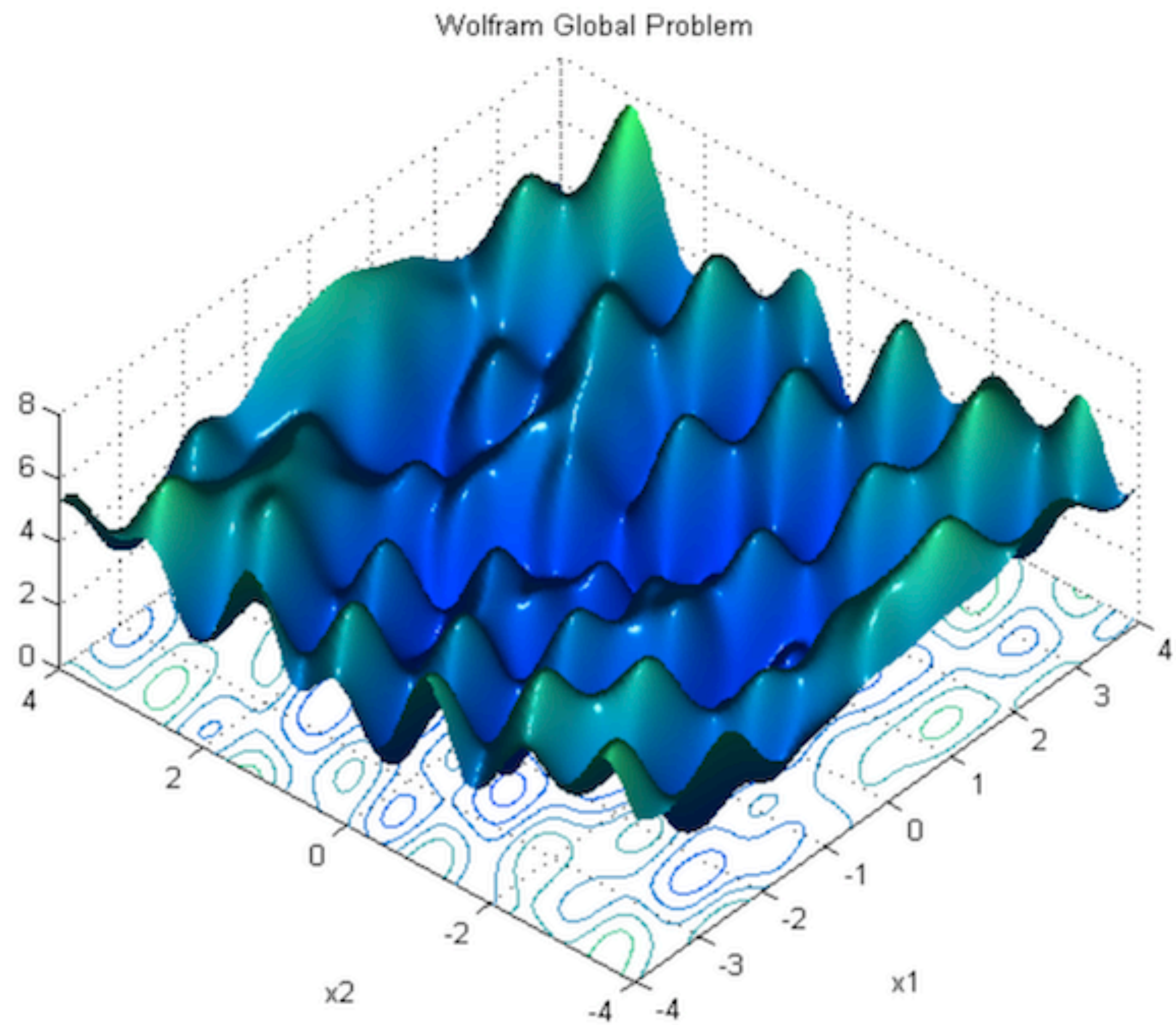


Optimisation Problems

Leandro L. Minku

Outline

- What are optimisation problems?
- How to formulate optimisation problems?
- Relationship between search and optimisation.
- Familiarising with maths.

Announcements

- Tutorials should be showing in your timetables. If not, please contact the Education Support Office to add you to a tutorial group.
- Do interact with the PGTAs in the tutorials. They need your help to be able to help you :-)
- Do attend the office hours if you have questions before the tutorials or if there was not enough time to go through all your questions in the tutorials.
- Engagement with the content is essential.

Optimisation Problems

- Optimisation problems: to find a solution that minimises/maximises one or more pre-defined objective functions.
- Minimisation / maximisation problems.
- There may be some constraints that must be satisfied for a given solution to be feasible.

Examples of Optimisation Problems

- Bin packing problem:
  - Given bins with maximum volume, which cannot be exceeded.
  - We have *n* items to pack, each with a given volume.
  - We must pack all items.

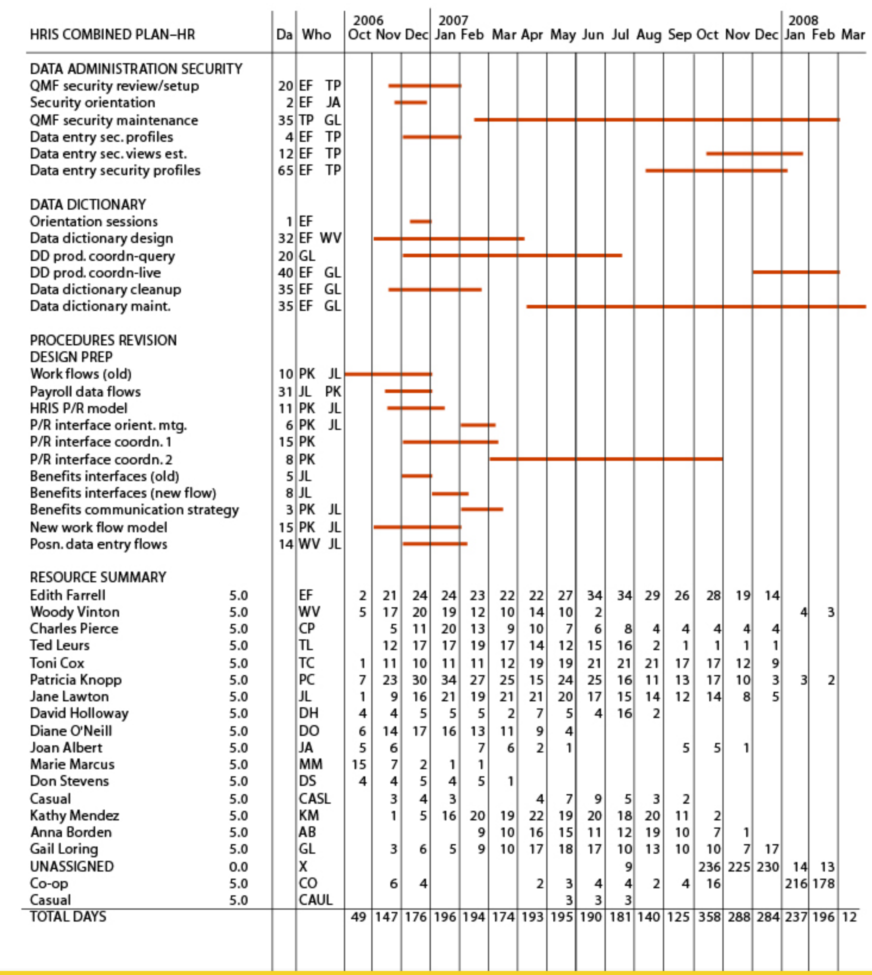
Problem: find an assignment of items to bins that minimises the number of bins used, ensuring that all items are packed and the maximum volume of the bins is not exceeded.

