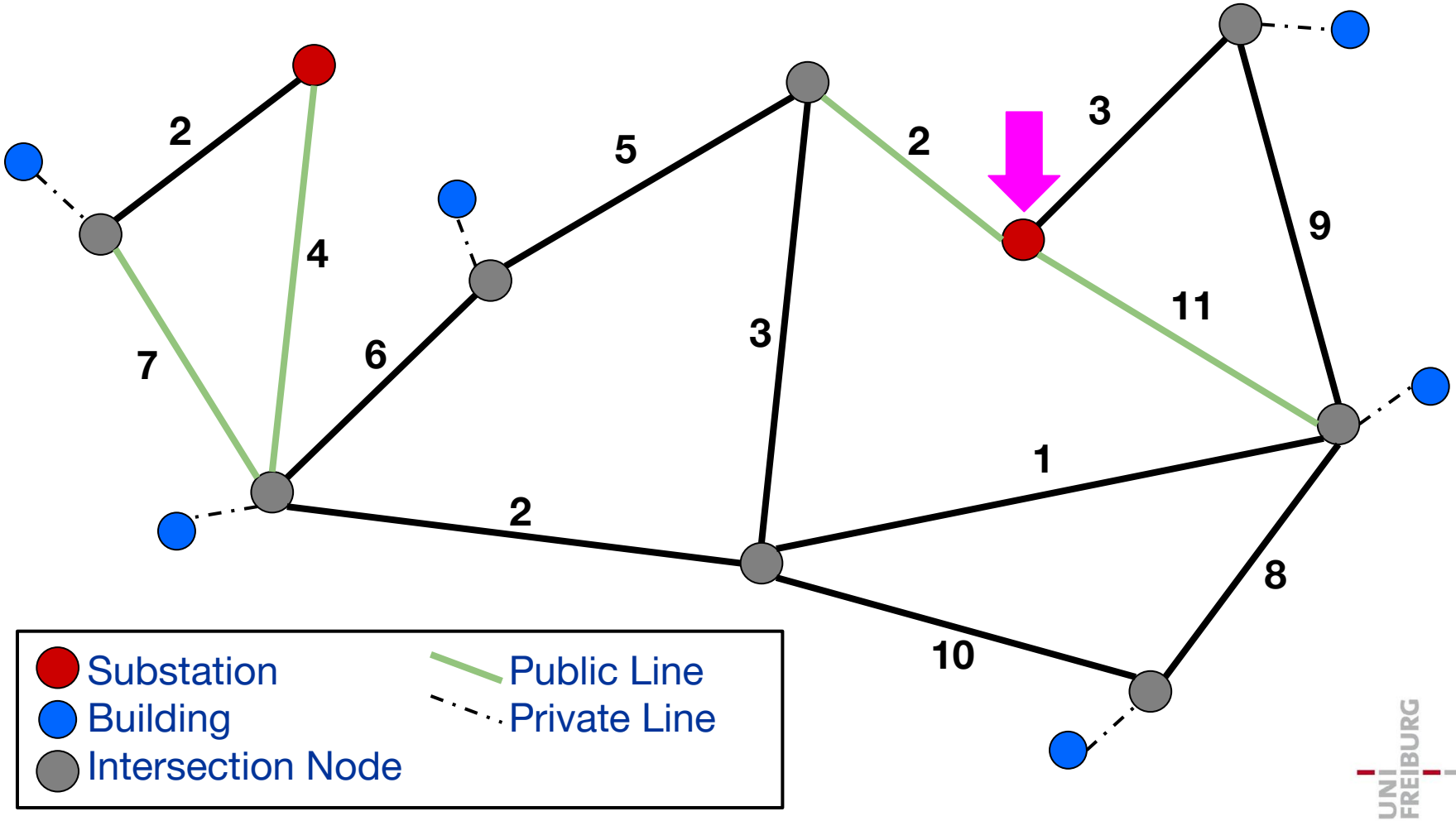
Random SB-Forest



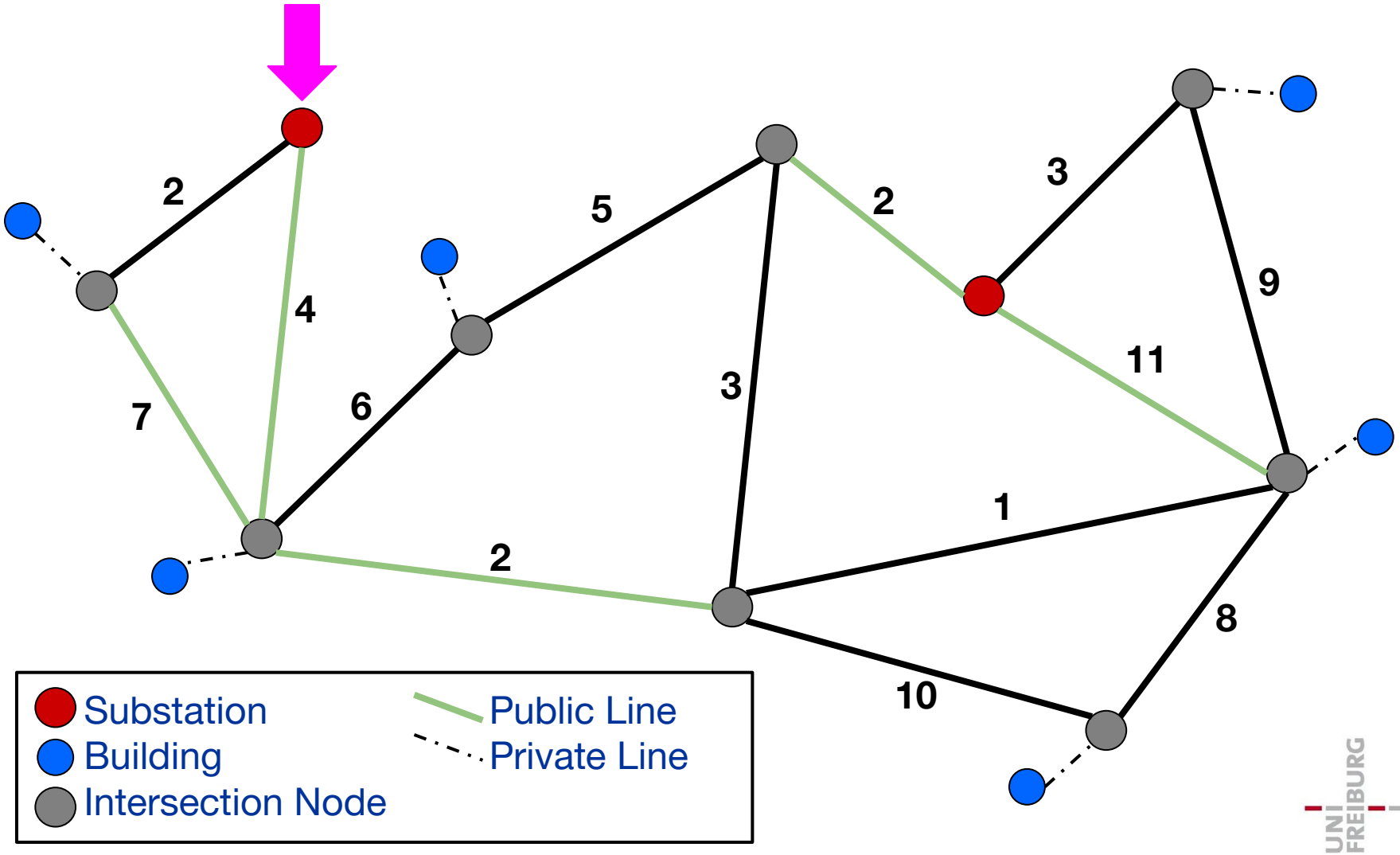
Random SB-Forest



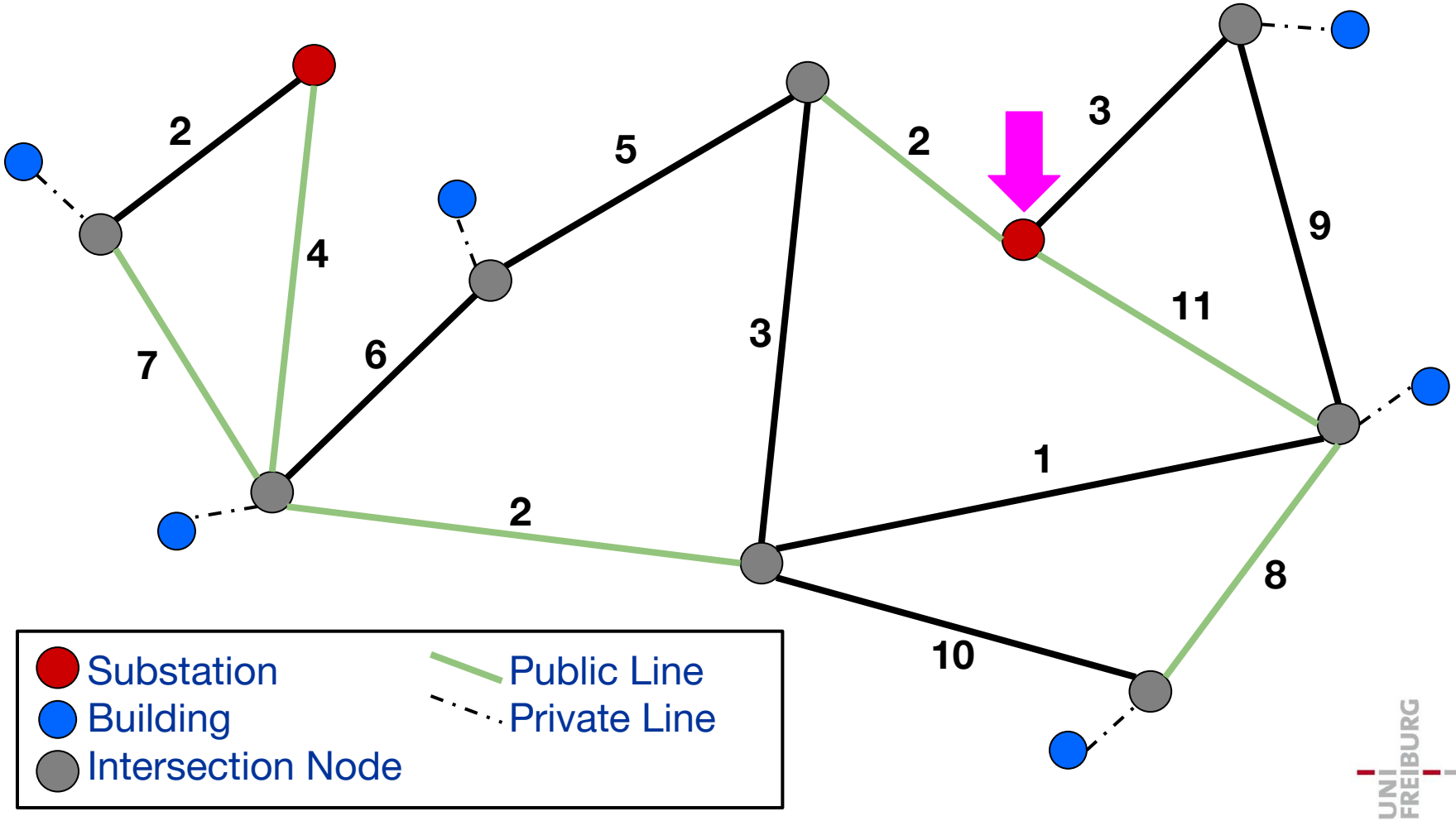
Random SB-Forest



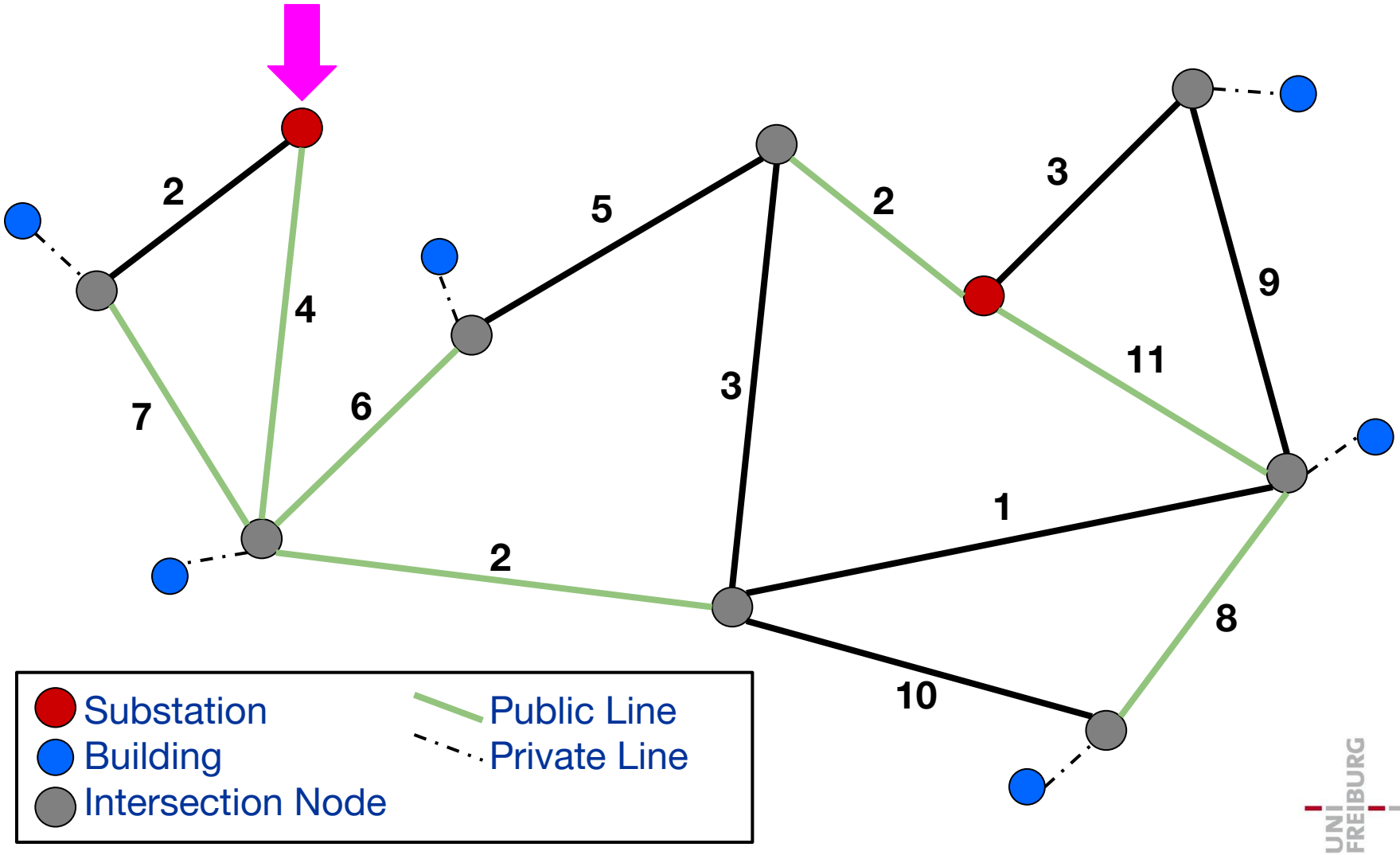
Random SB-Forest



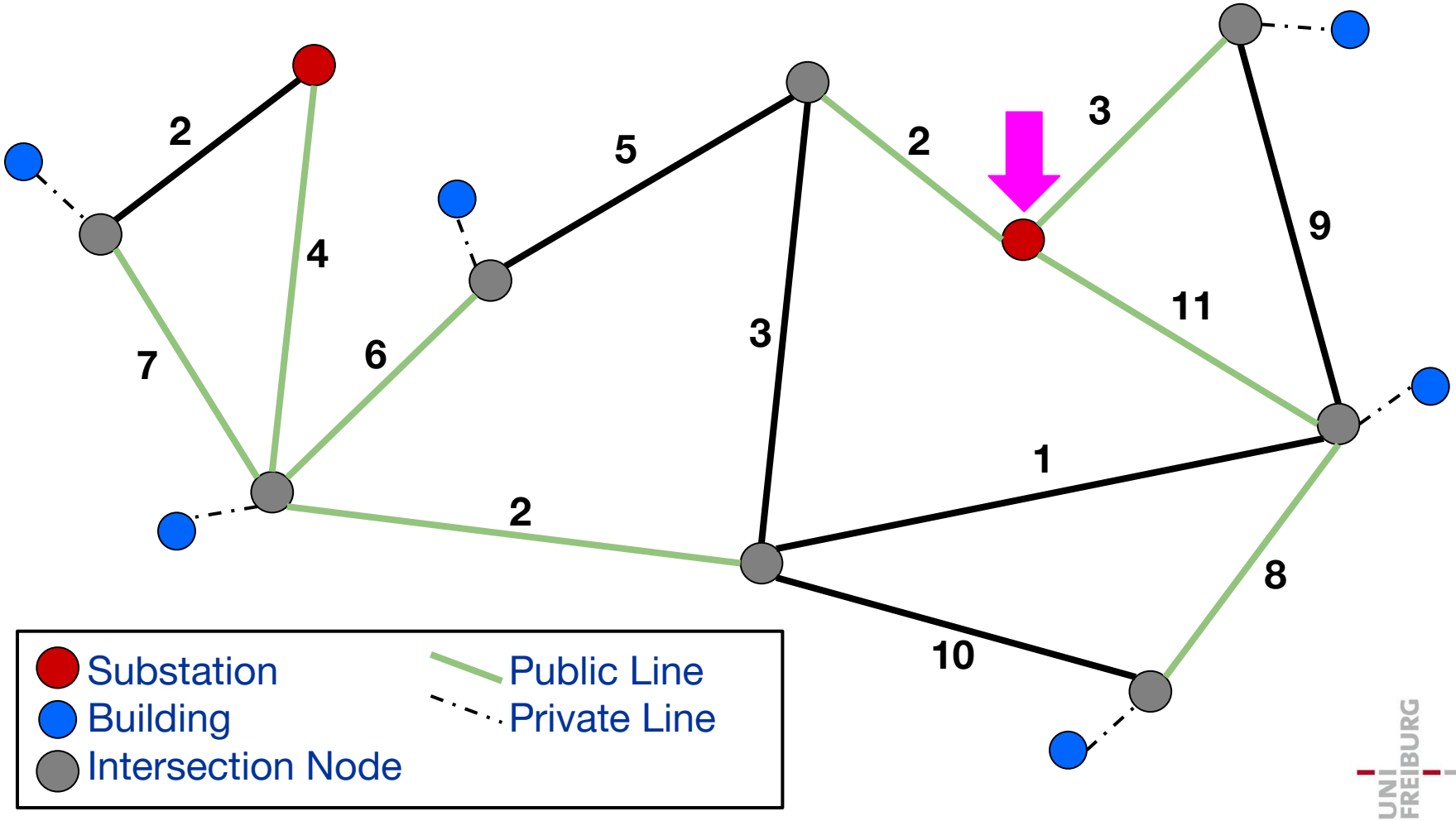
Random SB-Forest



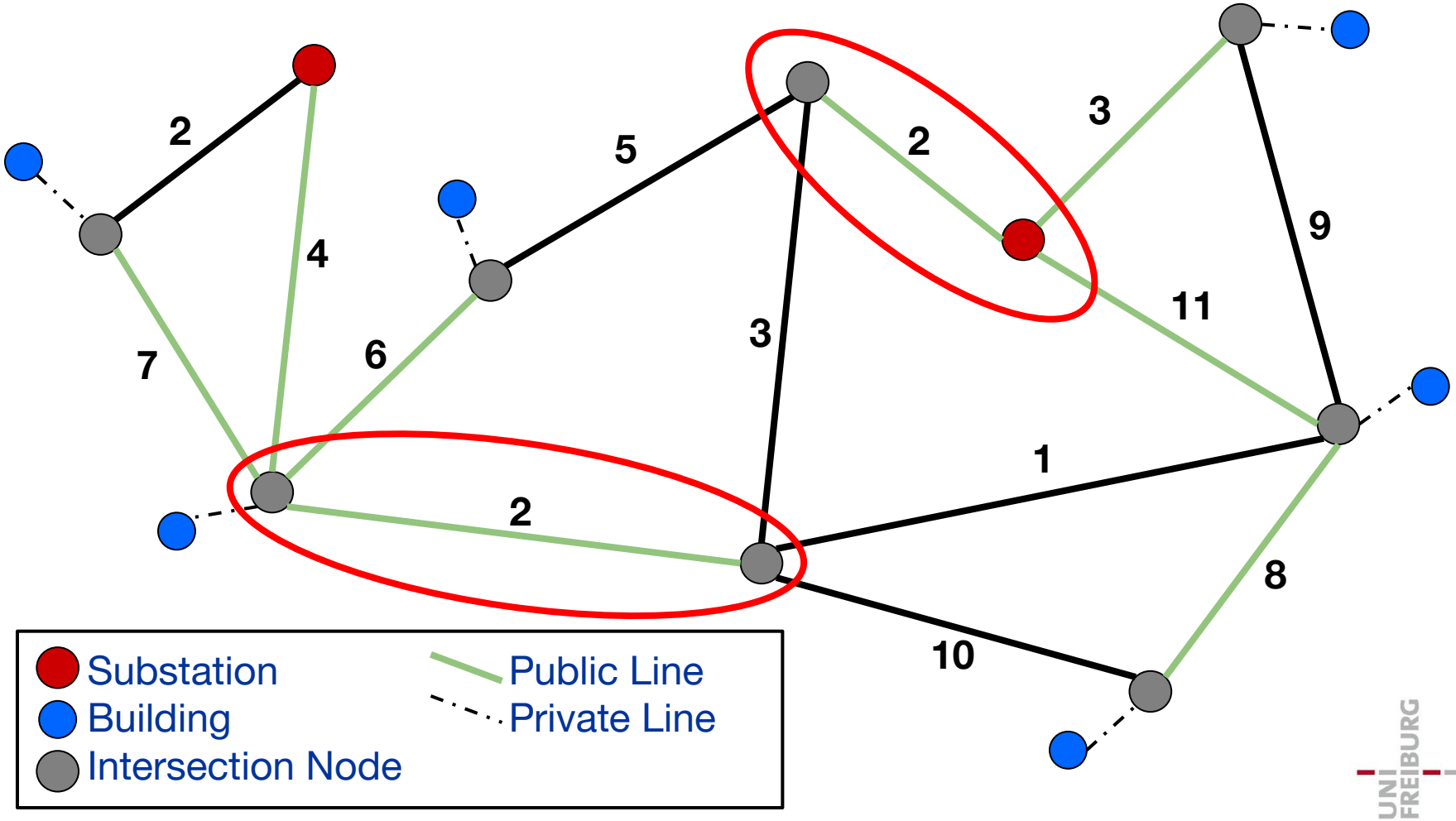
Random SB-Forest



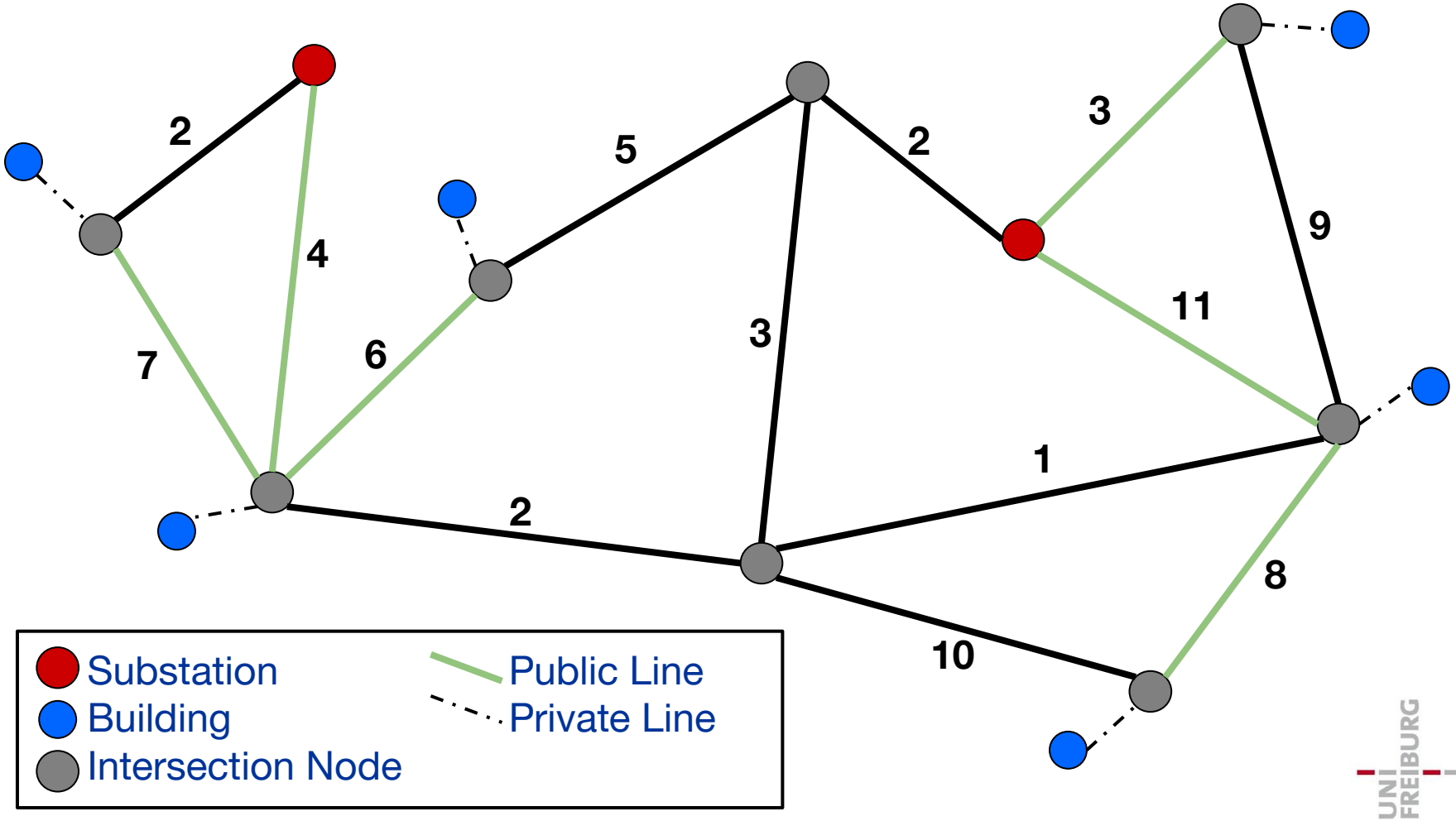
Random SB-Forest



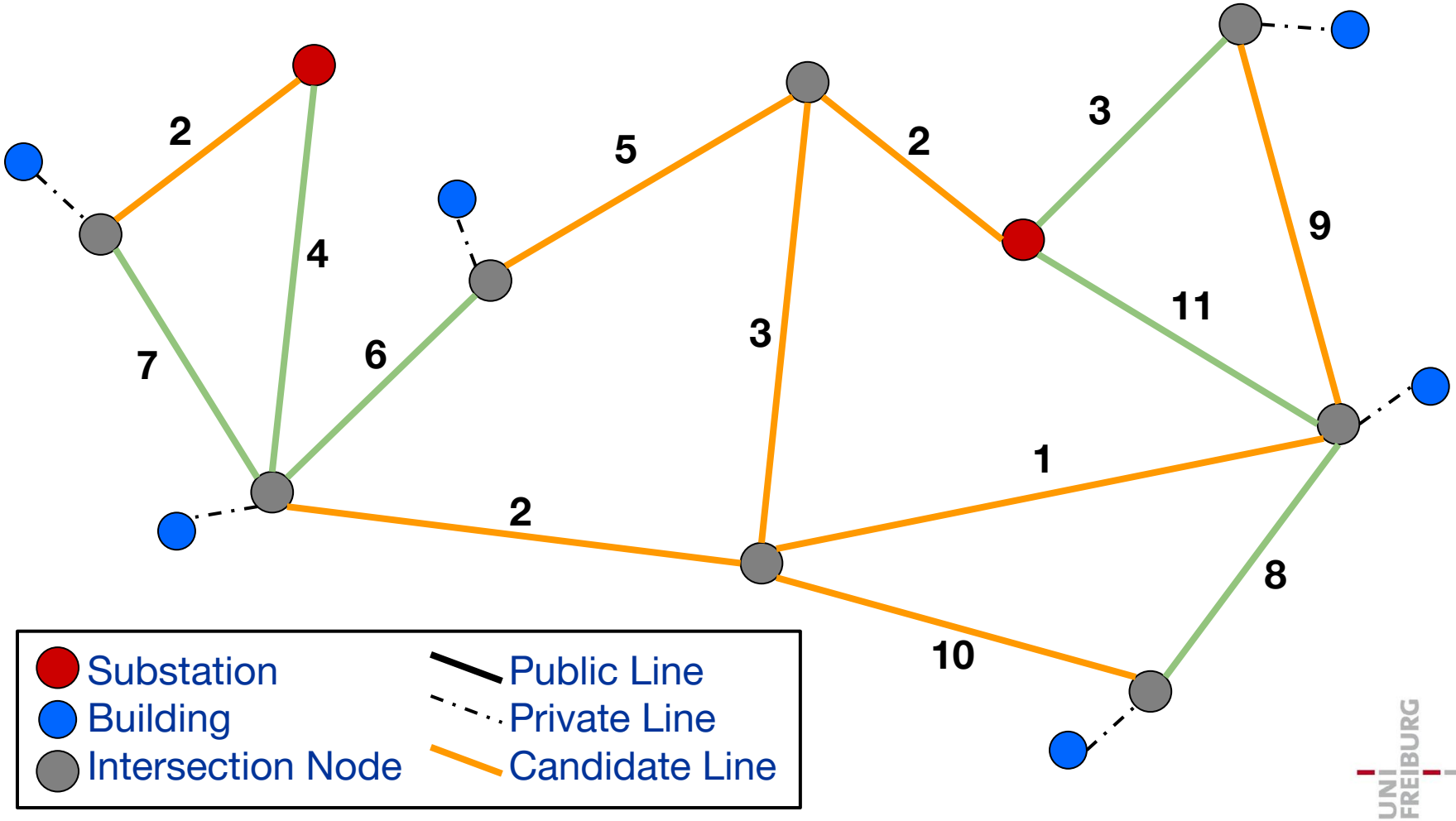
Random SB-Forest



