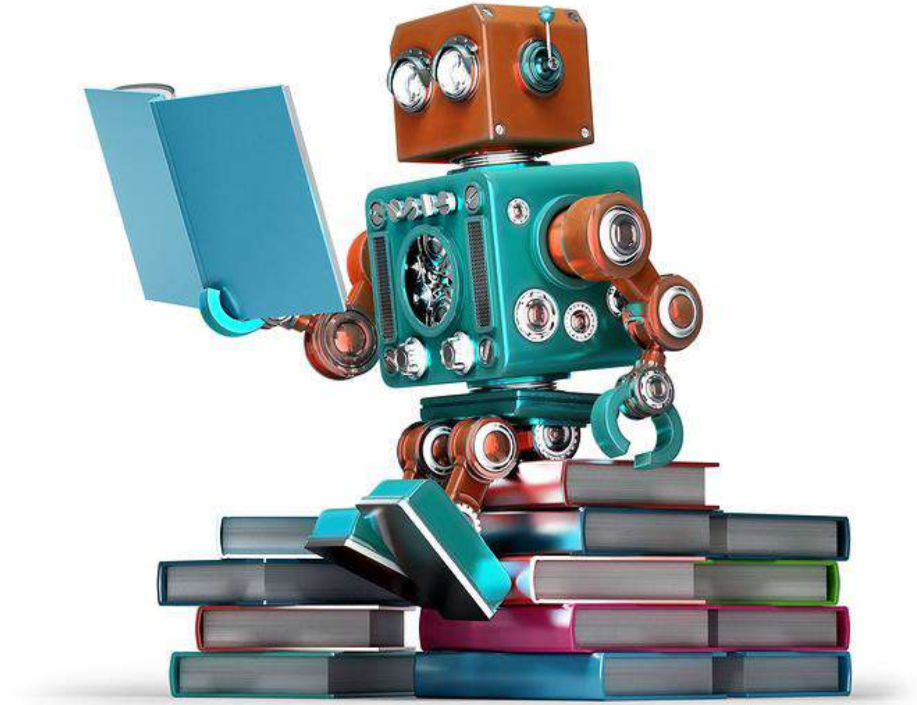
REASONING SYSTEMS
DAY 1



https://robohub.org/wp-content/uploads/2016/11/bigstock-Retro-Robot-Reading-A-Book-Is-110707406.jpg

DAY 1 AGENDA

1.1 Reasoning Systems Overview

1.2 Uninformed Search Techniques

1.3 Informed Search Techniques (part 1/2)

1.4 Search Representation **Workshop**

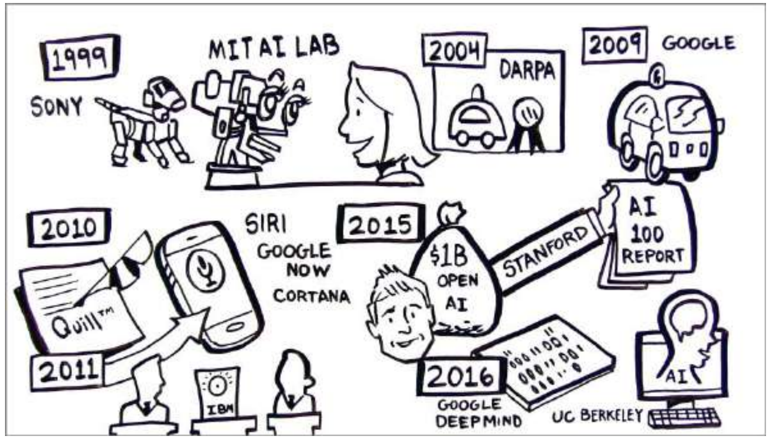
DAY 1 TIMETABLE

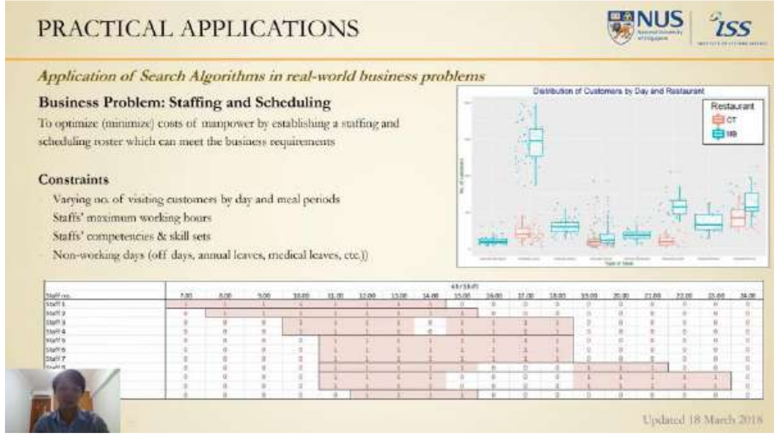
No	Time	Topic	By Whom	Where
1	9 am	Welcome and Introduction	GU Zhan (Sam)	Class
2	9.30 am	1.1 Reasoning Systems Overview	GU Zhan (Sam)	Class
3	10.10 am	Morning Break		
4	10.30 am	1.2 Uninformed Search Techniques	GU Zhan (Sam)	Class
5	12.10 pm	Lunch Break		
6	1.30 pm	1.3 Informed Search Techniques (part 1/2)	All	Class
7	3.10 pm	Afternoon Break		
8	3.30 pm	1.4 Search Representation Workshop	All	Class
9	4.50 pm	Summary and Review	All	Class
10	5 pm	End		

1.1
REASONING SYSTEMS OVERVIEW

1.1 REASONING SYSTEMS OVERVIEW

- ## AI History





PRACTICAL APPLICATIONS

Application of Search Algorithms in real-world business problems

Business Problem: Staffing and Scheduling

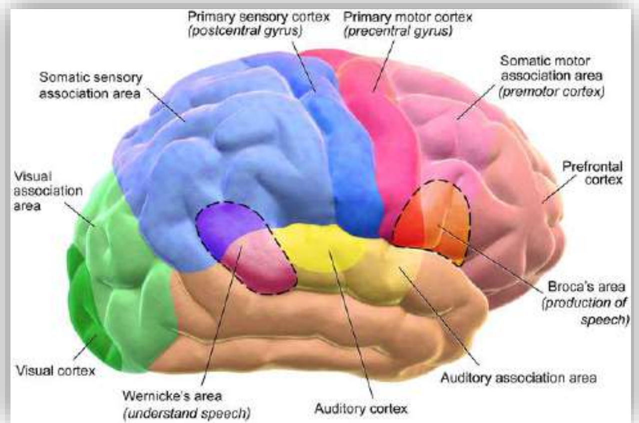
To optimize (minimize) costs of manpower by establishing a staffing and scheduling roster which can meet the business requirements

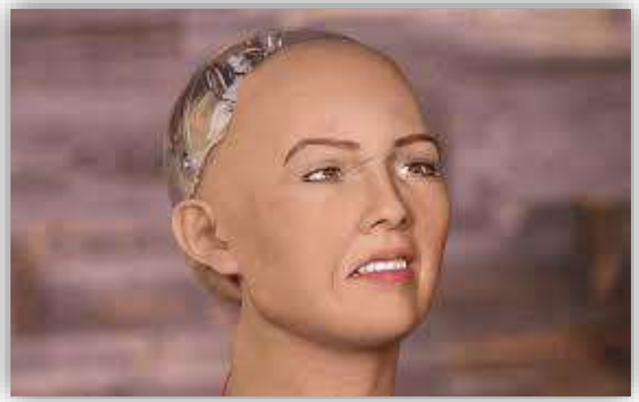
Constraints

- Varying no. of visiting customers by day and meal periods
- Staffs' maximum working hours
- Staffs' competencies & skill sets
- Non-working days (off days, annual leaves, medical leaves, etc.))

Updated 18 March 2018

1.1 REASONING SYSTEMS OVERVIEW

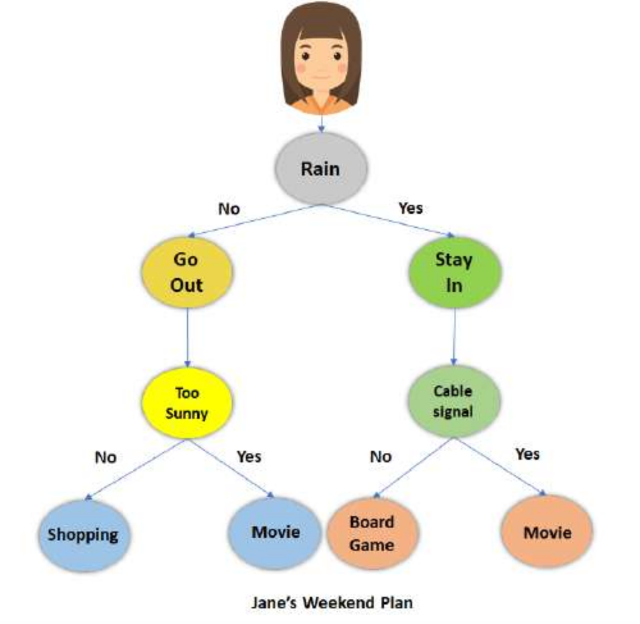

Primary sensory cortex (postcentral gyrus)
Primary motor cortex (precentral gyrus)
Somatic sensory association area
Somatic motor association area (premotor cortex)
Visual association area
Prefrontal cortex
Broca's area (production of speech)
Visual cortex
Auditory association area
Wernicke's area (understand speech)
Auditory cortex



Thinking vs Acting (acting = behaviour)

Human vs Rational (rationality = doing the right thing)

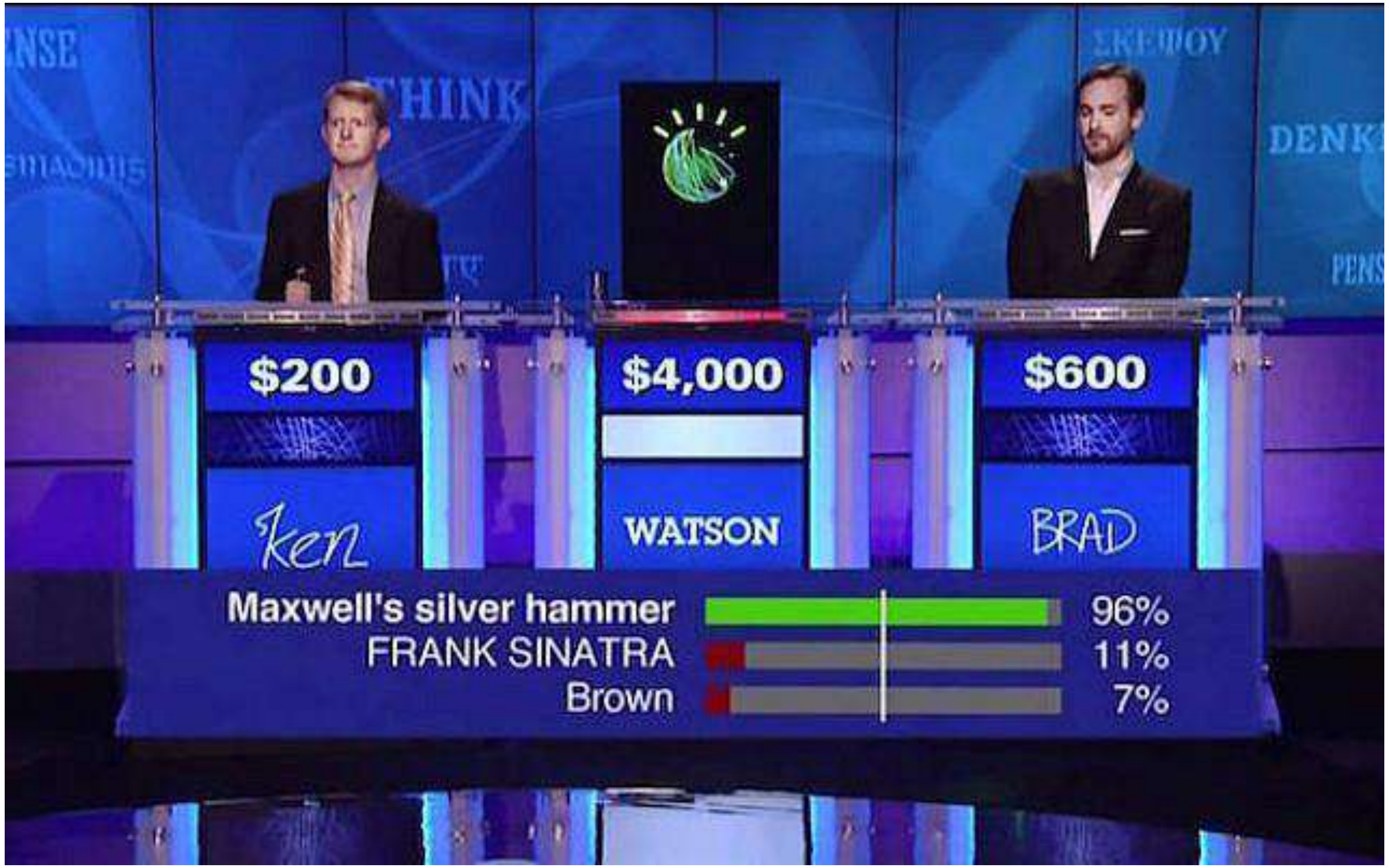
Systems that think like humans (cognitive science)	Systems that think rationally (logic/laws of thought)
Systems that act like humans (c.f. Turing test)	Systems that act rationally (agents)



Source https://slideplayer.com/slide/4644026/15/images/20/Systems+that+think+like+humans+%28cognitive+science%29.jpg

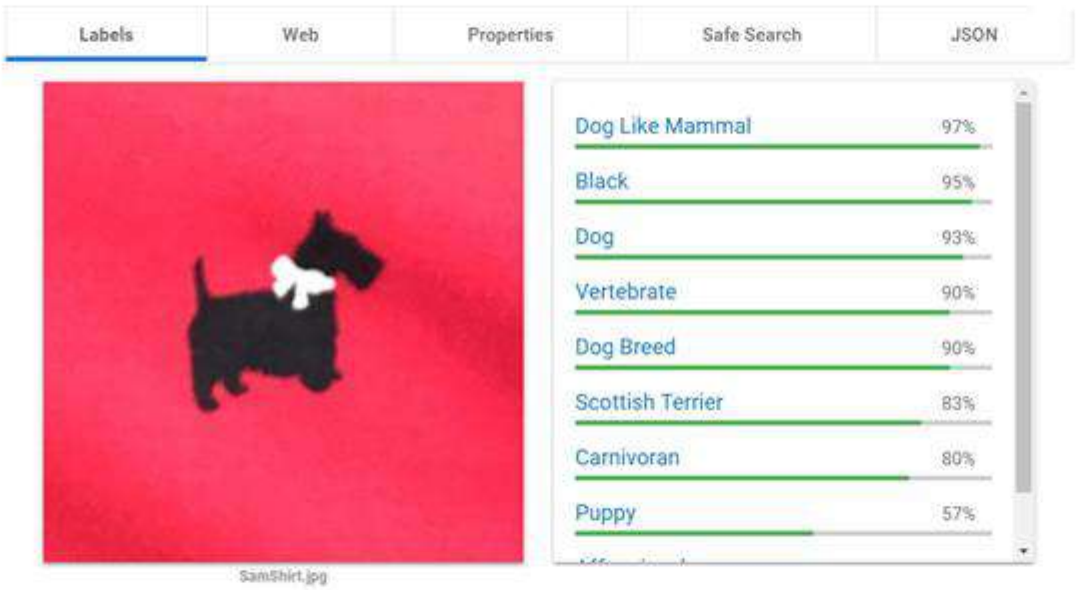
1.1 REASONING SYSTEMS OVERVIEW

- Question Answering System: IBM Watson



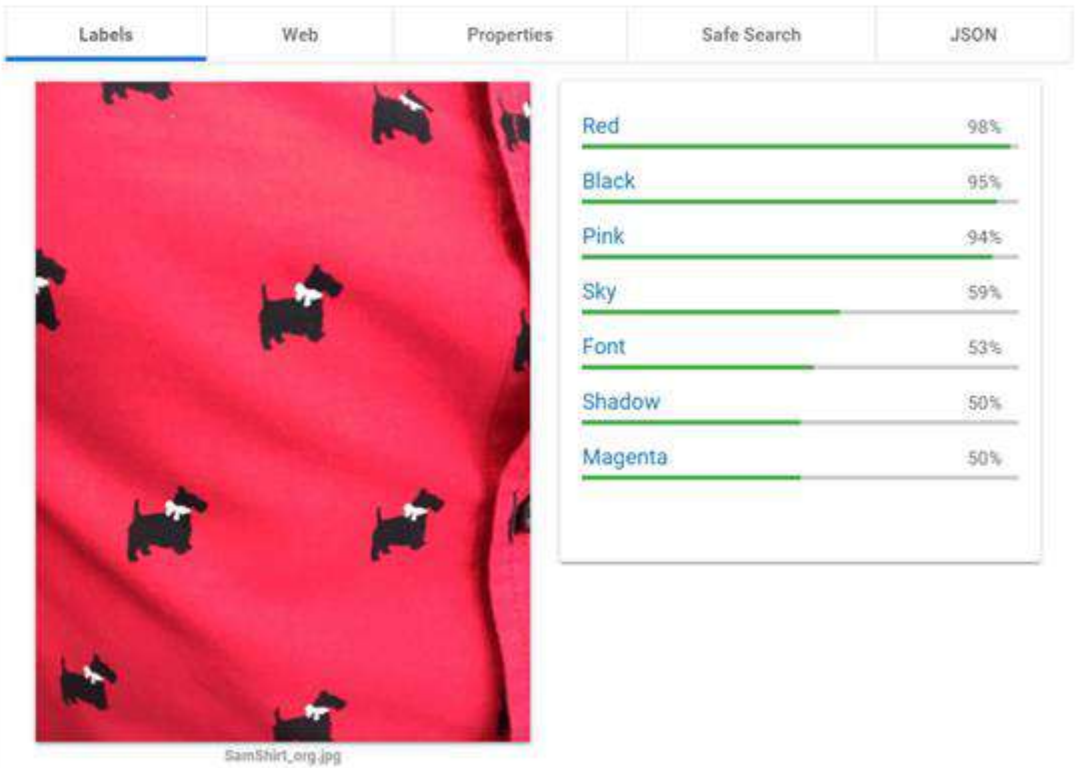
1.1 REASONING SYSTEMS OVERVIEW

- Image Object Recognition: Google Vision



Dog Like Mammal	97%
Black	95%
Dog	93%
Vertebrate	90%
Dog Breed	90%
Scottish Terrier	83%
Carnivoran	80%
Puppy	57%

SamShirt.jpg

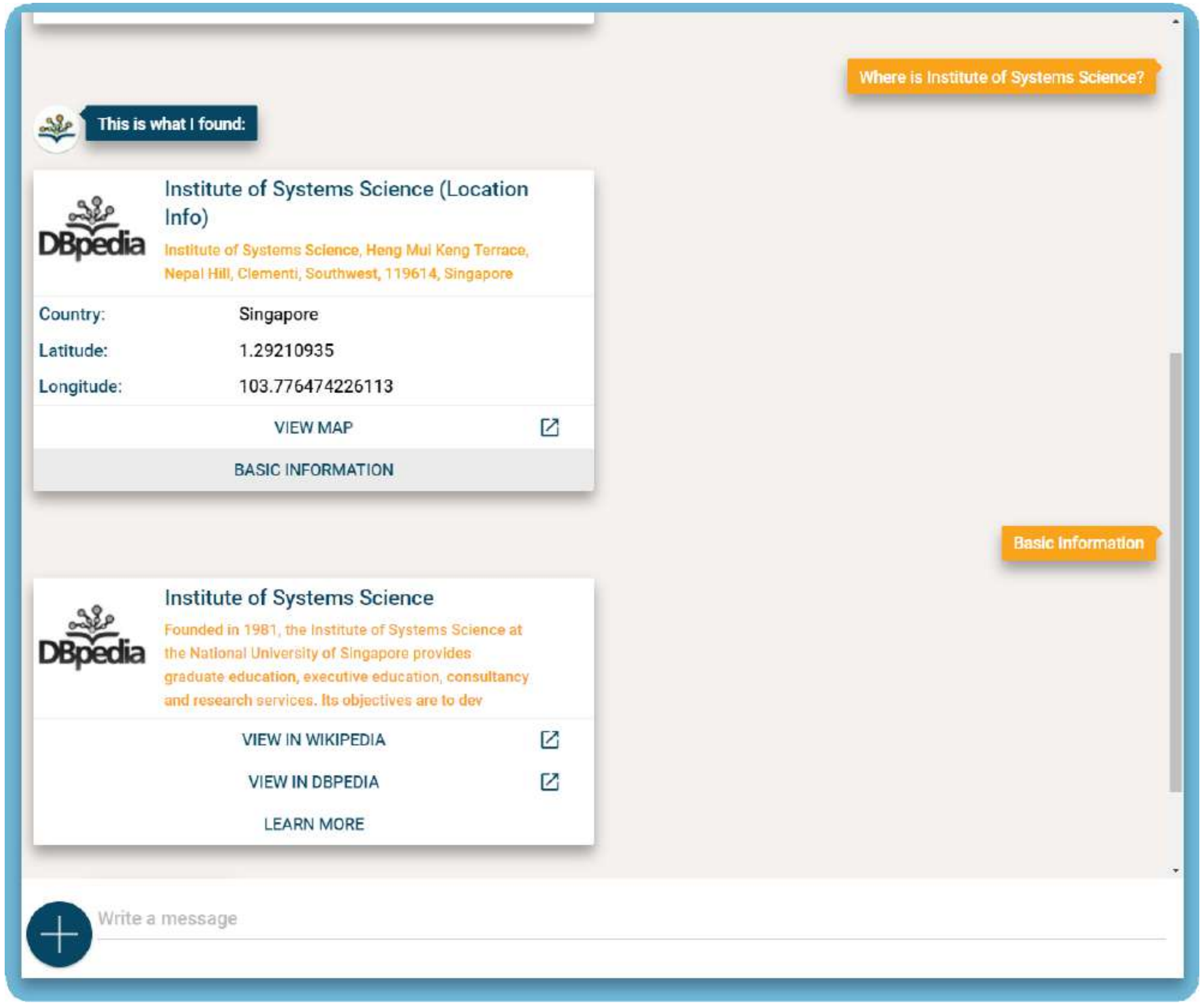


Red	98%
Black	95%
Pink	94%
Sky	59%
Font	53%
Shadow	50%
Magenta	50%

SamShirt_org.jpg

1.1 REASONING SYSTEMS OVERVIEW

- Chat-Bot: DBpedia



Where is Institute of Systems Science?