Photo from: http://maritime-connector.com/images/container-ship-16-wiki-19057.jpg

Photo from: http://www.tscargo.ca/images/cargo1.jpg

Examples of Software Engineering Optimisation Problems

- Software Project Scheduling:
  - Given *E* employees and *T* tasks.
  - Each task requires an effort in hours and certain skills.
  - Each employee has a salary, a set of skills and can work a maximum number of hours.
  - Tasks have precedence relationships.

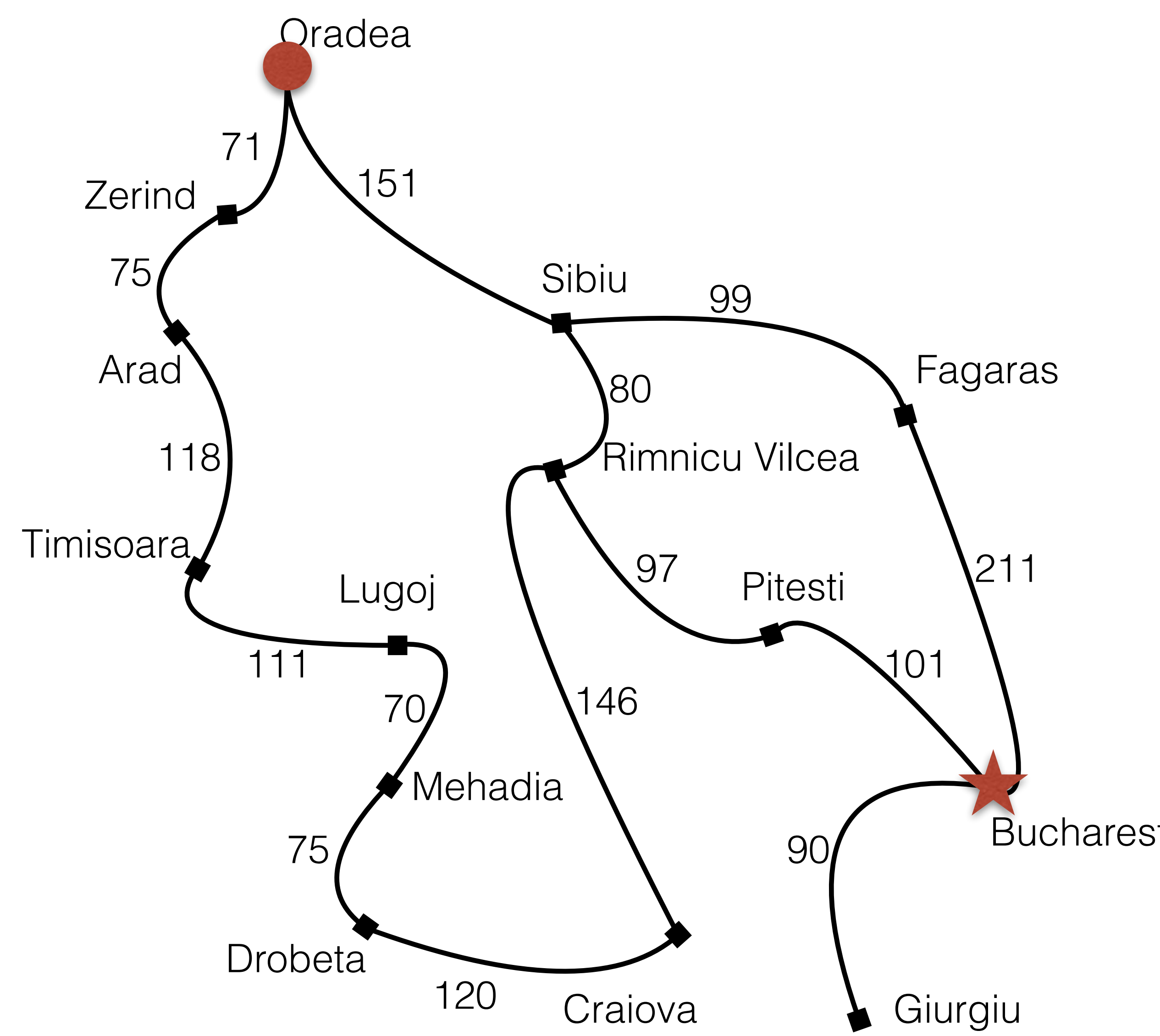


Problem: find an allocation of employees to tasks that minimises the cost and the duration of the software project, while ensuring that:

- employees are only assigned to tasks for which they have the required skills,
- they work only up to a maximum number of hours, and
- that the task precedences are respected.

Examples of Optimisation Problems

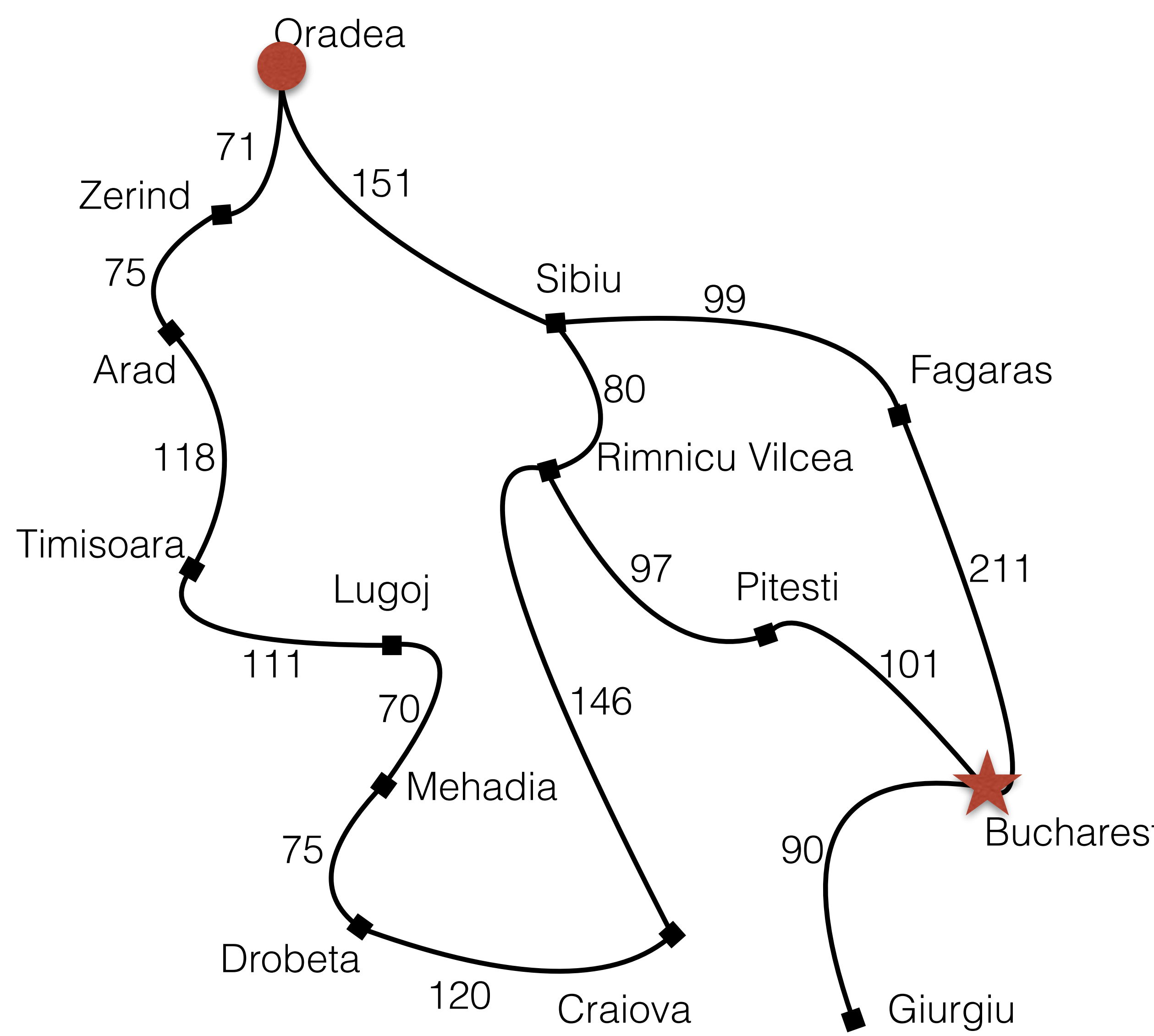
- Routing problem:
  - Given a motorway map containing N cities.
  - The map shows the distance between connected cities.
  - We have a city of origin and a city of destination.



