



# Základy umělé inteligence

## B4B36ZUI, LS 2019

**Martin Schaefer, Jaromír Janisch, Tomáš Krajník**

`{name.surname}@agents.fel.cvut.cz`

Artificial Intelligence Center, Czech Technical University

# Course Outline



## Lectures

- Tuesday 14:30 - 16:00
- Prof. Michal Pěchouček, Doc. Jiří Kléma, Tomáš Krajník

## Seminars

- Krajník, Schaefer, Janisch; Kléma, Kunc; VINTR
- Assignments – **a lot of programming**, Java knowledge required

## CourseWare

- <https://cw.fel.cvut.cz/wiki/courses/b4b36zui/start>
- Course materials
- Course requirements

## Contacts

- Forum, emails of lecturers on cw

# Seminars Credit Requirements



Participation and active work at all seminars (up to 2 absences without an excuse are allowed).

Gathering at least 25 points from the assignments (out of 50).

Submitting all assignments during the term and receiving at least 50% of points from each of the assignment for the content

- You can choose **one assignment**, for which you **don't have to get 50% of points**.
- The points for the fulfilling 50% of the assignment are computed before the deduction of points for late submission penalty
- The late submission penalty is 1 point for each day after deadline

# Homework Assignments

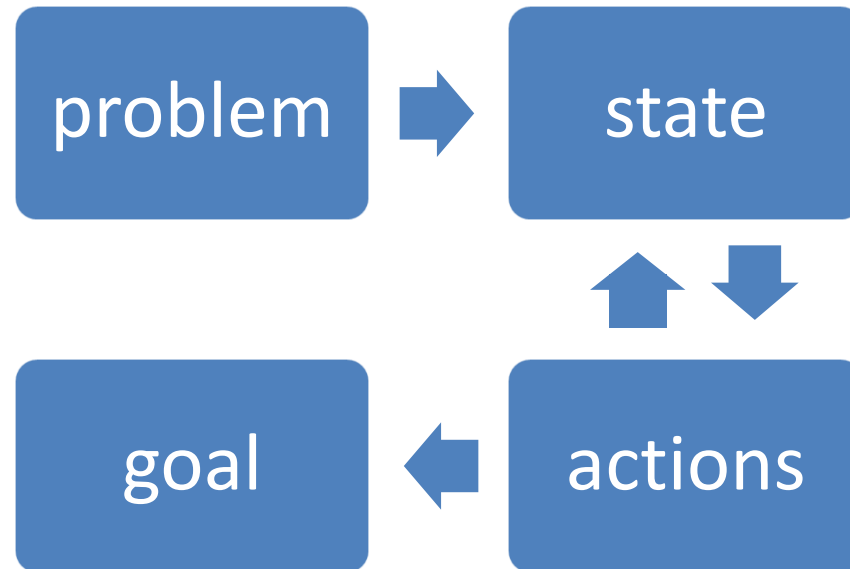


A\* - finding an optimal path

Constraint Satisfaction Programming

Two-player games

# Problem Solving



# Search Problem

State space

---



## Problem

**Initial state** :  $s_0$

**Successor function** :  $x \in S \rightarrow succ(x) \in 2^S$

**Goal test** :  $x \in S \rightarrow goal(x) = T \mid F$

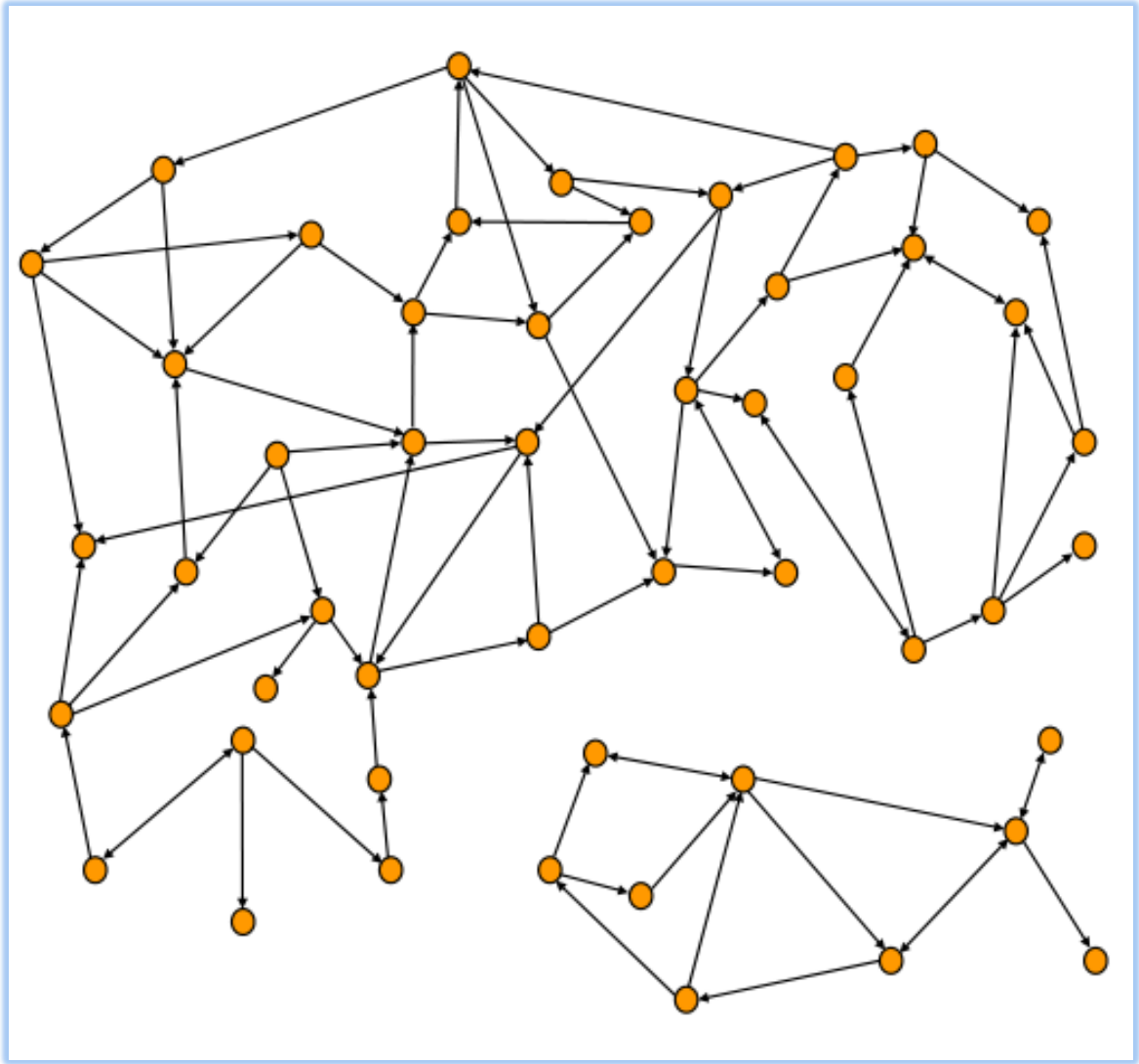