This is what I found:

Institute of Systems Science (Location Info)

Institute of Systems Science, Heng Mui Keng Terrace, Nepal Hill, Clementi, Southwest, 119614, Singapore

Country:	Singapore
Latitude:	1.29210935
Longitude:	103.776474226113

VIEW MAP

BASIC INFORMATION

Basic Information

Institute of Systems Science

Founded in 1981, the Institute of Systems Science at the National University of Singapore provides graduate education, executive education, consultancy and research services. Its objectives are to dev

VIEW IN WIKIPEDIA

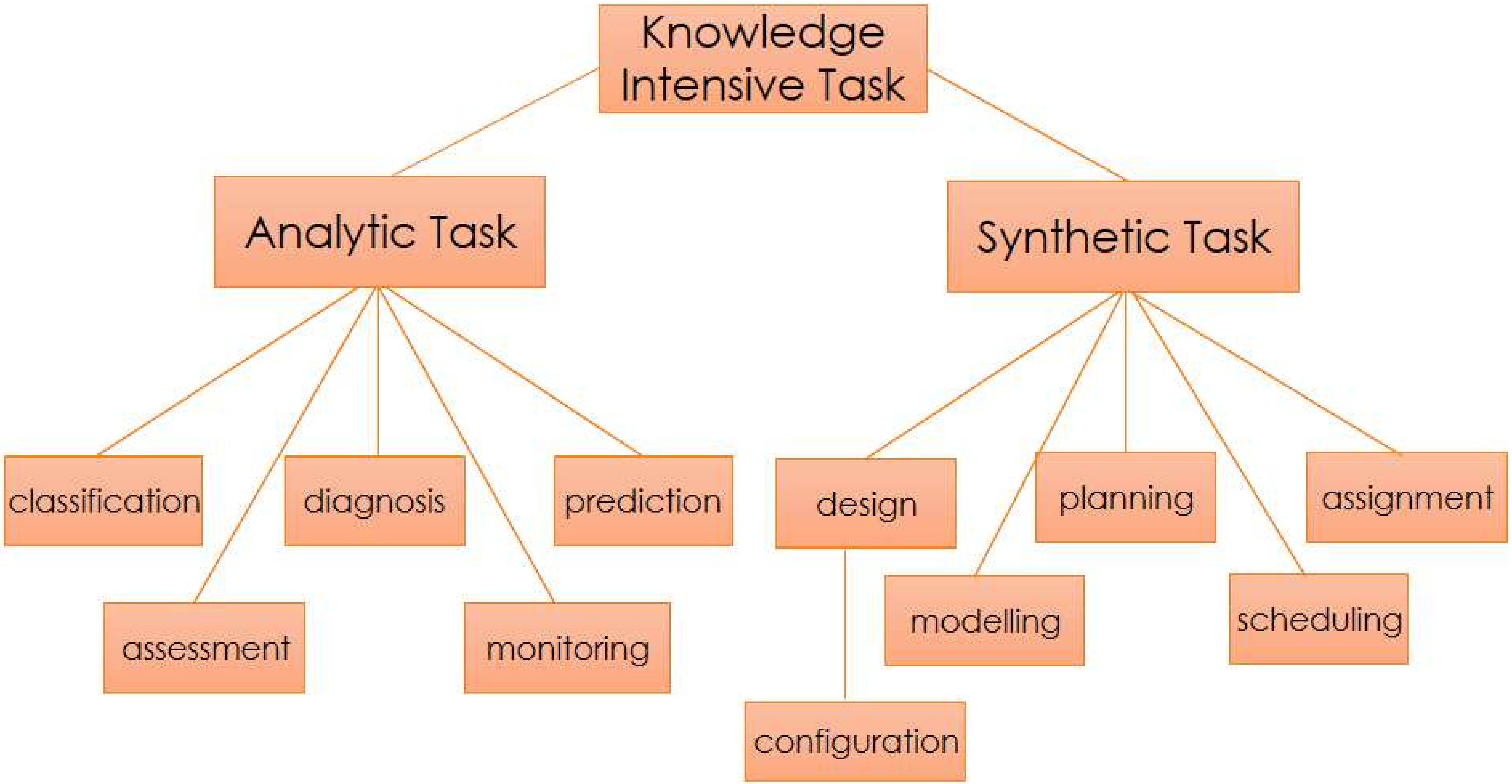
VIEW IN DBPEDIA

LEARN MORE

Write a message

1.1 REASONING SYSTEMS OVERVIEW

Problem Solving Task Hierarchy



1.1 REASONING SYSTEMS OVERVIEW

Problem Solving Task Types

- **Analytic Tasks**
  - System/Solution to be analysed pre-exists, but usually not completely "known“.
  - Input: some data to trigger the system (e.g. patient symptoms)
  - Output: some characterization or behaviours about the system (e.g. cause of illness)
- **Synthetic Tasks**
  - System/Solution does not yet exist.
  - Input: requirements about system to be constructed
  - Output: constructed system description

1.1 REASONING SYSTEMS OVERVIEW

Problem Solving of Analytic Tasks

- **Analytic Tasks**

  Identification, Classification, Prediction, Clustering/Grouping, …

- **Techniques (Machine Reasoning)**

  Heuristic Business Rules

  Decision Trees

  Case Based Reasoning

  Fuzzy Logic

  Rule Induction

  Machine Learning

  …

1.1 REASONING SYSTEMS OVERVIEW

Problem Solving of Synthetic Tasks

- **Synthetic Tasks**

  Planning, Scheduling, Optimisation, Design, …

- **Techniques (Reasoning Systems)**

  Uninformed (brute force / blind) Search

  Informed (heuristic) Search

  Simulations

  Genetic Algorithms

  Reinforcement Learning

  Data Mining

  …

1.2

UNINFORMED SEARCH TECHNIQUES

1.2 UNINFORMED SEARCH TECHNIQUES

- Solving Problem by Search
- Search Tree Representation
- Depth First Search (DFS)
- Breadth First Search (BFS)

https://modernmarketingtoday.com/wp-content/uploads/2013/02/search-marketing.jpg

1.2 UNINFORMED SEARCH TECHNIQUES

Solving Problem by Search

- Synthesis of a new valid solution is performed by searching through the (search/solution) space, which contains all possible solutions
- Each possible solution is evaluated to see whether it is valid and/or the optimum (best solution found by now), e.g. a valid employee schedule, a valid vehicle delivery route, an optimal (shortest) vehicle delivery route,
- Validity of solution involves satisfaction of a set of constraints on the solution variables
- Optimality is measured by a user-defined function which measures the “goodness” of the solution, e.g. the shorter delivery route the better.

1.2 UNINFORMED SEARCH TECHNIQUES

Solving Problem by Search

(1) Create a pool of solution candidates (search space)

(2) Pick up one candidate solution from pool

(3) Check whether this candidate is valid (constraints satisfied?)

(3)=True If valid, continue

(3)=False If not valid, go to (2)

(4) Check whether this candidate is the best till now (optimal solution?)

(4)=True If best, save this solution as the best then continue

(4)=False If not best, discard this solution then continue

(5) Go to (2). Repeat the cycle until a stopping criteria is met.

1.2 UNINFORMED SEARCH TECHNIQUES

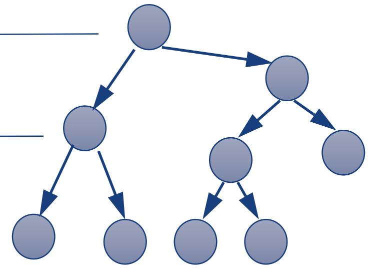
Search Tree Representation

- **Search is illustrated using a search space with a particular restricted structure**
- **Solutions (search space) can be represented as a Tree**
  - Nodes in tree represent

    an initial state

    an intermediate state

    a final state (feasible solution, or failure)



  - Connection between nodes represents a search step

Depth First Search (DFS)

- Always prefers to search deeper in the search tree rather than wider.

1.2 UNINFORMED SEARCH TECHNIQUES

Depth First Search (DFS)

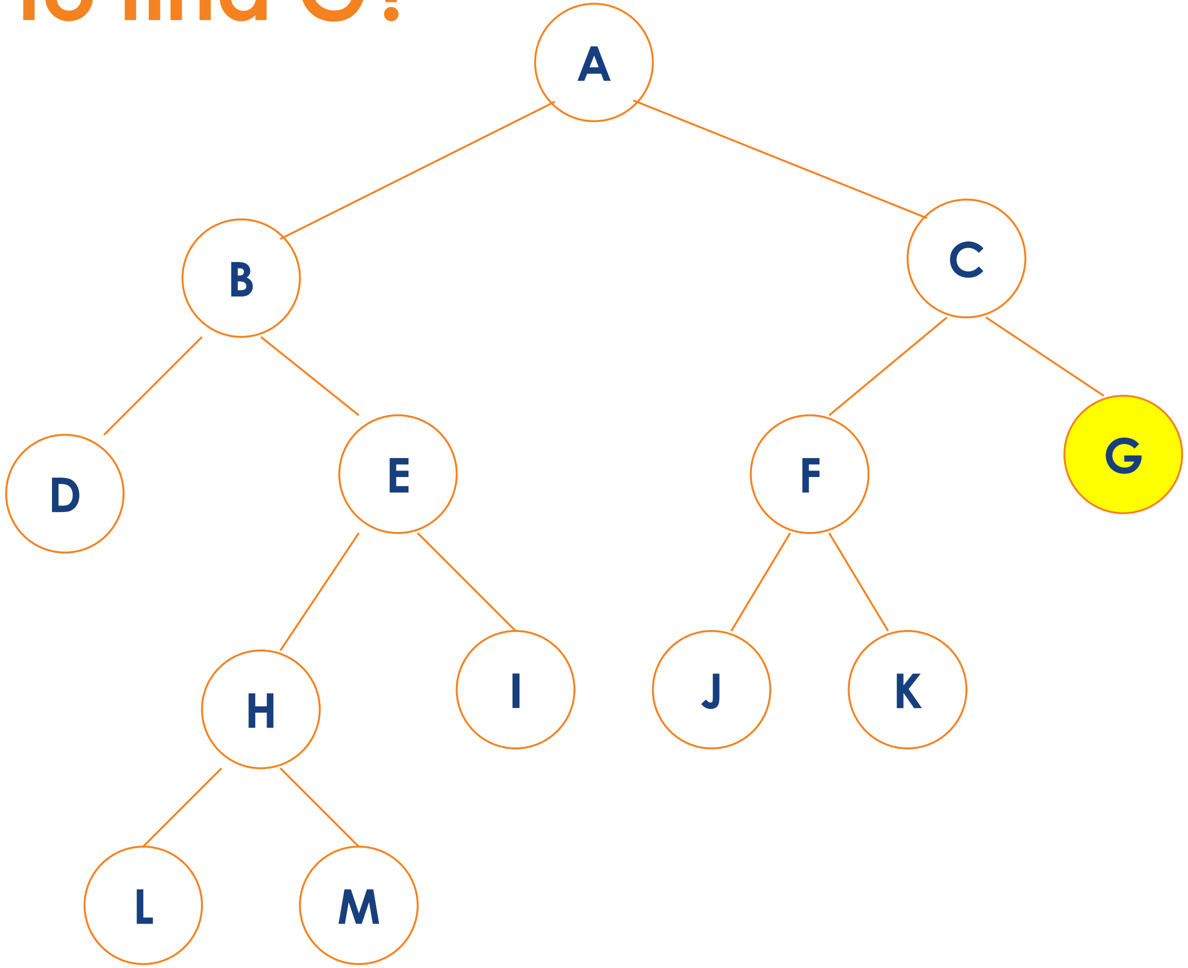
Algorithm Pseudo Code

```
(1) Set N to be a list of initial nodes
(2) If N is empty, then exit and signal failure
(3) Set n to be the first node in N, and remove n from N
(4) Check n:
    (4.1) If n is a goal node, then exit and signal success
    (4.2) Otherwise, add the children of n to the front of N then
    go to step (2)
```

1.2 UNINFORMED SEARCH TECHNIQUES

Depth First Search (DFS)

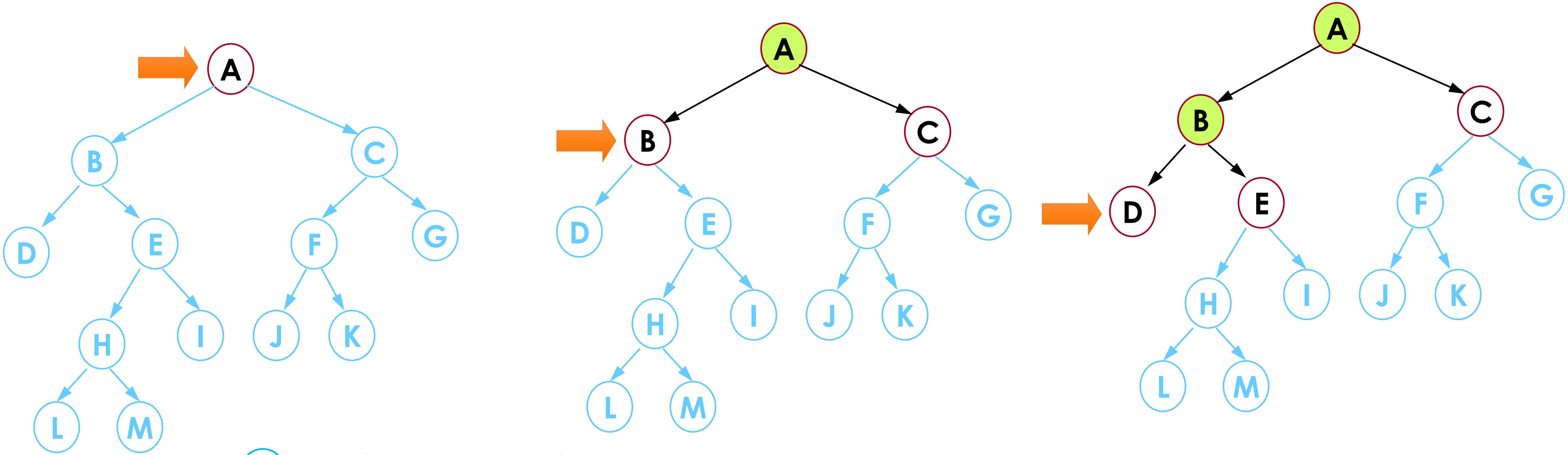
- Visit order to find G?



1.2 UNINFORMED SEARCH TECHNIQUES

Depth First Search (DFS)

- Keep track of nodes



○ Node unexplored

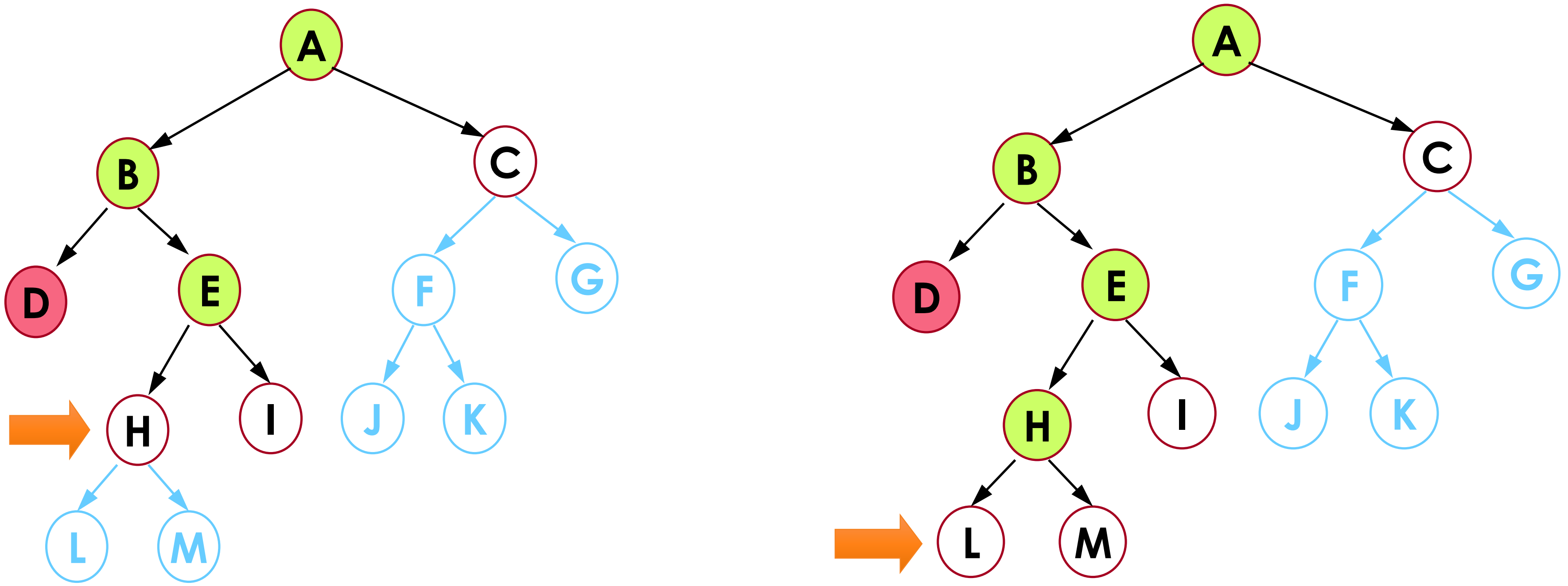
○ Node waiting to be explored

● Node already explored

1.2 UNINFORMED SEARCH TECHNIQUES

Depth First Search (DFS)

- Keep track of nodes

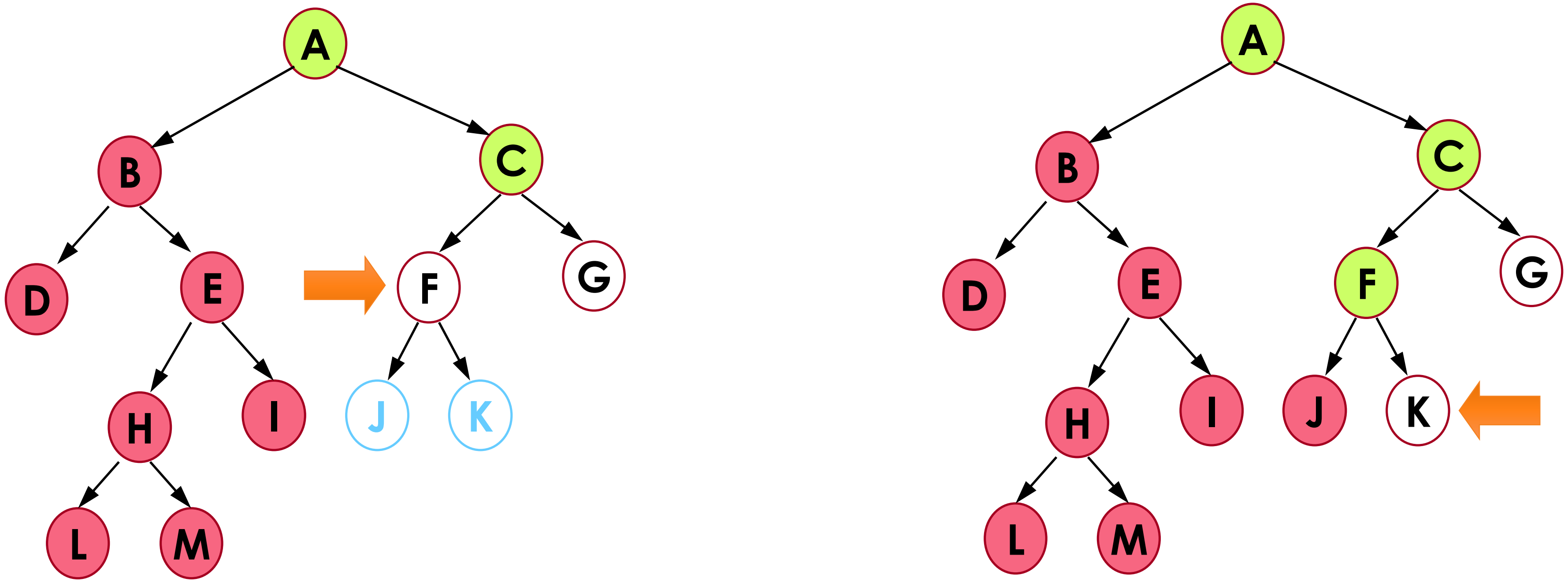


● Explored non-solution node can be removed

1.2 UNINFORMED SEARCH TECHNIQUES

Depth First Search (DFS)

- Keep track of nodes

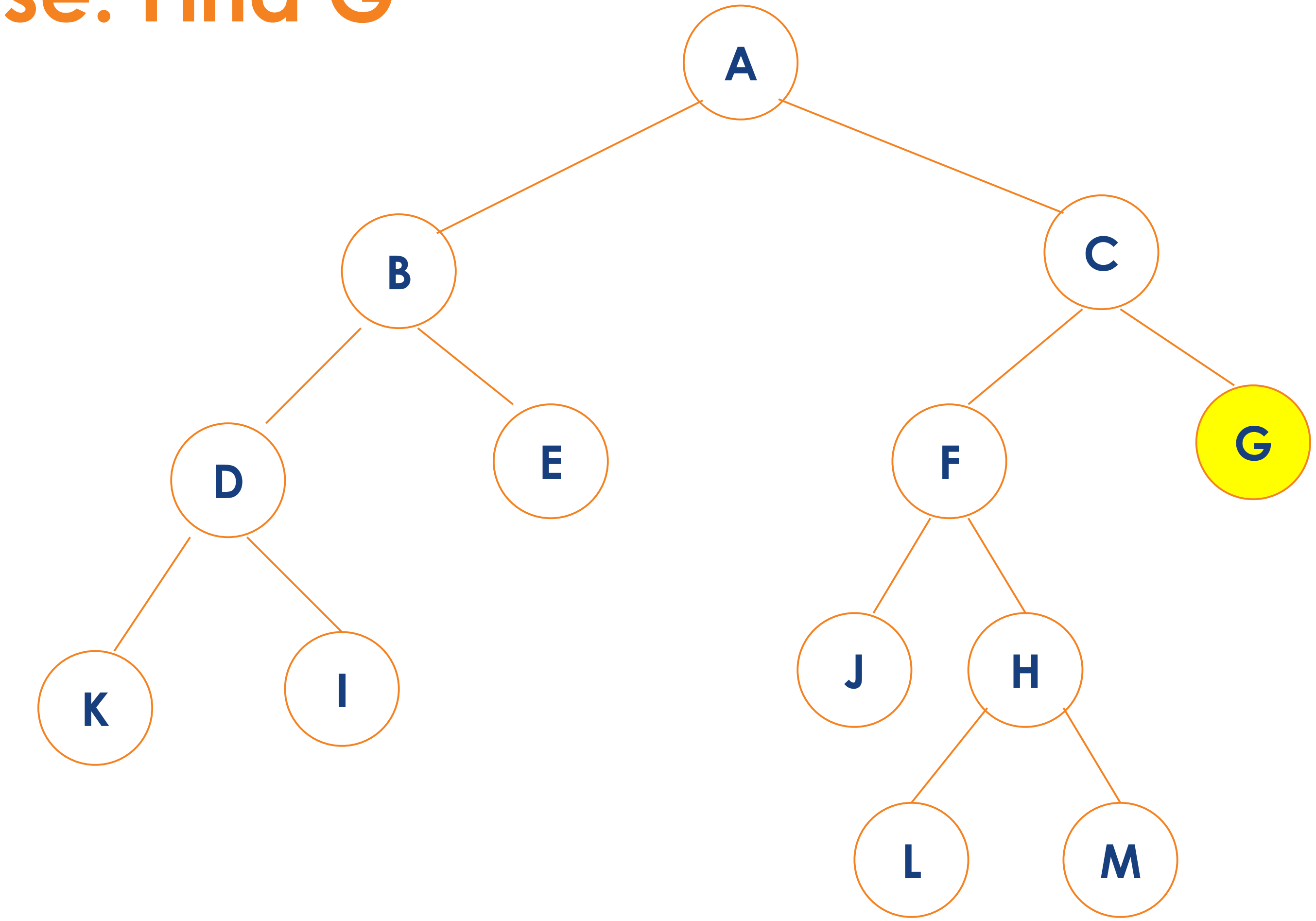


● Explored non-solution node/branch can be removed

1.2 UNINFORMED SEARCH TECHNIQUES

Depth First Search (DFS)

- Exercise: Find G



1.2 UNINFORMED SEARCH TECHNIQUES

Breadth First Search (BFS)

- Explores all the nodes at a given depth before processing deeper in the search tree.