Problem: find a solution that optimises an objective function, while satisfying some constraints.

Examples of Optimisation Problems

- Routing problem:
  - Given a motorway map containing N cities.
  - The map shows the distance between connected cities.
  - We have a city of origin and a city of destination.



Problem: find a path from the origin to the destination that minimises the distance travelled, while ensuring that direct paths between non-neighbouring cities are not used.

Optimisation Problems: Canonical Representation

Objective function

Design variables

Minimise?
Maximise?

Minimise $f(\mathbf{x})$

Subject to $g_i(\mathbf{x}) \leq 0, \quad i = 1, \cdots, m$

$h_j(\mathbf{x}) = 0, \quad j = 1, \cdots, n$

Constraints

Search space: space of all possible $\mathbf{x}$ values.

Multi-Objective Optimisation Problems

Minimise $f_k(\mathbf{x}), \quad k = 1\cdots, p$

Subject to $g_i(\mathbf{x}) \leq 0, \quad i = 1, \cdots, m$

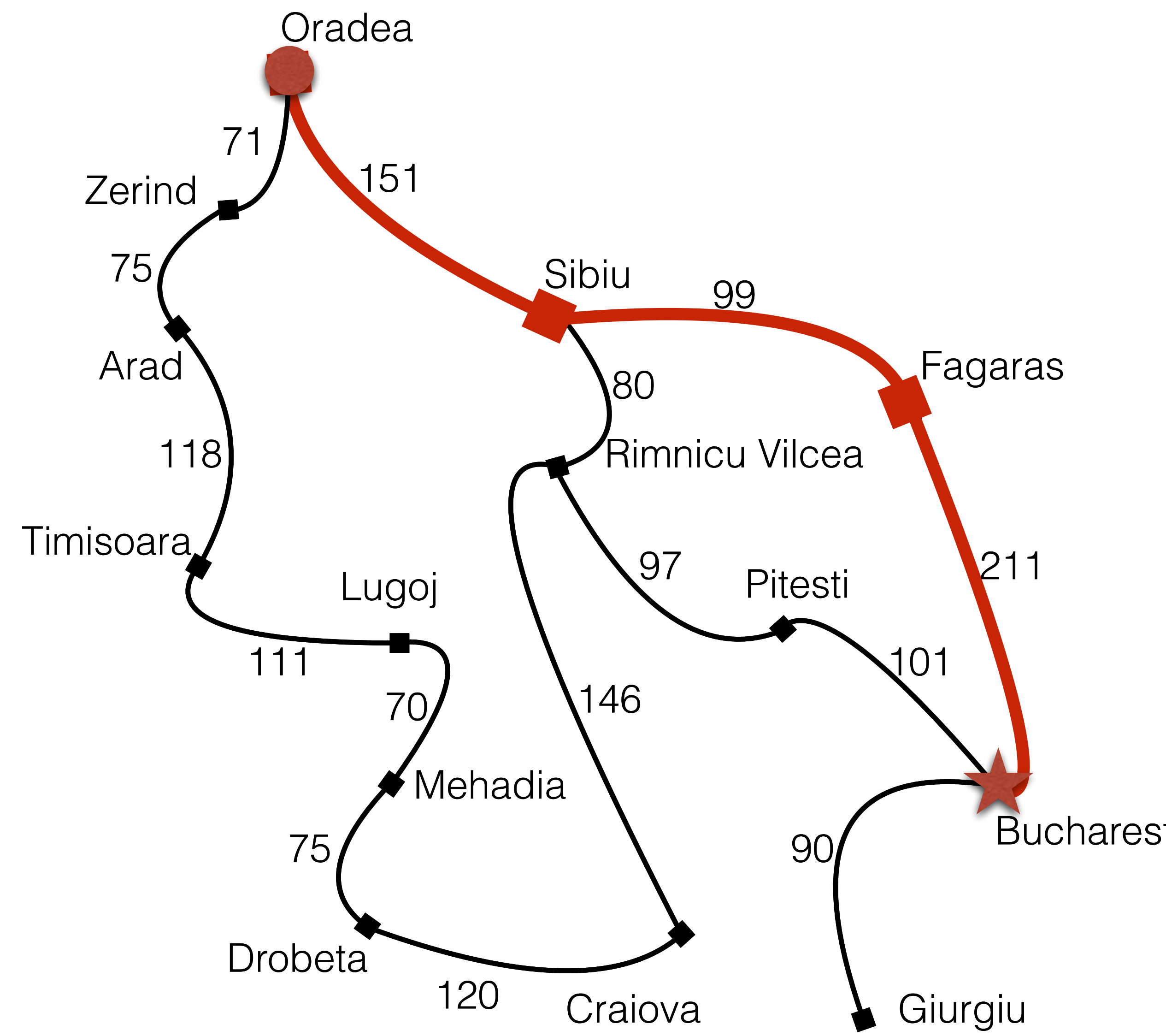
$h_j(\mathbf{x}) = 0, \quad j = 1, \cdots, n$

Formulating Optimisation Problems

- Design variables represent a candidate solution.
  - Design variables define the search space of candidate solutions.
- Objective function defines the cost (or quality) of a solution.
  - Function to be optimised (minimised or maximised).
- [Optional] Solutions must satisfy certain constraints, which define solution feasibility.
  - Candidate solutions may be feasible or infeasible.

Routing Problem: Design Variable

- Design variables represent a candidate solution.
  - 1-d array $\mathbf{x}$ of any size containing the sequence of cities (from the map)…
  - …to be visited from the city of origin to the city of destination (excluding the cities of origin and destination).
  - The search space consists of all possible sequences of cities.