Random SB-Forest



Random SB-Forest

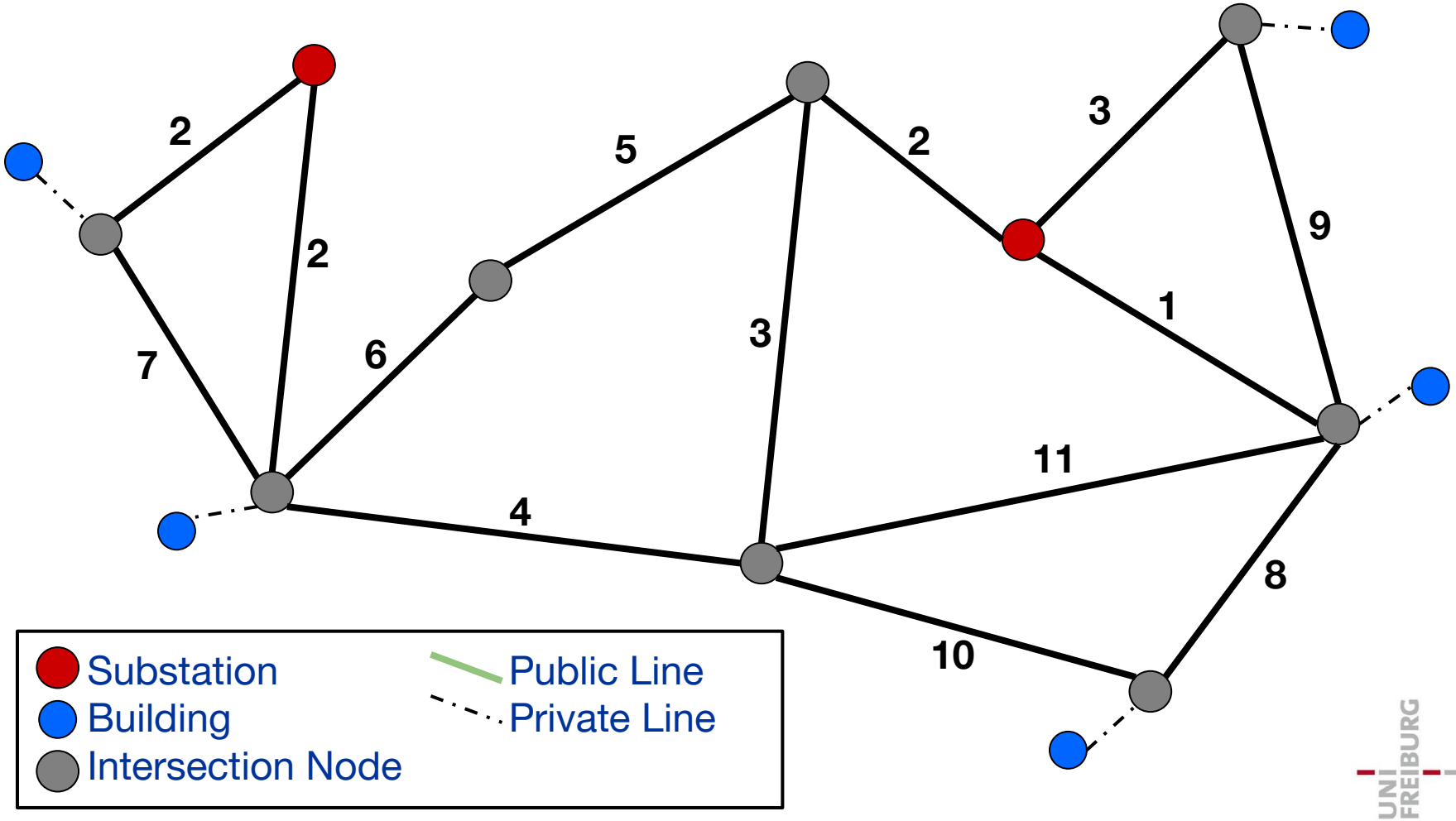


Minimum SB-Forest

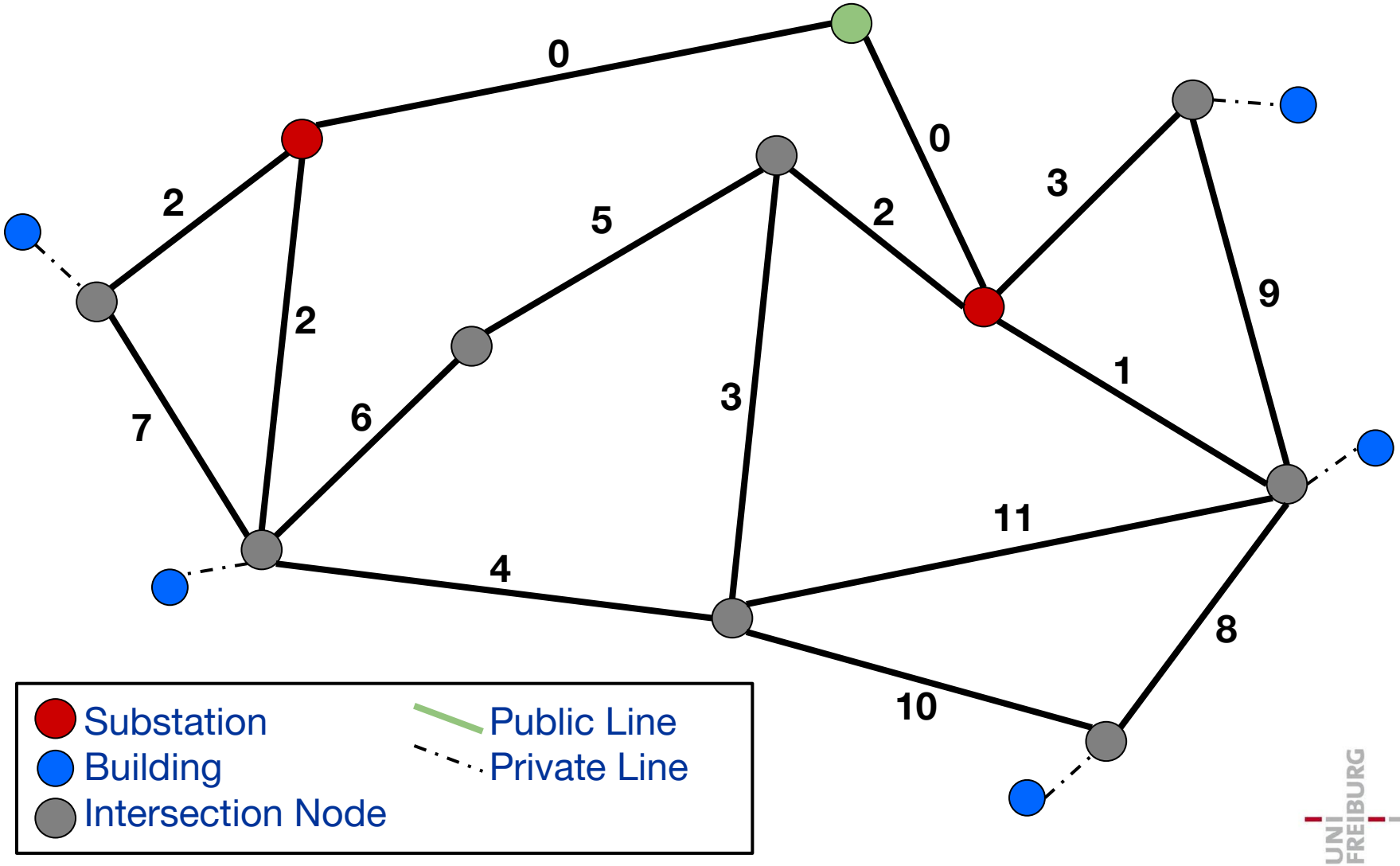
- Add an additional node that is connected by weight 0 to all substation nodes
- Sort the edges by their weights
- Successively add edge if it does not form a cycle (Kruskal's algorithm)
- Trim the spanning forest, to get a sb-forest

Note: Maximum SB-Forest does work the same exact way. The only difference is that we invert the weights while sorting the edges.