1.2 UNINFORMED SEARCH TECHNIQUES

Breadth First Search (BFS)

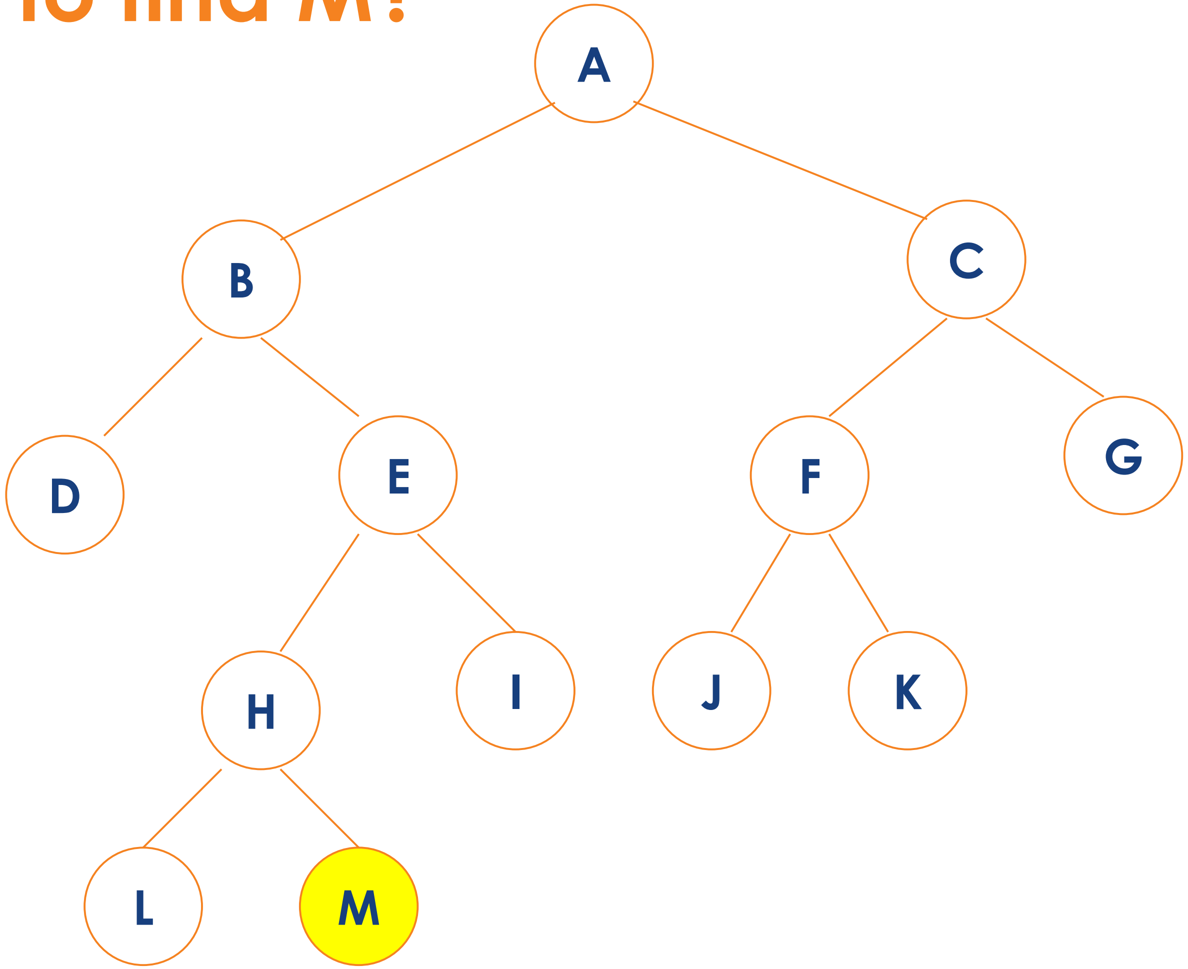
Algorithm Pseudo Code

```
(1) Set N to be a list of initial nodes
(2) If N is empty, then exit and signal failure
(3) Set n to be the first node in N, and remove n from N
(4) Check n:
    (4.1) If n is a goal node, then exit and signal success
    (4.2) Otherwise, add the children of n to the end of N then
    go to step (2)
```

1.2 UNINFORMED SEARCH TECHNIQUES

Breadth First Search (BFS)

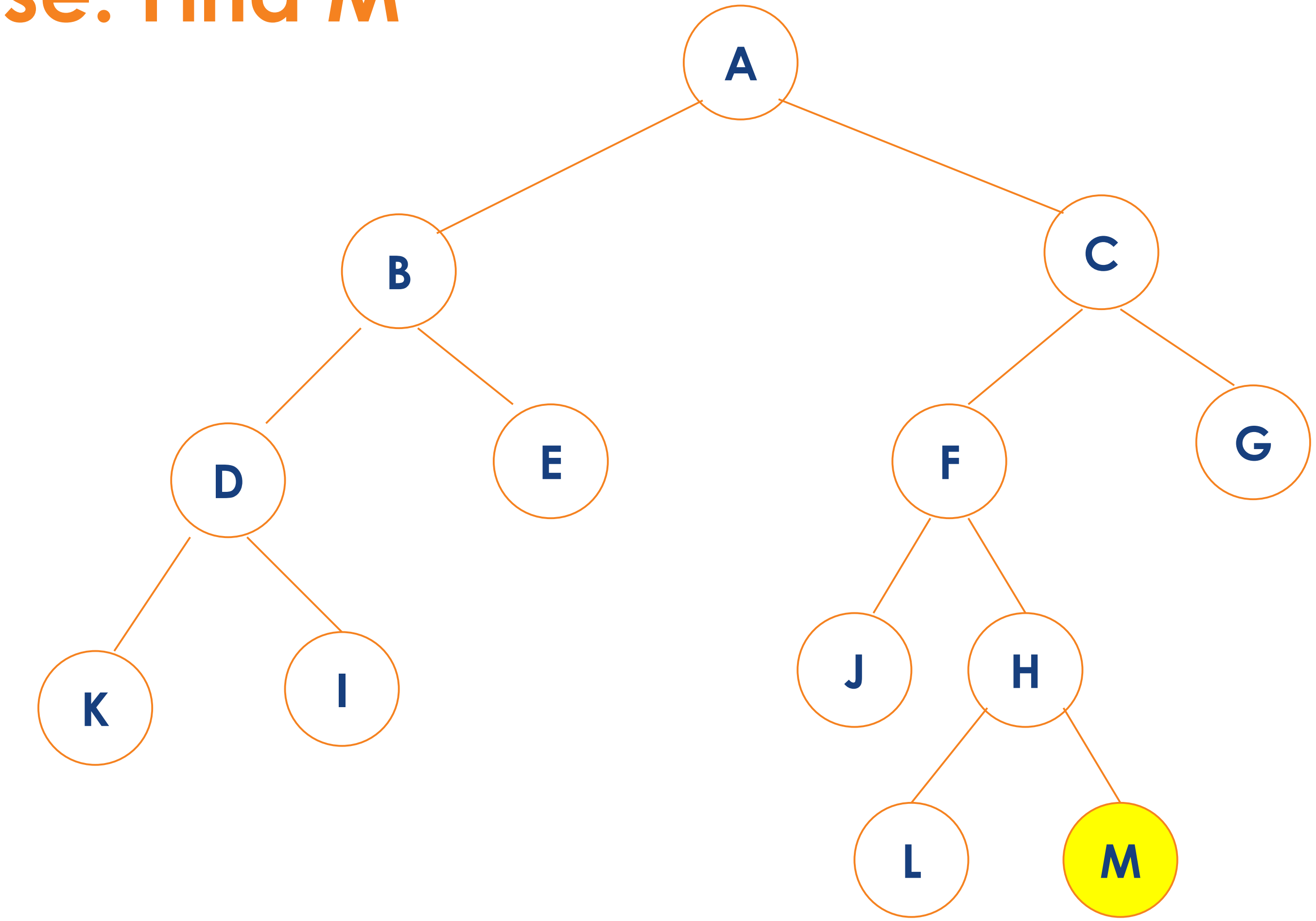
- Visit order to find M?



1.2 UNINFORMED SEARCH TECHNIQUES

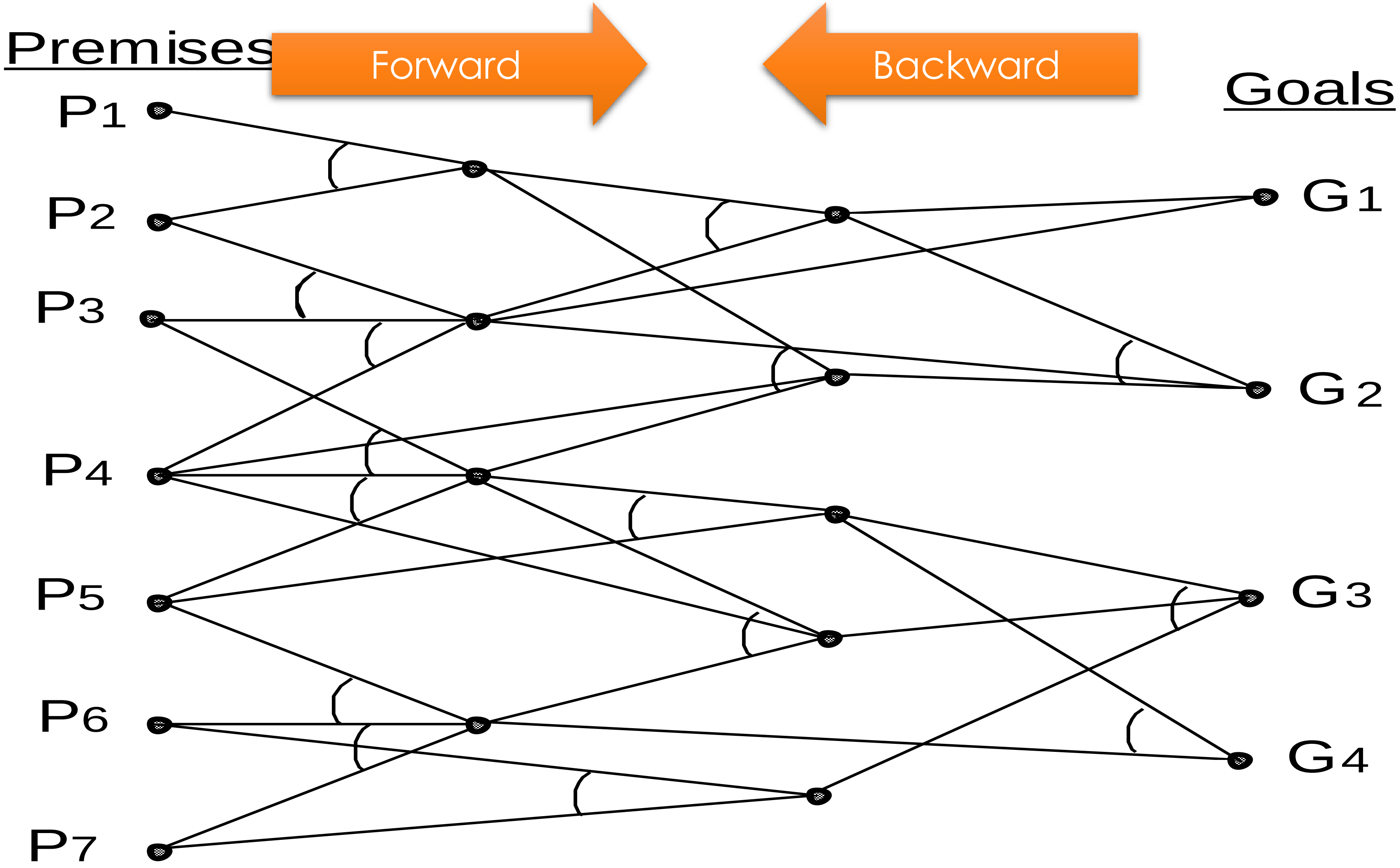
Breadth First Search (BFS)

- Exercise: Find M



1.2 UNINFORMED SEARCH TECHNIQUES

Forward Chaining (BFS) vs. Backward Chaining (DFS)



1.3

INFORMED SEARCH TECHNIQUES

(PART 1/2)

1.3 INFORMED SEARCH TECHNIQUES (1/2)

- Use Heuristics
- Hill Climbing Search (HC)
- A Star Search (A*)
- Tabu Search (TS)
- Simulated Annealing (SA)
- Informed Search Use Case

https://modernmarketingtoday.com/wp-content/uploads/2013/02/search-marketing.jpg

1.3 INFORMED SEARCH TECHNIQUES (1/2)

Use Heuristics

- **Basic Idea**
  - It works by firstly sorting the list of nodes, then explore them orderly, according to their **optimality** (best score) determined by an evaluation function ***f*(*n*)**
- **Typical Best-first Strategies**
  - Use heuristics only : Hill Climbing, Tabu search
  - Use heuristics and past cost : A*, Late Acceptance
  - Use heuristics and randomness : Simulated Annealing, GA

1.3 INFORMED SEARCH TECHNIQUES (1/2)

Use Heuristics

- A key component of evaluation function *f(n)* is a heuristic function: *h(n)* = estimated cost of the cheapest path from node n to a goal node. Or estimated degree of difference between the current states/solutions and ultimate goal state.
- By convention, the lower the heuristic value the more promising the node: better to check first. h(n) = 0 when n is goal
- ☺ When there is no other information available but only the heuristics, *f(n)* = *h(n)*

1.3 INFORMED SEARCH TECHNIQUES (1/2)

Use Heuristics

- **Heuristic Function (Knowledge)**
  - ***h(n)*** = Correct or Incorrect positions

Initial=

1	3	2
8		4
5	6	7

Goal=

1	2	3
8		4
7	6	5

  - ***h_1(n)*** = number of tiles in their **correct** goal state positions

1	3	2
8		4
5	6	7

= 4

  - ***h_2(n)*** = number of tiles in their **incorrect** goal state positions

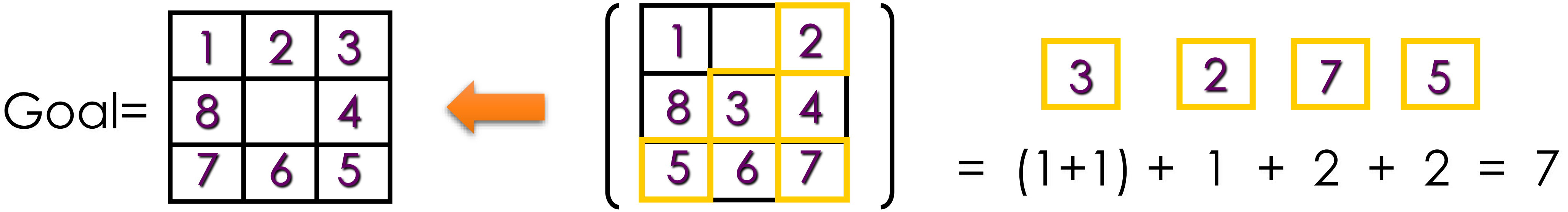
1	3	2
8		4
5	6	7

= 4

1.3 INFORMED SEARCH TECHNIQUES (1/2)

Use Heuristics

- **Heuristic Function (Knowledge)**
  - ***h(n)*** = Manhattan distance (sum of the horizontal & vertical distance each tile is away from its goal state position)



Goal=

1	2	3
8		4
7	6	5

1		2
8	3	4
5	6	7

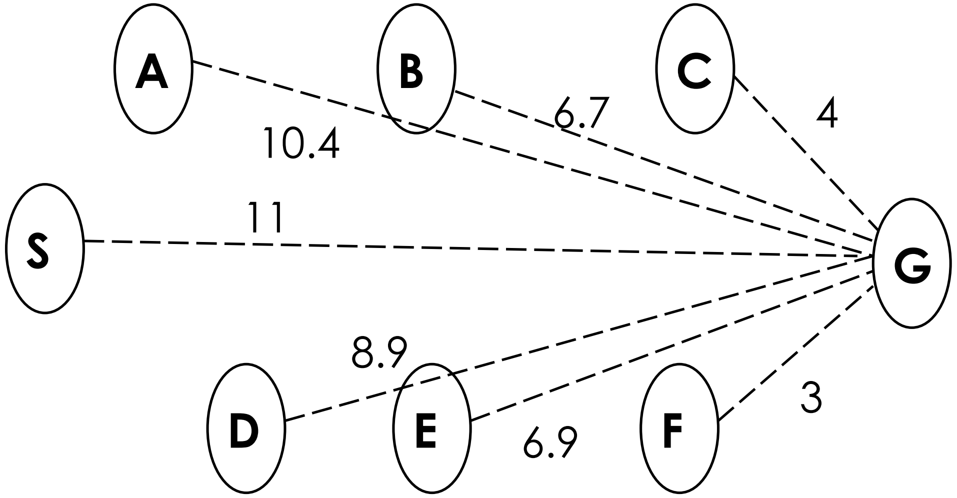
$$= \overset{\boxed{3}}{(1+1)} + \overset{\boxed{2}}{1} + \overset{\boxed{7}}{2} + \overset{\boxed{5}}{2} = 7$$

☺ Manhattan distance gives a better estimate of the distance to the Goal state

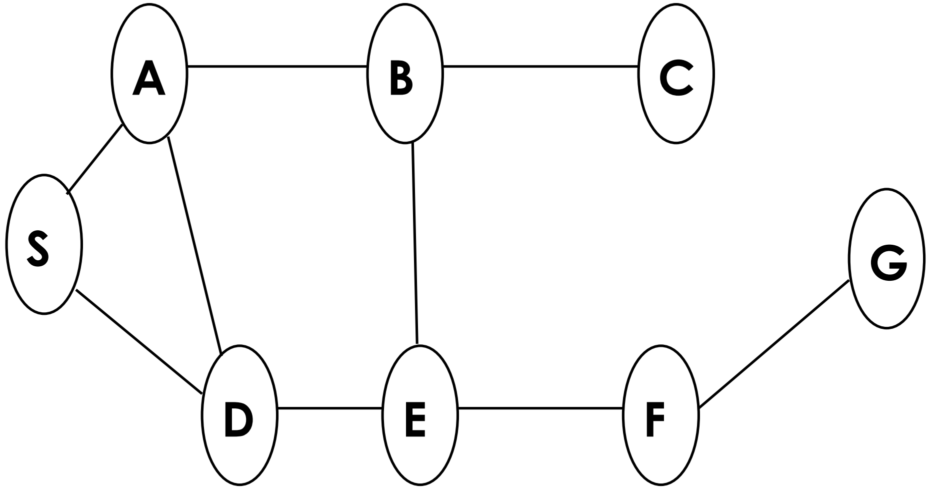
1.3 INFORMED SEARCH TECHNIQUES (1/2)

Use Heuristics

- **Heuristic Function (Knowledge)**
  - *h(n)* = "straight-line" distance between each city & the goal (This is useful estimation or heuristic.)