Design variable:

Oradea $\underline{\text{Sibiu}}_{x_1}$ $\underline{\text{Fagaras}}_{x_2}$ Bucharest

Routing Problem: Objective Function

- Objective function defines the cost (or quality) of a solution.

  Minimise the sum of the distances between consecutive cities in the path obtained through the solution.

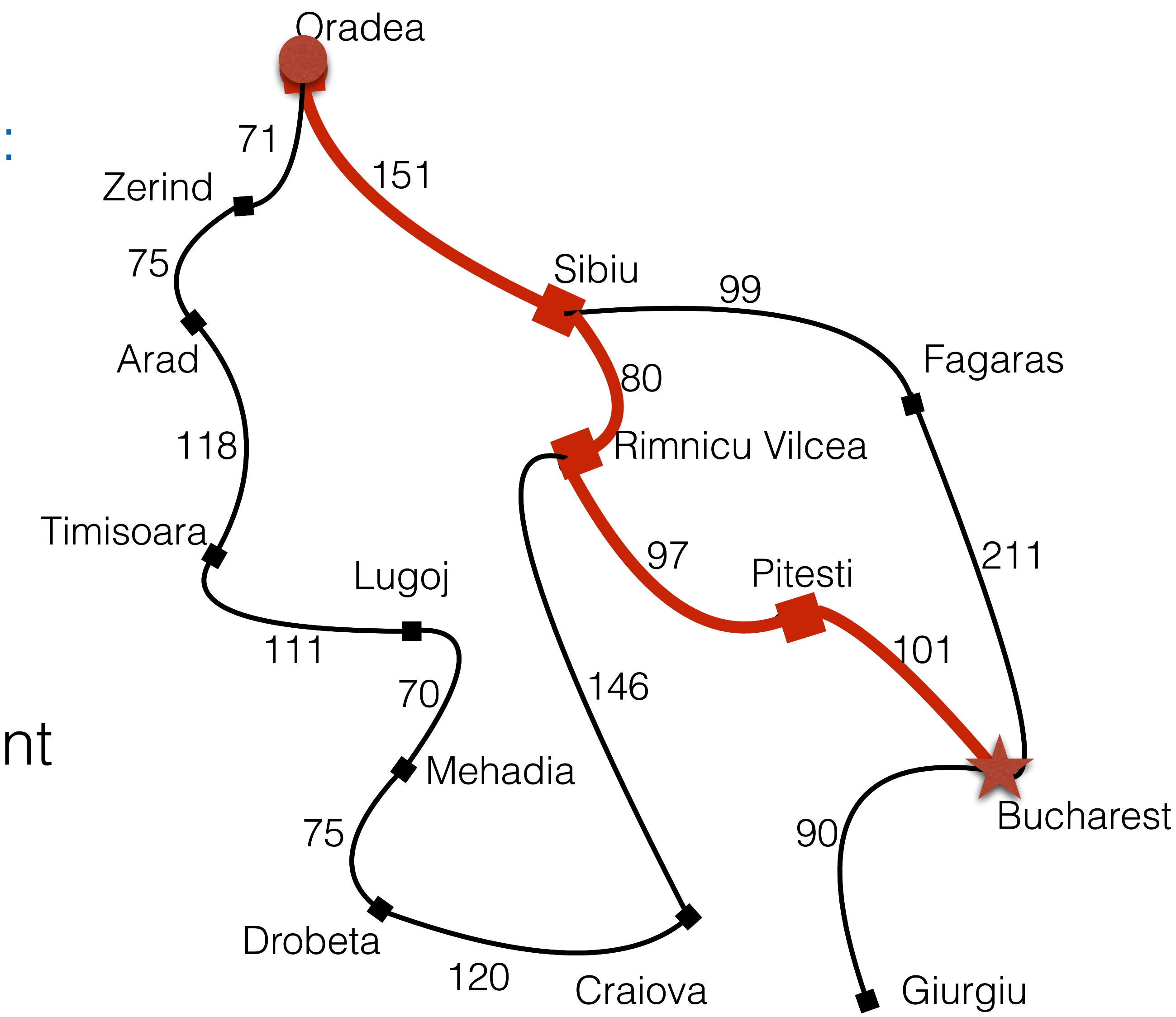
Oradea $\underset{x_1}{\underline{\text{Sibiu}}}$ $\underset{x_2}{\underline{\text{Fagaras}}}$ Bucharest

Routing Problem: Constraints

Example of infeasible candidate solution:
Oradea [Rimnicu, Mehadia] Bucharest
(moves to non-neighbouring cities)

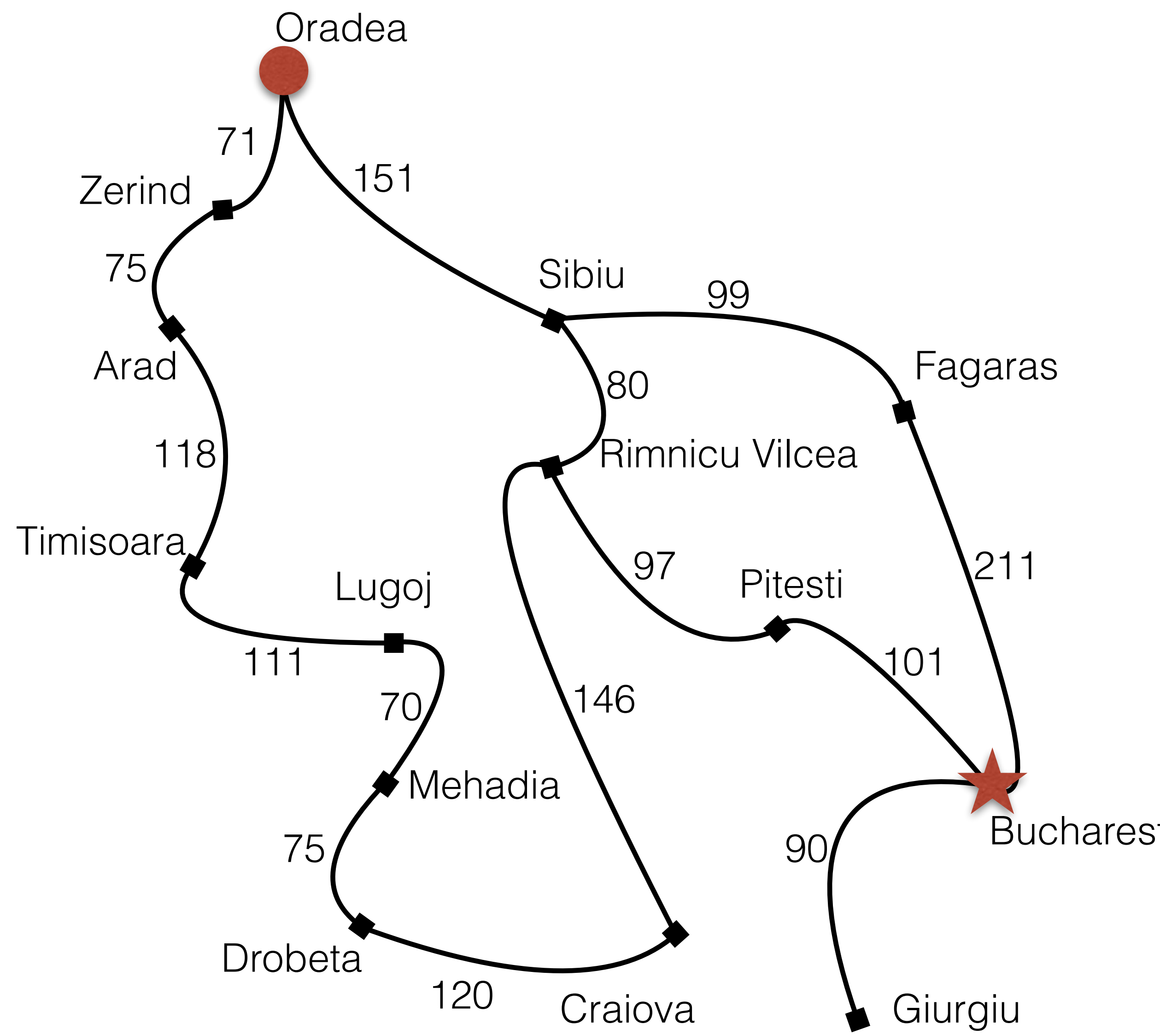
Example of feasible candidate solution:
Oradea [Sibiu, Fagaras] Bucharest
(non-optimal solution that takes the agent from the origin to the destination)