**Arc cost** :  $c(x, succ(x))$

**Solution** is set of actions leading from initial state to a goal state.

**State space** is defined by the initial state and successor function

**Formalization of the state space is the key to solve  
(almost) all symbolic AI problems!**

# State Graph



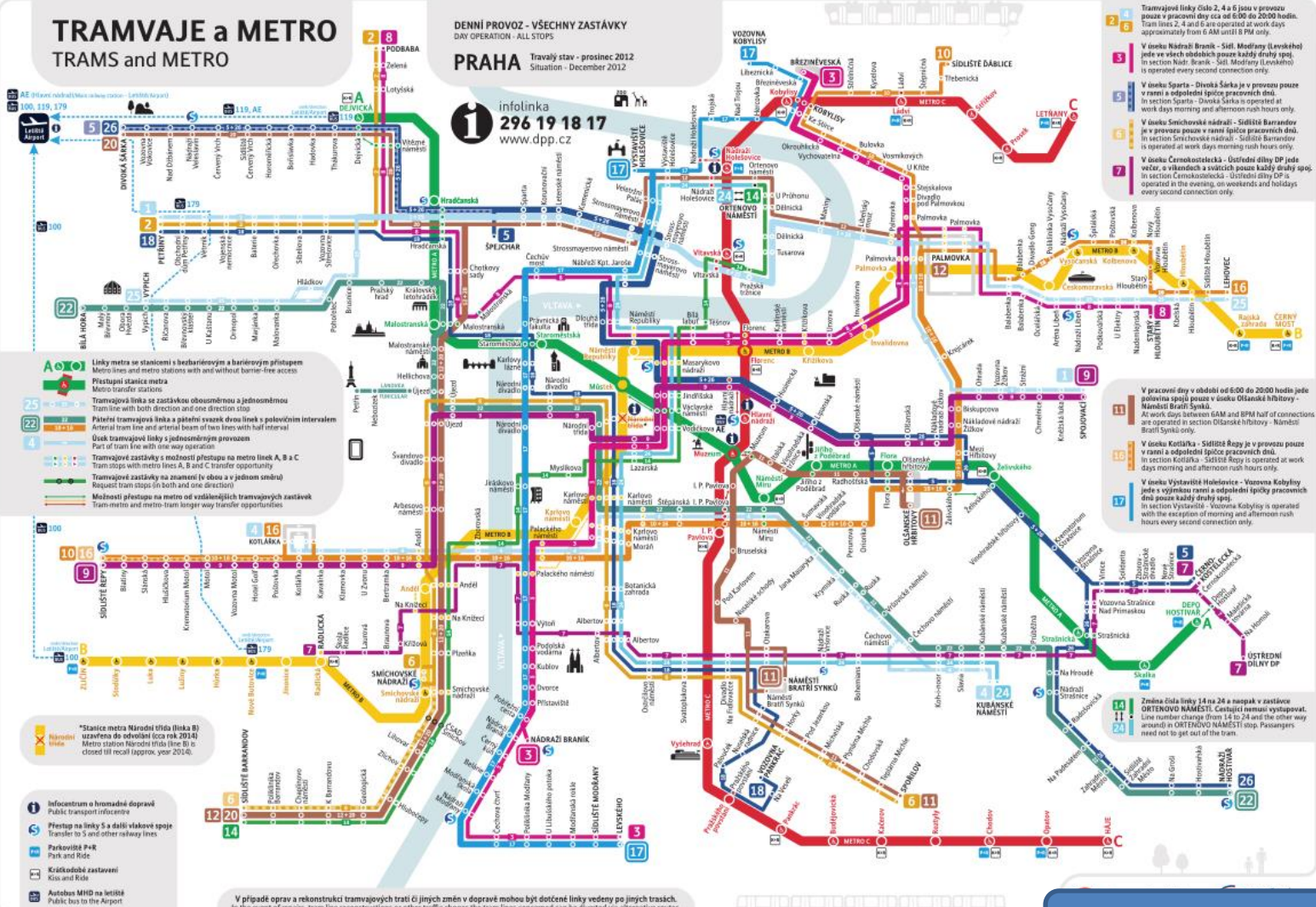
# State Graph

## TRAMVAJE a METRO TRAMS and METRO

DENNÍ PROVOZ - VŠECHNY ZASTÁVKY  
DAY OPERATION - ALL STOPS

PRAHA Trvalý stav - prosinec 2012  
Situation - December 2012

infolinka  
296 19 18 17  
www.dpp.cz



- A** Linky metra se stanicemi s bezbariérovým a bariérovým přístupem  
Metro lines with metro stations with and without barrier-free access
- P** Přestupní stanice metra  
Metro transfer stations
- 1** Tramvajová linka se zastávkou obousměrnou a jednosměrnou  
Tram line with both directions and one direction stop
- 2** Patřící tramvajová linka a páteřní svazek dvou linek s poločim intervalem  
Paternal tram line and external beam of two lines with half interval
- 3** Dvuk tramvajová linka s jednosměrným provozem  
Part of tram line with one way operation
- 4** Tramvajové zastávky s možností přestupu na metru linek A, B a C  
Tram stops with metro lines A, B and C transfer opportunity
- 5** Dvuk tramvajové linky s jednosměrným provozem  
Part of tram line with one way operation
- 6** Tramvajové zastávky na omezení (v obou a v jednom směru)  
Request tram stops (in both and one direction)
- 7** Možnosti přestupu na metru od vzdálených tramvajových zastávek  
Tram-metro and metro-tram longer way transfer opportunities

\*Stanice metra Národní třída (linka B) uzavřena do otevření čca rok 2014  