Heuristic Distances



Actual Roads

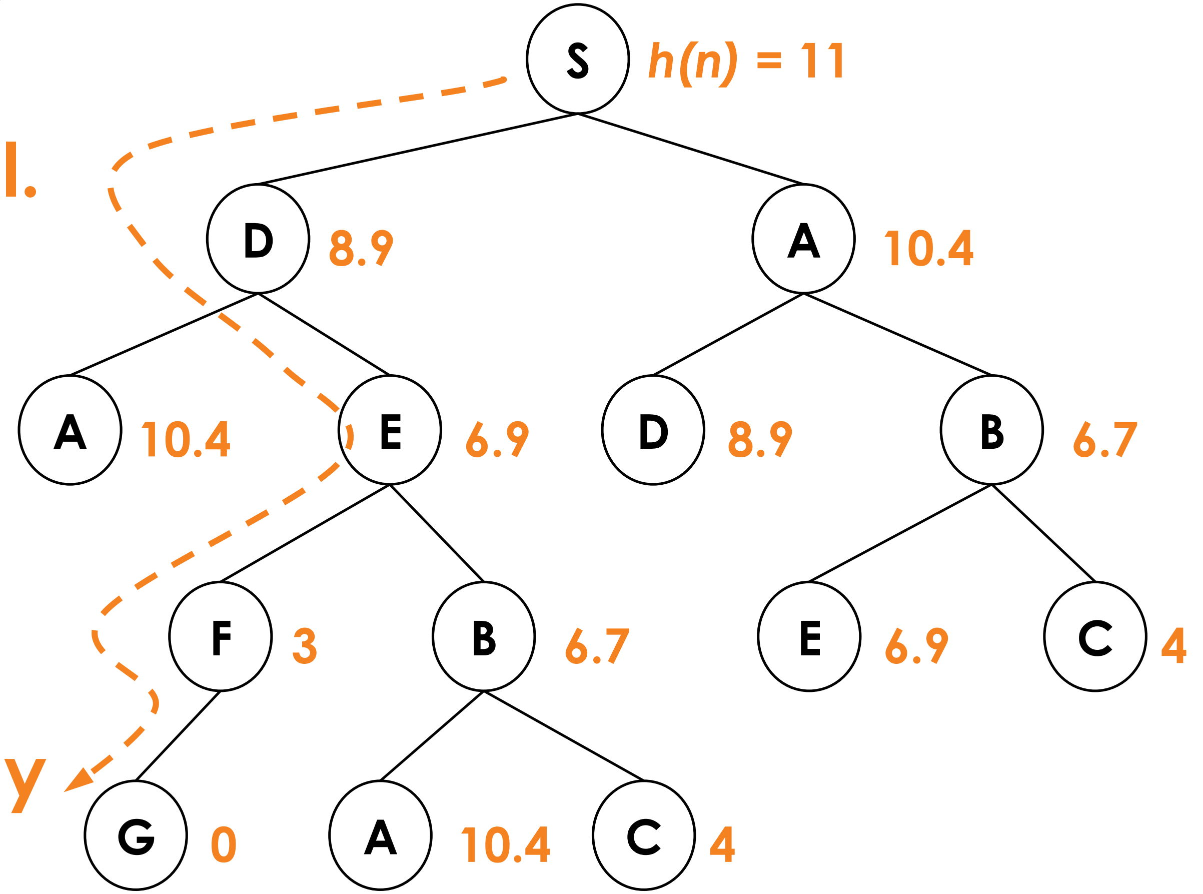
1.3 INFORMED SEARCH TECHNIQUES (1/2)

Hill Climbing (Greedy Best First Search)

- Minimize the cost to reach the goal state by expanding the node that is closest to the goal.
- Using only the heuristic values for evaluation function:

$$f(n) = h(n)$$

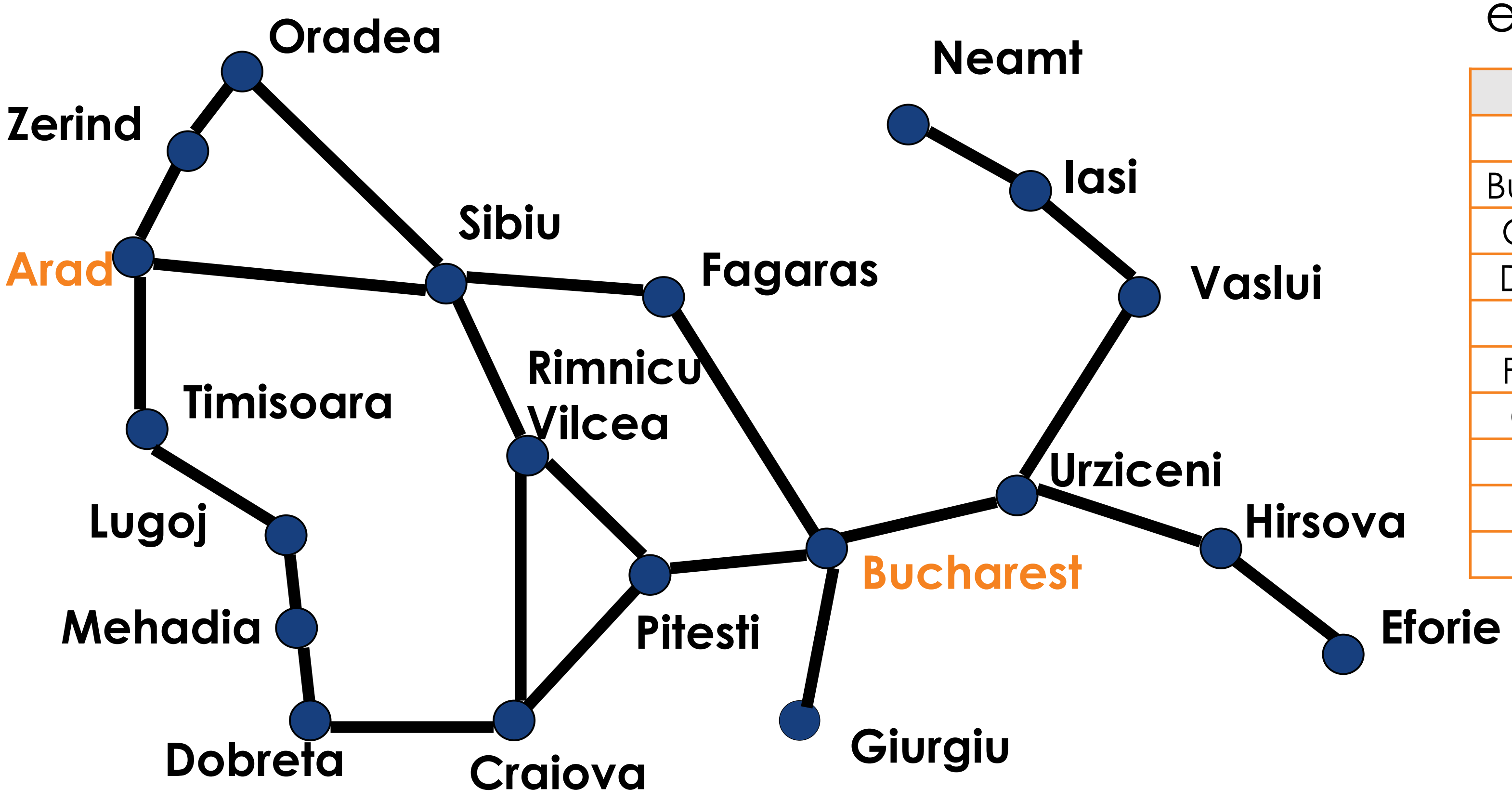
- Select search node with min($f(n)$) at each step.
- Follow a single path all the way to a goal, but can back track when it hits a dead end.



1.3 INFORMED SEARCH TECHNIQUES (1/2)

Hill Climbing (Greedy Best First Search)

- Arad → Bucharest

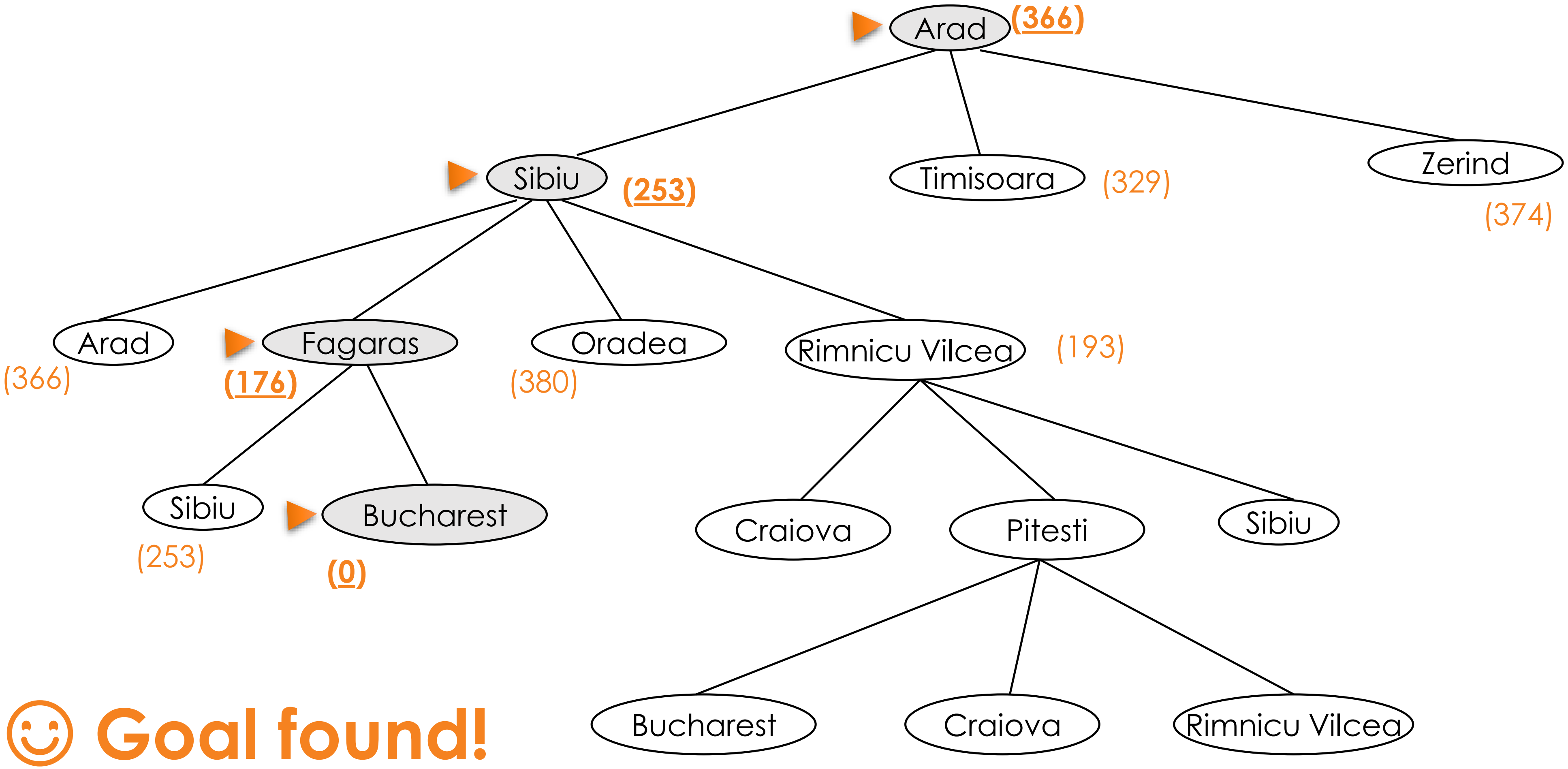


h(n) =Straight-line distance from each city to **Bucharest**

City	Distance	City	Distance
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

1.3 INFORMED SEARCH TECHNIQUES (1/2)

Hill Climbing (Greedy Best First Search)

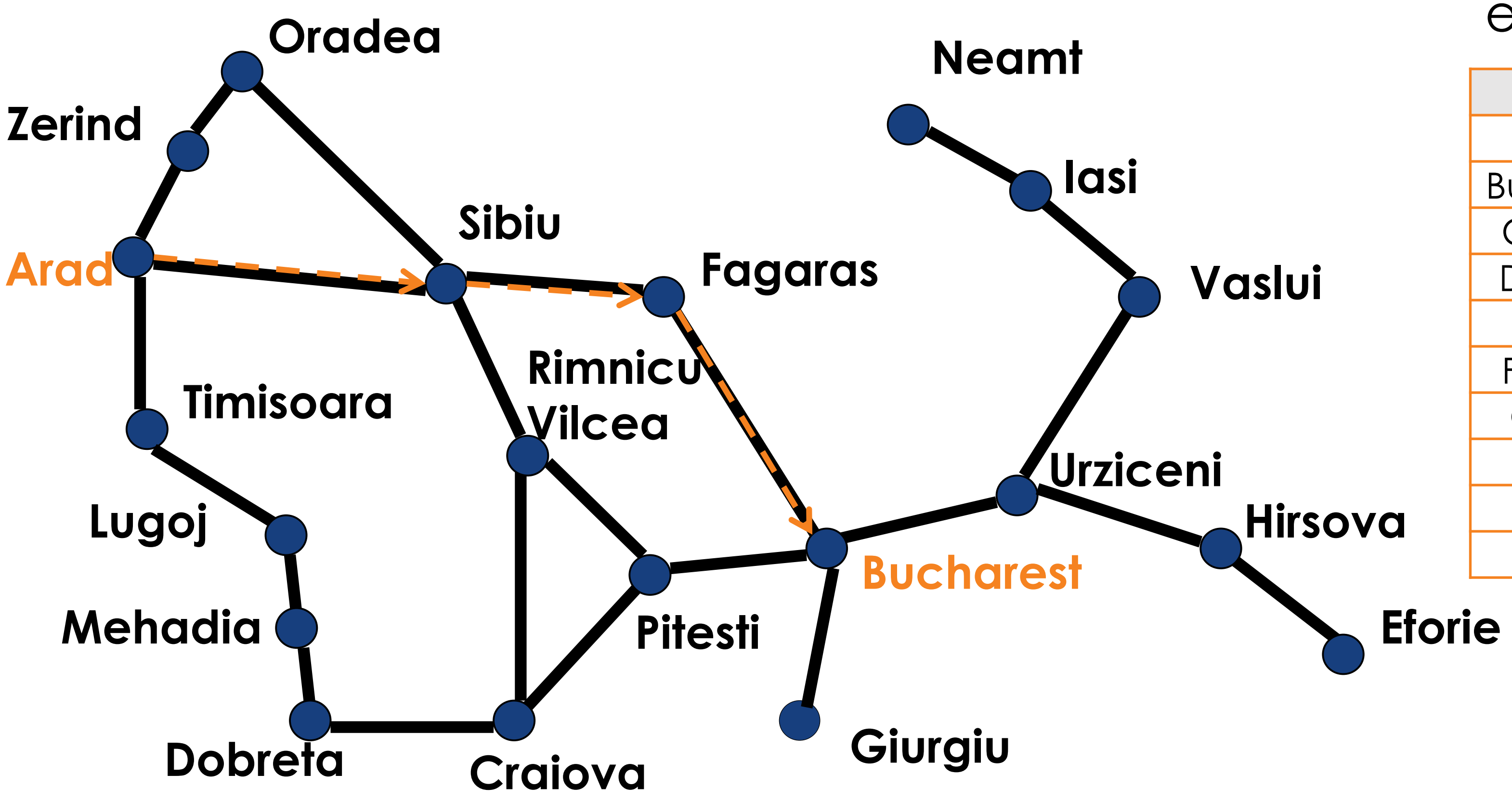


☺ Goal found!

1.3 INFORMED SEARCH TECHNIQUES (1/2)

Hill Climbing (Greedy Best First Search)

- Arad → Bucharest



h(n) =Straight-line distance from each city to **Bucharest**

City	Distance	City	Distance
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search

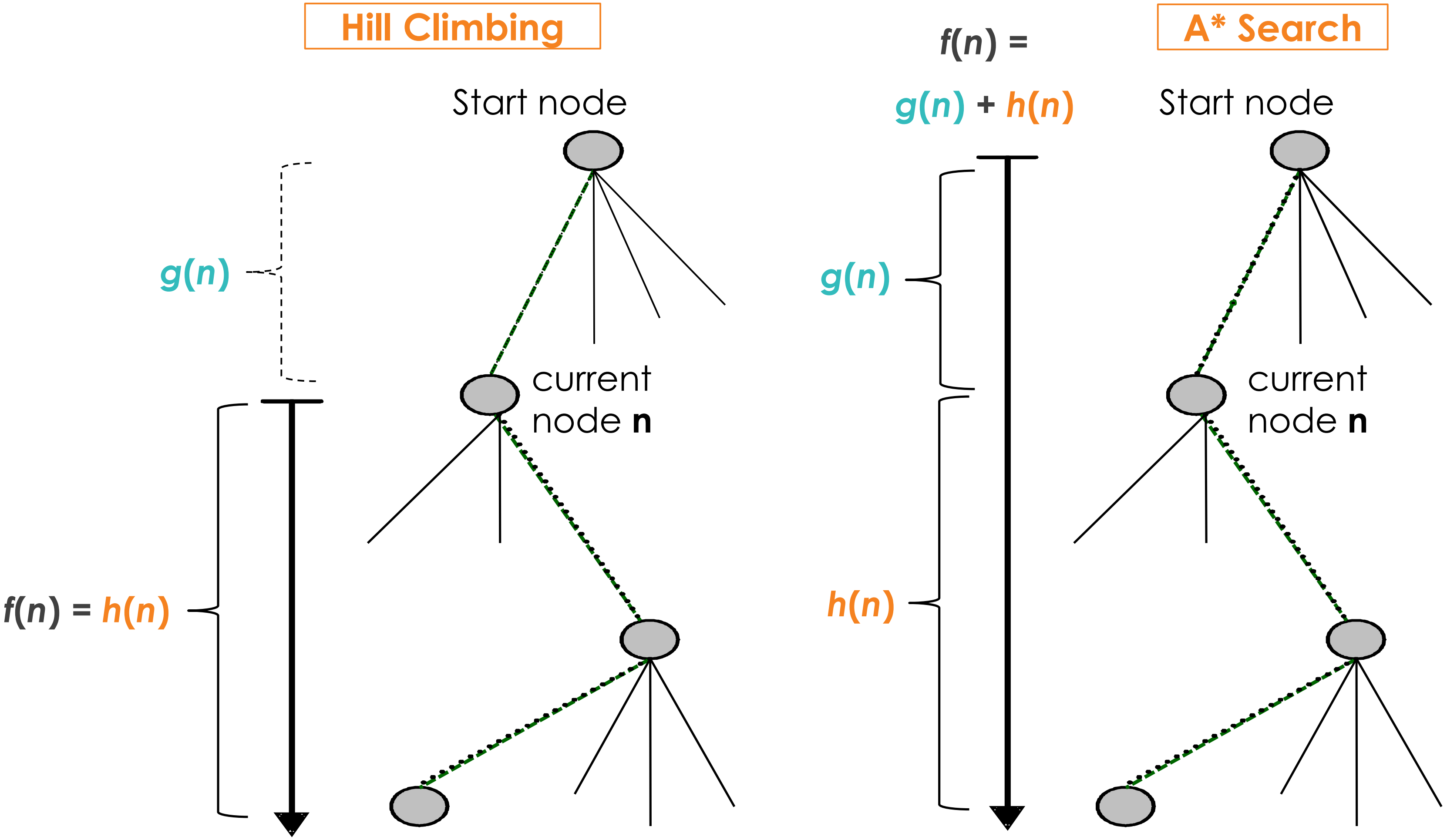
- **A* search is the most widely-known form of best-first search**
  - This strategy evaluates each search node by combining ***g(n)***, the past (path) cost from the start node to current node ***n***, and ***h(n)***, the estimated future (path) cost: the cheapest path/cost from current node ***n*** to a goal node
  - Estimated total cost of the cheapest solution through ***n***

$$f(n) = g(n) + h(n)$$

☺ ***g(n)*** **is exactly known, but** ***h(n)*** **is only an estimation with possible error.**

1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search vs. Hill Climbing



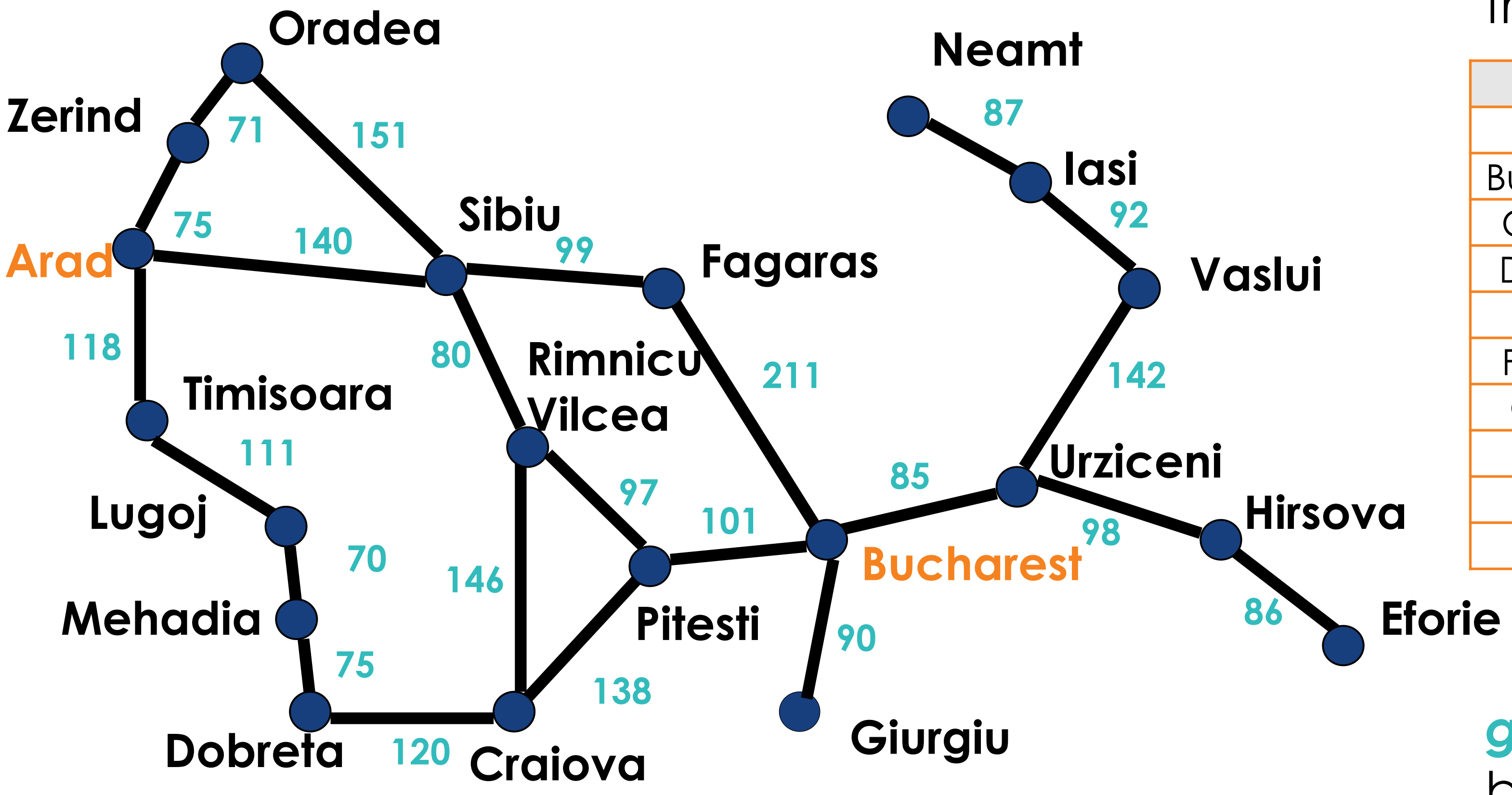
Select then expand "best-path-**from-n**-to-goal" child-node at each layer

Select then expand "best-path-**from-start**-to-goal" child-node at each layer

1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search

- Arad → Bucharest



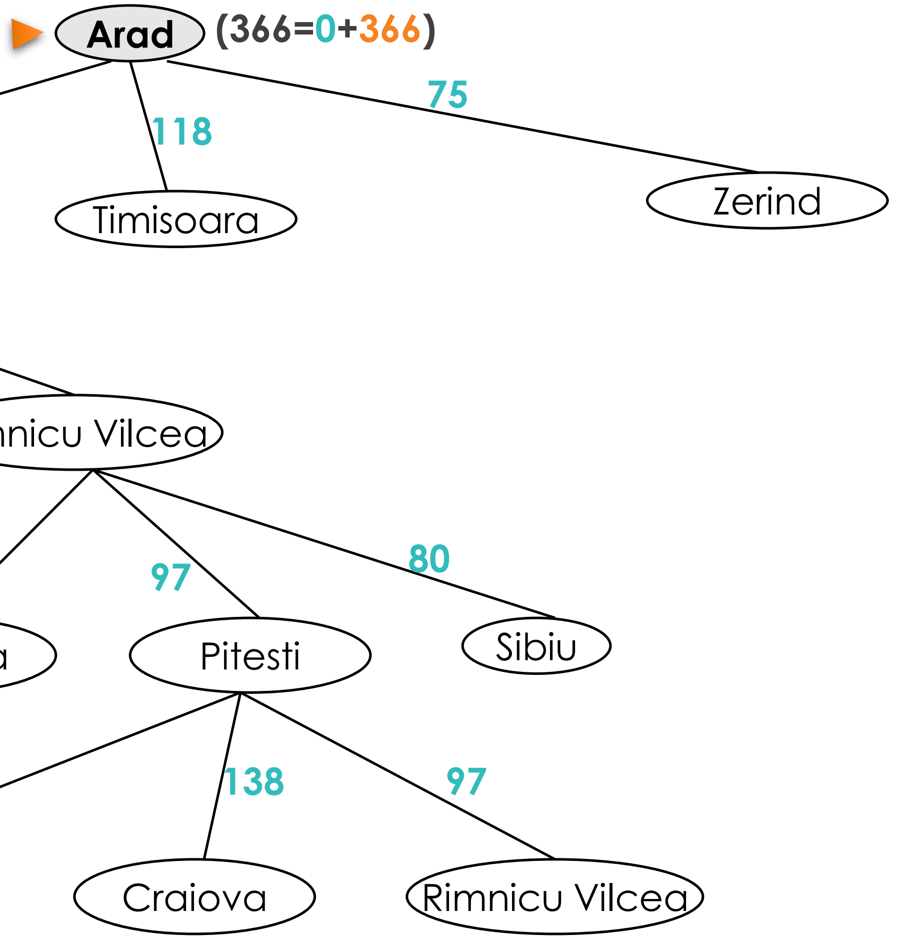
h(n) = Straight-line distance from each city to **Bucharest**