Optimal solution:
Oradea [Sibiu, Rimnicu, Pitesti] Bucharest



Routing Problem: Constraints

- Solutions must satisfy certain constraints, which define solution feasibility.
  - (Inexistent) direct paths between non-neighbouring cities must not be used (explicit constraint).
  - We must start at the city of origin and end at the city of destination (implicit constraint).
  - Only cities from the map can be used (implicit constraint).



Design variable: 1-d array $\mathbf{x}$ (of any size) containing the sequence of cities (from the map) to be visited from the city of origin to the city of destination (excluding the cities of origin and destination).

Routing Problem: Formulation as an Optimisation Problem

- Design variables represent a candidate solution.
  - 1-d array **x** of any size containing the sequence of cities (from the map) to be visited from the city of origin to the city of destination (excluding the cities of origin and destination).
- Objective function defines the cost (or quality) of a solution.

  Minimise the sum of the distances between consecutive cities in the path obtained through the solution.
- [Optional] Solutions must satisfy certain constraints, which define solution feasibility.
  - (Inexistent) direct paths between non-neighbouring cities must not be used (explicit constraint).
  - We must start at the city of origin and end at the city of destination (implicit constraint).
  - Only cities in the map can be used (implicit constraint).

Routing Problem: Formalising the Design Variable

Design variable:

1-d array $\mathbf{x}$ of any size containing the sequence of cities (from the map) to be visited from the city of origin to the city of destination (excluding the cities of origin and destination).

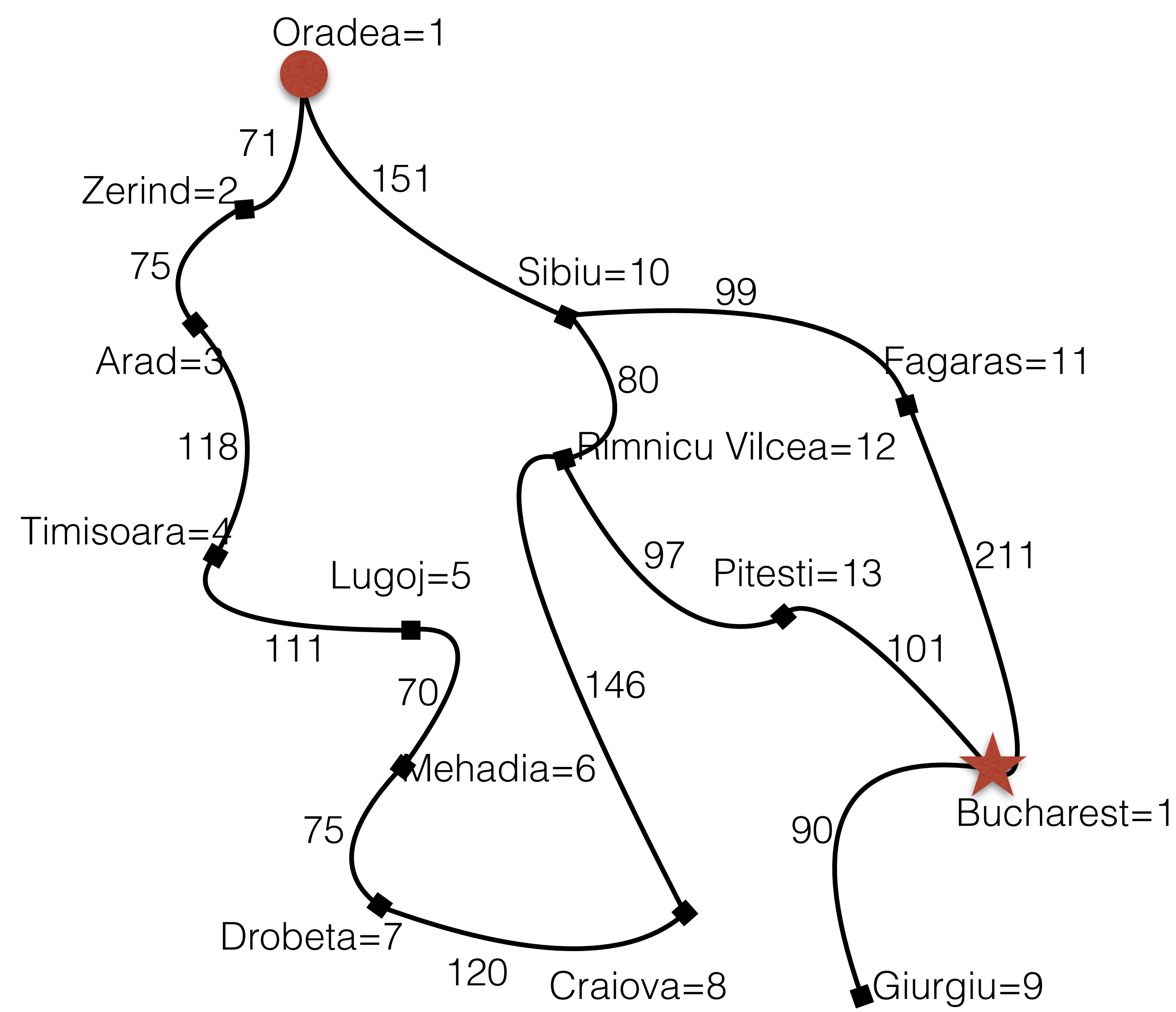
Routing Problem: Formalising the Design Variable

Design variable:

1-d array $\mathbf{x}$ of any size containing the sequence of cities (from the map) to be visited from the city of origin to the city of destination (excluding the cities of origin and destination).