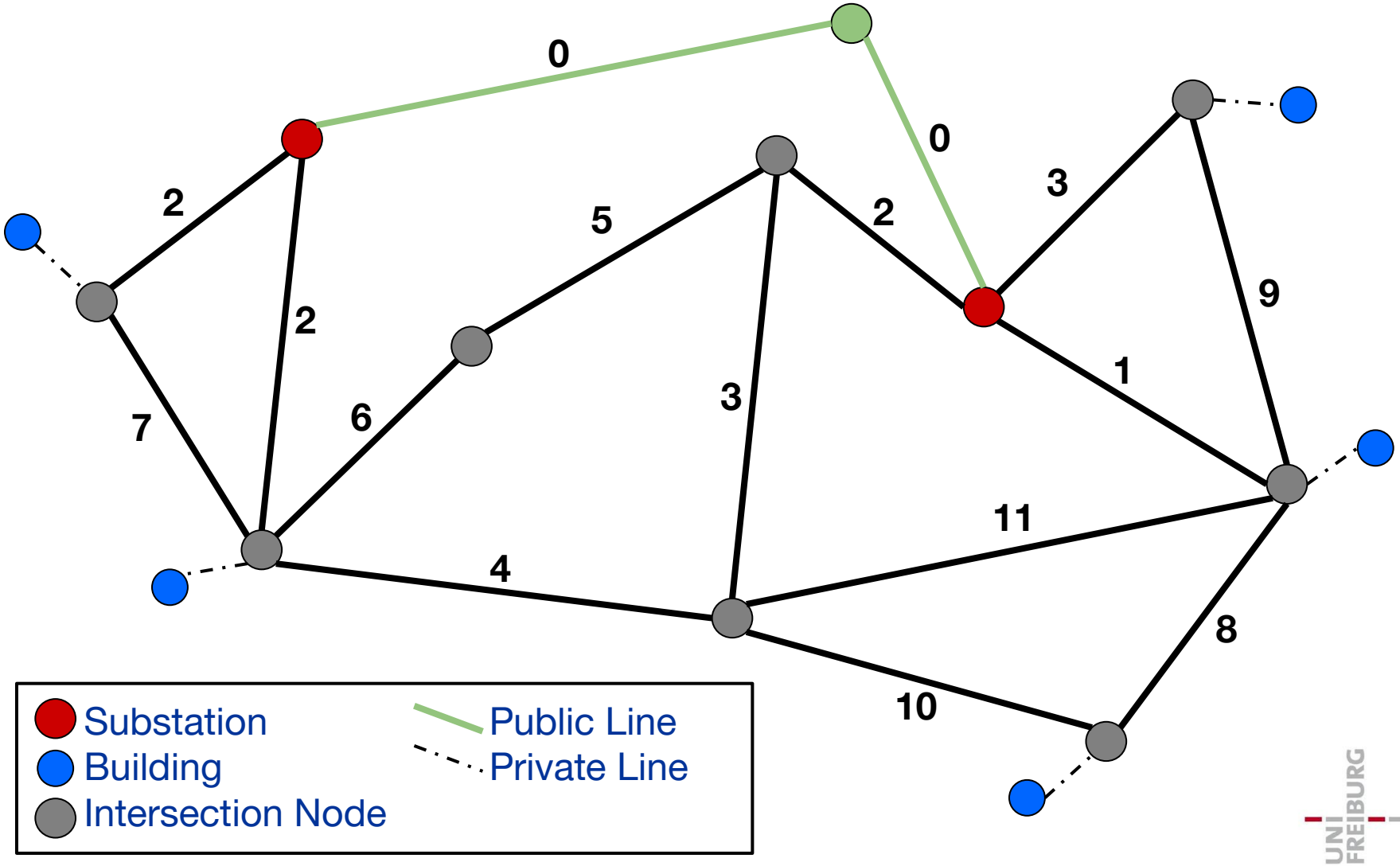
Minimum SB-Forest



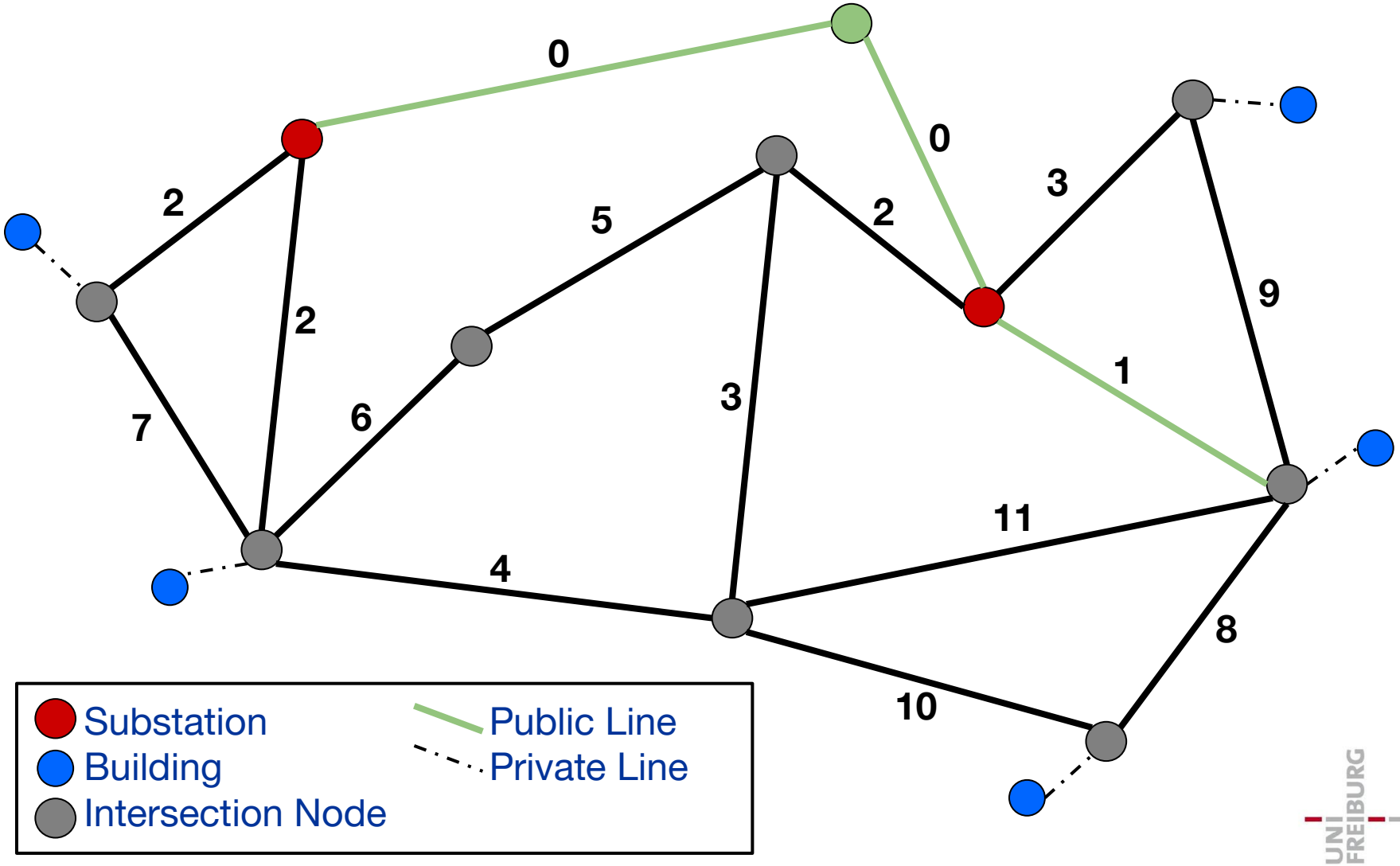
Minimum SB-Forest



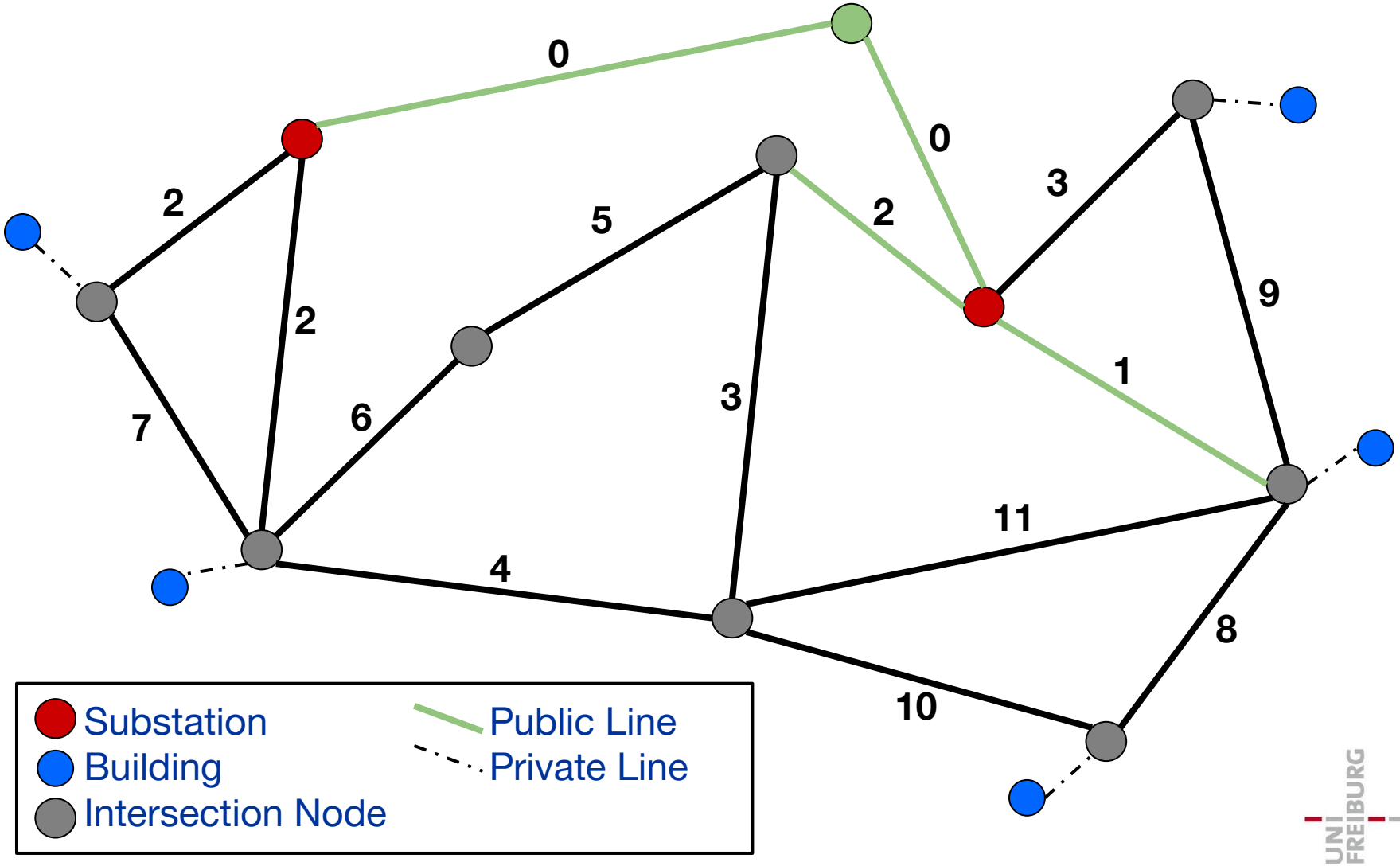
Minimum SB-Forest



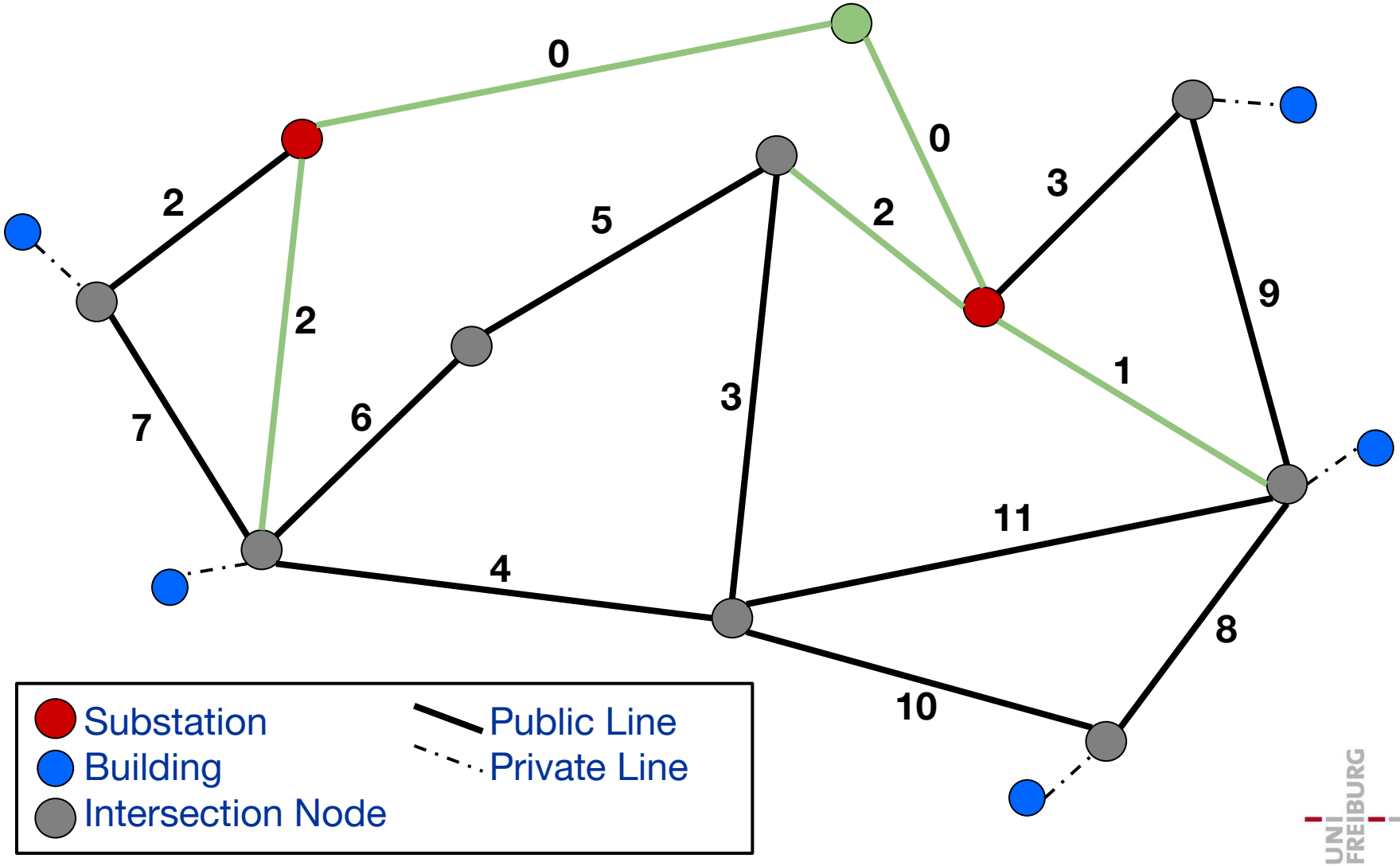
Minimum SB-Forest



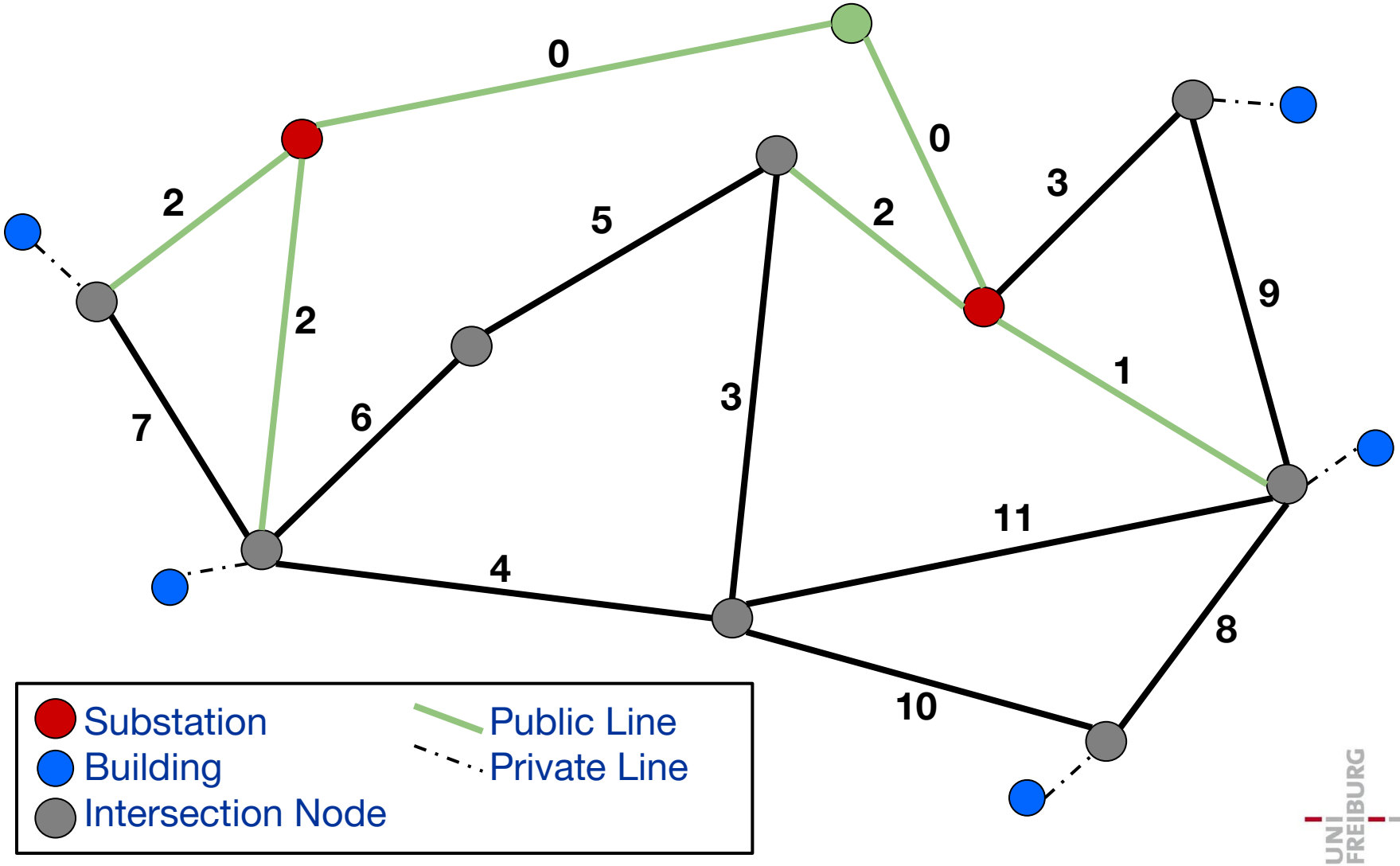
Minimum SB-Forest



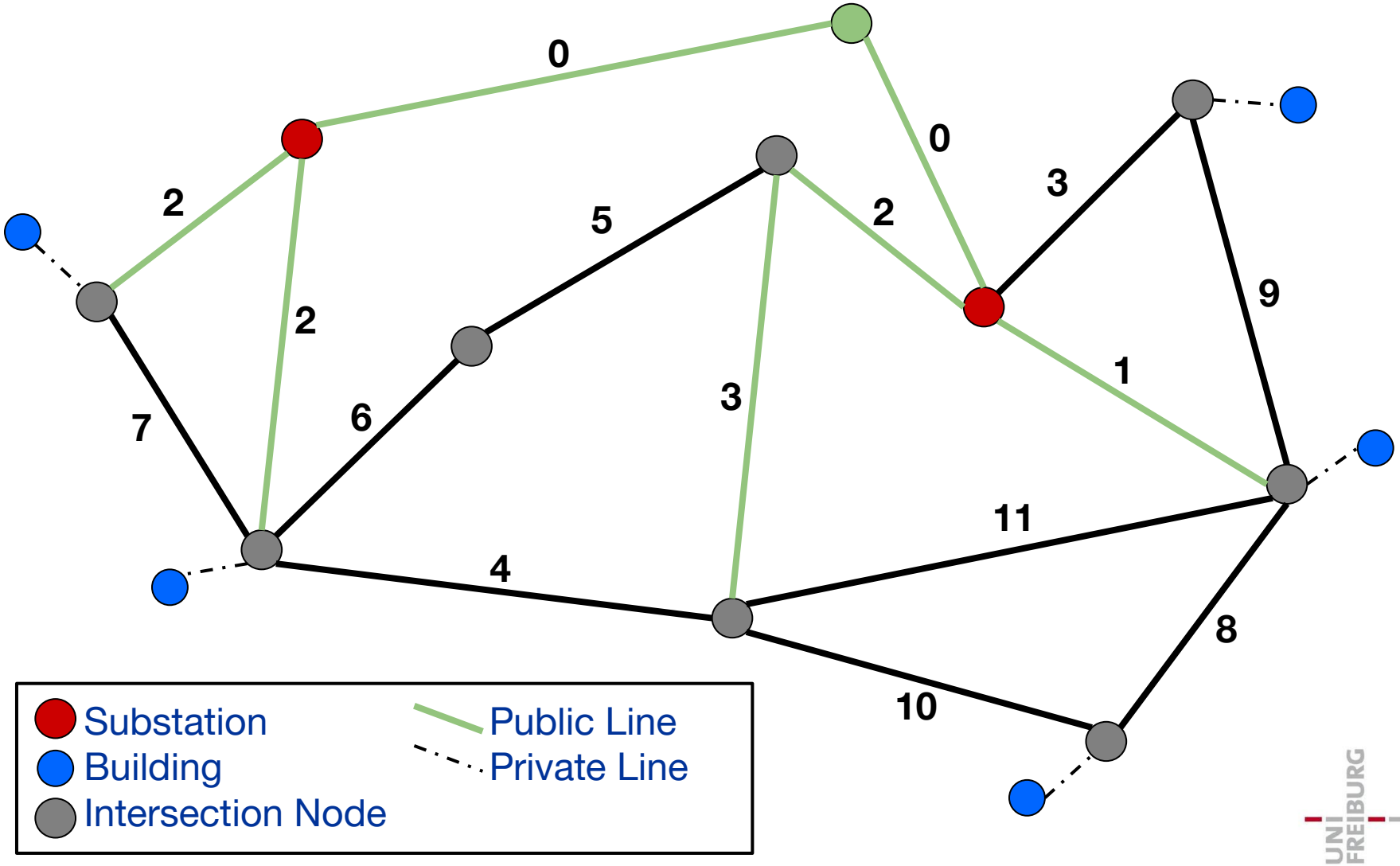
Minimum SB-Forest



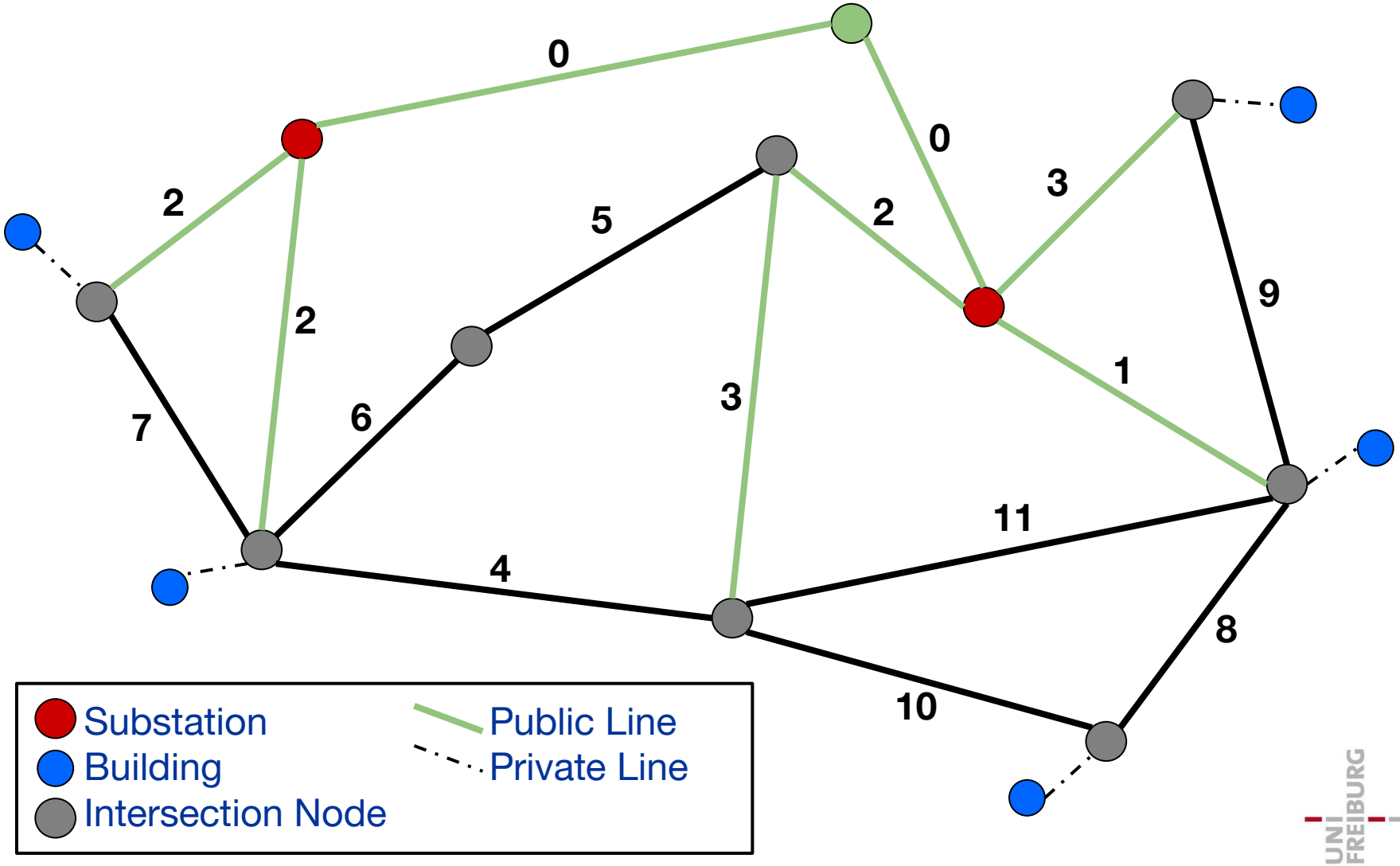
Minimum SB-Forest



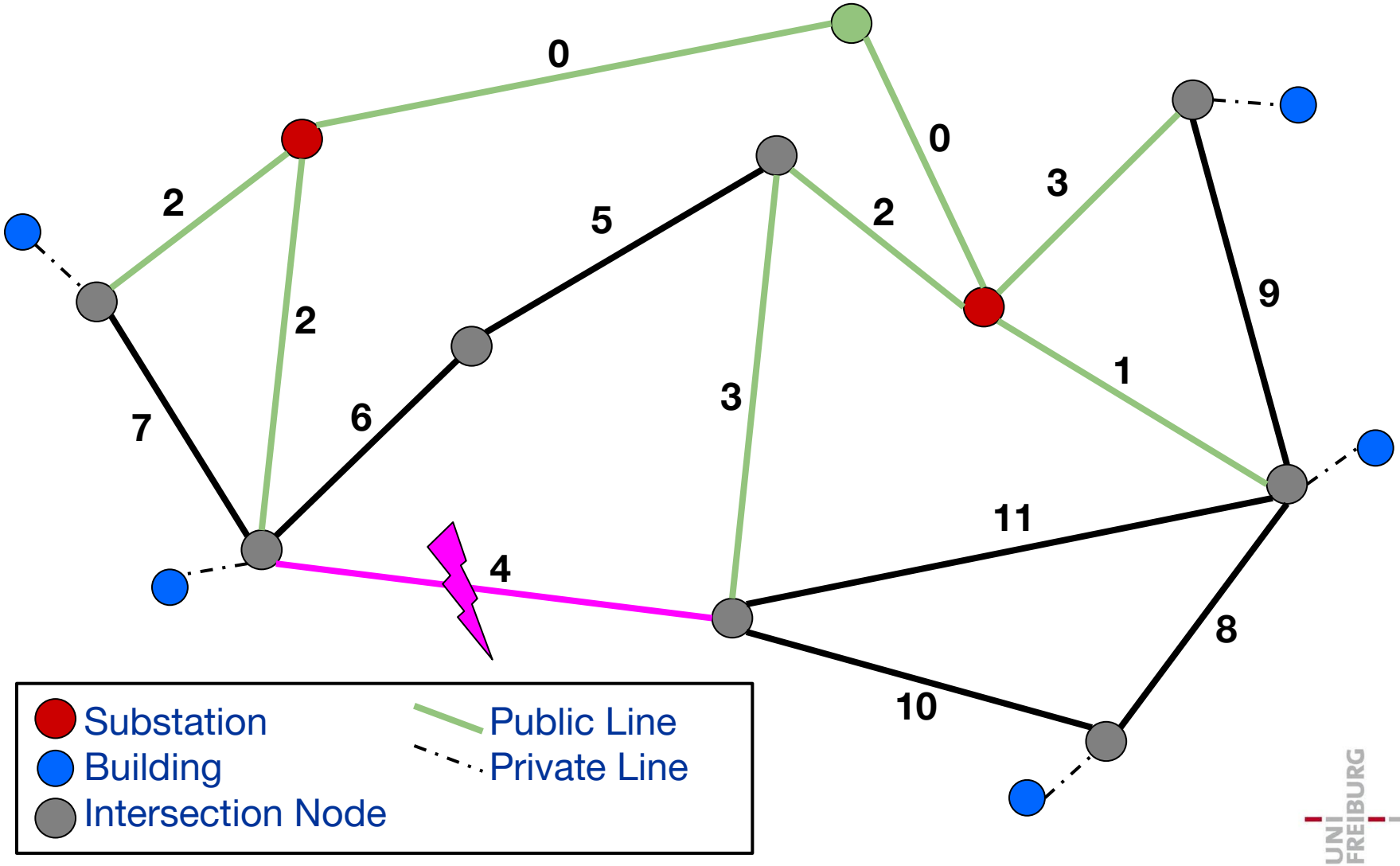
Minimum SB-Forest



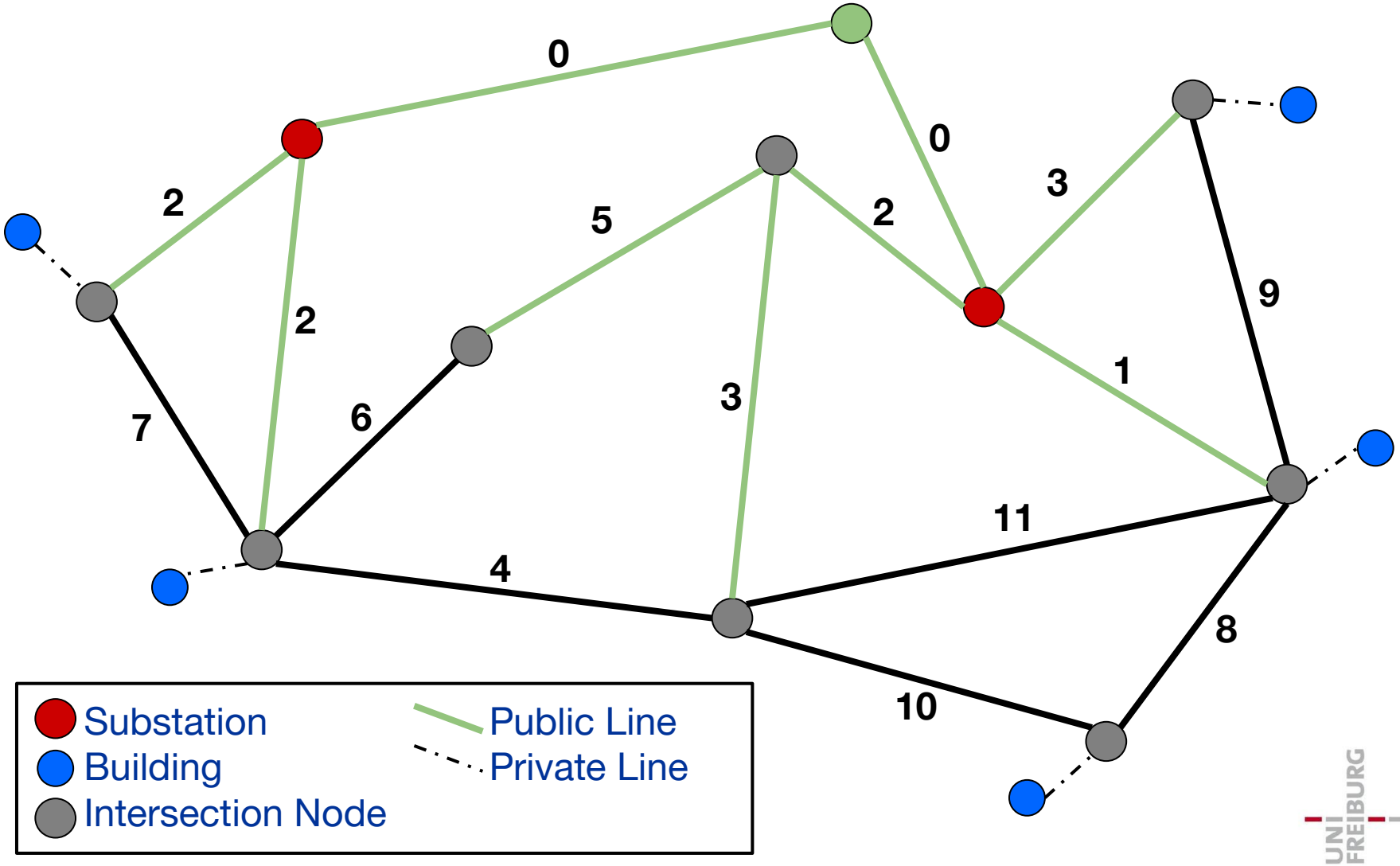