City	Distance	City	Distance
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

g(n) = Actual path distance between different cities

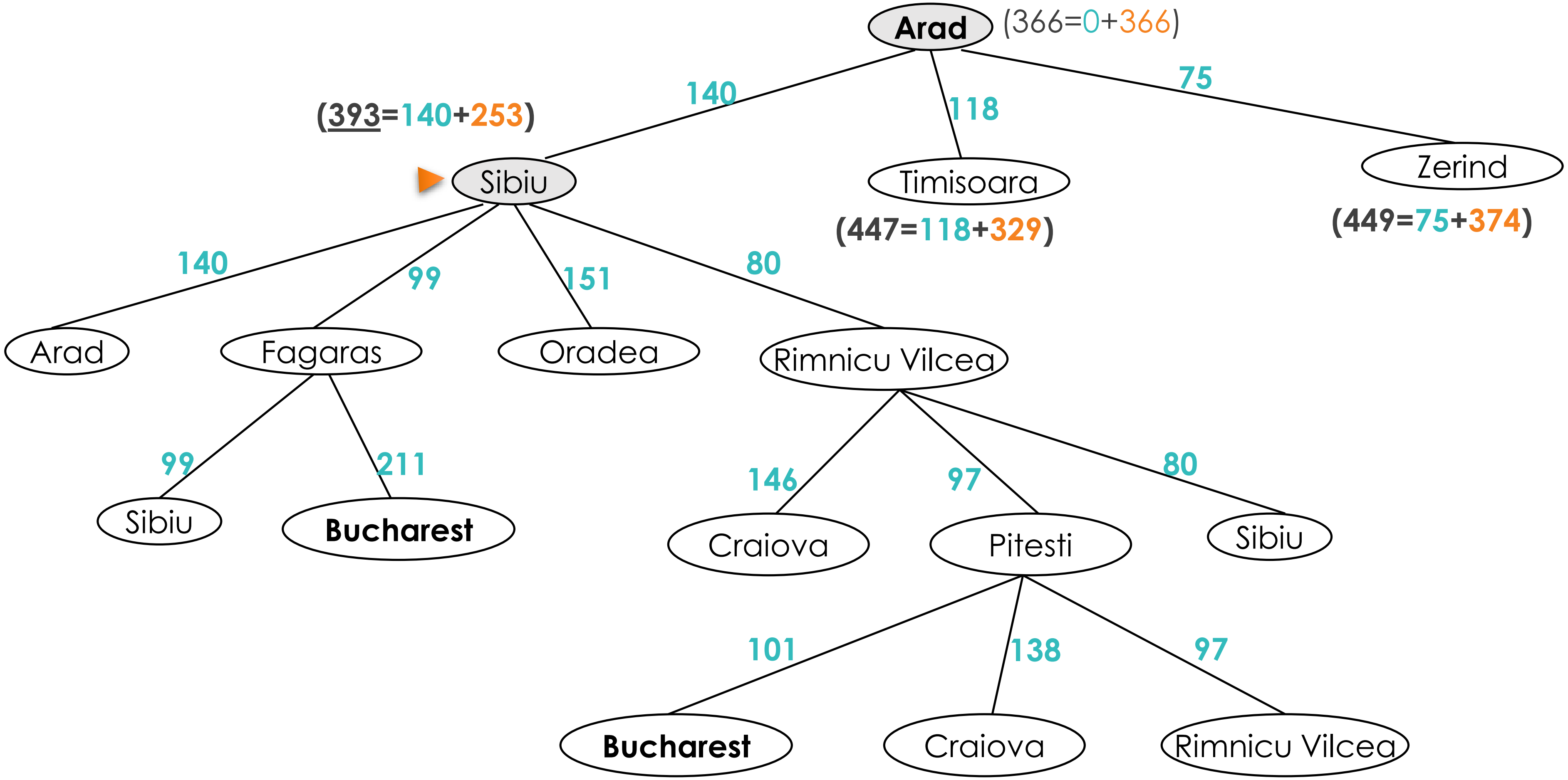
1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search



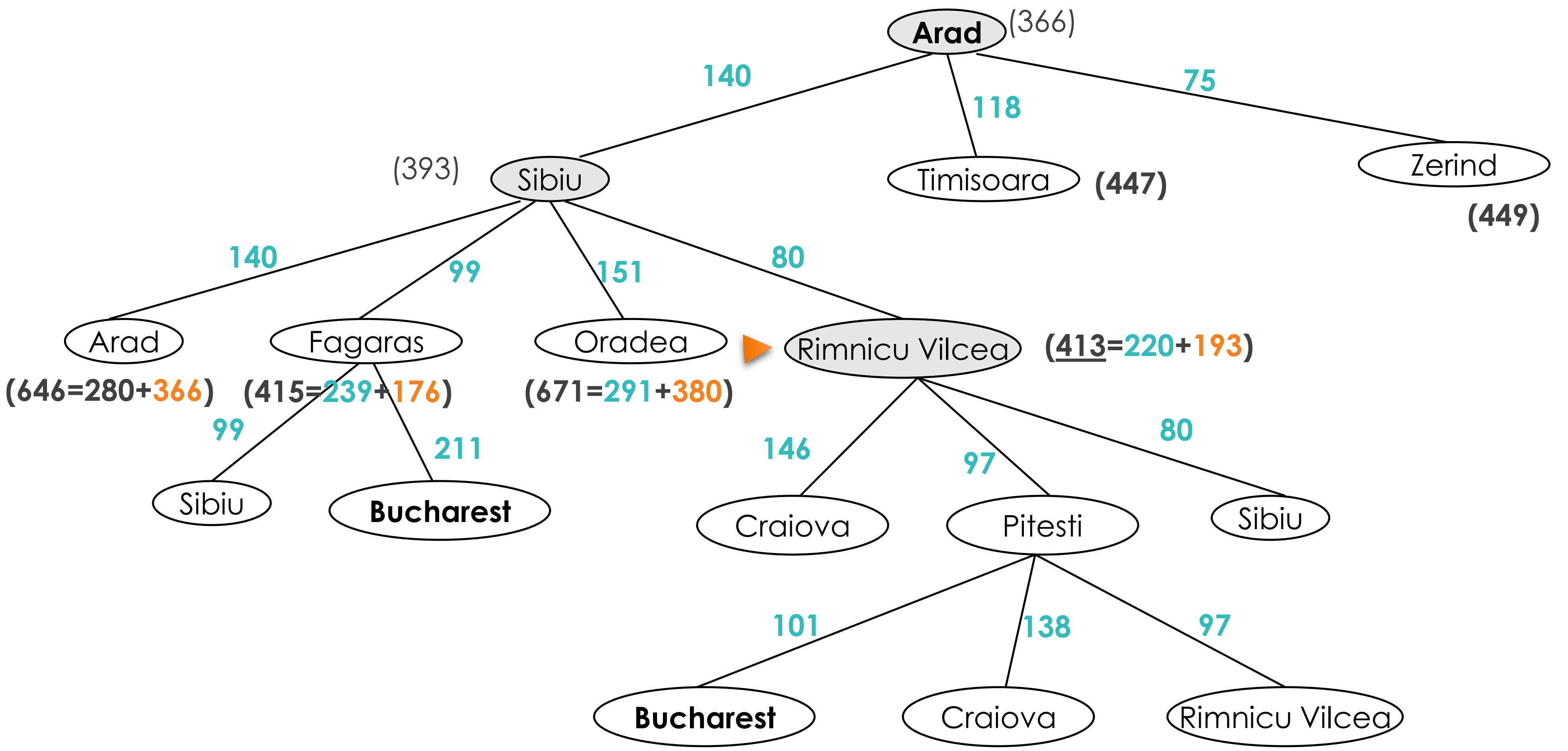
1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search



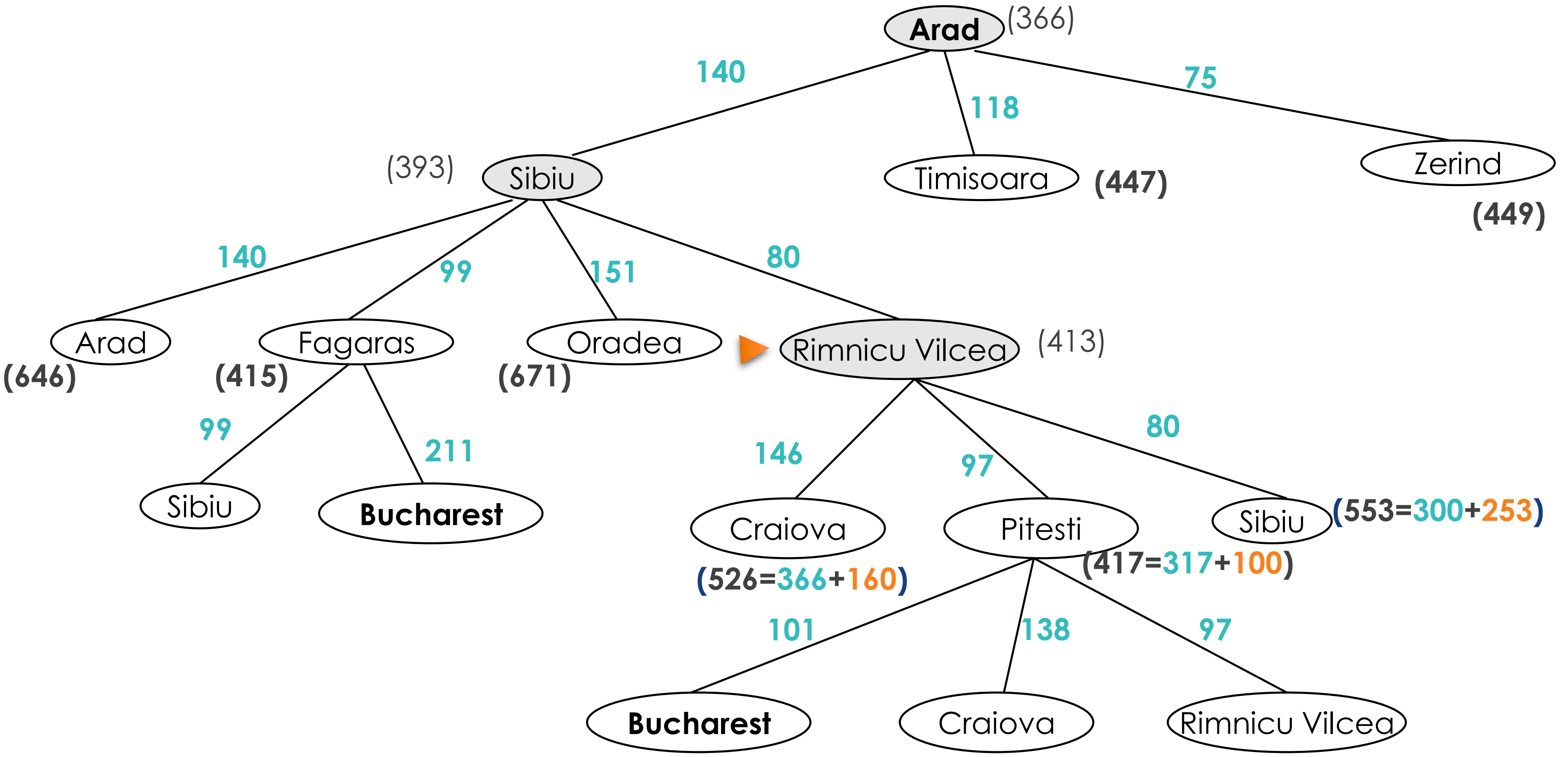
1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search



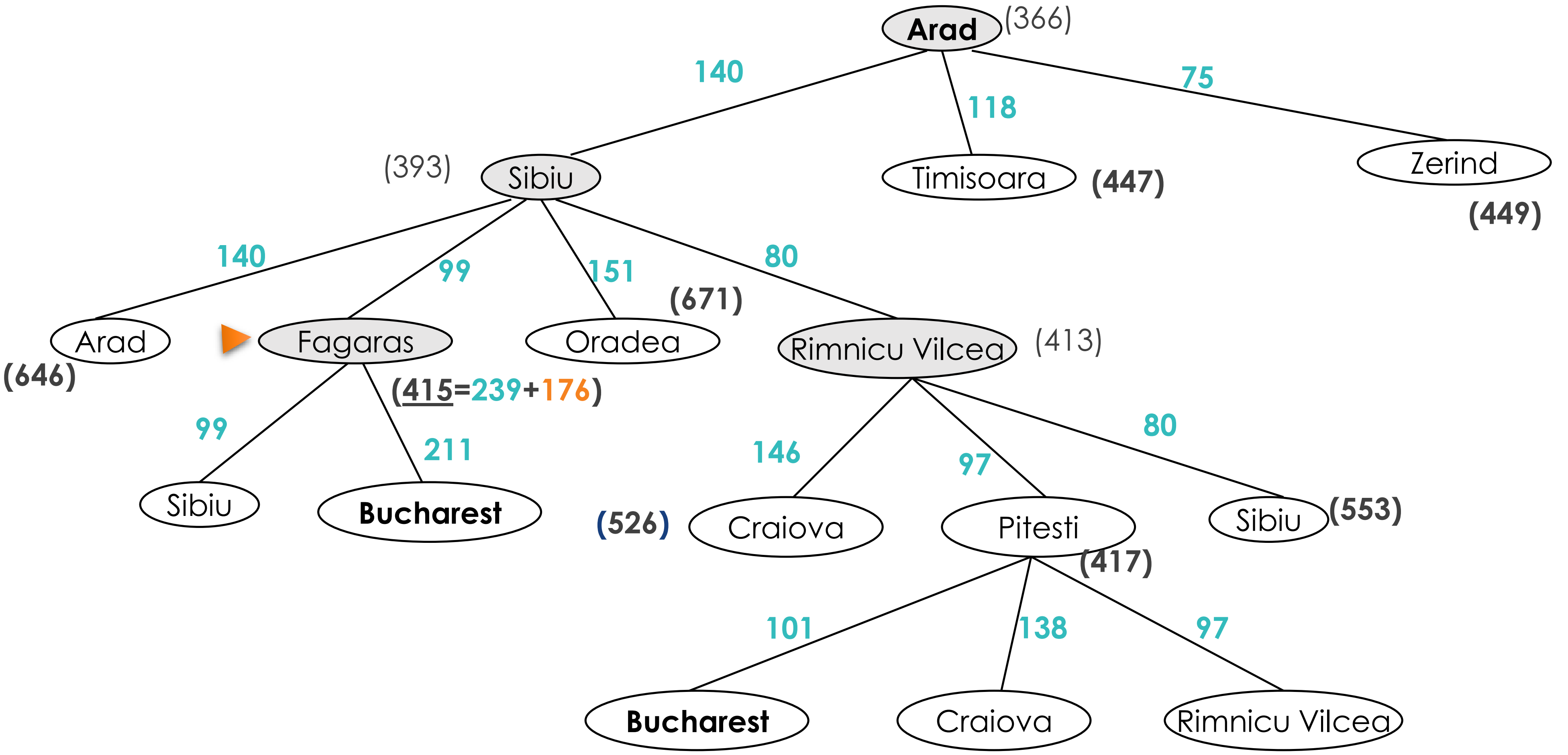
1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search



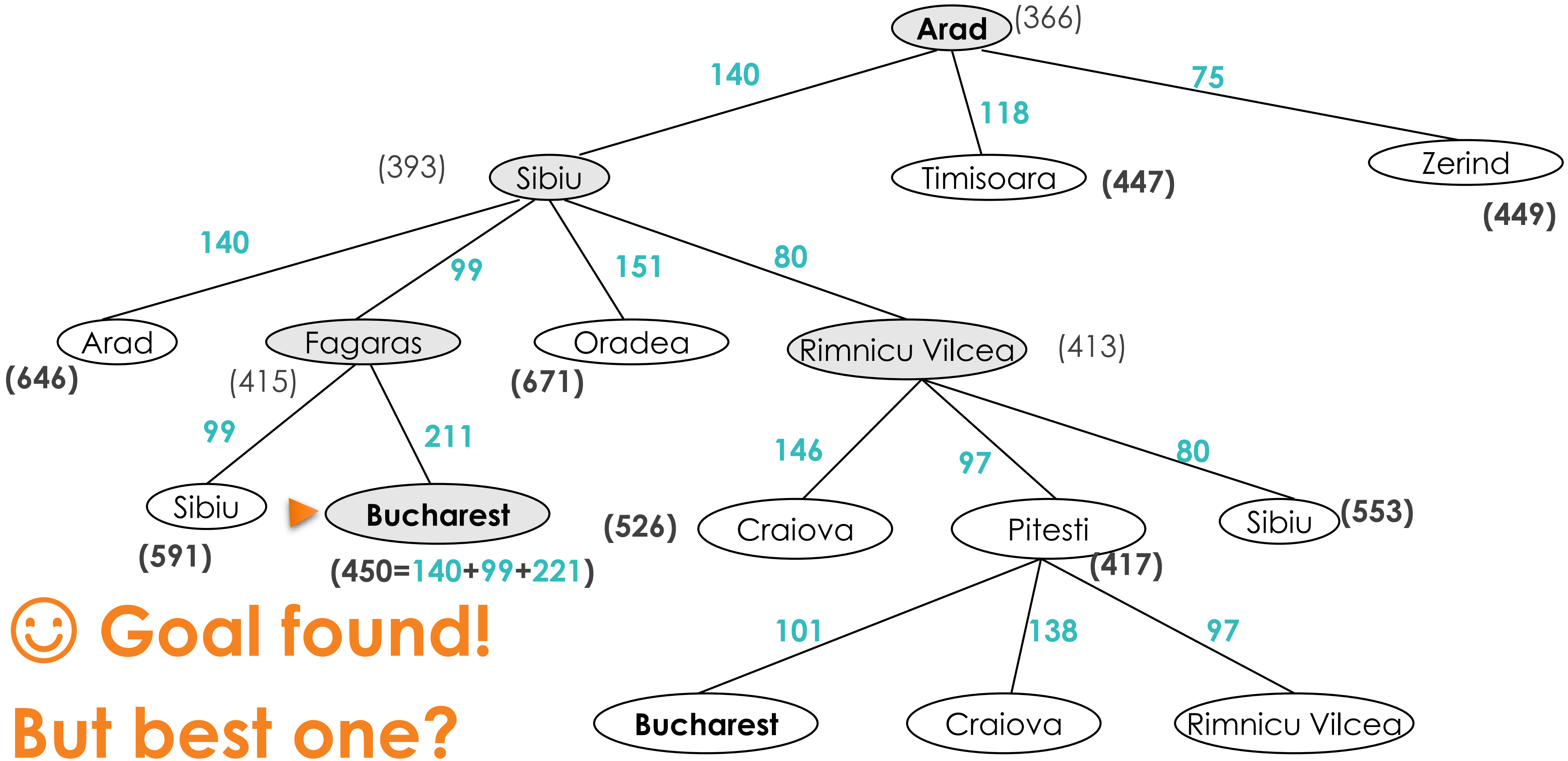
1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search



1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search

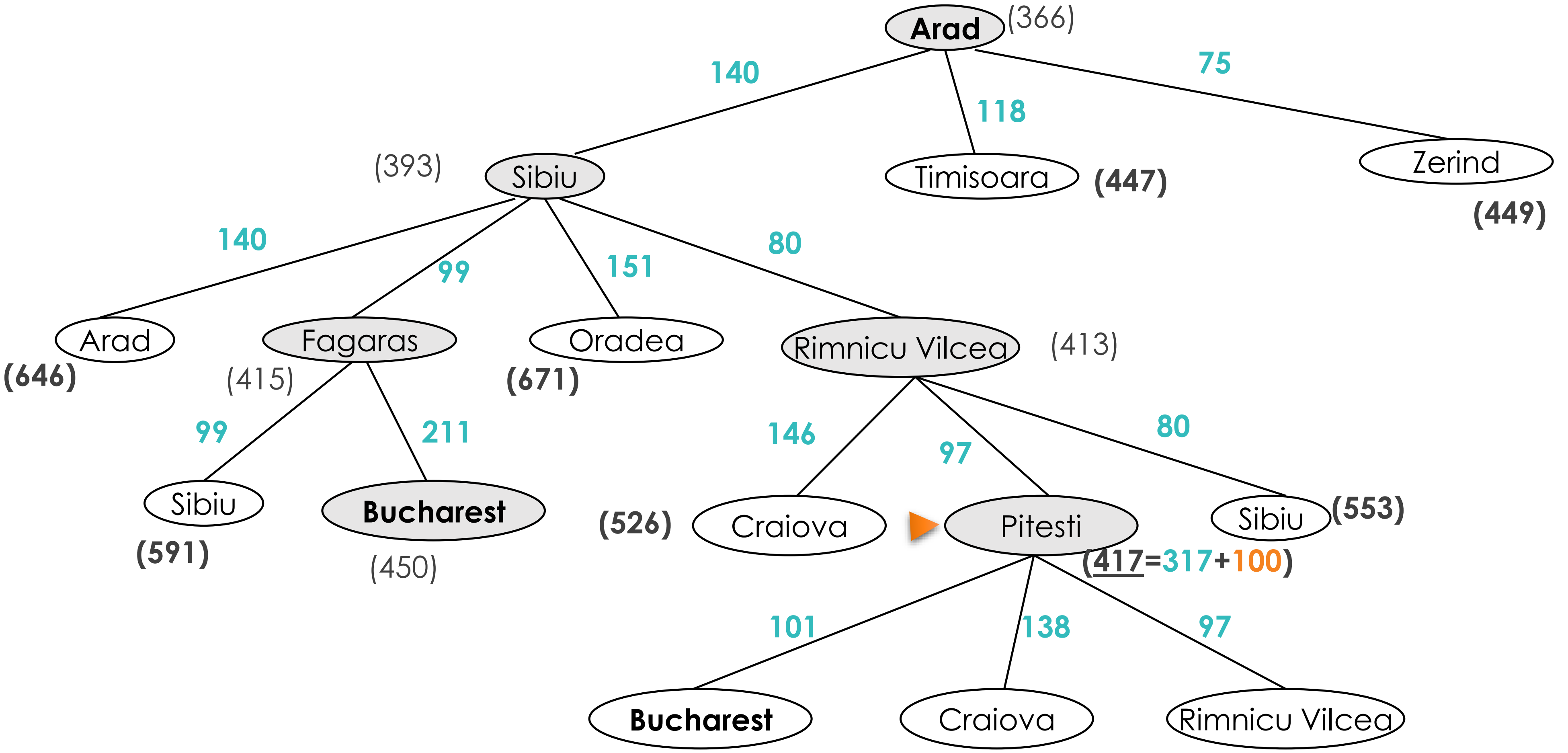


☺ Goal found!