Set of cities from the map:

$$C = \{1, \cdots, N\}$$



Routing Problem: Formalising the Design Variable

Design variable:

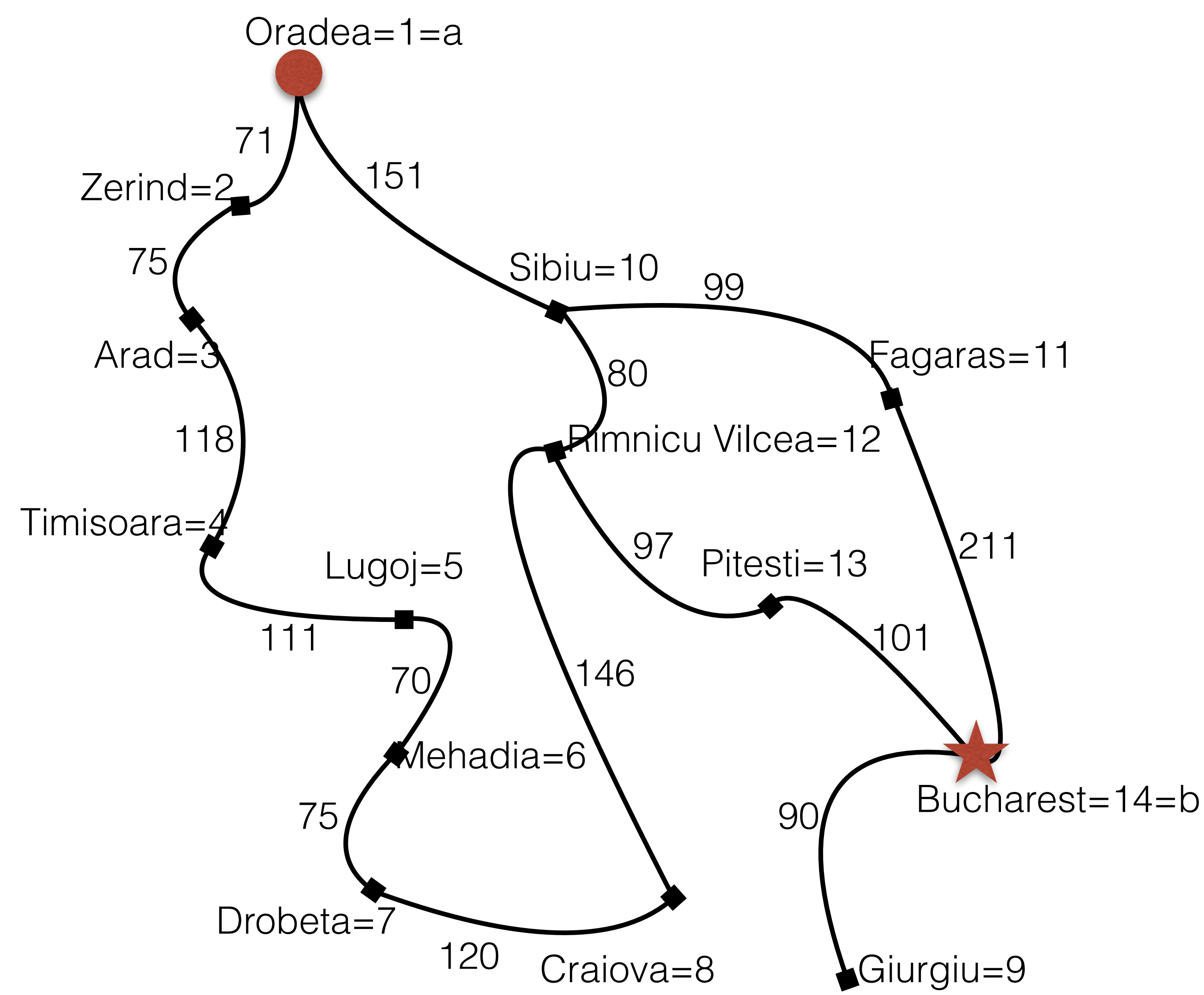
1-d array $\mathbf{x}$ of any size containing the sequence of cities (from the map) to be visited from the city of origin to the city of destination (excluding the cities of origin and destination).

City of origin: $a \in C$

City of destination: $b \in C$

Set of cities from the map:

$$C = \{1, \cdots, N\}$$



Routing Problem: Formalising the Design Variable

Design variable:

1-d array $\mathbf{x}$ of any size containing the sequence of cities (from the map) to be visited from the city of origin to the city of destination (excluding the cities of origin and destination).

Set of cities from the map:

$$C = \{1, \cdots, N\}$$

City of origin: $a \in C$

City of destination: $b \in C$

Routing Problem: Formalising the Design Variable

Design variable:

1-d array $\mathbf{x}$ of any size containing the sequence of cities (from the map) to be visited from the city of origin to the city of destination (excluding the cities of origin and destination).

Set of cities from the map:

$$C = \{1, \cdots, N\}$$

City of origin: $a \in C$

City of destination: $b \in C$

Design variable:

Vector $\mathbf{x}, \text{size}(\mathbf{x}) > 0$

$\forall i \in \{1, \cdots, \text{size}(\mathbf{x})\}, \quad x_i \in C$

Routing Problem: Formalising the Design Variable

Design variable:

1-d array $\mathbf{x}$ of any size containing the sequence of cities (from the map) to be visited from the city of origin to the city of destination (excluding the cities of origin and destination).

Set of cities from the map:

$$C = \{1, \cdots, N\}$$

City of origin: $a \in C$

City of destination: $b \in C$

Design variable:

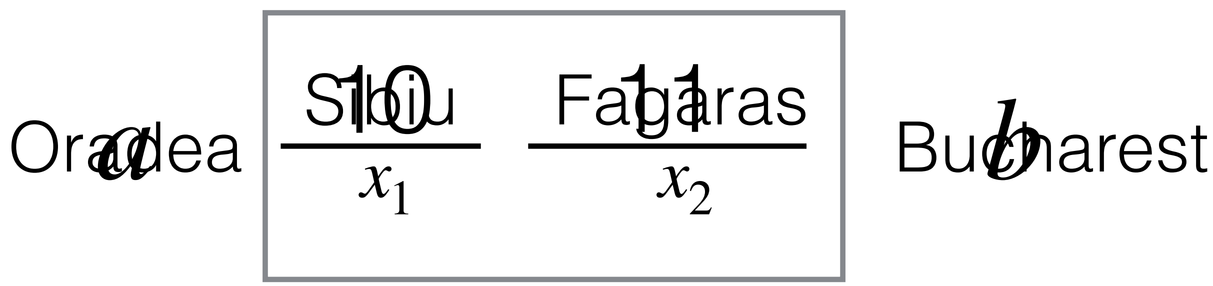
Vector $\mathbf{x}, \text{size}(\mathbf{x}) > 0$