Minimum SB-Forest



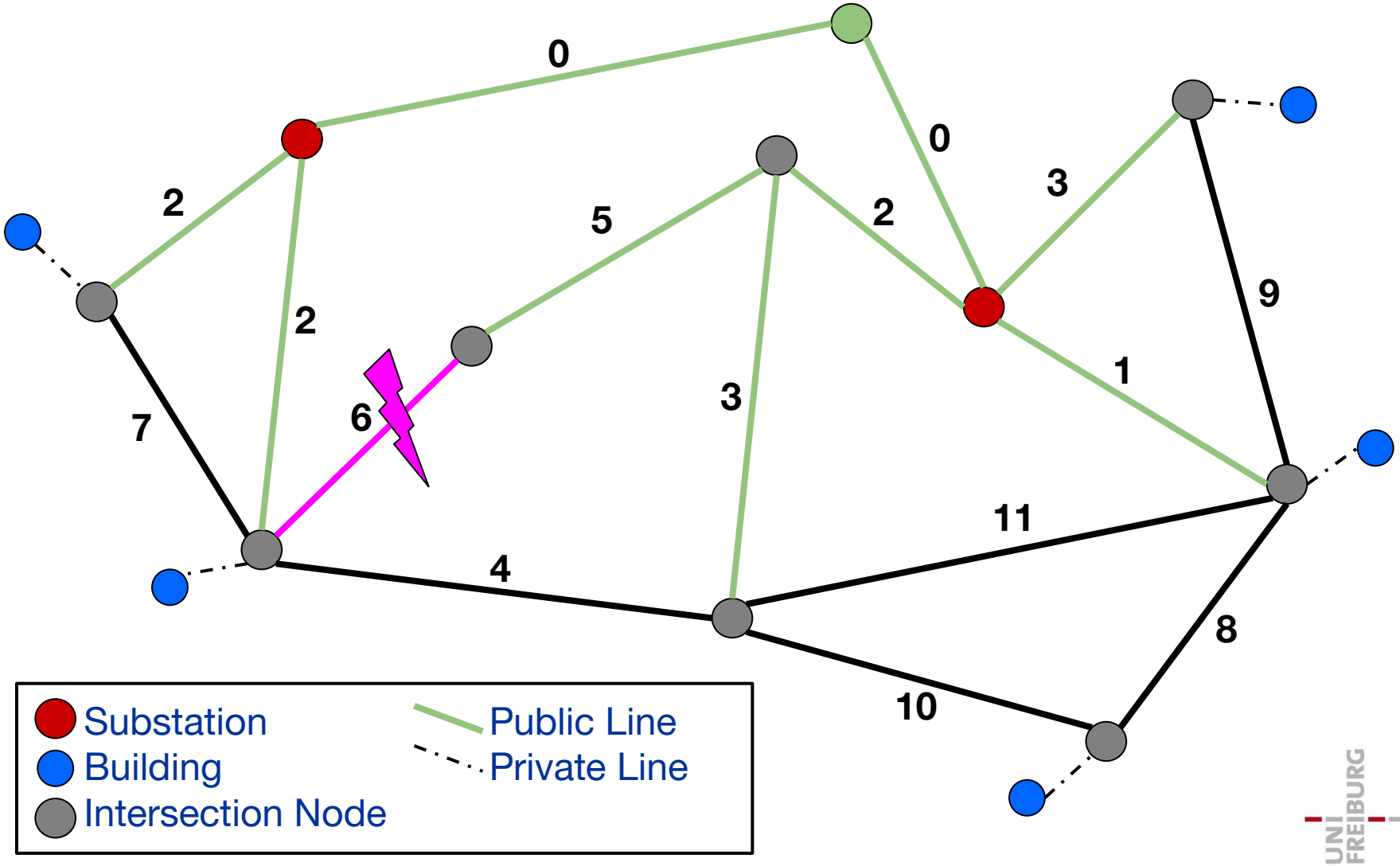
Minimum SB-Forest



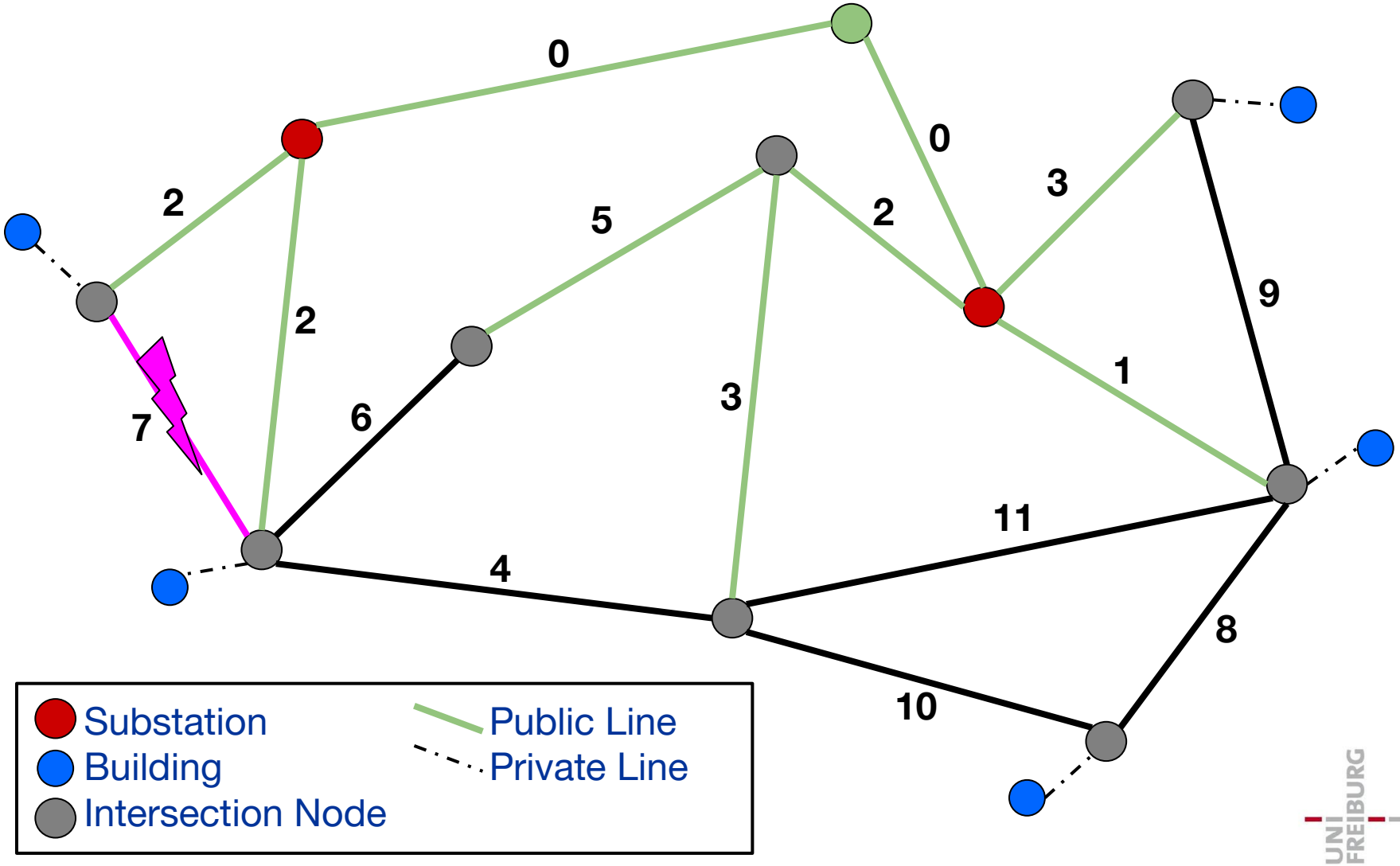
Minimum SB-Forest



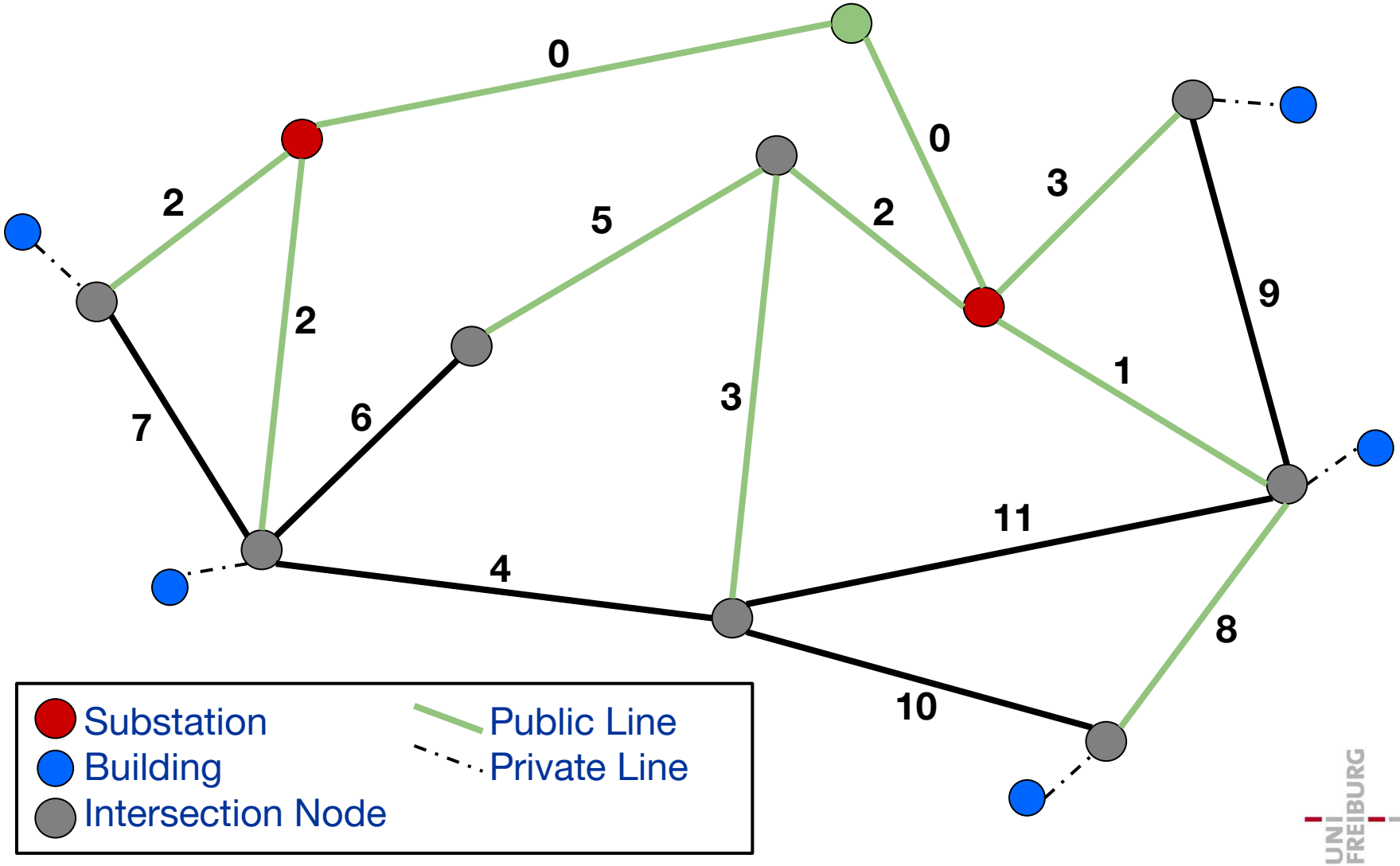
Minimum SB-Forest



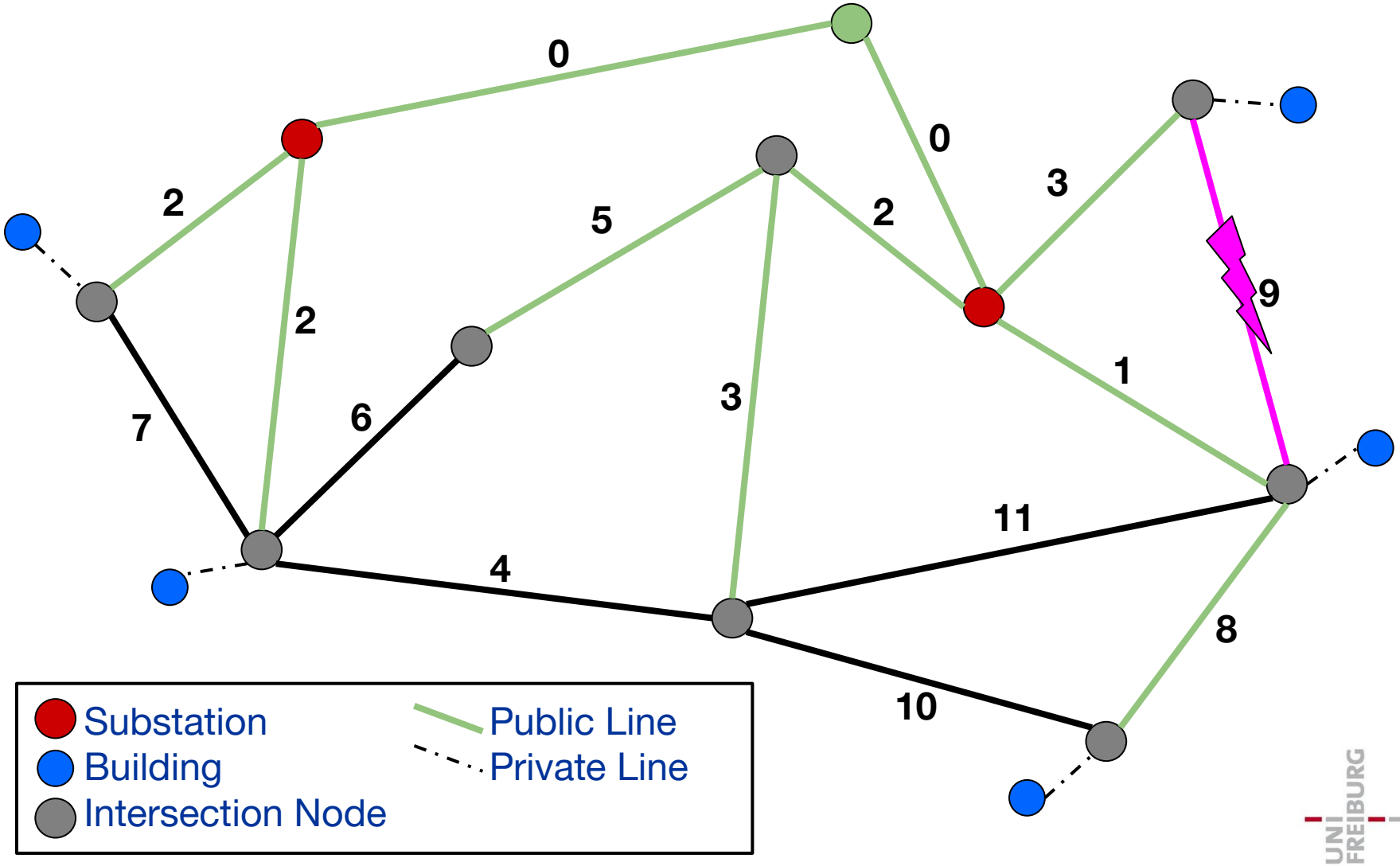
Minimum SB-Forest



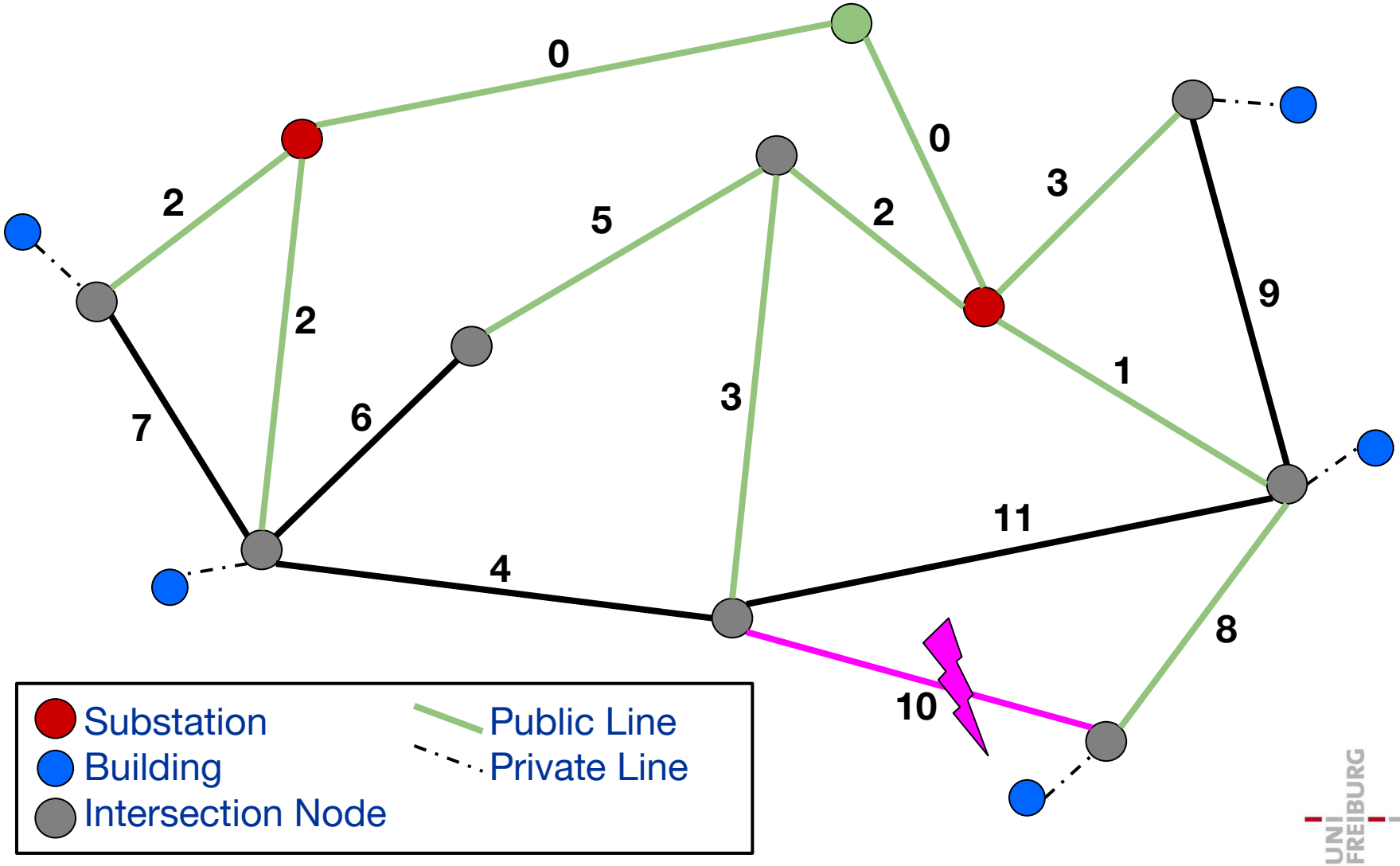
Minimum SB-Forest



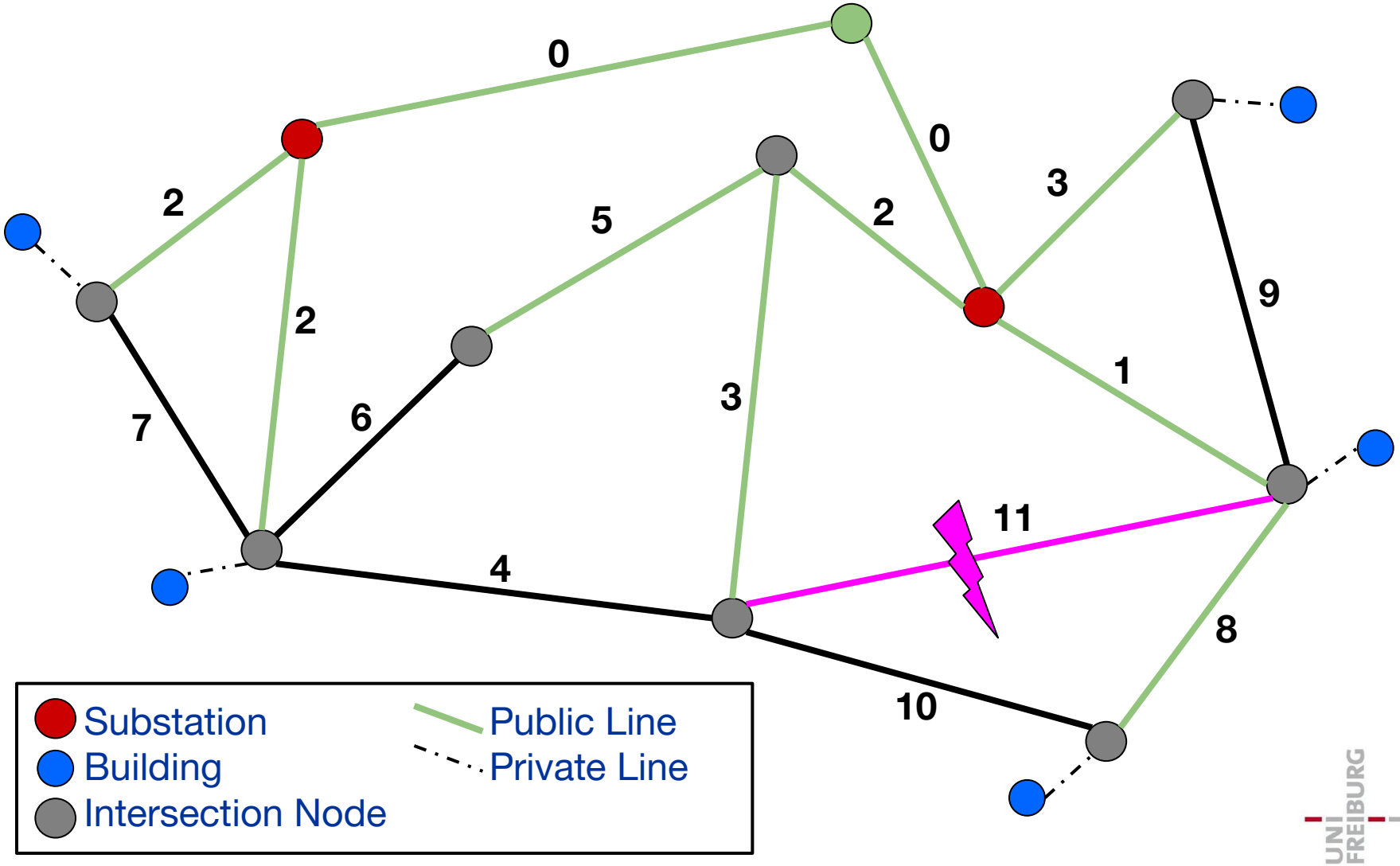
Minimum SB-Forest



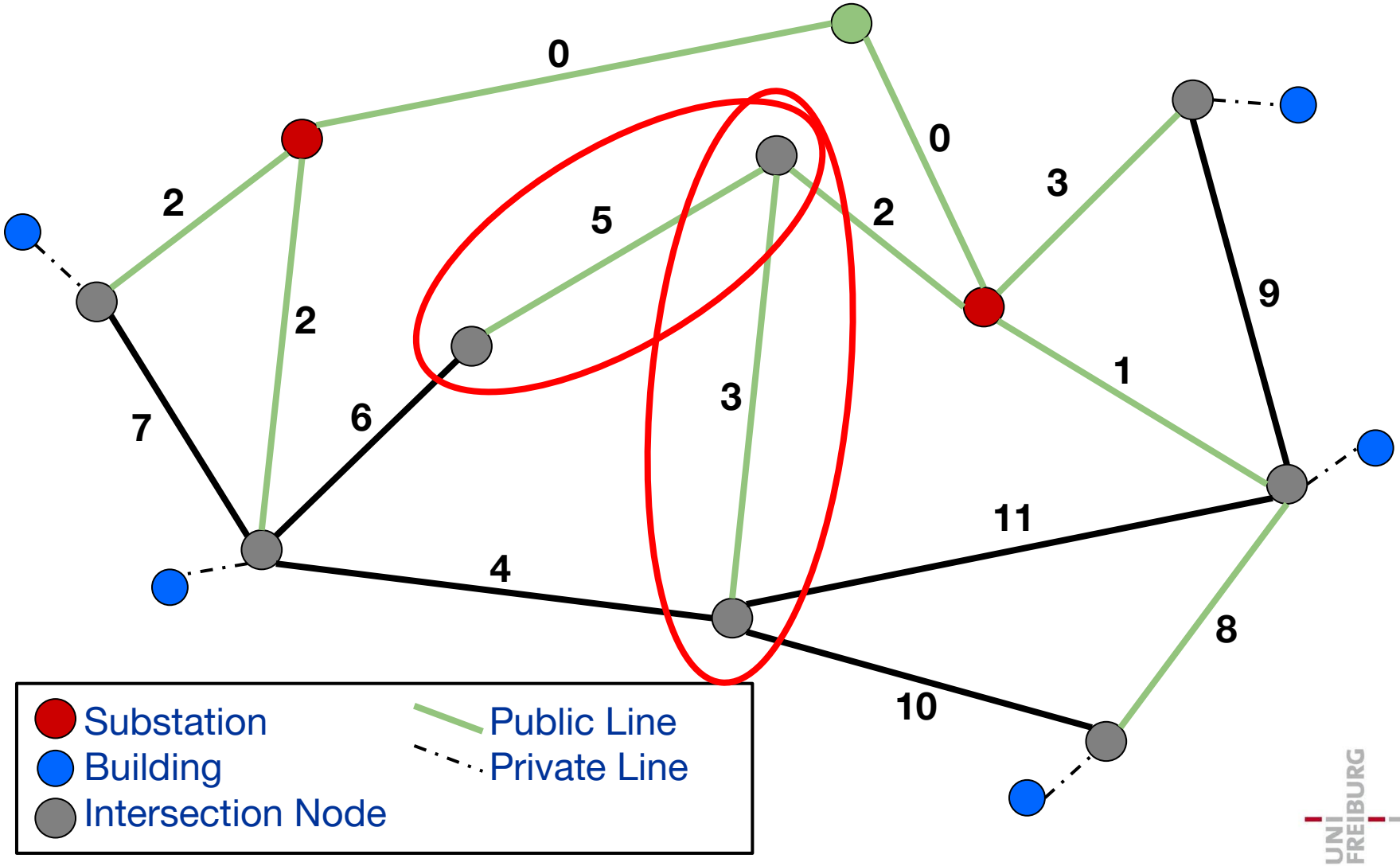
Minimum SB-Forest



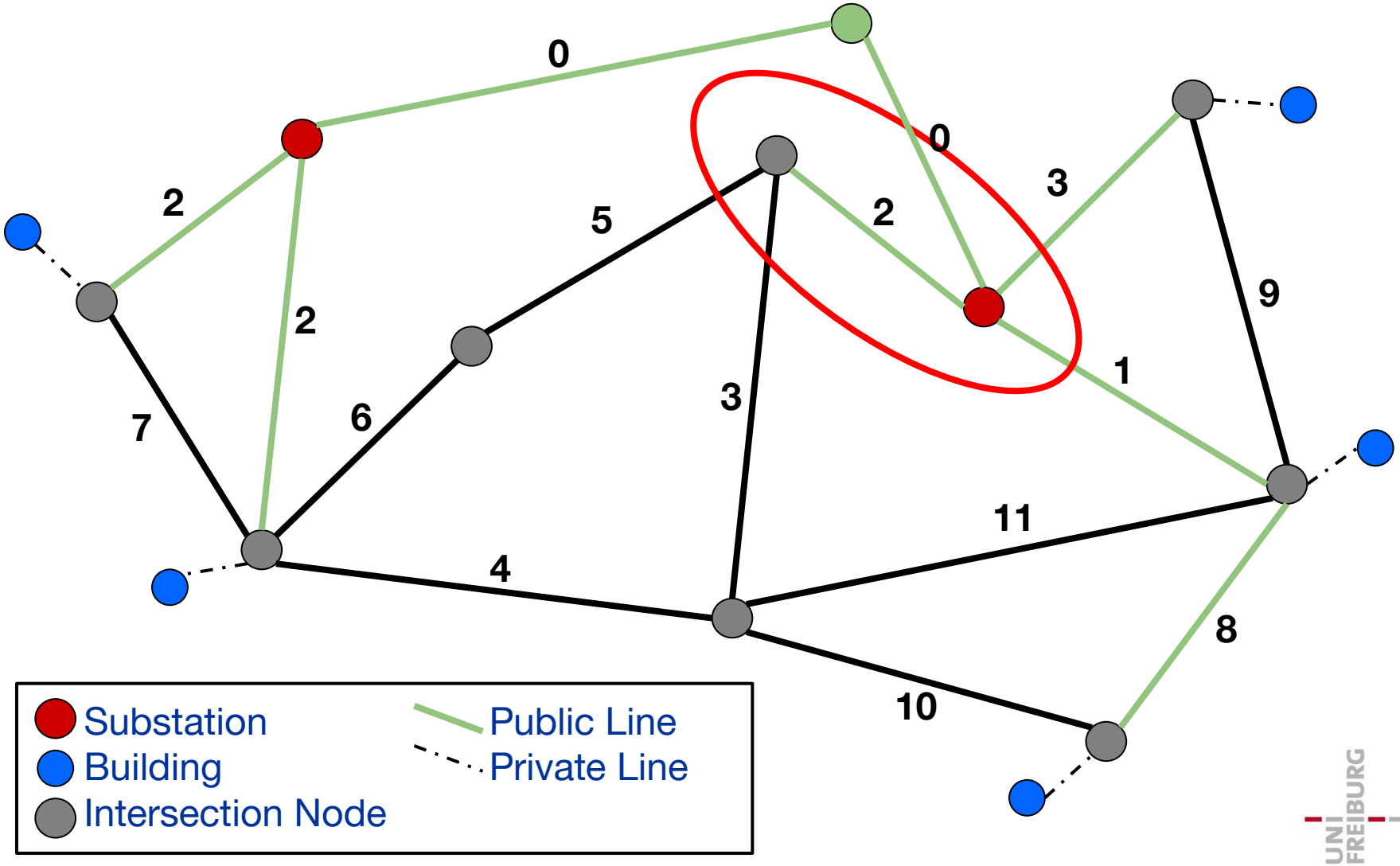
Minimum SB-Forest