Metro station Národní třída (line B) is closed till re-open approx. year 2014

- I** Infolinkou o hromadné dopravě  
Public transport infocentre
- T** Přístup na linky S a další vlakové spoje  
Transfer to S and other railway lines
- P** Parkoviště P+R  
Park and Ride
- K** Krátkodobé zastavení  
Kiss and Ride
- A** Autobus MHD na letišti  
Public bus to the Airport

V případě oprav a rekonstrukcí tramvajových tratí či jiných změn v dopravě mohou být dotčené linky vedeny po jiných trasách.  
In the event of repairs, tram line reconstructions or other traffic changes the tram lines concerned can be diverted via alternative routes.

- 2** Tramvajové linky čísla 2, 4 a 6 jsou v provozu pouze v pracovní dny od 6:00 do 20:00 hodin.  
Tram lines 2, 4 and 6 are operated at work days approximately from 6 AM until 8 PM only.
- 3** V lince Nádraží Braník - Sídliště Modřany (L vlnkové) jede v všech obědích směrech každý druhý spoj.  
In section Naď: Braník - Sídliště Modřany (L vlnkové) is operated every second connection only.
- 5** V lince Sparta - Divoká Šárka je v provozu pouze v ranní a odpolední špičce pracovních dnů.  
In section Sparta - Divoká Šárka is operated at work days morning and afternoon rush hours only.
- 6** V lince Smíchovské nádraží - Sídliště Barrandov je v provozu pouze v ranní špičce pracovních dnů.  
In section Smíchovské nádraží - Sídliště Barrandov is operated at work days morning rush hours only.
- 7** V lince Černokostelecká - Ořtřední dílny DP jede večer, o víkendech a svátcích pouze každý druhý spoj.  
In section Černokostelecká - Ořtřední dílny DP, operated in the evening, on weekends and holidays every second connection only.