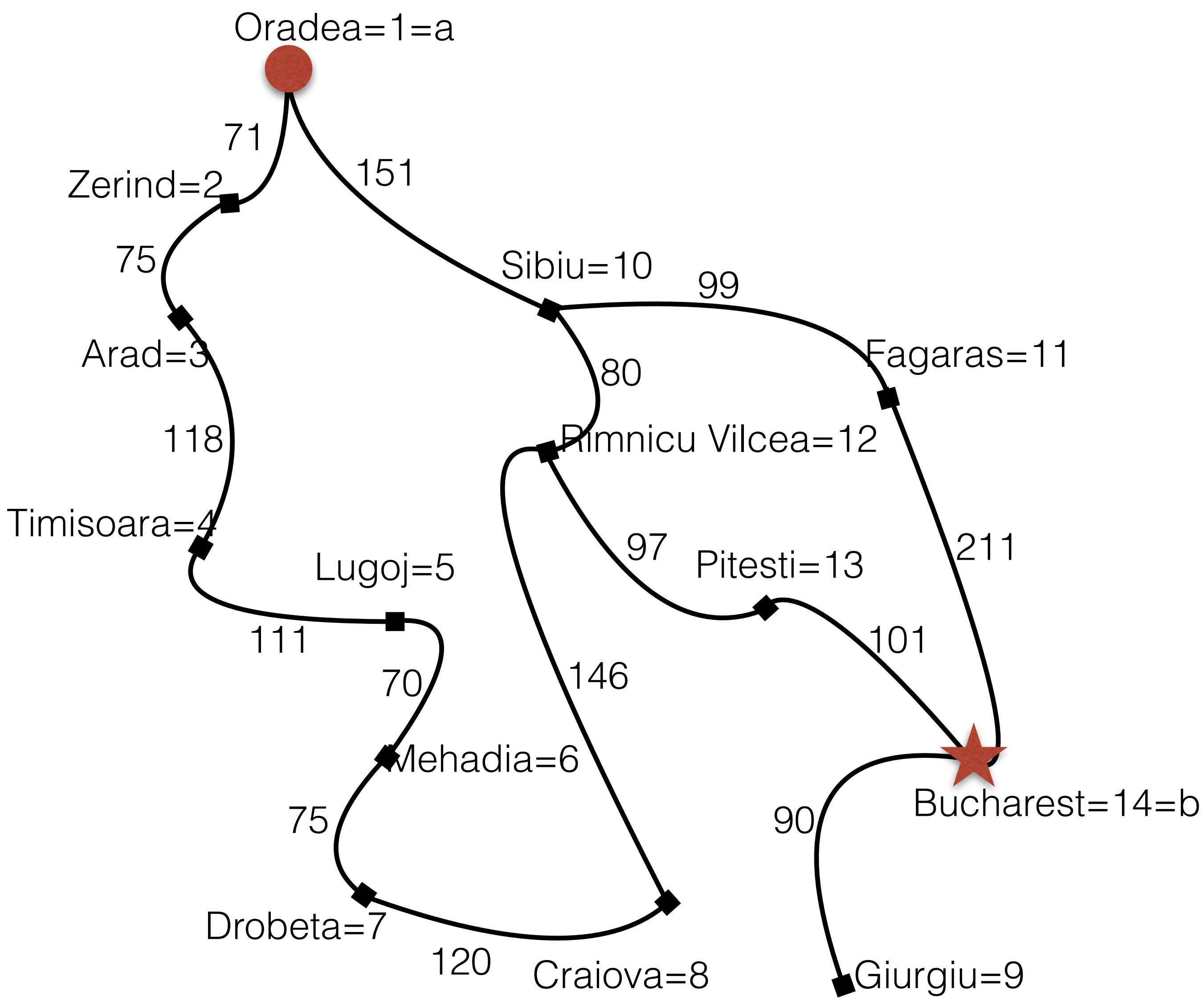
$$\forall i \in \{1, \cdots, \text{size}(\mathbf{x})\}, \quad x_i \in C \backslash \{a, b\}$$

We could potentially use $\geq$, but this will complicate our problem formulation

Routing Problem: Formalising the Design Variable

Design variable:





Set of cities from the map:

$$C = \{1, \cdots, N\}$$

City of origin: $a \in C$

City of destination: $b \in C$

Design variable:

Vector $\mathbf{x}, \text{size}(\mathbf{x}) > 0$

$\forall i \in \{1, \cdots, \text{size}(\mathbf{x})\}, \quad x_i \in C \backslash \{a, b\}$

Routing Problem: Formalising the Objective Function

Objective function:

Minimise the sum of the distances between consecutive cities in the path obtained through the solution.

function pathDistance($\mathbf{x}, a, b, D$)

dist = D_{a,x_1}

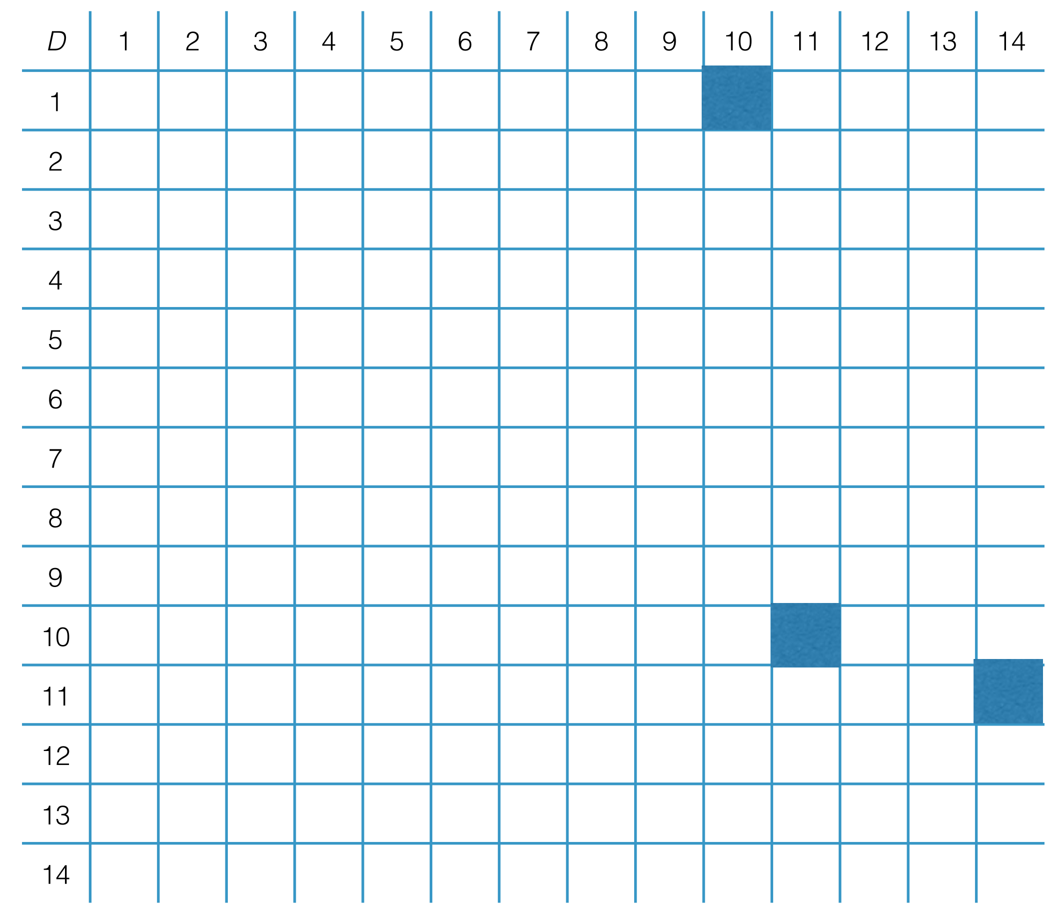
for i=1 to $size(\mathbf{x}) - 1$

dist = dist + $D_{x_i,x_{i+1}}$

dist = dist + $D_{x_{size(\mathbf{x})},b}$

return dist

$a \quad \underset{x_1}{\underline{10}} \quad \underset{x_2}{\underline{11}} \quad b$



Assume we have a matrix (2-d array) D of distances, with each position $D_{i,j}$ containing

- the distance in km to travel directly between city i and j, or
- -1 if such direct path does not exist.