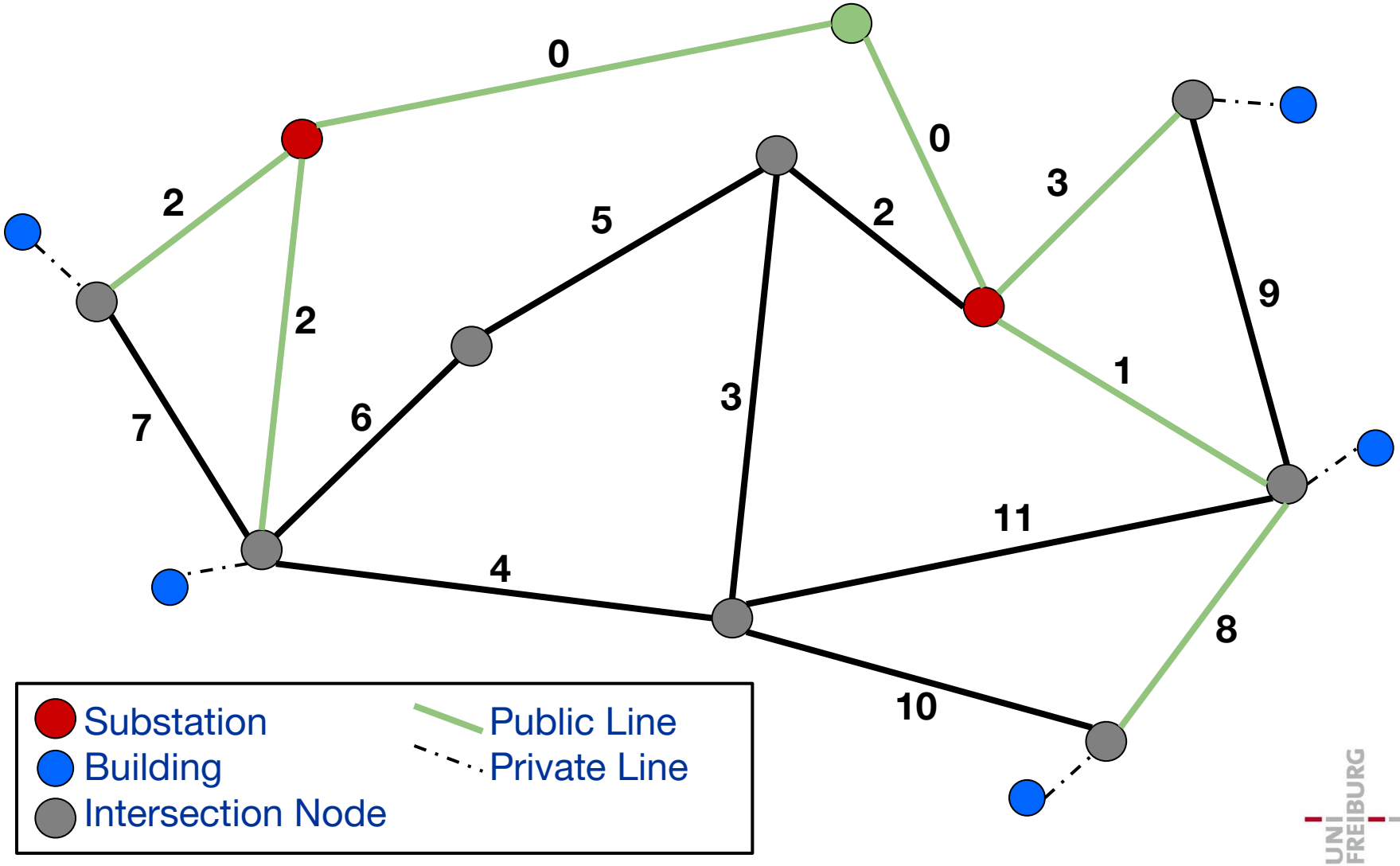
Minimum SB-Forest



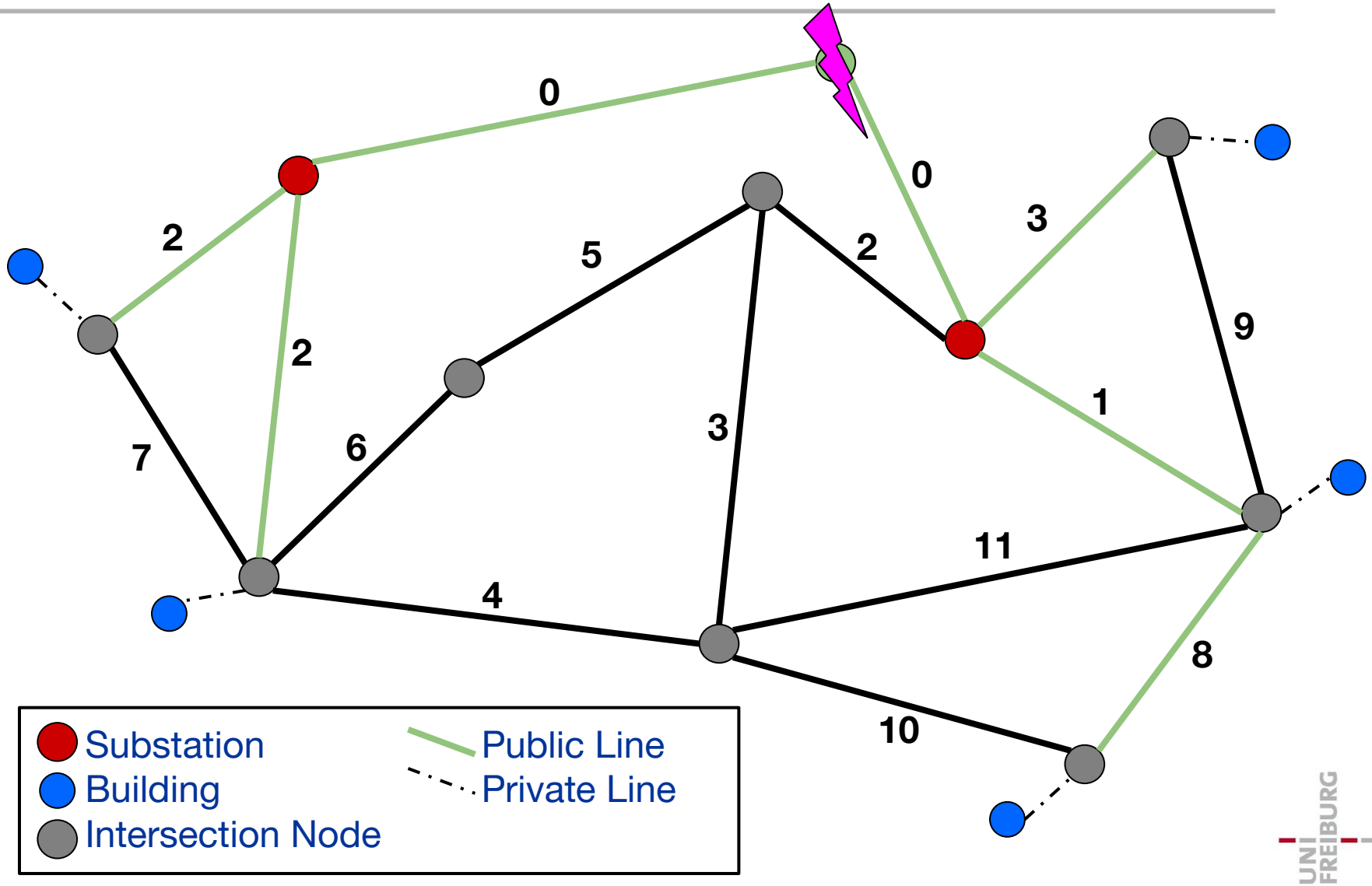
Minimum SB-Forest



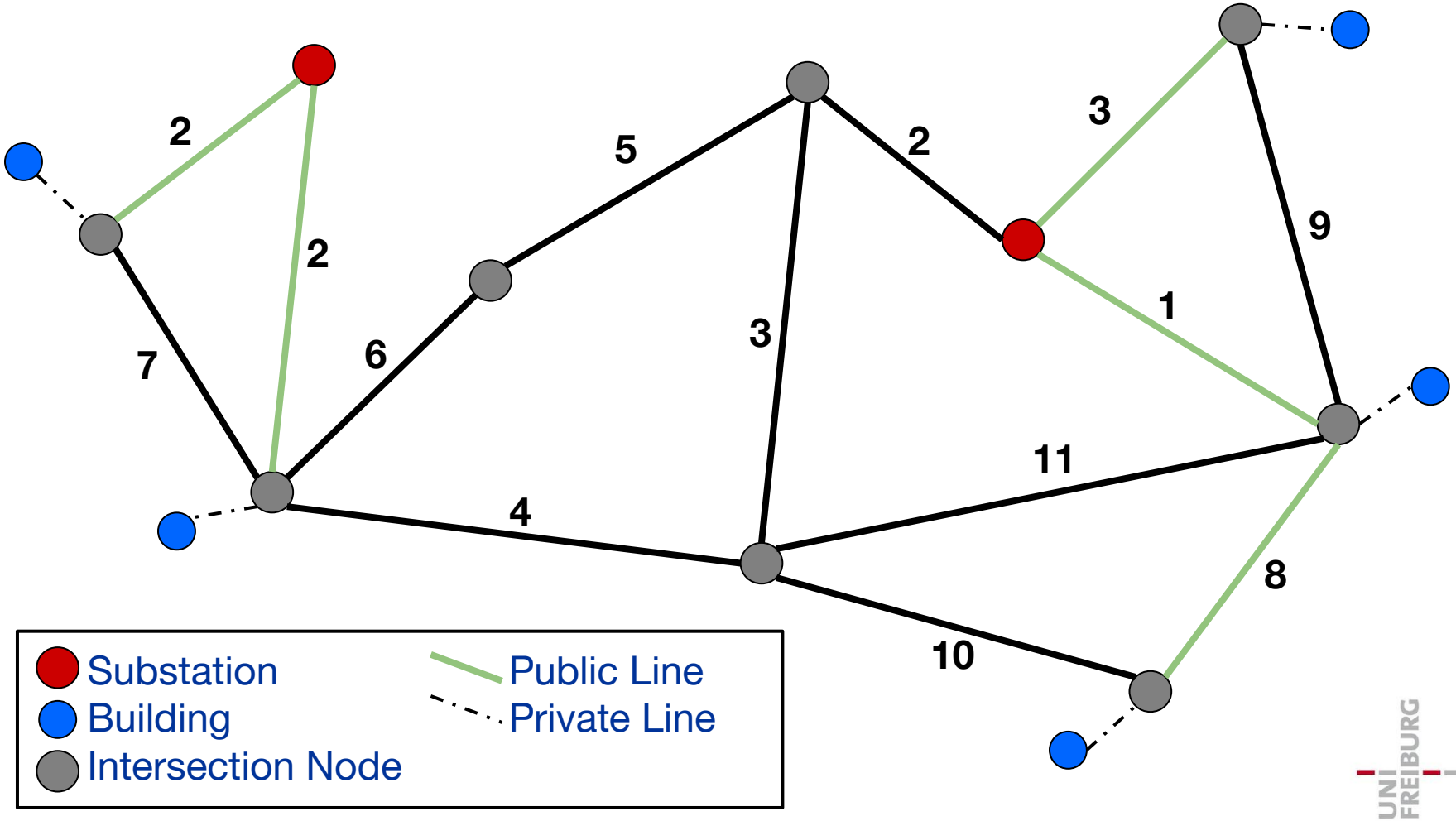
Minimum SB-Forest



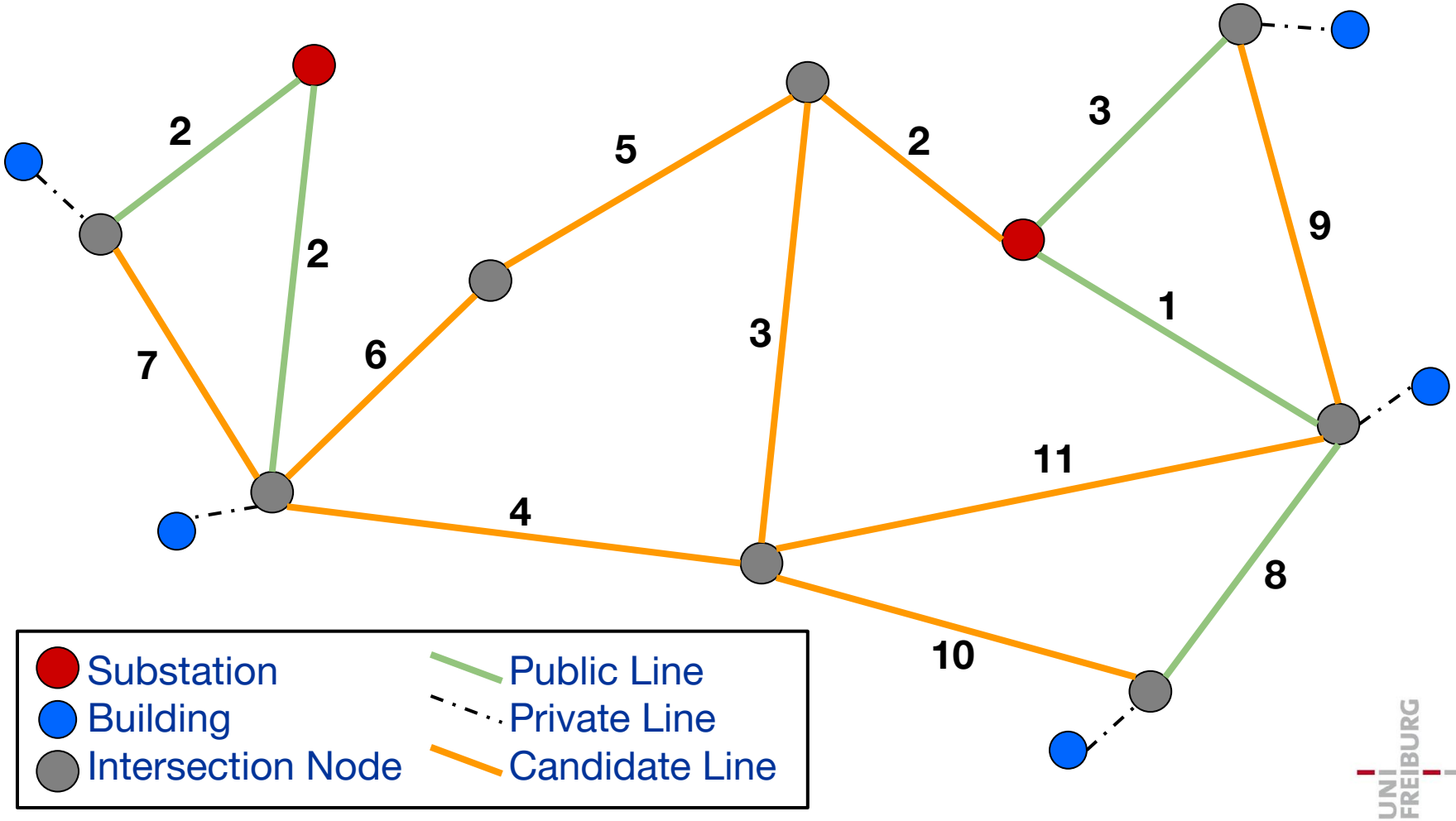
Minimum SB-Forest



Minimum SB-Forest



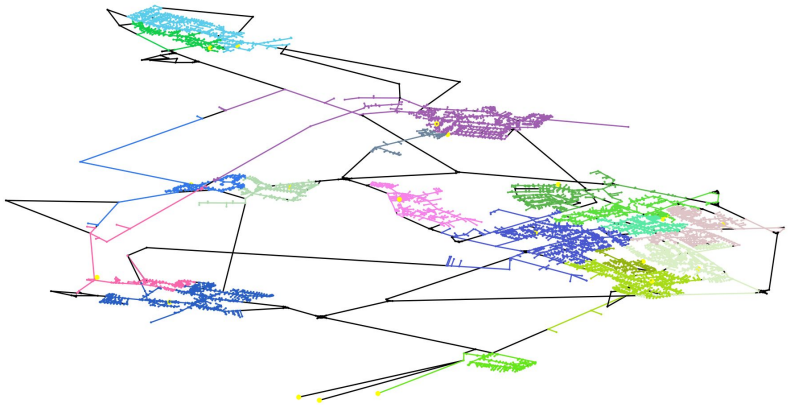
Minimum SB-Forest



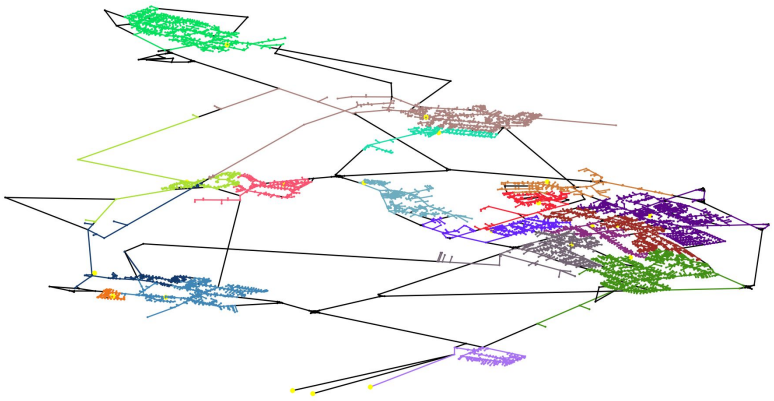
Output as CSV

- Currently installed grid aswell as new candidate lines are returned as a folder of CSV files
- Meet the PyPSA standard
- Ready to be used by powerflow