But best one?

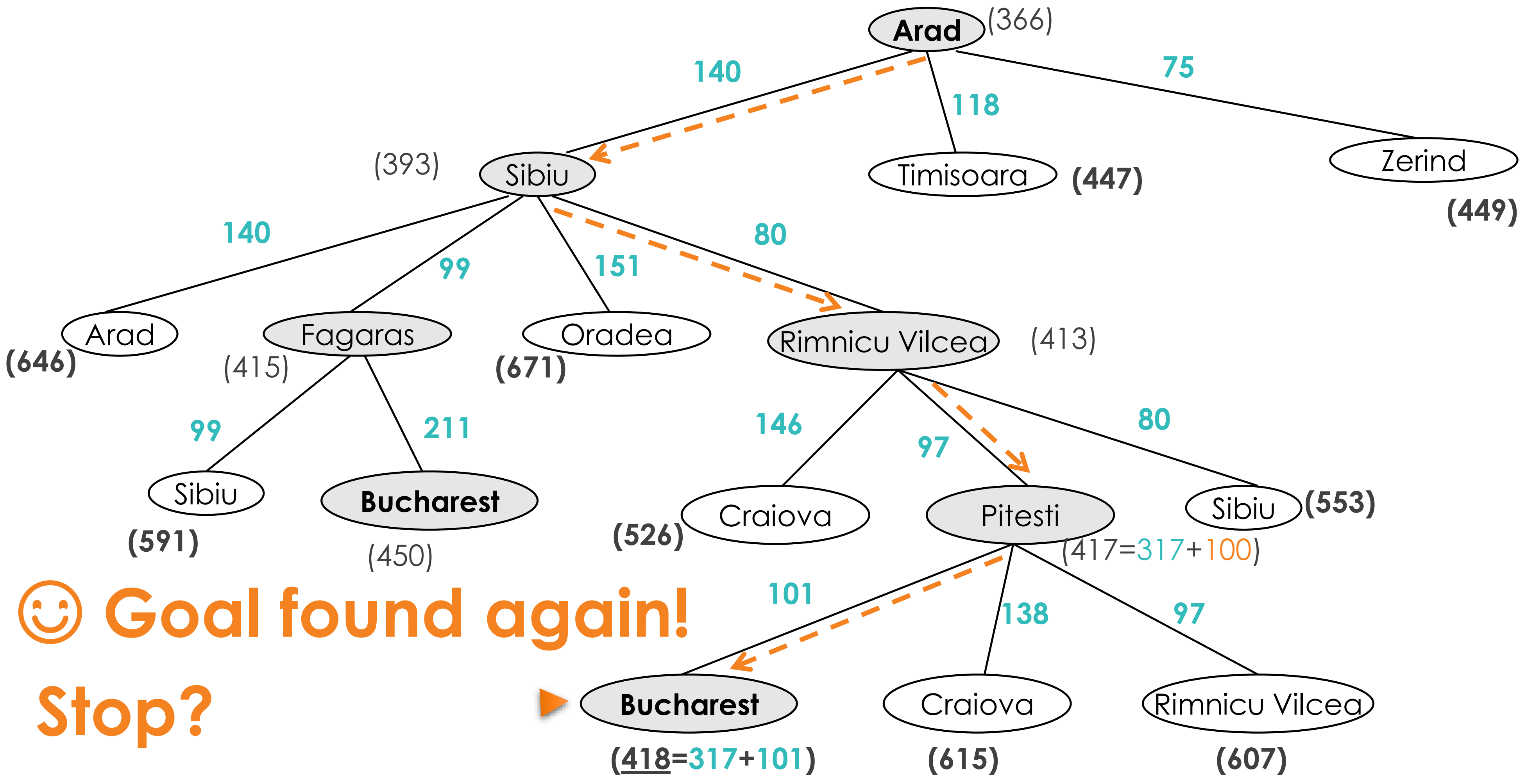
1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search



1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search



☺ Goal found again!

Stop?

1.3 INFORMED SEARCH TECHNIQUES (1/2)

A* Search vs. Hill Climbing

Exercise

- Compare the results between A* Search & Hill Climbing.

- Are they the same? If not, why?

1.4 WORKSHOP

SEARCH REPRESENTATION

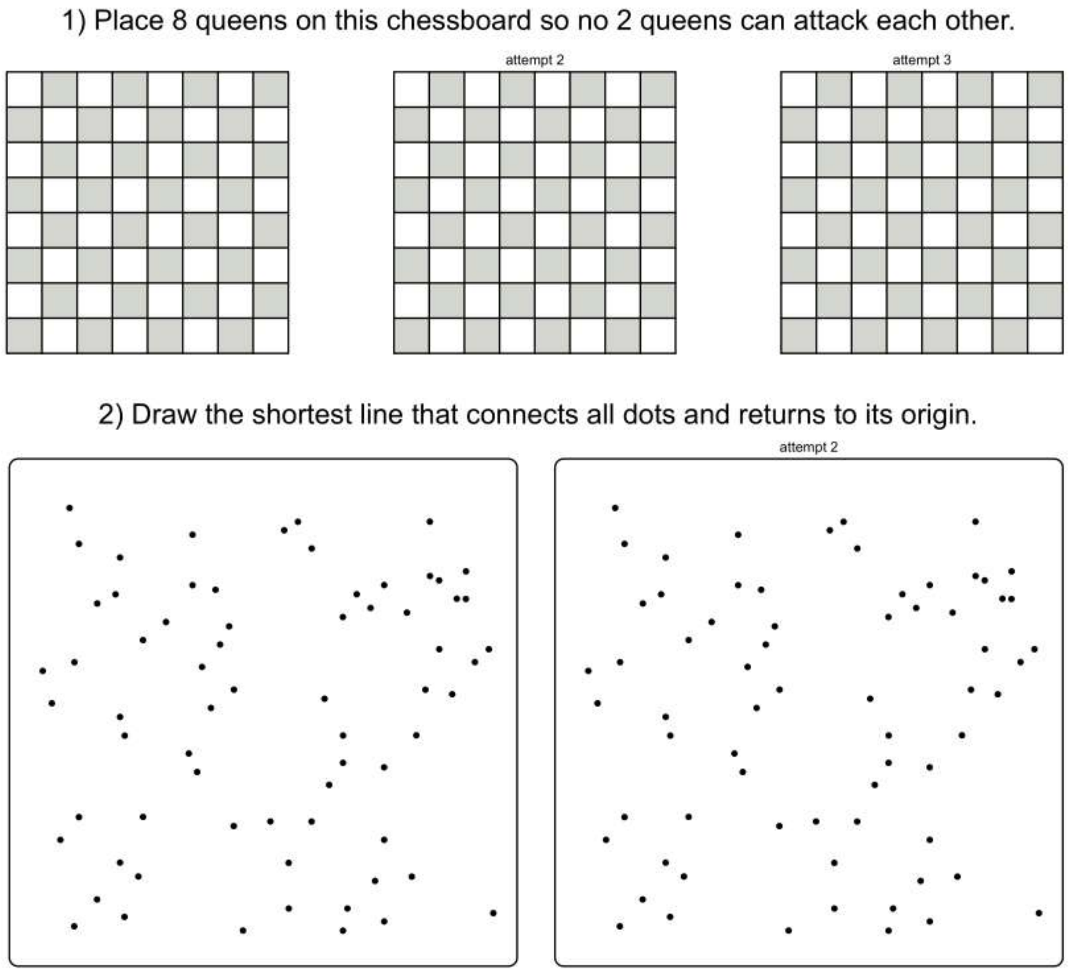
1.4 WORKSHOP SEARCH REPRESENTATION

- **Search Modelling & Representation**
  - Pen & Paper Planning
  - Robot Navigation
- **KIE OptaPlanner Tutorial**
  - Optimizing Vehicle Route Planning (VRP)
  - Optimizing Europe Travelling Sales Person (TSP)

1.4 WORKSHOP SEARCH REPRESENTATION

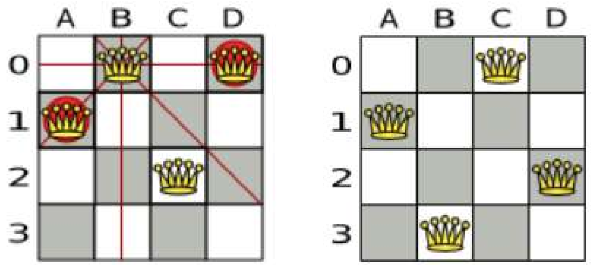
Search Modelling & Representation

- **Pen & Paper Planning**

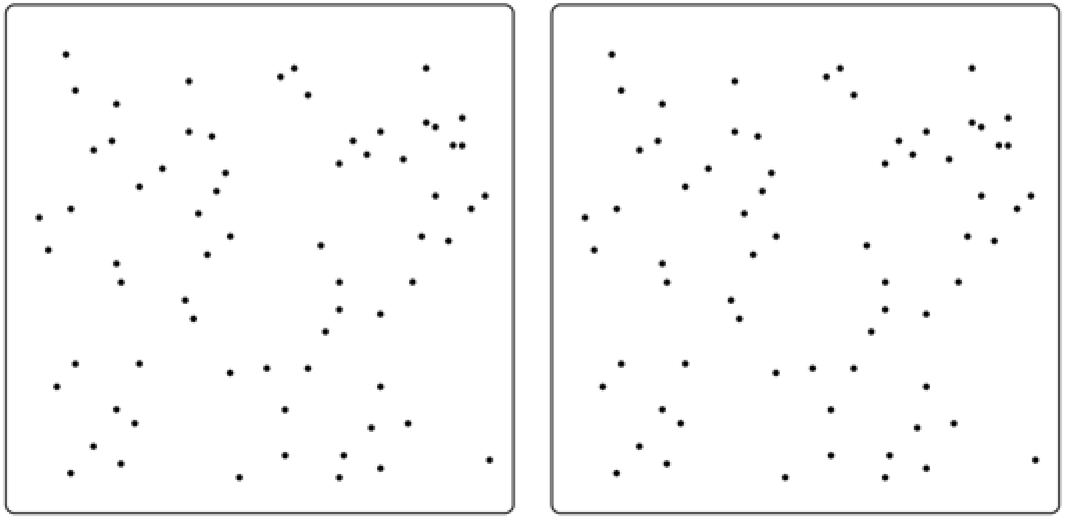


1) Place 8 queens on this chessboard so no 2 queens can attack each other.

2) Draw the shortest line that connects all dots and returns to its origin.



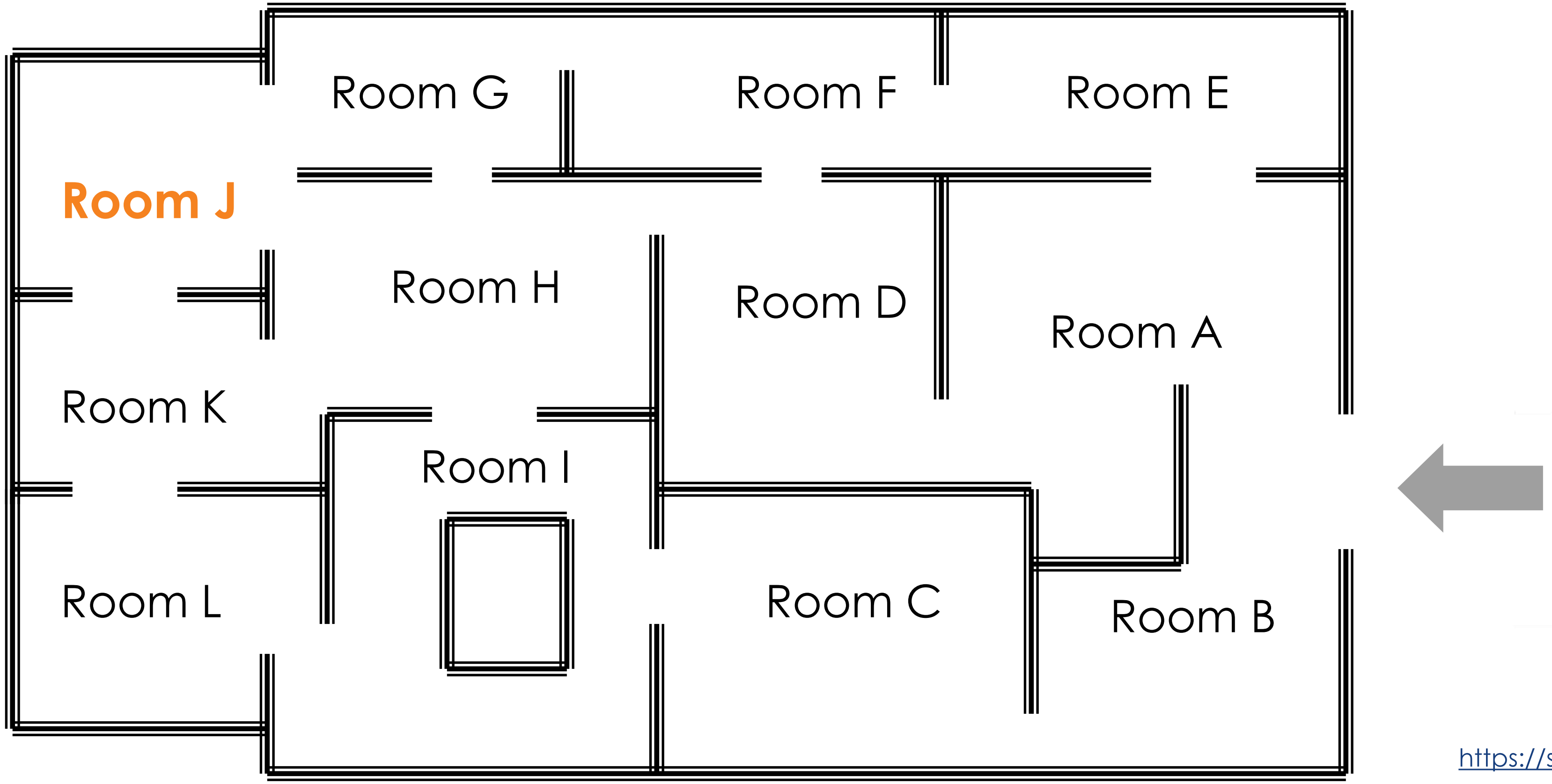
Bad Good



1.4 WORKSHOP SEARCH REPRESENTATION

Search Modelling & Representation

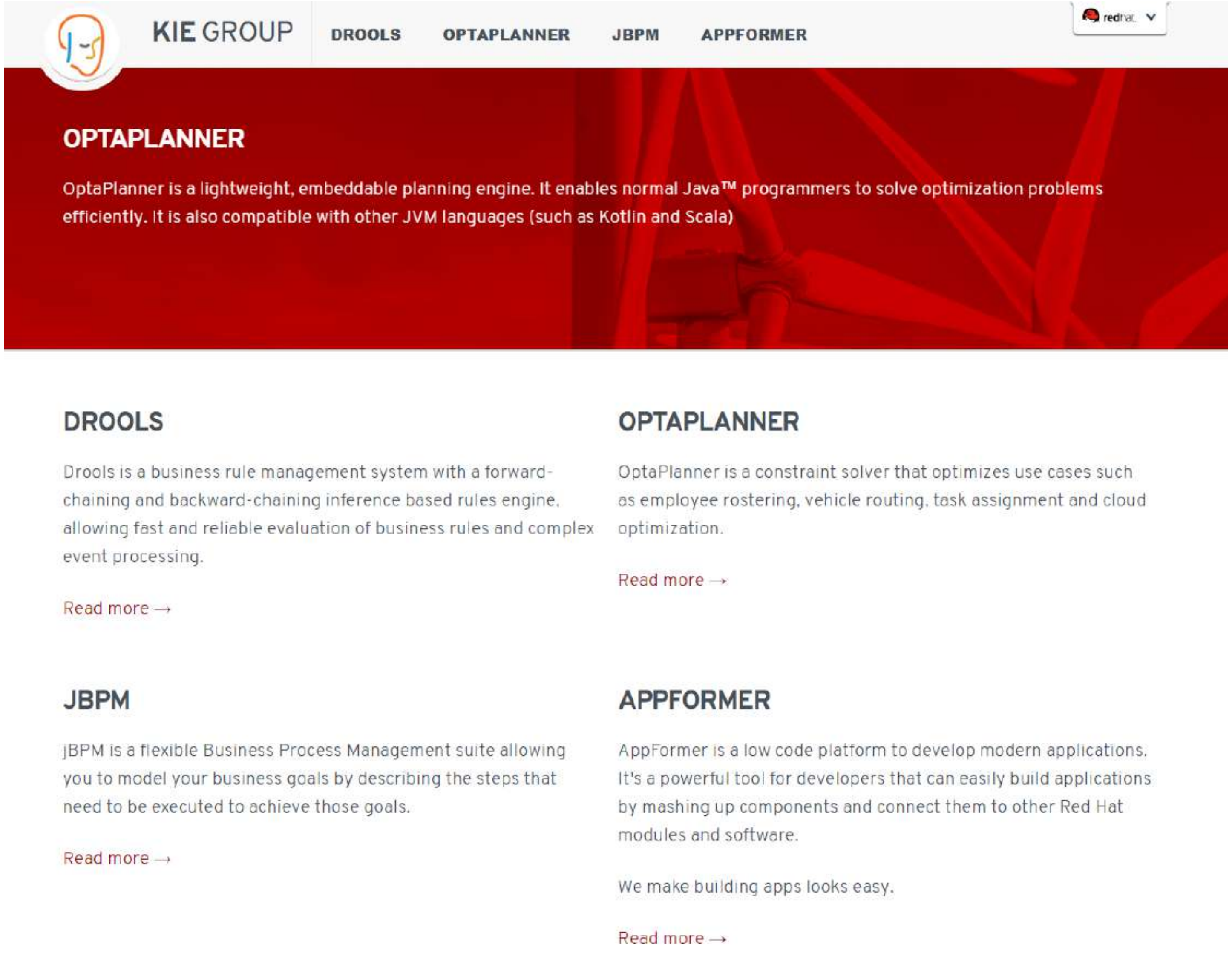
- Robotics: How to rapidly navigate to Room J ?



https://static1.squarespace.com/static/57c8a68a20099ef23fb19e90/t/5a32a9804192022be97f9d10/1513269668629/Atlas.png

1.4 WORKSHOP SEARCH REPRESENTATION

KIE OptaPlanner Tutorial



OPTAPLANNER

OptaPlanner is a lightweight, embeddable planning engine. It enables normal Java™ programmers to solve optimization problems efficiently. It is also compatible with other JVM languages (such as Kotlin and Scala)

DROOLS

Drools is a business rule management system with a forward-chaining and backward-chaining inference based rules engine, allowing fast and reliable evaluation of business rules and complex event processing.

Read more →

OPTAPLANNER

OptaPlanner is a constraint solver that optimizes use cases such as employee rostering, vehicle routing, task assignment and cloud optimization.

Read more →

JBPM

jBPM is a flexible Business Process Management suite allowing you to model your business goals by describing the steps that need to be executed to achieve those goals.

Read more →

APPFORMER

AppFormer is a low code platform to develop modern applications. It's a powerful tool for developers that can easily build applications by mashing up components and connect them to other Red Hat modules and software.

We make building apps looks easy.