Routing Problem: Formalising the Objective Function

Objective function:

Minimise the sum of the distances between consecutive cities in the path obtained through the solution.

function pathDistance($\mathbf{x}, a, b, D$)
    dist = D_{a,x_1}
    for i=1 to $size(\mathbf{x}) - 1$
        dist = dist + $D_{x_i,x_{i+1}}$
    dist = dist + $D_{x_{size(\mathbf{x})},b}$
    return dist

$$f(\mathbf{x}) = D_{a,x_1} + \left(\sum_{i=1}^{\text{size}(\mathbf{x})-1} D_{x_i,x_{i+1}} \right) + D_{x_{size(\mathbf{x})},b}$$

$$f(\mathbf{x}) = D_{a,x_1} + D_{x_{size(\mathbf{x})},b} + \sum_{i=1}^{\text{size}(\mathbf{x})-1} D_{x_i,x_{i+1}}$$

Note that this objective function doesn't work well when an inexistent direct path is used, but this is ok because constraints will be defined next.

Routing Problem: Formalising the Constraints

- Solutions must satisfy certain constraints, which define solution feasibility.
  - (Inexistent) direct paths between non-neighbouring cities must not be used (explicit constraint).
  - We must start at the city of origin and end at the city of destination (implicit constraint).
  - Only cities in the map can be used (implicit constraint).

Minimise $f(\mathbf{x})$

Subject to $g_i(\mathbf{x}) \leq 0, \quad i = 1, \cdots, m$

$h_j(\mathbf{x}) = 0, \quad j = 1, \cdots, n$

Routing Problem: Formalising the Constraints

Explicit constraint: (Inexistent) direct paths between non-neighbouring cities must not be used.

function violateNeighbourConstraint($\mathbf{x}, a, b, D$)

    if D_{a,x_1}== -1 or $D_{x_{size(\mathbf{x})},b}$==-1

        return 1 // true

    for i=1 to $size(\mathbf{x}) - 1$

        if $D_{x_i,x_{i+1}}$== -1

            return 1 // true

    return 0 // false

$$a \quad \underset{x_1}{\underline{\ 10\ }} \quad \underset{x_2}{\underline{\ 11\ }} \quad b$$

Routing Problem: Formalising the Constraints

(Inexistent) direct paths between non-neighbouring cities must not be used (explicit constraint).

function violateNeighbourConstraint($\mathbf{x}, a, b, D$)

- if D_{a,x_1}== -1 or $D_{x_{size(\mathbf{x})},b}$==-1
  - return 1 // true
- for i=1 to $size(\mathbf{x}) - 1$
  - if $D_{x_i,x_{i+1}}$== -1
    - return 1 // true
- return 0 // false

$$h(\mathbf{x}) = \begin{cases} 1, & \text{if } D_{a,x_1} = -1 \text{ or } D_{x_{size(\mathbf{x})},b} = -1 \\ 1, & \text{if } \exists i \in \{1,\ldots,\text{size}(\mathbf{x}) - 1\}, D_{x_i,x_{i+1}} = -1, \\ 0, & \text{otherwise} \end{cases}$$

$$a \quad \underset{x_1}{\underline{\ 10\ }} \quad \underset{x_2}{\underline{\ 11\ }} \quad b$$

Routing Problem: Formalising the Constraints

Solutions must satisfy certain constraints, which define solution feasibility.

- (Inexistent) direct paths between non-neighbouring cities must not be used (explicit constraint). $h(\mathbf{x}) = 0$
- We must start at the city of origin and end at the city of origin at the city of destination (implicit constraint).
- Only cities in the map can be used (implicit constraint).

Vector $\mathbf{x}, \text{size}(\mathbf{x}) \geq 0$

$$\forall i \in \{1, \cdots, \text{size}(\mathbf{x})\}, \quad x_i \in C \backslash \{a, b\}$$

$$a \quad \underset{x_1}{\underline{10}} \quad \underset{x_2}{\underline{11}} \quad b$$

$$f(\mathbf{x}) = D_{a,x_1} + D_{x_{size(\mathbf{x})},b} + \sum_{i=1}^{\text{size}(\mathbf{x})-1} D_{x_i,x_{i+1}}$$

Routing Problem Formulation

- Design variable:
  - Consider the set of cities from the map $C = \{1,\cdots,N\}$, where the city of origin is $a \in C$ and the city of destination is $b \in C$.
  - The design variable is a vector $\mathbf{x}$, size$(\mathbf{x}) > 0$, where $\forall i \in \{1,\cdots,\text{size}(\mathbf{x})\},\ \ x_i \in C \backslash \{a, b\}$.

- Objective function:

$$\text{minimise} \quad f(\mathbf{x}) = D_{a,x_1} + D_{x_{size(\mathbf{x})},b} + \sum_{i=1}^{\text{size}(\mathbf{x})-1} D_{x_i,x_{i+1}}$$

where D is an $N \times N$ matrix where $\forall i, j \in \{1,\cdots,N\}$, $D_{i,j}$ is the distance in km to travel from city i to city j, or -1 if no such direct path exists.

- Constraint:

$$h(\mathbf{x}) = 0 \text{ where } \quad h(\mathbf{x}) = \begin{cases} 1, & \text{if } D_{a,x_1} = \text{-1 or } D_{x_{size(\mathbf{x})},b} = \text{-1} \\ 1, & \text{if } \exists i \in \{1,\ldots,\text{size}(\mathbf{x}) - 1\}, D_{x_i,x_{i+1}} = -1, \\ 0, & \text{otherwise} \end{cases}$$

minimise $f(\mathbf{x}) = D_{a,x_1} + D_{x_{size(\mathbf{x})},b} + \sum_{i=1}^{\text{size}(\mathbf{x})-1} D_{x_i,x_{i+1}}$

subject to $h(\mathbf{x}) = 0$

where $\text{size}(\mathbf{x}) > 0$; $\forall i \in \{1,\cdots,\text{size}(\mathbf{x})\},\ x_i \in C \backslash \{a,b\}$;

$C = \{1,\cdots,N\}$ are the cities in the map

a and b are the cities of origin and destination, respectively;

D is an $N \times N$ matrix where $\forall i,j \in \{1,\cdots,N\}$, $D_{i,j}$ is the distance in km to travel from city i to city j, or -1 if no direct path exists between these cities; and

$$h(\mathbf{x}) = \begin{cases} 1, & \text{if } D_{a,x_1} = -1 \text{ or } D_{x_{size(\mathbf{x})},b} = -1 \\ 1, & \text{if } \exists i \in \{1,\ldots,\text{size}(\mathbf{x})-1\}, D_{x_i,x_{i+1}} = -1, \\ 0, & \text{otherwise} \end{cases}$$

This is one way to formulate this problem. There are other formulations that are equivalent to this.

Search and Optimisation

- In search, we are interested in searching for a goal state by taking feasible actions (possibly, while minimising cost).
- In optimisation, we are interested in searching for an optimal solution (possibly, while satisfying constraints).
- As many search problems have a cost associated to actions, they can also be formulated as optimisation problems.
- Similarly, optimisation problems can frequently be formulated as search problems associated to a cost function.
- Many search algorithms will "search" for optimal solutions (see A* as an example).
- Optimisation algorithms may also be used to solve search problems if they can be associated to an appropriate function to be optimised.

Summary

- We can formulate an optimisation problem by specifying:
  - Design variables.
  - Objective functions.
  - Constraints.

- There is a close relationship between search and optimisation problems.

Next

- How to solve optimisation problems?