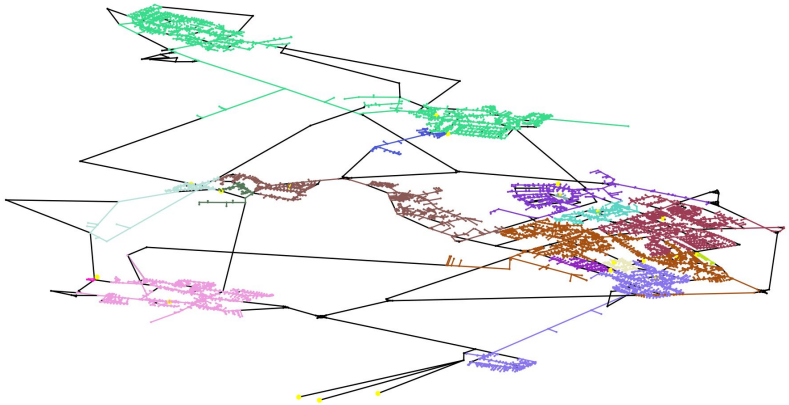
Q&A



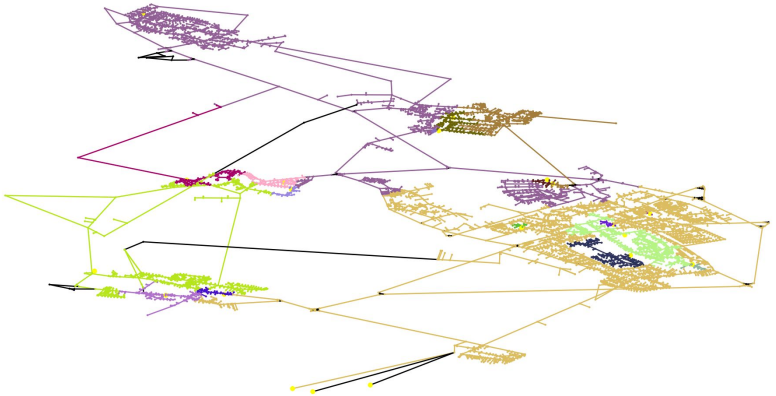
Shortest Path SB-Forest



Random SB-Forest



Minimum SB-Forest

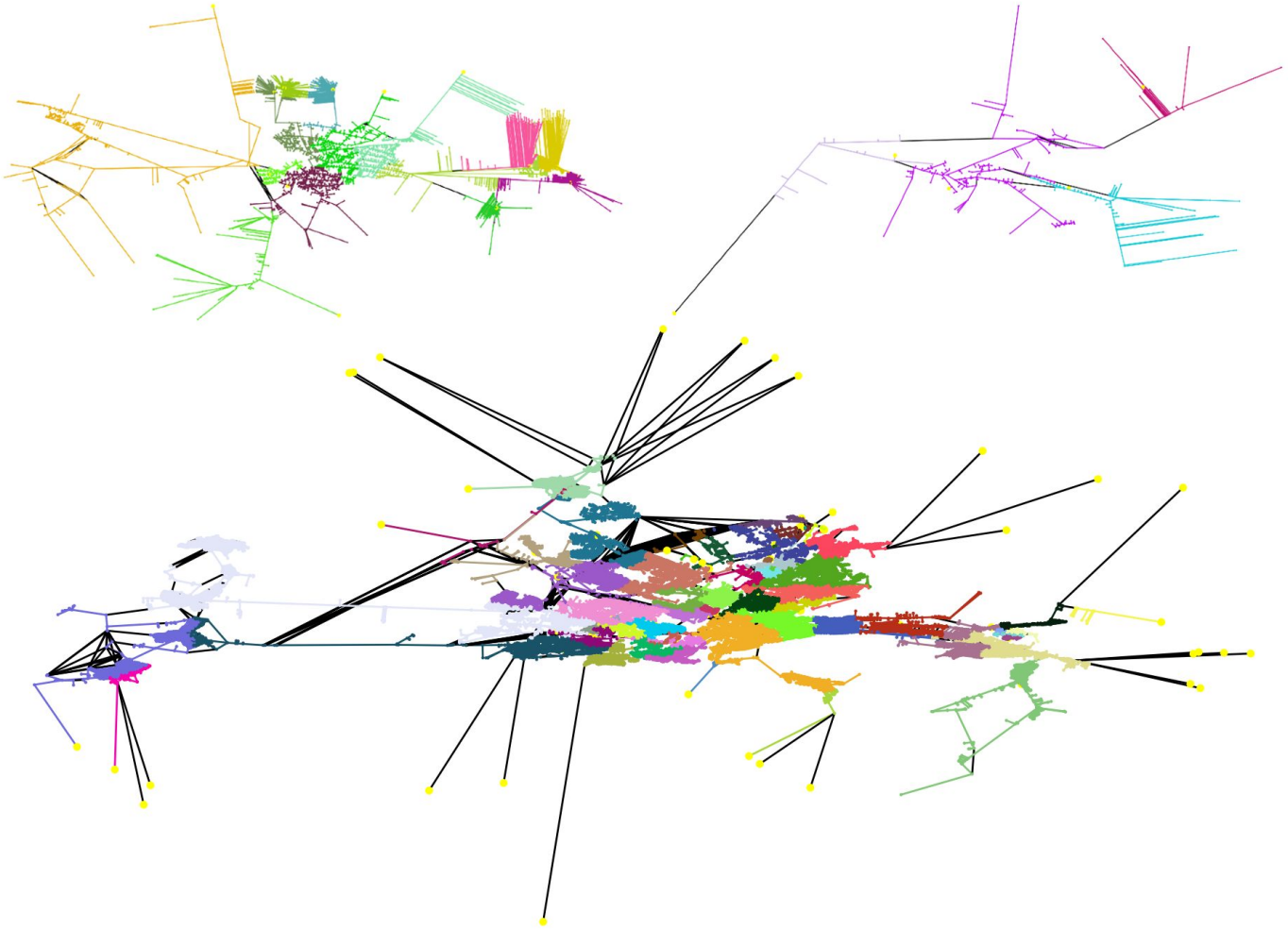


Maximum SB-Forest

Evaluation

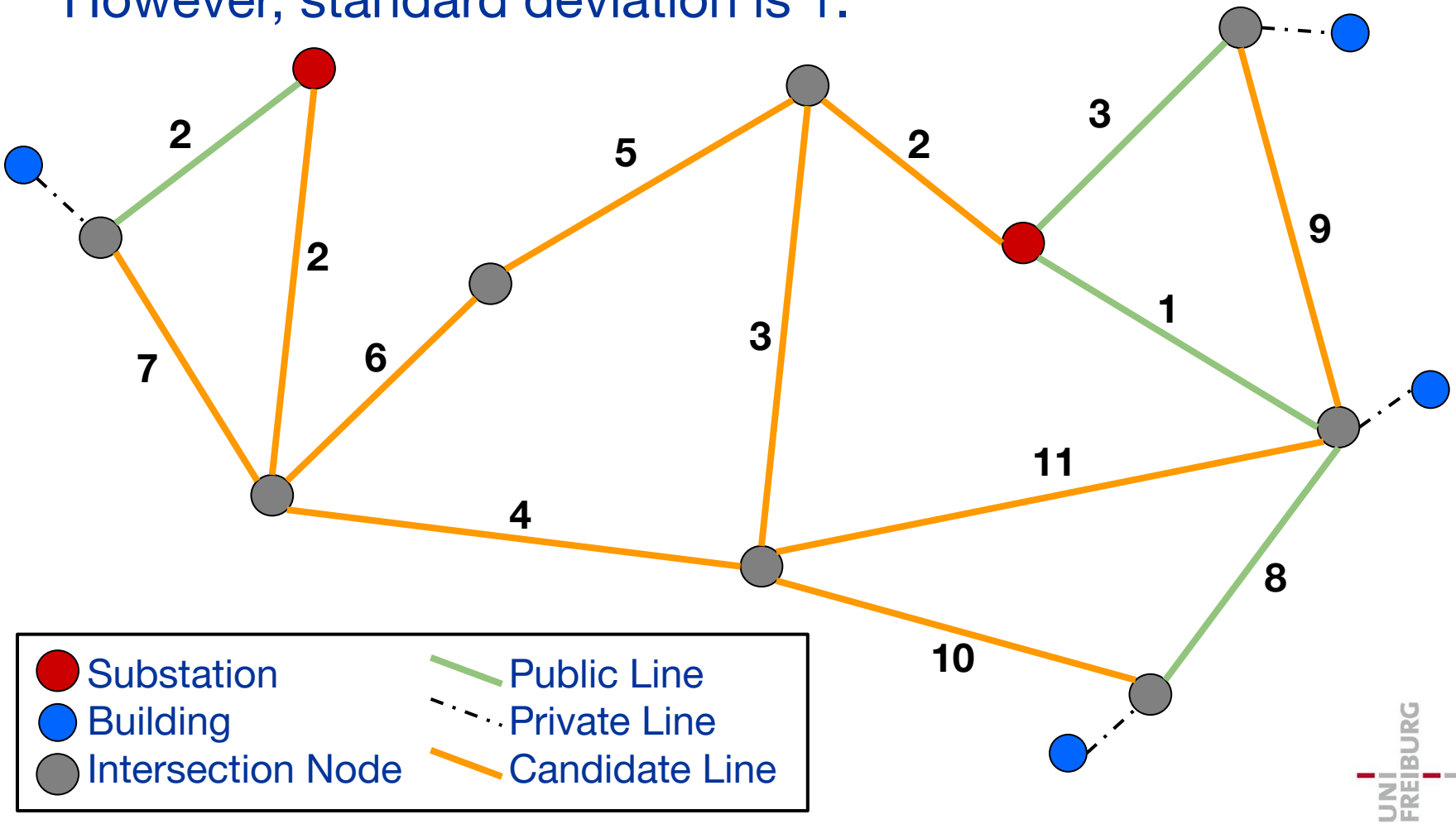
Village	Nodes	Edges	Substations	Buildings
Wieden	434	439	1	206
Kirchzarten	4669	4732	14	2457
Bad Krozingen	9254	9510	11	4353
Freiburg	60942	62490	91	29650

Evaluation



1. Standard deviation example

On average every substation **should** get two buildings.
However, standard deviation is 1.

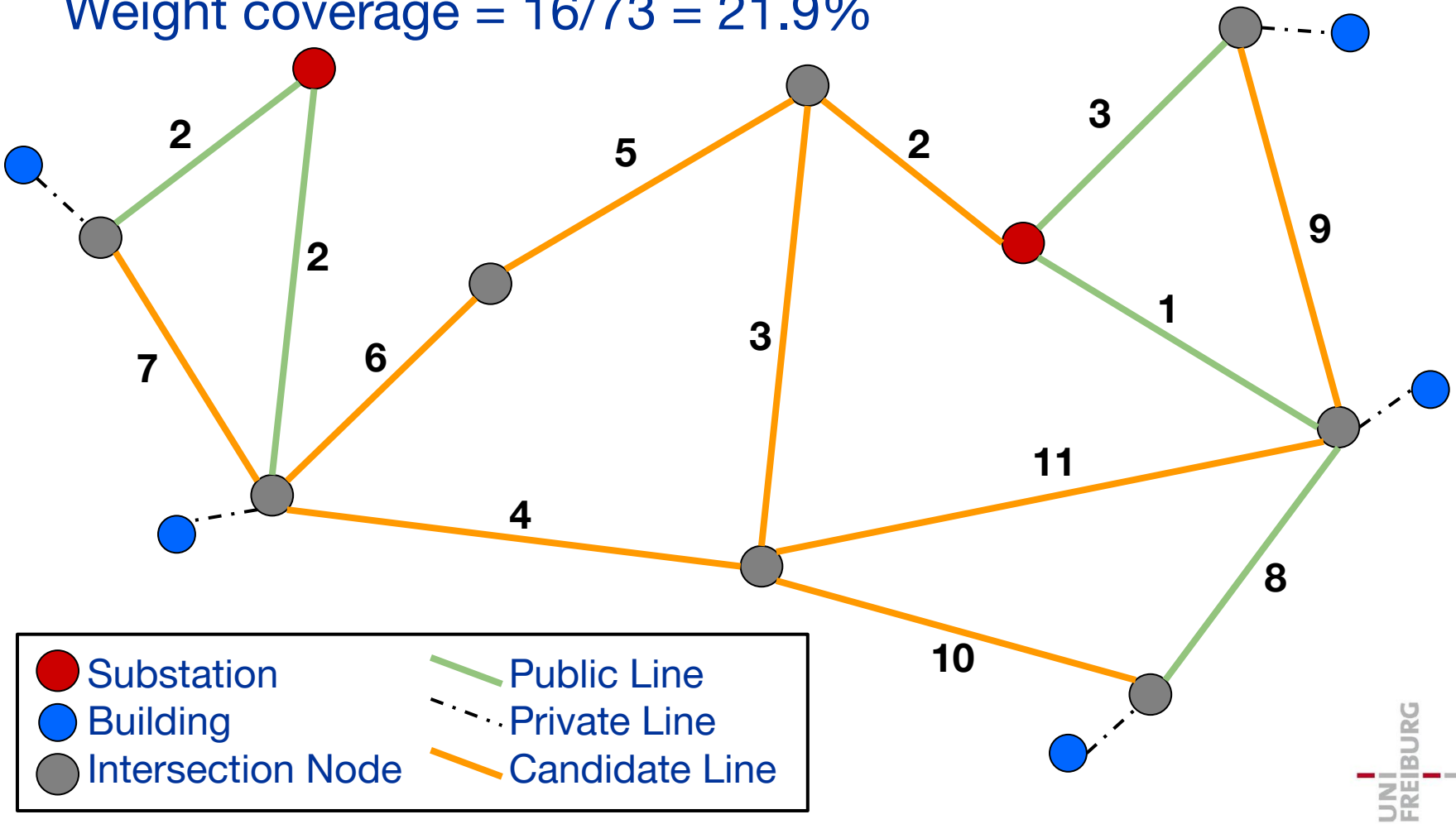


Average Standard Deviation for Buildings per Substation

Village	σ(Shortest Path)	σ(Random)	σ(Minimum)	σ(Maximum)
Wieden	10.03	6.75	12.40	13.37
Kirchzarten	143.39	72.03	211.20	254.24
Bad Krozingen	196.60	128.99	276.68	573.36
Freiburg	163.85	116.31	260.87	392.29

2. Weight coverage example

SUM[all edges] = 73, SUM[Public Lines] = 16
Weight coverage = 16/73 = 21.9%



Average Weight Coverage

Village	Shortest Path	Random	Minimum	Maximum
Wieden	83.76%	94.64%	80.88%	98.64%
Kirchzarten	96.12%	98.60%	94.11%	99.31%
Bad Krozingen	81.63%	93.07%	76.84%	95.35%
Freiburg	83.01%	93.35%	79.93%	95.81%

3. Average Runtimes *(in milliseconds)*

Village	Shortest Path	Random	Minimum	Maximum
Wieden	11.69	25.06	21.47	25.54
Kirchzarten	287.97	738.80	271.85	250.16
Bad Krozingen	582.05	2969.30	880.65	694.11
Freiburg	4108.22	37347.56	4902.45	4599.00

ANTPOWER[3] *(on Wieden)*

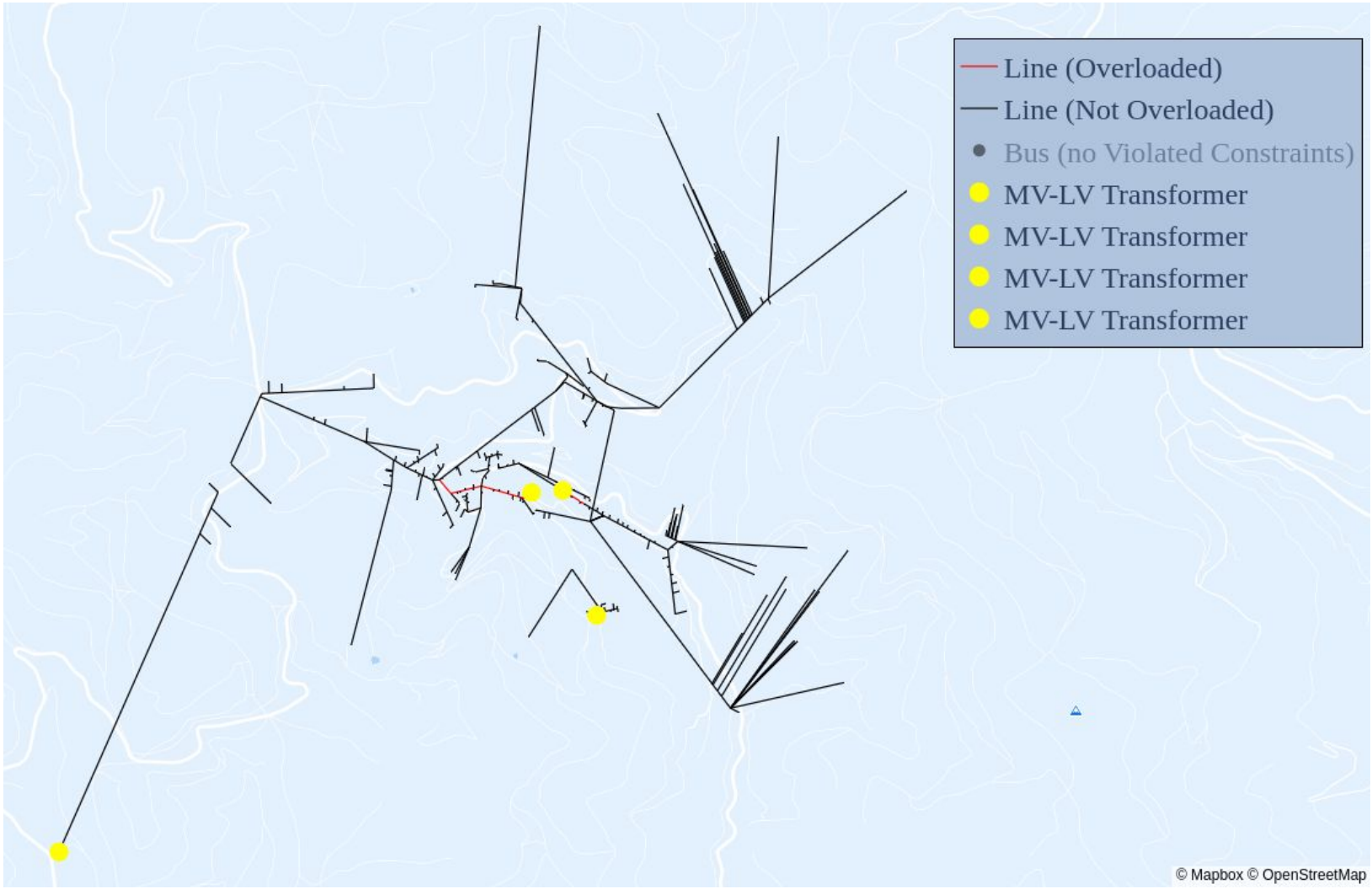
Grid Problem:

- 5 substation
- 50% ratio of generators per building
- Length factor of 0.1
- Powerflow states 10 constraint violations

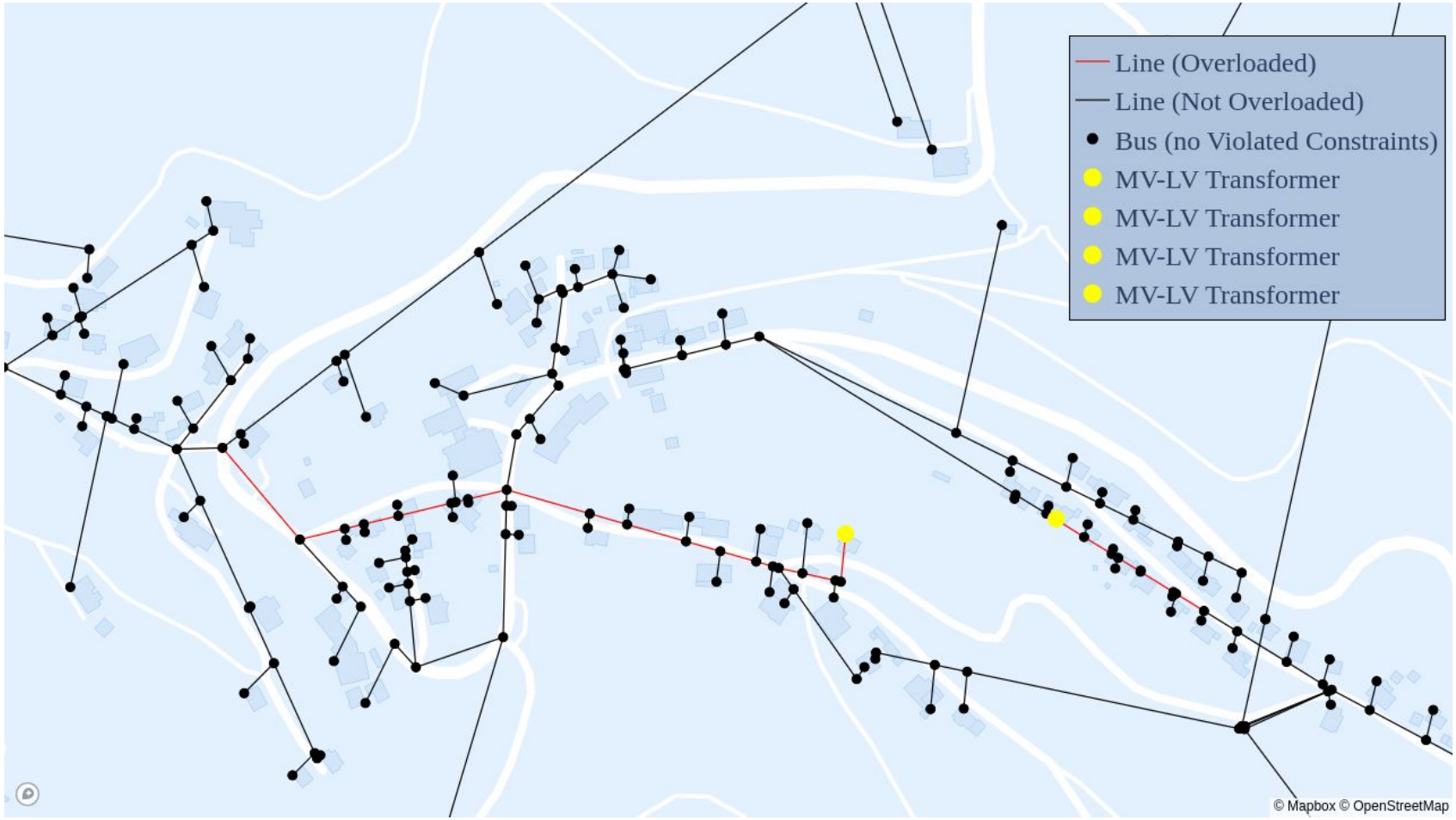
ANTPOWER solves to zero constraint violations

Note: The length factor is introduces as our initial electrical grid configuration may not be optimal. We expect grid operators to be able to properly adjust such values to test grids with their correct length of edges.