Read more →

JBoss KIE

http://www.kiegroup.org/

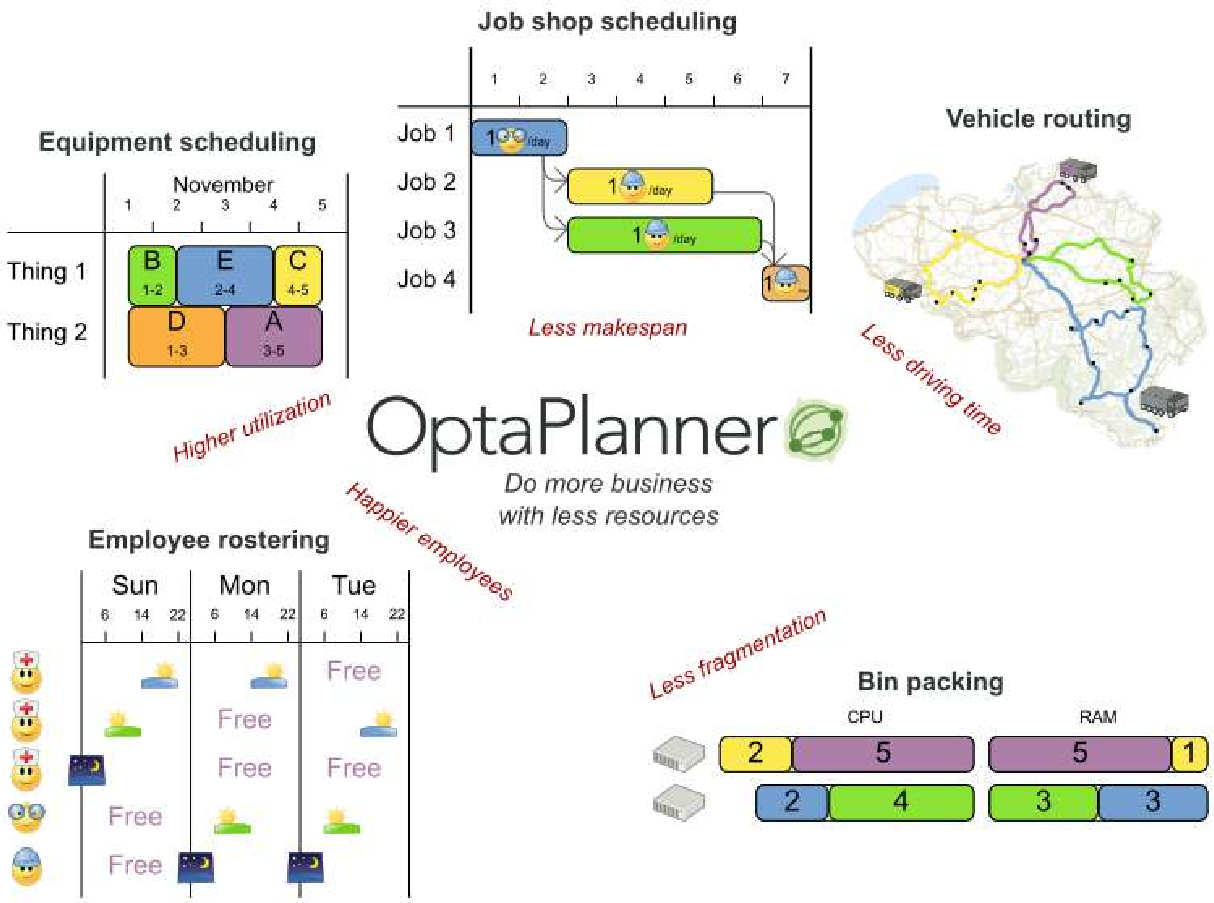
JBoss KIE OptaPlanner

http://www.optaplanner.org/

1.4 WORKSHOP SEARCH REPRESENTATION

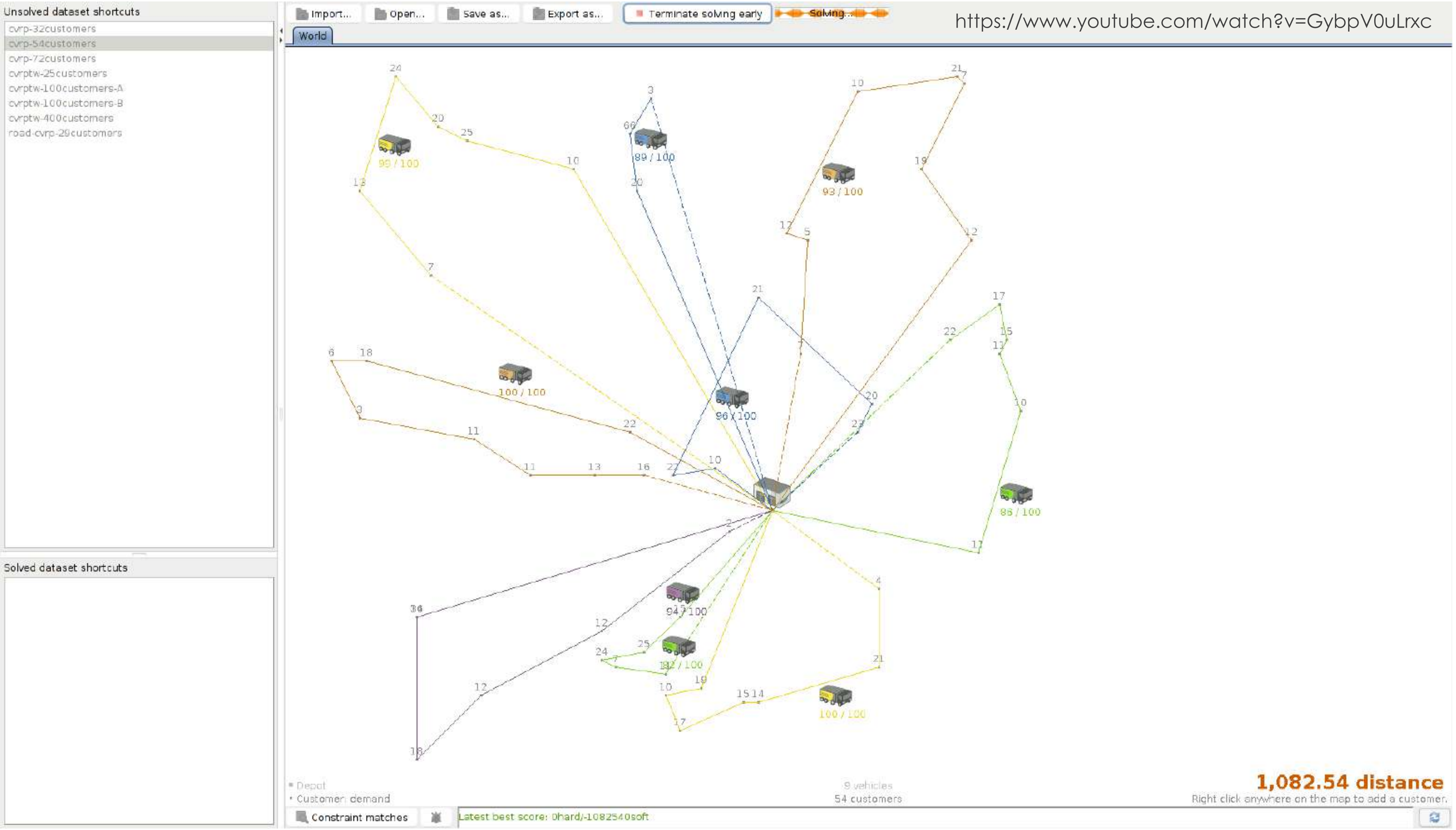
KIE OptaPlanner Tutorial

- Constrain Satisfaction: Business Resource Optimizer



1.4 WORKSHOP SEARCH REPRESENTATION

KIE OptaPlanner Tutorial – VRP: Customer demand (vehicle load)



https://www.youtube.com/watch?v=GybpV0uLrxc

Unsolved dataset shortcuts

- cvrp-32customers
- cvrp-54customers
- cvrp-72customers
- cvrptw-25customers
- cvrptw-100customers-A
- cvrptw-100customers-B
- cvrptw-400customers
- road-cvrp-29customers

Solved dataset shortcuts

- Depot
- Customer demand

9 vehicles
54 customers

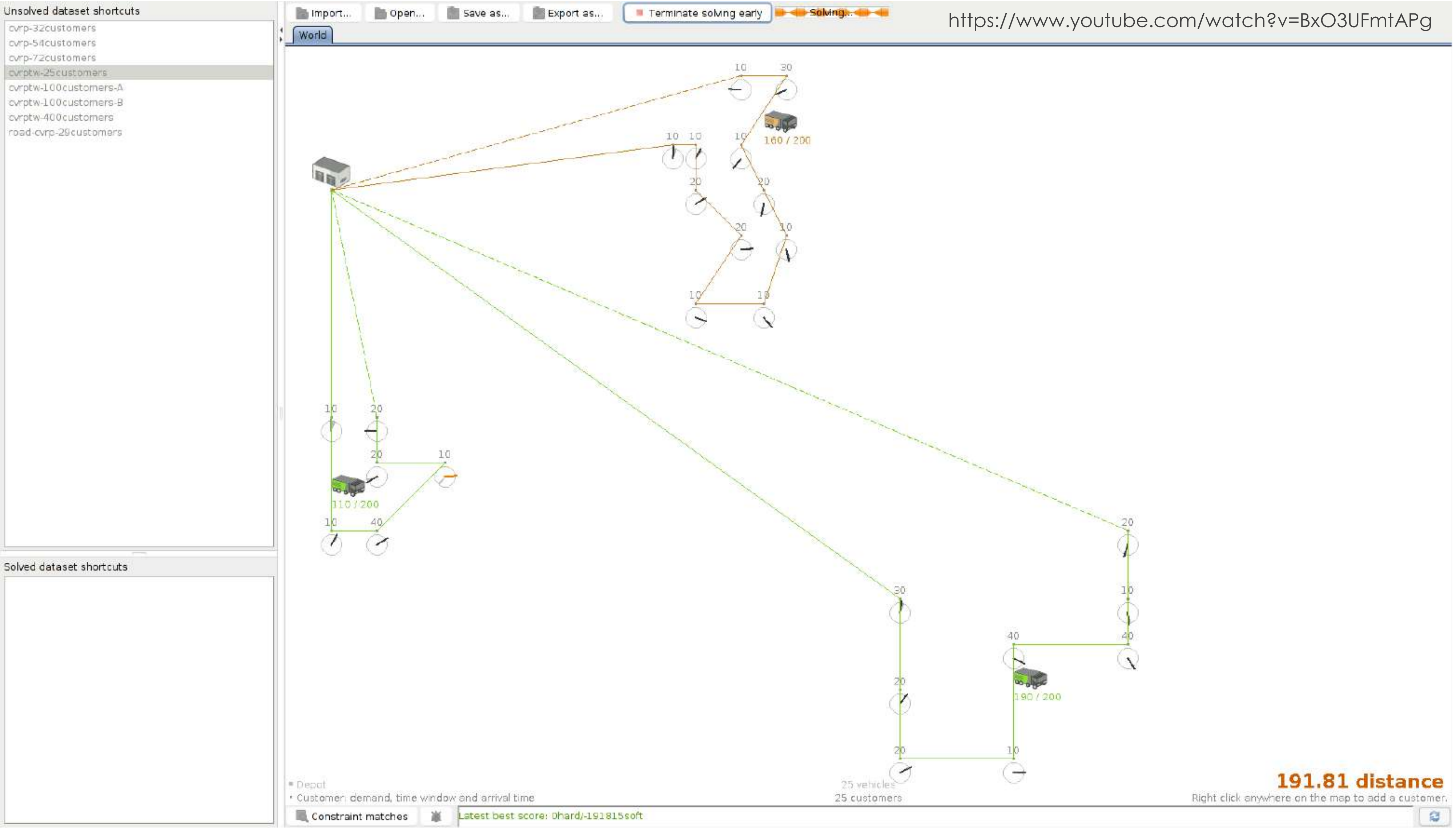
1,082.54 distance

Right click anywhere on the map to add a customer.

Latest best score: 0hard/-1082540soft

1.4 WORKSHOP SEARCH REPRESENTATION

KIE OptaPlanner Tutorial – VRP: Customer demand, Time window



https://www.youtube.com/watch?v=BxO3UFmtAPg

Unsolved dataset shortcuts

- cvrp-32customers
- cvrp-54customers
- cvrp-72customers
- cvrptw-25customers
- cvrptw-100customers-A
- cvrptw-100customers-B
- cvrptw-400customers
- road-cvrp-29customers

Solved dataset shortcuts

Customer: demand, time window and arrival time

25 customers

191.81 distance

Right click anywhere on the map to add a customer.

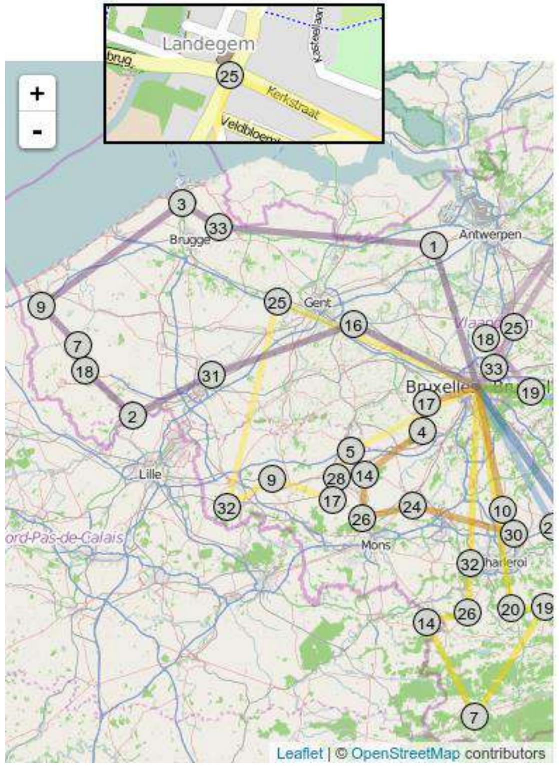
Latest best score: 0hard/-191815soft

1.4 WORKSHOP SEARCH REPRESENTATION

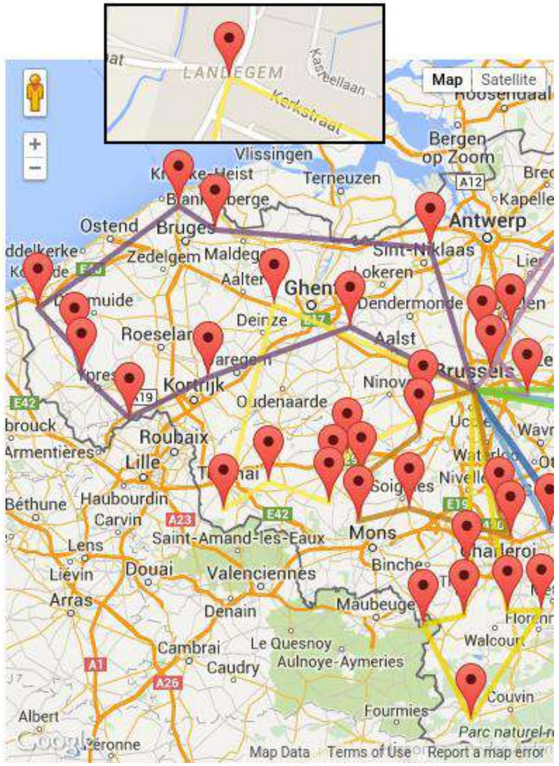
KIE OptaPlanner Tutorial – VRP with map integration

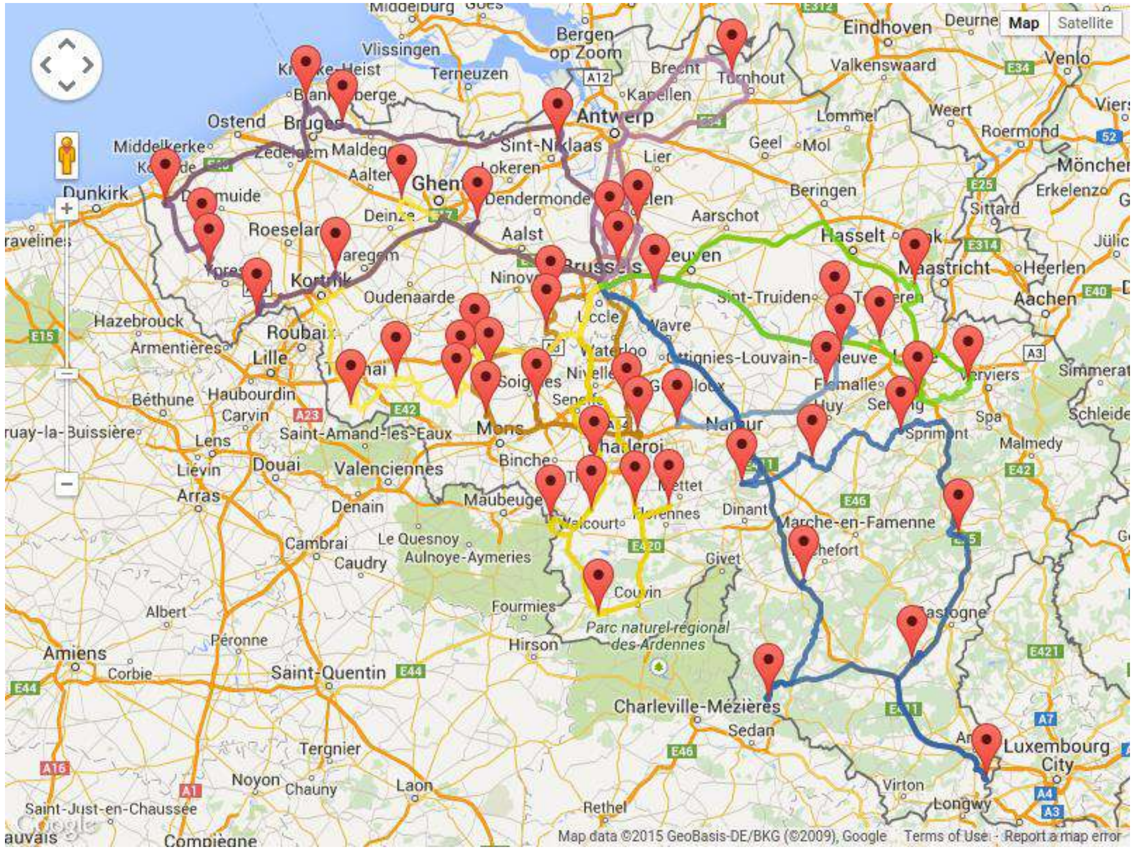
Visualizing Vehicle Routing with Leaflet and Google Maps

Leaflet.js



Google Maps



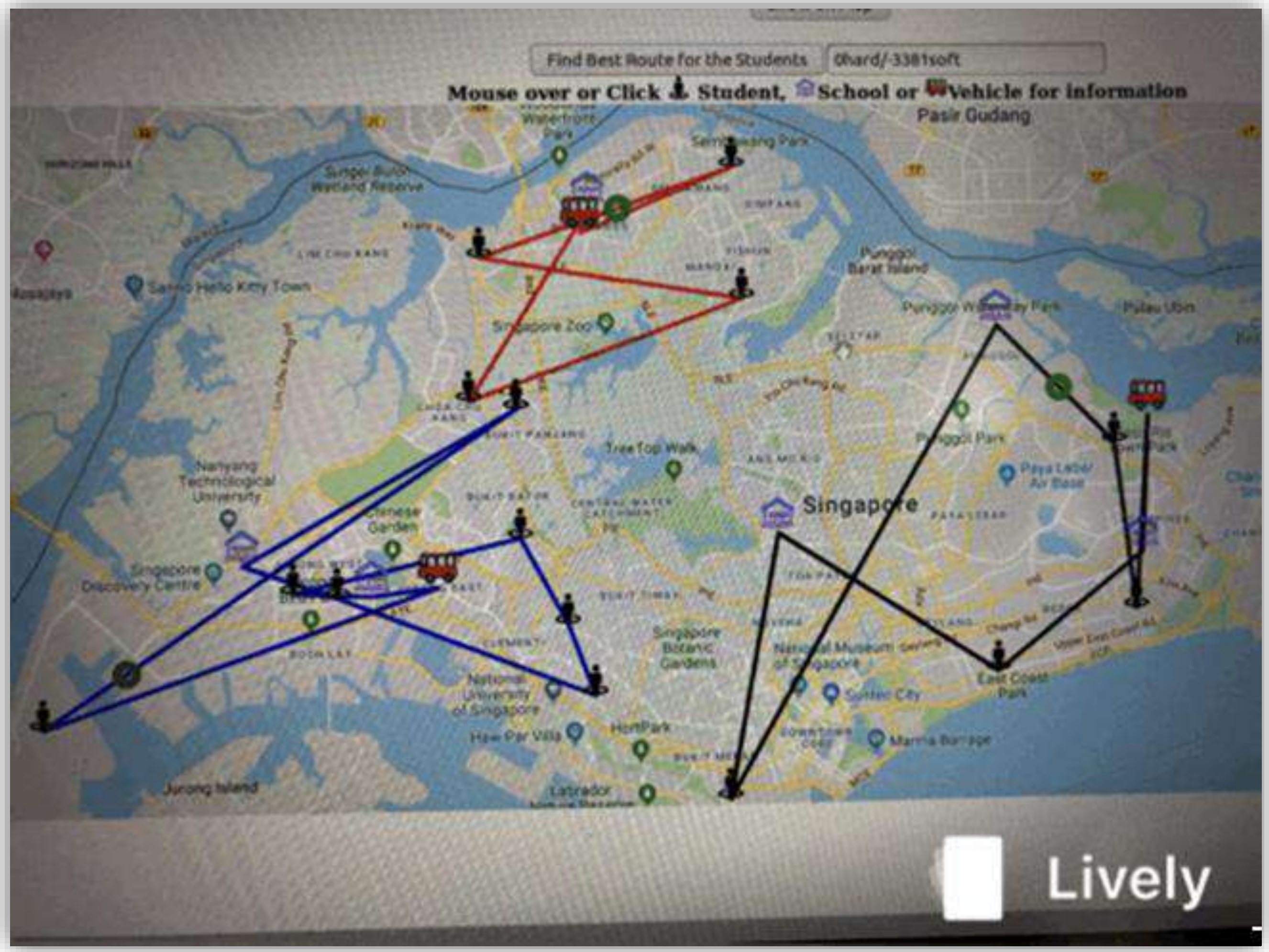


https://www.optaplanner.org/blog/2015/03/10/VisualizingVehicleRoutingWithLeafletAndGoogleMaps.html

1.4 WORKSHOP SEARCH REPRESENTATION

Past project – VRP with map integration

Intelligent Rapid Shuttle (IRS) System



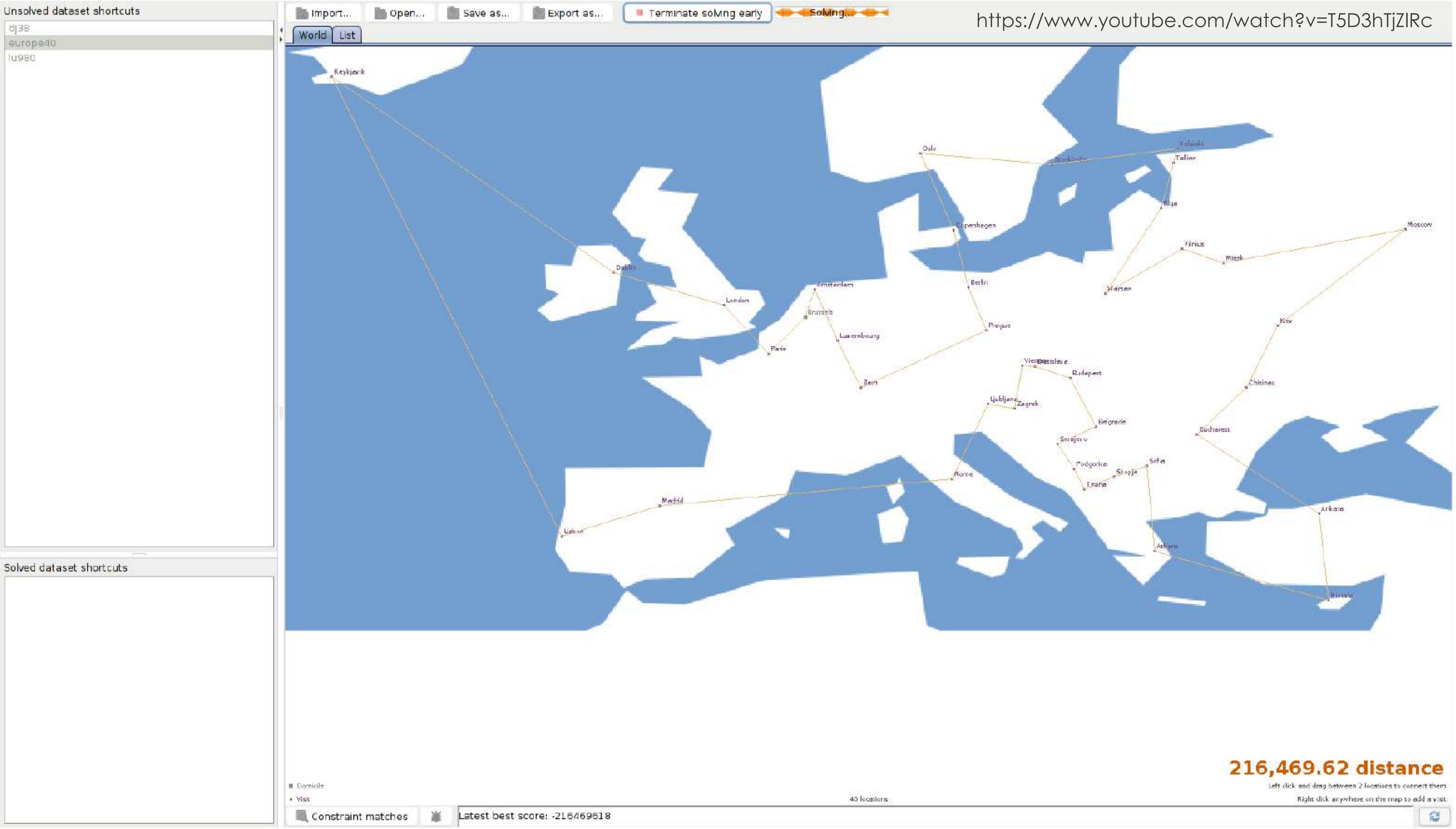
0hard/-3381soft

Mouse over or Click Student, School or Vehicle for information

Source https://github.com/IRS-RS/IRS-RS-2019-03-09-IS1PT-GRP-aiVoyagers-irs-Intelligent-Rapid-Shuttle

1.4 WORKSHOP SEARCH REPRESENTATION

KIE OptaPlanner Tutorial – TSP: Europe cities



https://www.youtube.com/watch?v=T5D3hTjZlRc

Unsolved dataset shortcuts
dj38
europe40
lu980

Solved dataset shortcuts

Import... | Open... | Save as... | Export as... | Terminate solving early | Solving...