ANTPOWER *(on Wieden)*



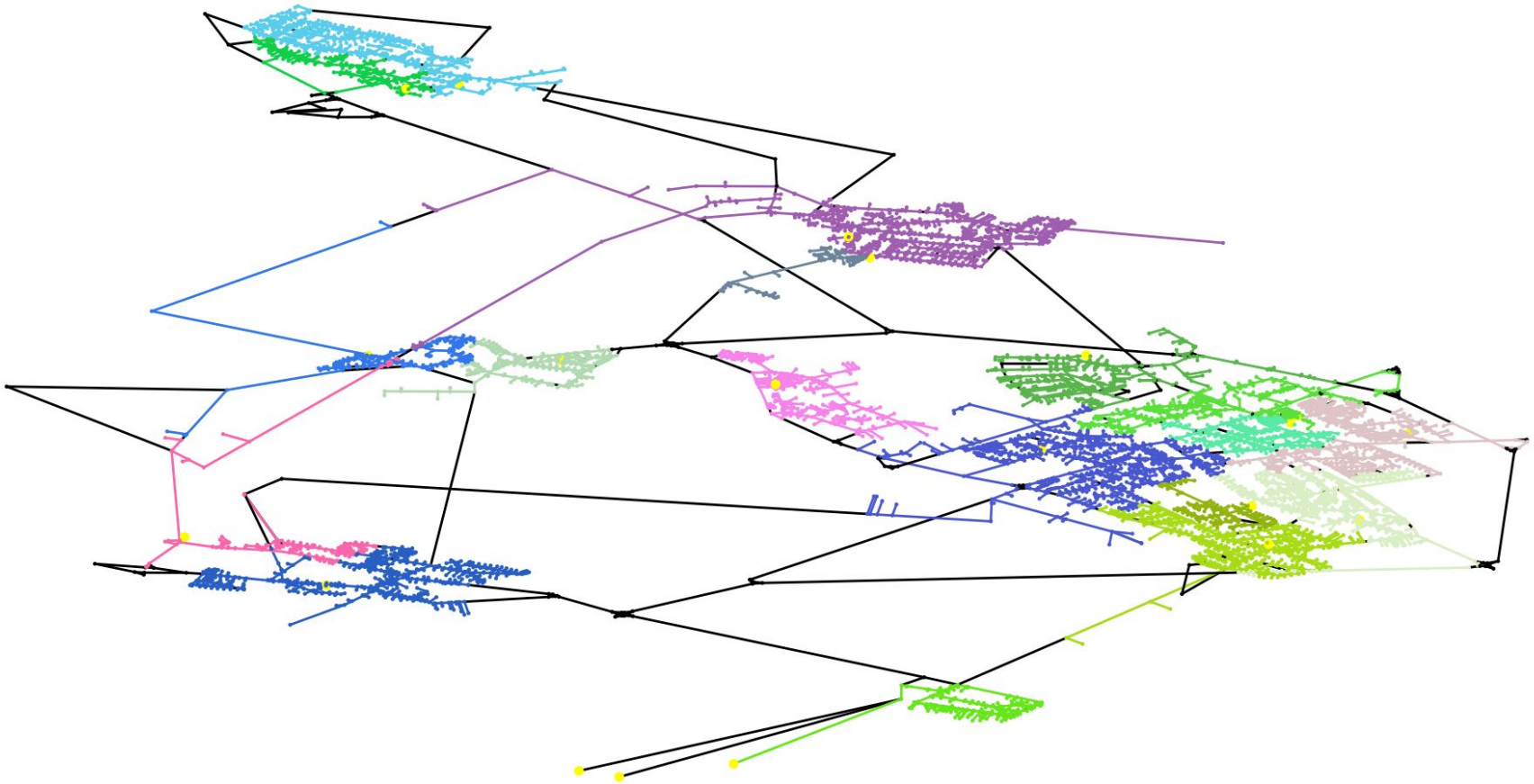
Q&A


Line (Overloaded)
Line (Not Overloaded)
Bus (no Violated Constraints)
MV-LV Transformer
MV-LV Transformer
MV-LV Transformer
MV-LV Transformer
© Mapbox © OpenStreetMap

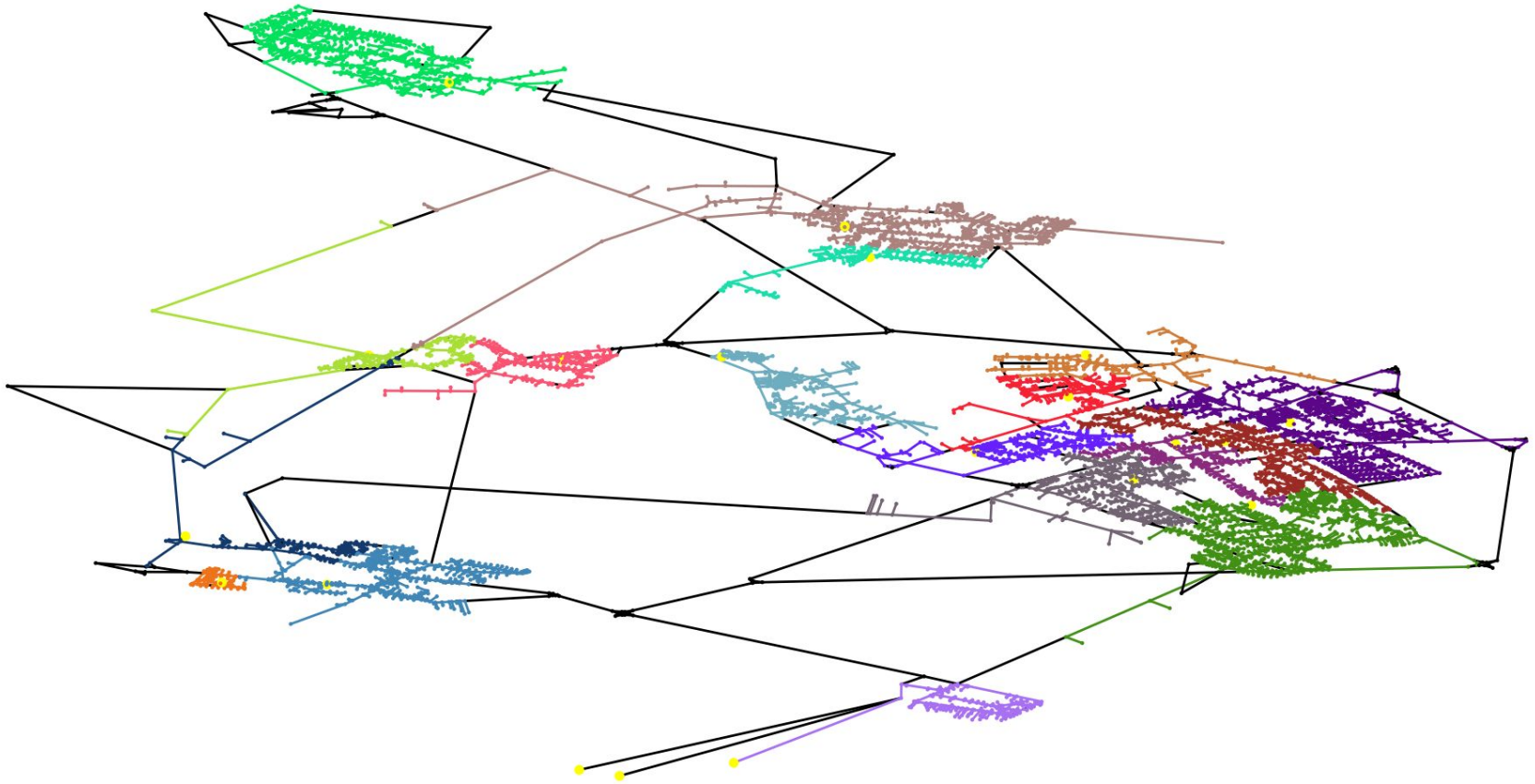
Sources

[1] G. Boeing, “Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks,” Computers Environment and Urban Systems, vol. 65, pp. 126–139, 07 2017.
[2] “Pyosmium,” [https://osmcode.org/pyosmium/; accessed 10-October-2022].
[3] L. Gebhard, “Expansion planning of low-voltage grids using ant colony optimization,” 2021. Chair of Algorithms and Data Structures, University of Freiburg.

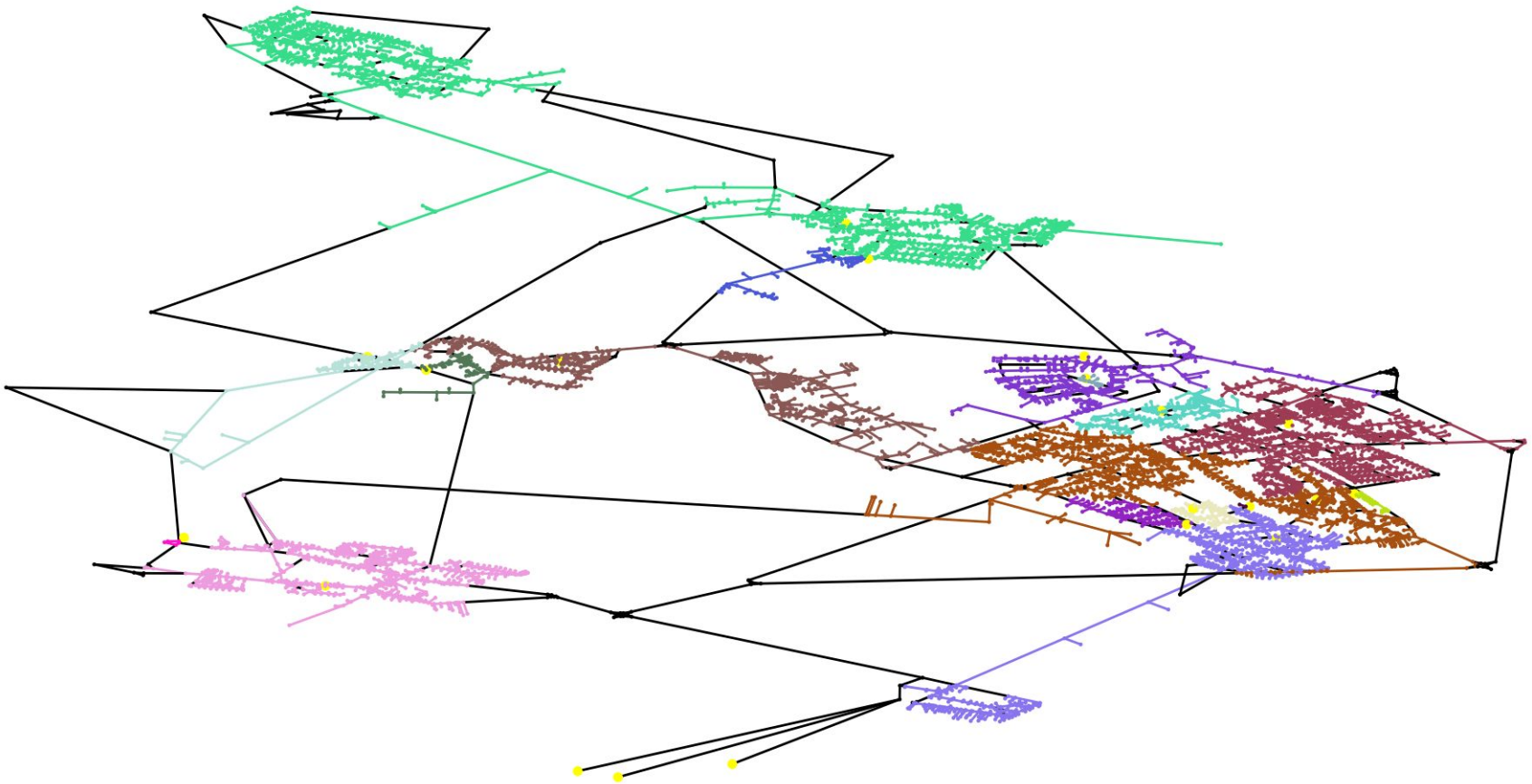
Shortest Path SB-Forest



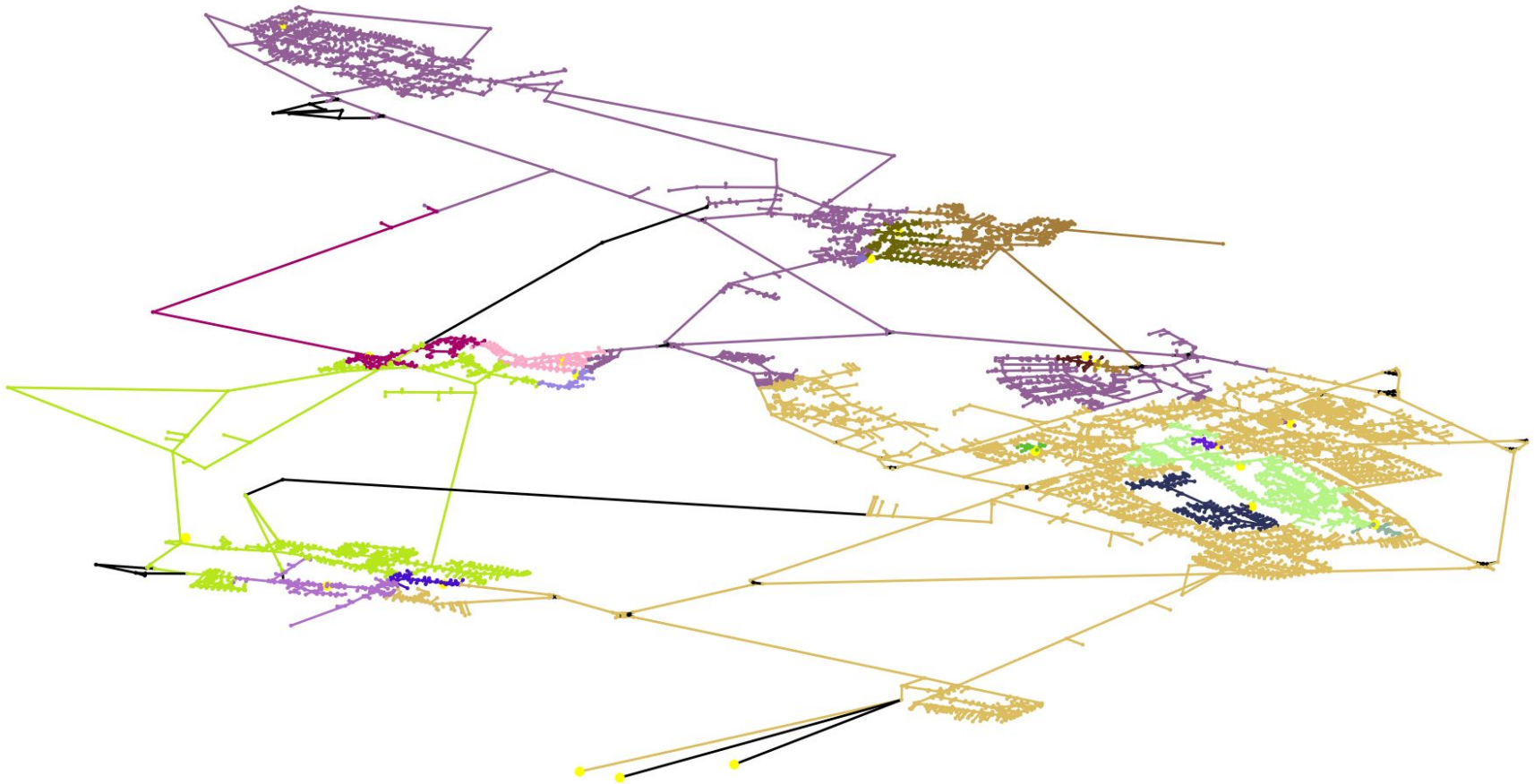
Random SB-Forest



Minimum SB-Forest



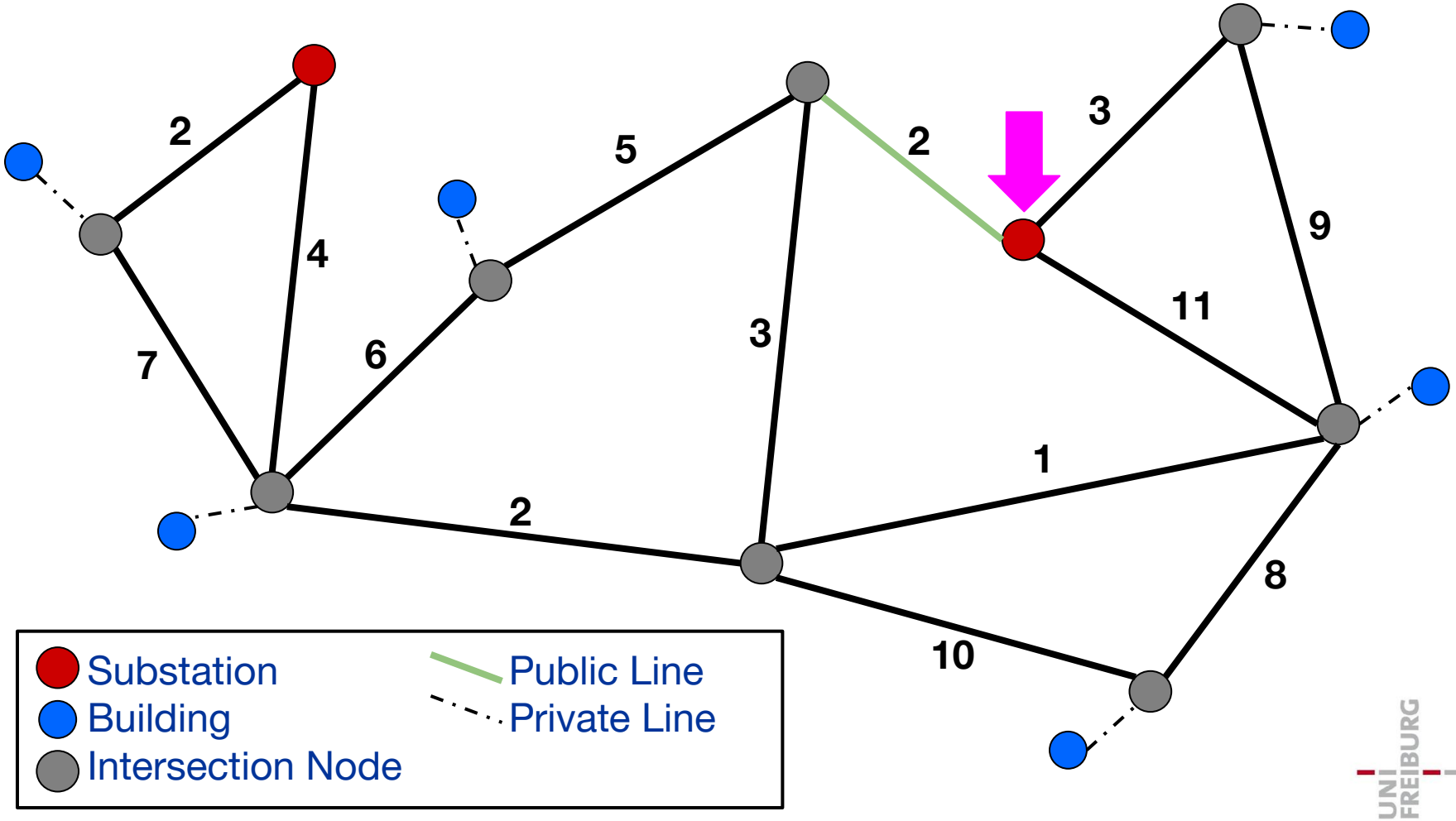
Maximum SB-Forest



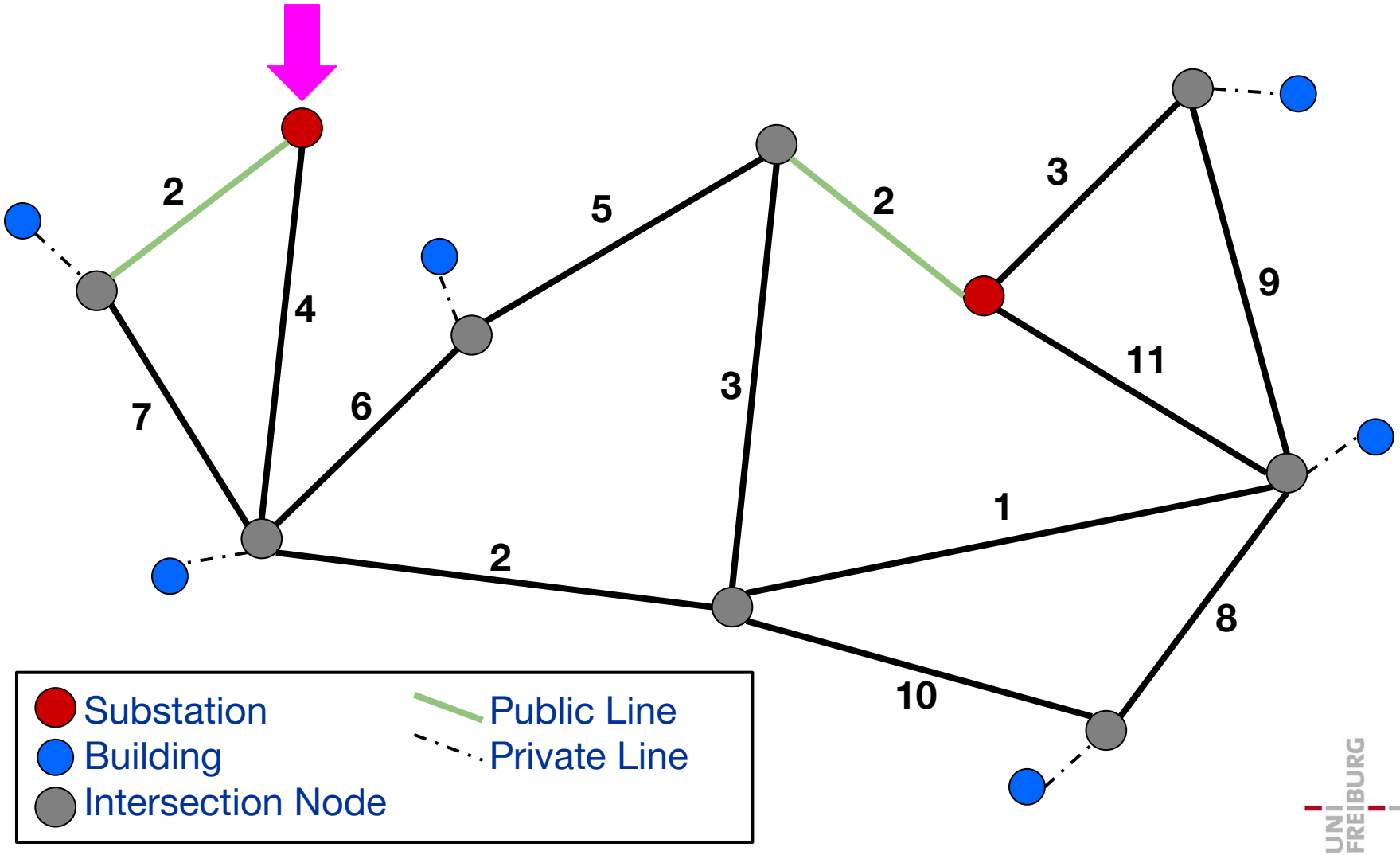
Random SB-Forest

Let's look at an other example:

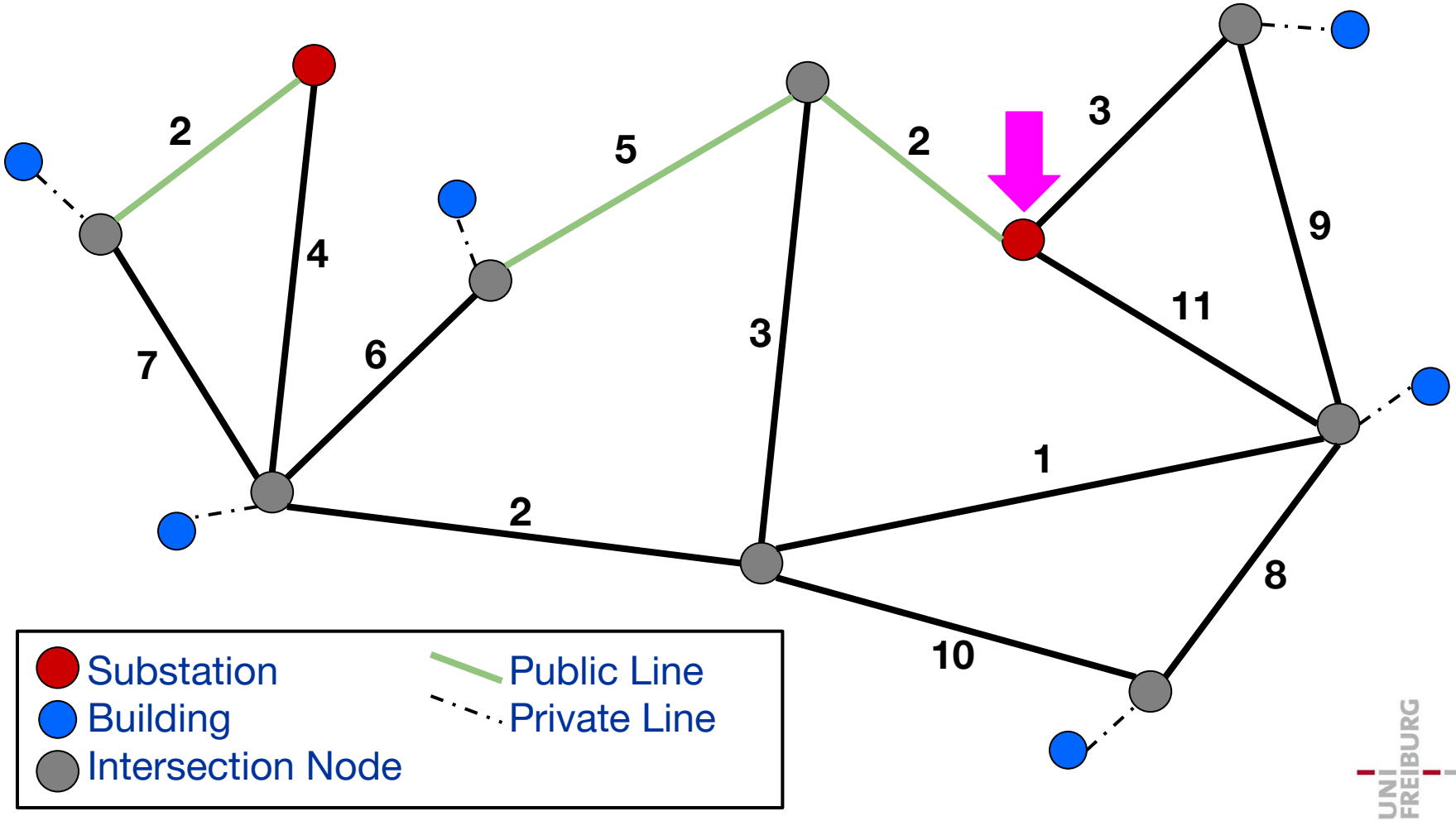
Random SB-Forest



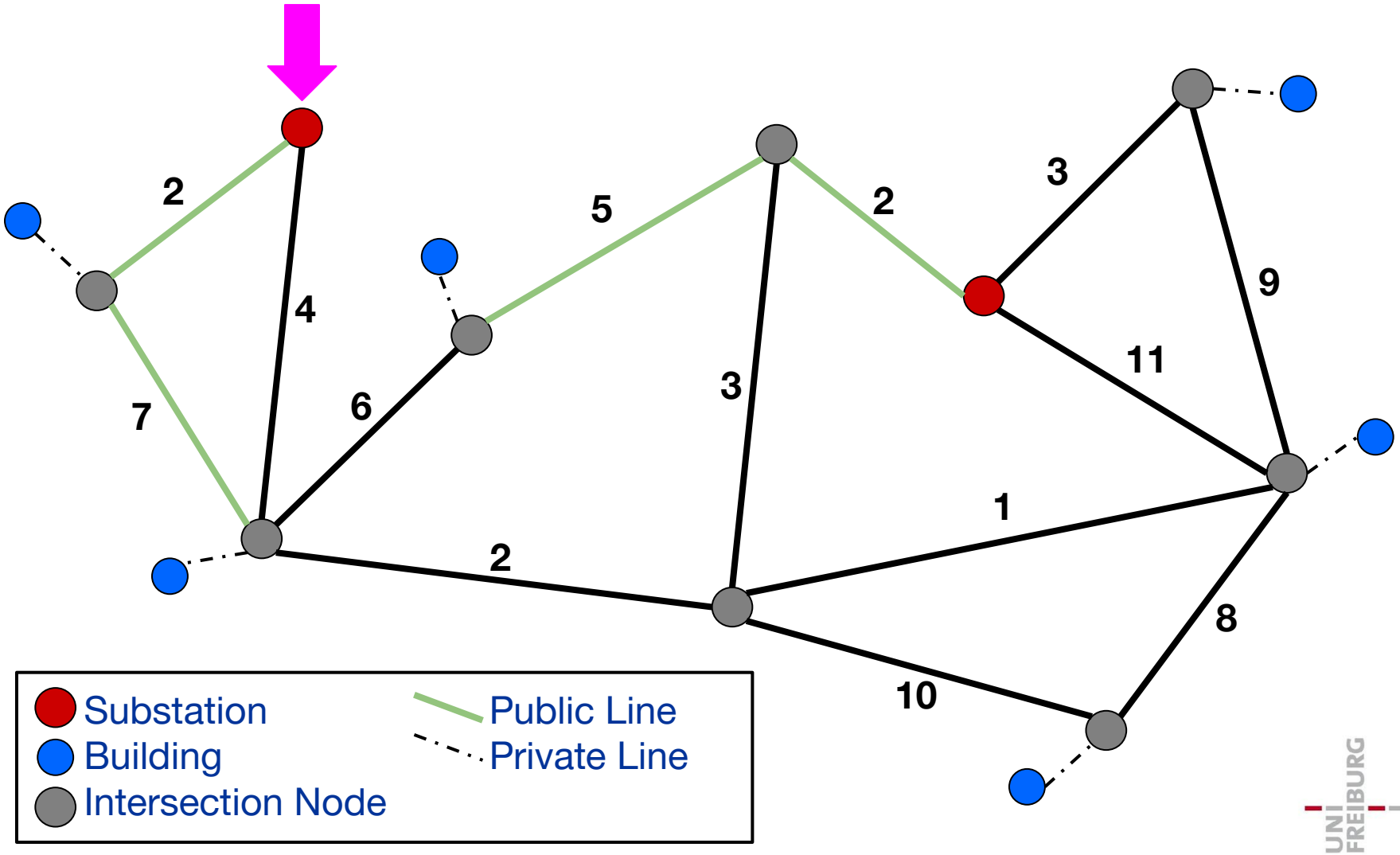
Random SB-Forest



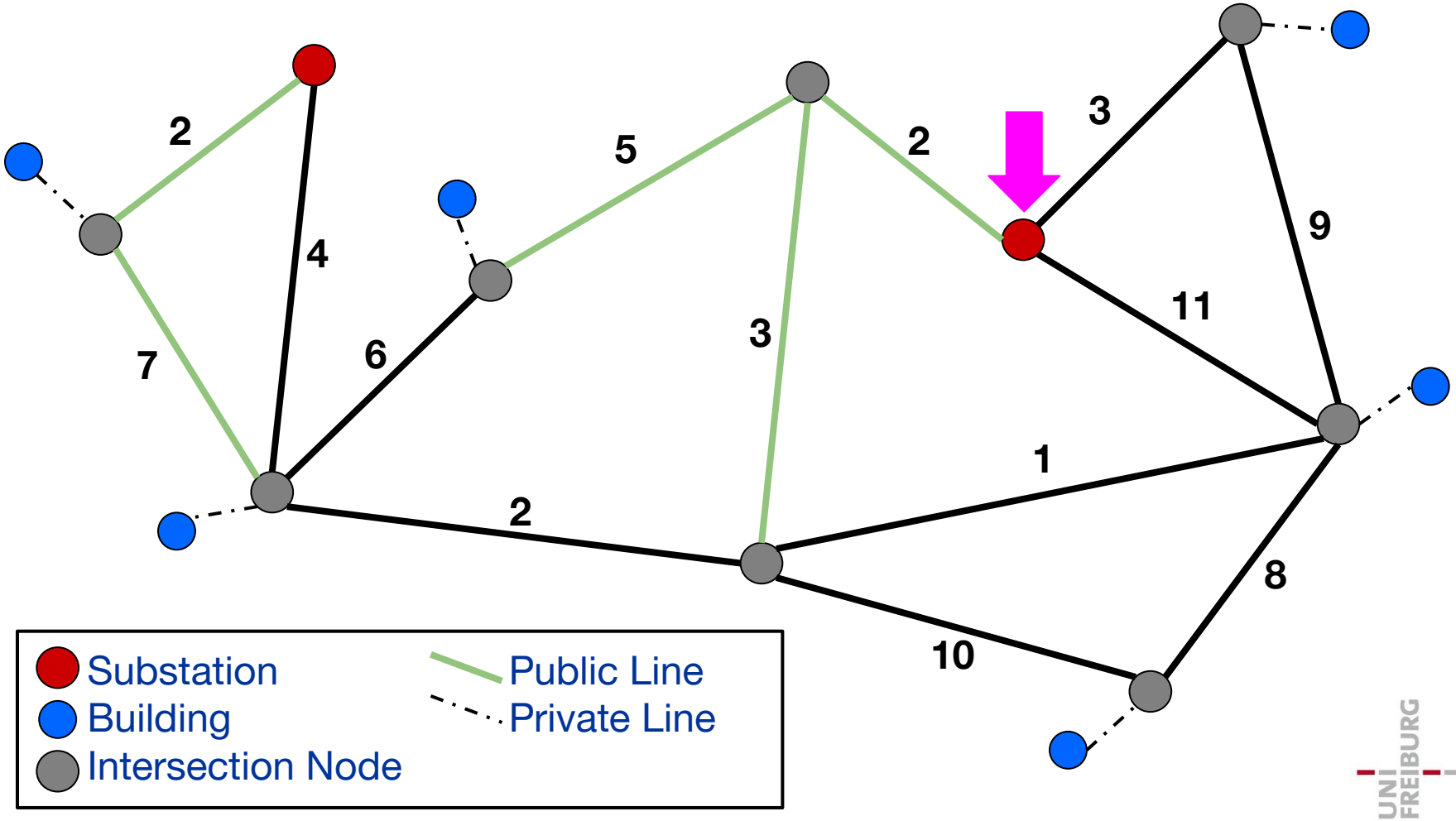
Random SB-Forest



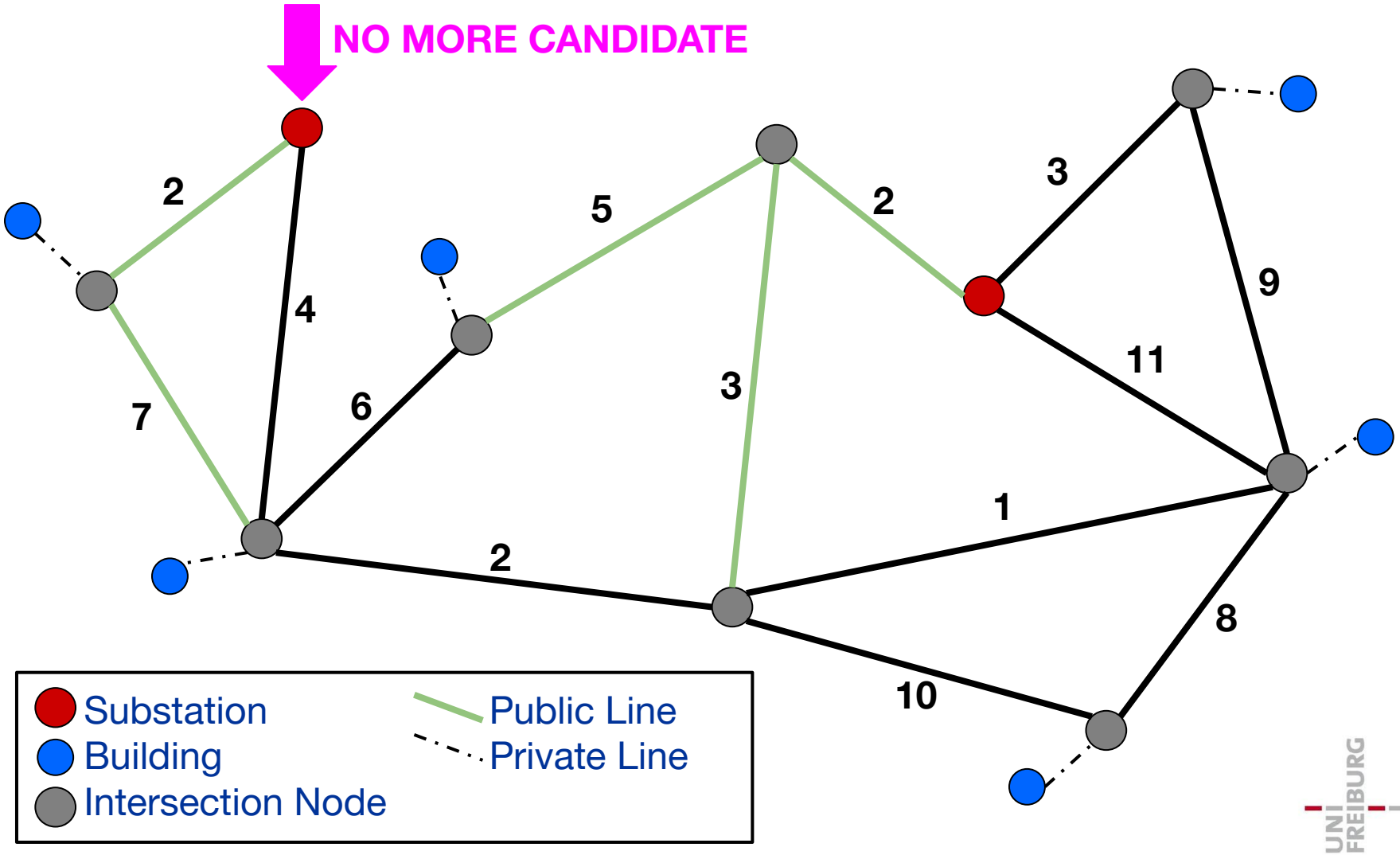
Random SB-Forest



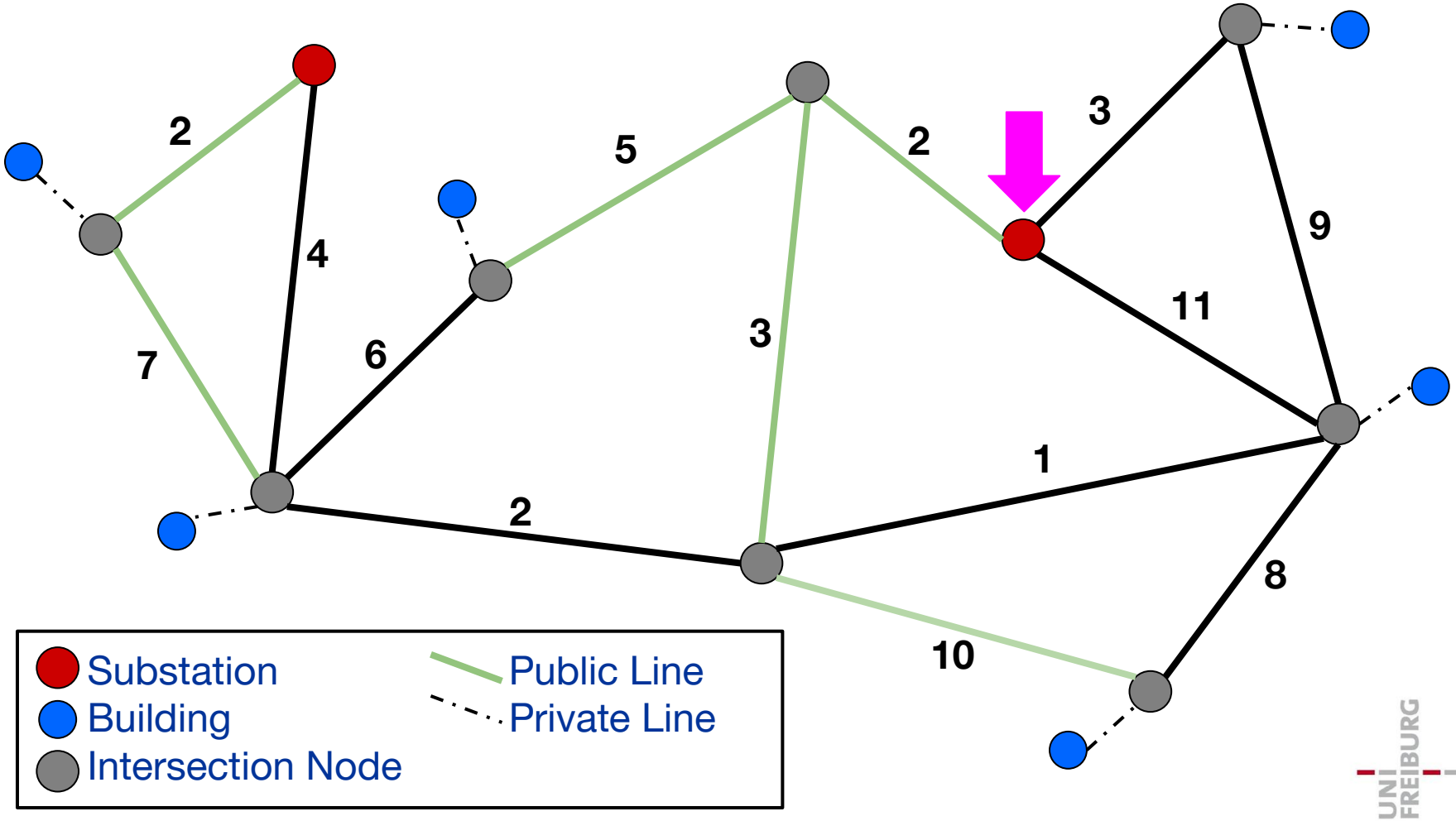
Random SB-Forest



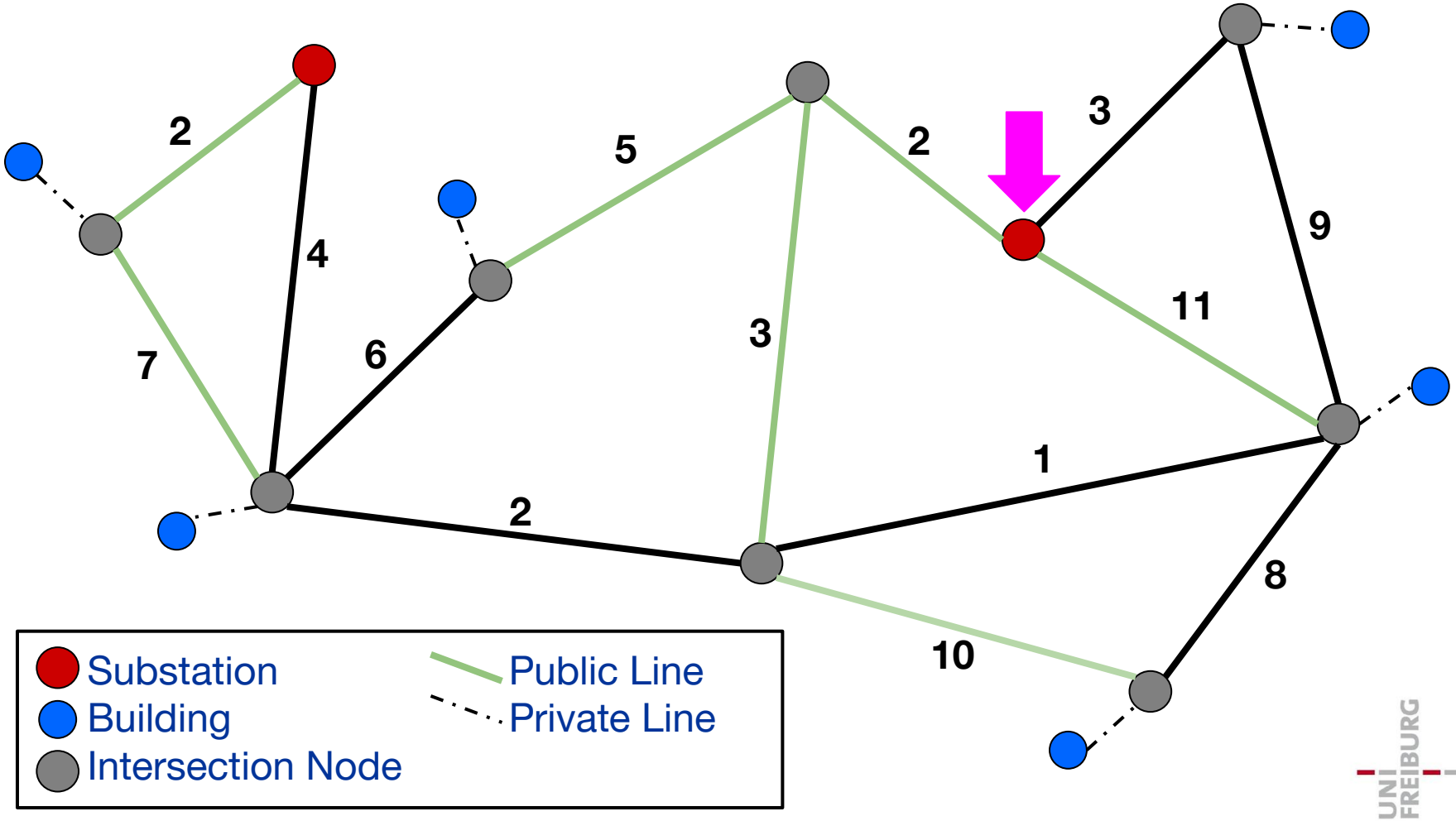
Random SB-Forest



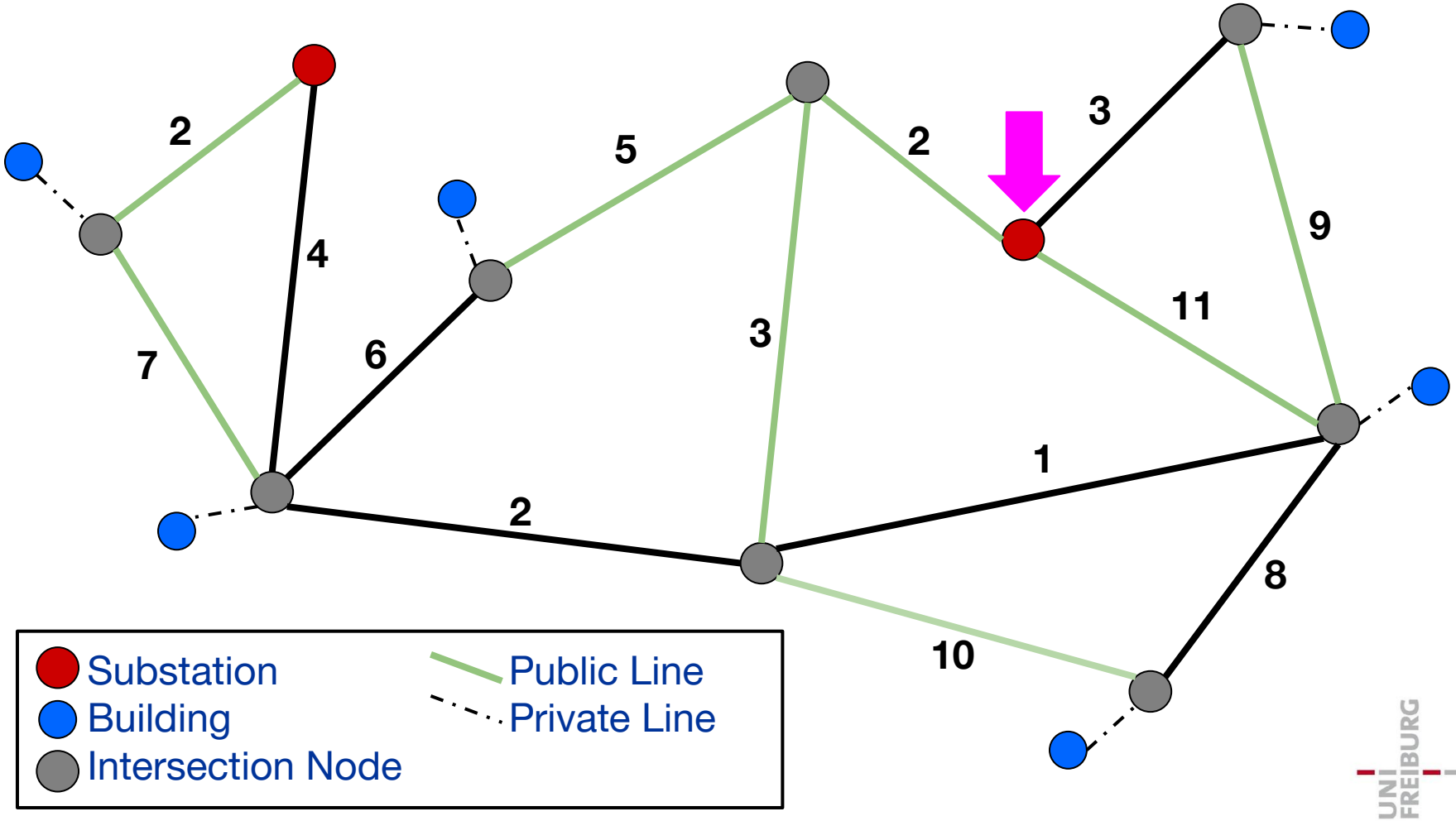
Random SB-Forest



Random SB-Forest



Random SB-Forest



Random SB-Forest