World | List

216,469.62 distance

Left click and drag between 2 locations to connect them
Right click anywhere on the map to add a visit.

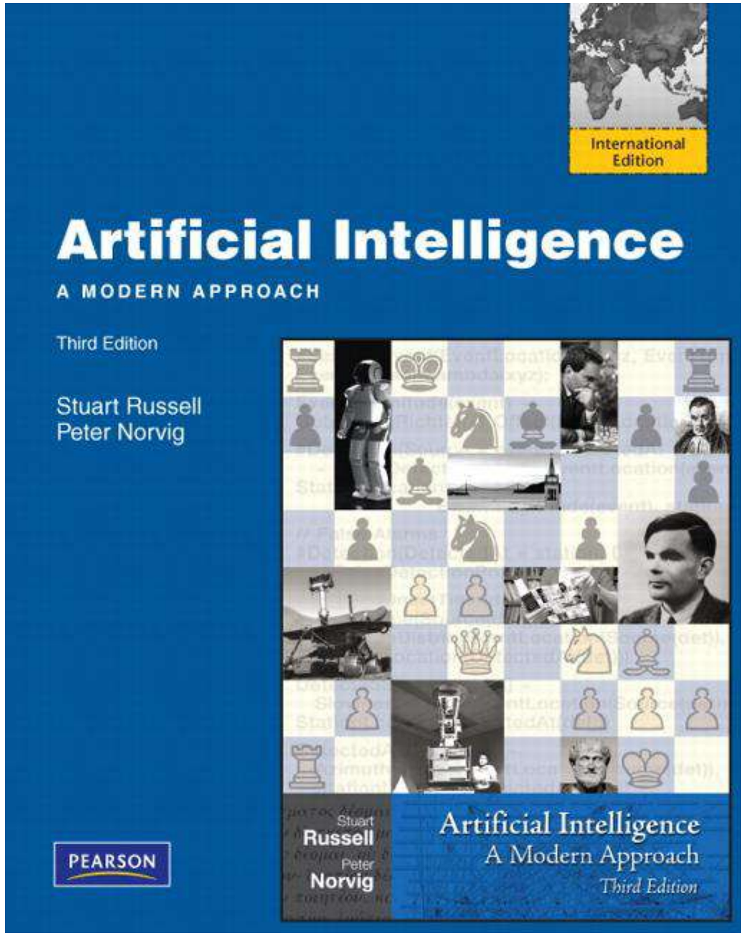
■ Domicile
• Visit

40 locations

Constraint matches

Latest best score: -216469618

DAY 1 REFERENCE



1. **OptaPlanner : Do more business with less recourses**

   http://www.optaplanner.org/learn/slides/optaplanner-presentation/index.html#/1

2. OptaPlanner

   https://www.optaplanner.org/

3. OptaPlanner Use Cases & Demo Videos

   http://www.optaplanner.org/learn/useCases/index.html

4. OptaPlanner Video Tutorials

   http://www.optaplanner.org/learn/video.html

   https://www.youtube.com/user/ge0ffrey2

5. Onne Beek. (2011). Efficient Local Search Methods For Vehicle Routing

   https://lib.ugent.be/fulltxt/RUG01/001/788/544/RUG01-001788544_2012_0001_AC.pdf

DAY 1 SUMMARY

1.1 Reasoning Systems Overview

- History of Artificial Intelligence
- Question Answering System; Image Object Recognition; Chat-Bot
- Problem Solving: Analytic Tasks vs. Synthetic Tasks
- Synthetic Techniques: Search; Simulations; Genetic Algorithms; Data Mining

1.2 Uninformed Search Techniques

- Search Representation
- Depth First Search (DFS)
- Breadth First Search (BFS)

1.3 Informed Search Techniques (1/2)

- Heuristic Knowledge
- Evaluation (Scoring) Function; Heuristic Function; Past Cost Function
- Hill Climbing Search (HC); A Star Search (A*)

1.4 Search Representation Workshop

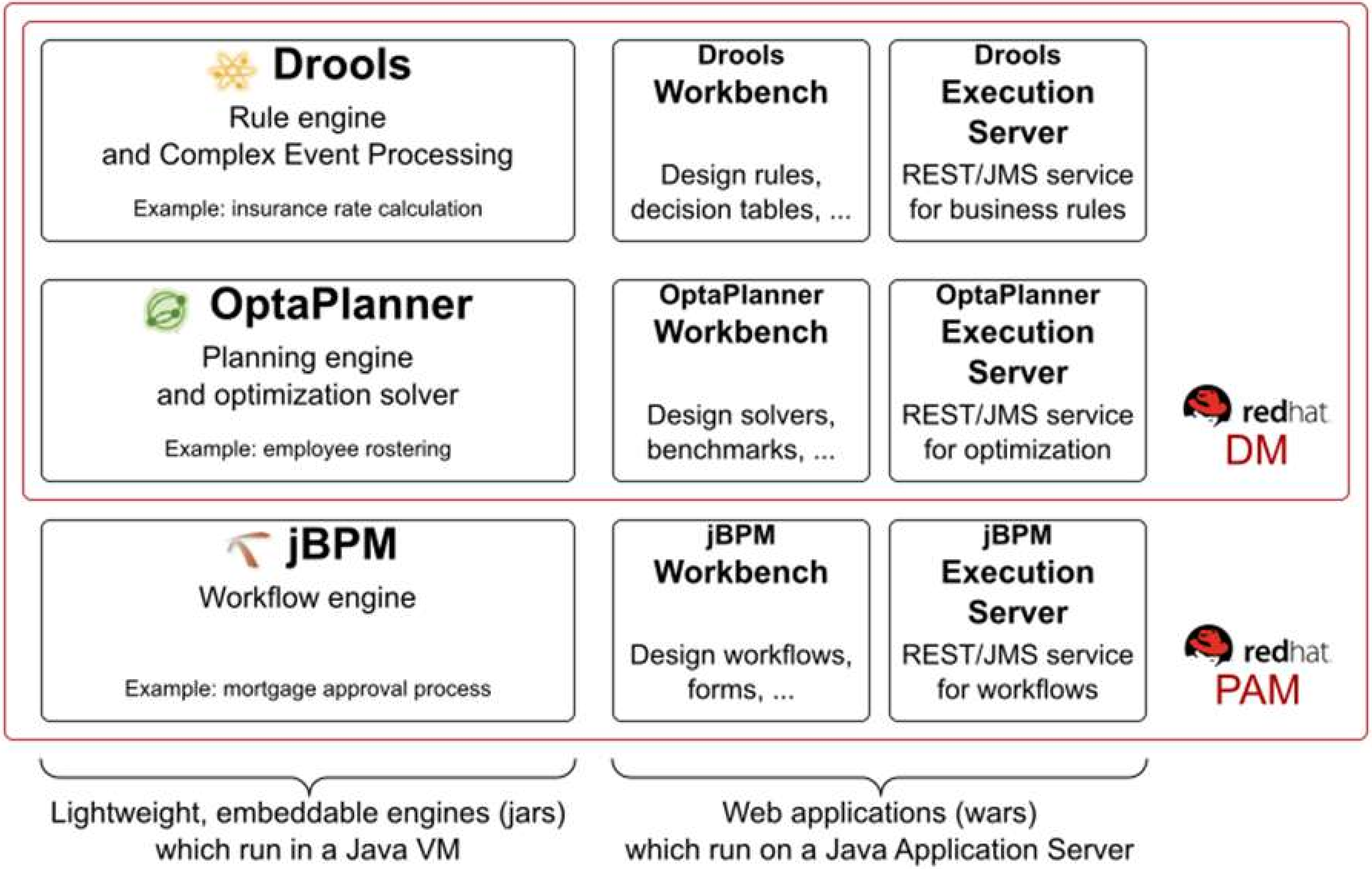
END OF LECTURE NOTES

APPENDICES

KIE System Architecture

KIE functionality overview

What are the KIE projects?



Drools

Rule engine
and Complex Event Processing

Example: insurance rate calculation

Drools Workbench

Design rules, decision tables, ...

Drools Execution Server

REST/JMS service for business rules

OptaPlanner

Planning engine
and optimization solver

Example: employee rostering

OptaPlanner Workbench

Design solvers, benchmarks, ...

OptaPlanner Execution Server

REST/JMS service for optimization

redhat DM

jBPM

Workflow engine

Example: mortgage approval process

jBPM Workbench

Design workflows, forms, ...

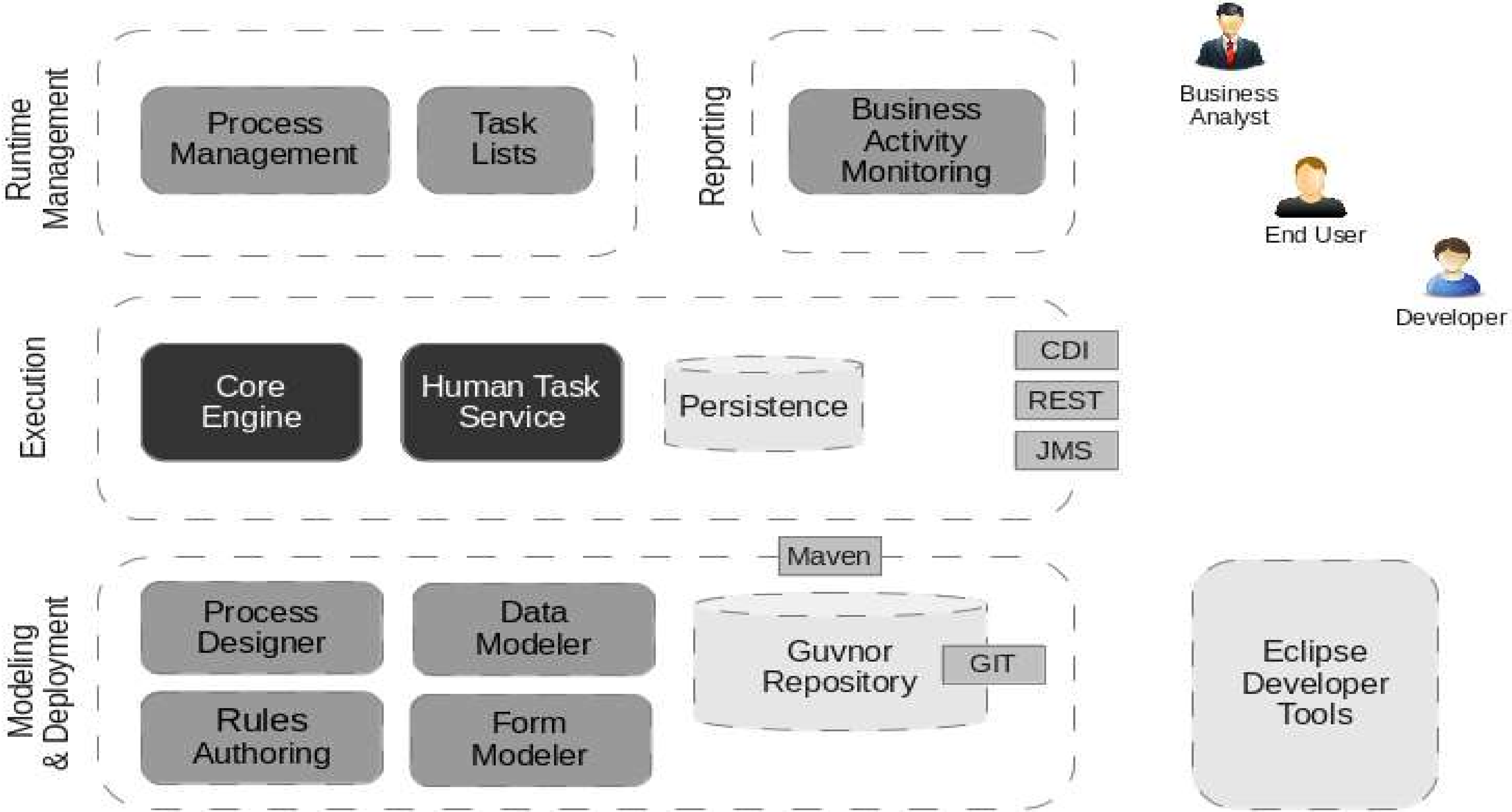
jBPM Execution Server

REST/JMS service for workflows

redhat PAM

Lightweight, embeddable engines (jars)
which run in a Java VM

Web applications (wars)
which run on a Java Application Server



Runtime Management

Process Management

Task Lists

Reporting

Business Activity Monitoring

Business Analyst

End User

Developer

Execution

Core Engine

Human Task Service

Persistence

CDI

REST

JMS

Maven

Modeling & Deployment

Process Designer

Data Modeler

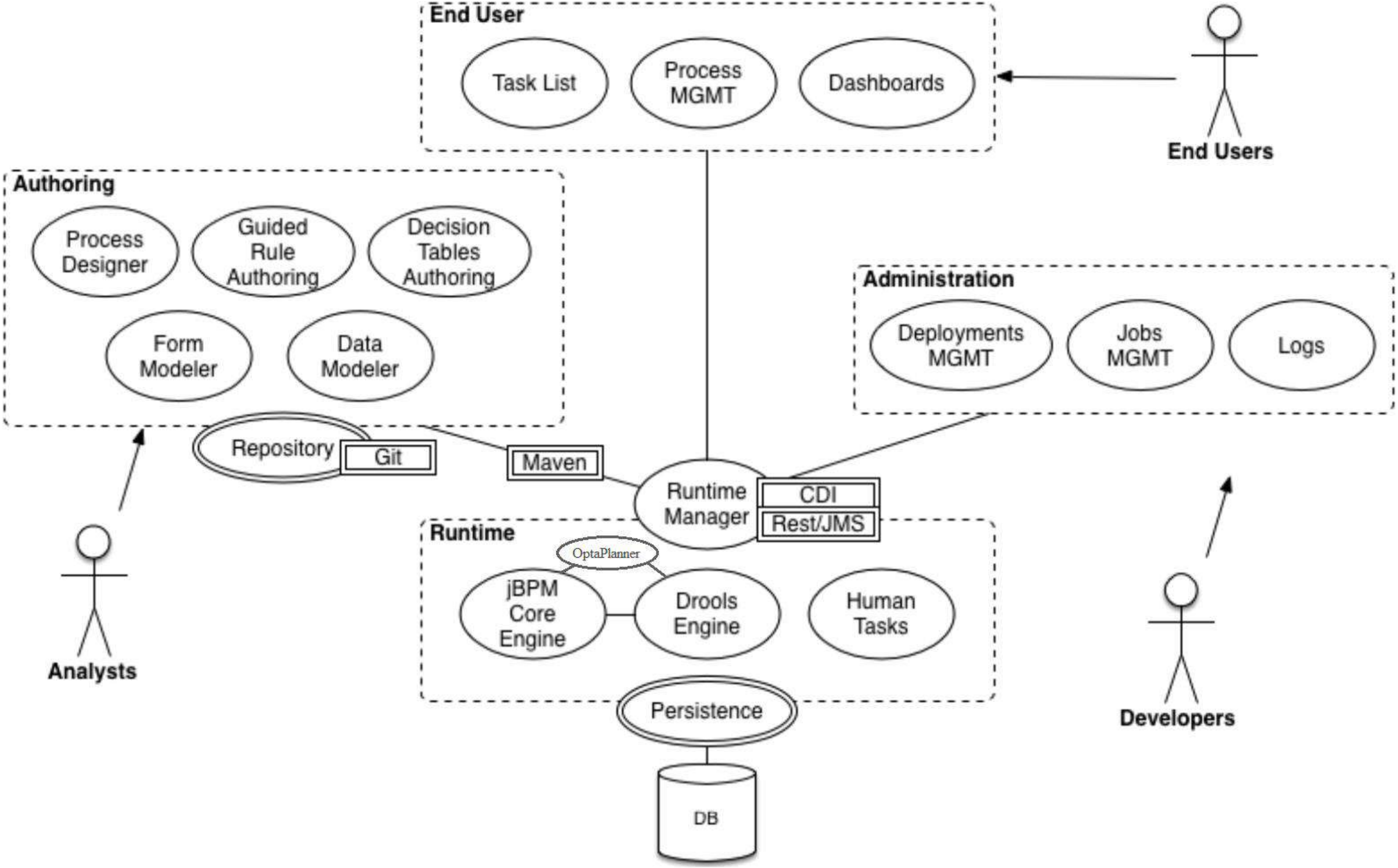
Rules Authoring

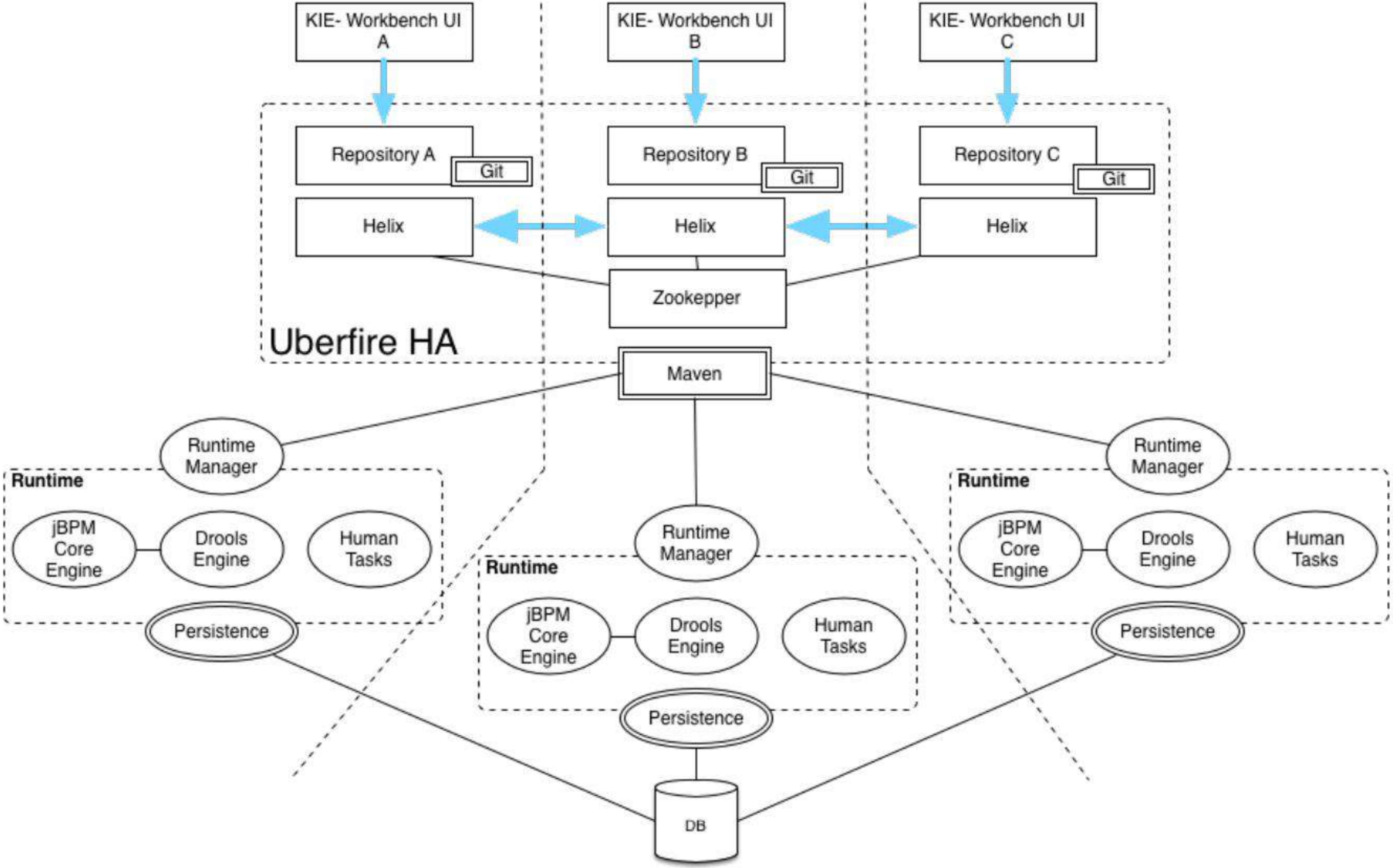
Form Modeler

Guvnor Repository

GIT

Eclipse Developer Tools


End User
Task List
Process MGMT
Dashboards
End Users
Authoring
Process Designer
Guided Rule Authoring
Decision Tables Authoring
Form Modeler
Data Modeler
Administration
Deployments MGMT
Jobs MGMT
Logs
Repository
Git
Maven
Runtime Manager
CDI
Rest/JMS
Runtime
OptaPlanner
jBPM Core Engine
Drools Engine
Human Tasks
Persistence
DB
Analysts
Developers


KIE- Workbench UI A
KIE- Workbench UI B
KIE- Workbench UI C
Repository A
Git
Repository B
Git
Repository C
Git
Helix
Helix
Helix
Zookepper
Uberfire HA
Maven
Runtime Manager
Runtime
jBPM Core Engine
Drools Engine
Human Tasks
Persistence
Runtime Manager
Runtime
jBPM Core Engine
Drools Engine
Human Tasks
Persistence
Runtime Manager
Runtime
jBPM Core Engine
Drools Engine
Human Tasks
Persistence
DB

END OF APPENDICES