- 11** V pracovní dny v období od 6:00 do 20:00 hodin jede jediná spoje pouze v lince Otánská hřbitovy - Nádraží Bratří Sýnků.  
At work days between 6AM and 8PM half of connections are operated in section Otánská hřbitovy - Nádraží Bratří Sýnků only.
- 16** V lince Kotlářka - Sídliště Rejpy je v provozu pouze v ranní a odpolední špičce pracovních dnů.  
In section Kotlářka - Sídliště Rejpy is operated at work days morning and afternoon rush hours only.
- 17** V lince Vystřiželk Holesovice - Vozovna Kobylisy jede v pracovních dnech a odpolední špičce pracovních dnů pouze každý druhý spoj.  
In section Vystřiželk Holesovice - Vozovna Kobylisy is operated at work days morning and afternoon rush hours only with the exception of morning and afternoon rush hours every second connection only.

Změna čísla linky 14 na 24 a nespánek v zastávce OŘTŘEDNÍ NÁMĚSTÍ. Číslo linky nemusí vystupovat. Line number change from 14 to 24 and the other way around in OŘTŘEDNÍ NÁMĚSTÍ stop. Passengers need not to get off the tram.

Properties?



# State Space

## Formulation

---



- **Problem** – shortest path in MHD from KN to Dejvice
- **Initial state** –  $s_0$ =Karlovo Namesti
- **Successor function** –  $succ(x)$  →all connected stations
- **Goal test** –  $x$ =Dejvicka (explicit)
- **Arc cost** –  $c(x, y)$ =time of transit between stations  $x, y$
  
- **Solution** – (Karlak-MustekB),(MustekB-MustekA), (MustekA-Staromestska),..., (Hradcanska,Dejvicka)

# State Space

## Examples

---



- Traveling problem
- from Karlak to Dejvice
- from Prague to Snezka
- from Prague to Sydney

# Roomba Robot path planning

---



# The ferryman problem



# Escaping the World Trade Center



- Imagine a huge skyscraper with several elevators. As the input you have:
- set of elevators, where for each you have:
  - - range of the floors that this elevator is operating in
  - - how many floors does this elevator skip (e.g. an elevator can stop only on every second floor, or every fifth floor, etc.)
  - - speed (time in seconds to go up/down one floor)
  - - starting position (number of the floor)



# Escaping the World Trade Center



- Let us assume, that transfer from one elevator to another one takes the same time (given as input -  $t$ ).
  - You are starting in  $k$ th floor and you want to find the quickest way to the ground floor.
  - You can assume that you are alone in the building and elevators do not run by themselves.
- 
1. What are the states?
  2. What is the initial state and the goal state?
  3. What is the cost function?

# Search Problem

State space

---



## Problem

**Initial state** :  $s_0$

**Successor function** :  $x \in S \rightarrow succ(x) \in 2^S$

**Goal test** :  $x \in S \rightarrow goal(x) = T \mid F$

**Arc cost** :  $c(x, succ(x))$

**Solution** is set of actions leading from initial state to a goal state.

**State space** is defined by the initial state and successor function

# Tree Search Algorithm

## Basic Idea

---



Basic idea:

offline, simulated exploration of state space  
by generating successors of already-explored states  
(a.k.a. **expanding** states)

```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
  end
```



# Tree Search Algorithm

## Formulation



**function** TREE-SEARCH(*problem*, *fringe*) **returns** a solution, or failure

*fringe*  $\leftarrow$  INSERT(MAKE-NODE(INITIAL-STATE[*problem*]), *fringe*)

**loop do**

**if** *fringe* is empty **then return** failure

*node*  $\leftarrow$  REMOVE-FRONT(*fringe*)

**if** GOAL-TEST(*problem*, STATE(*node*)) **then return** *node*

*fringe*  $\leftarrow$  INSERTALL(EXPAND(*node*, *problem*), *fringe*)

---

**function** EXPAND(*node*, *problem*) **returns** a set of nodes

*successors*  $\leftarrow$  the empty set

**for each** *action*, *result* **in** SUCCESSOR-FN(*problem*, STATE[*node*]) **do**

*s*  $\leftarrow$  a new NODE

    PARENT-NODE[*s*]  $\leftarrow$  *node*; ACTION[*s*]  $\leftarrow$  *action*; STATE[*s*]  $\leftarrow$  *result*

    PATH-COST[*s*]  $\leftarrow$  PATH-COST[*node*] + STEP-COST(*node*, *action*, *s*)

    DEPTH[*s*]  $\leftarrow$  DEPTH[*node*] + 1

    add *s* to *successors*

**return** *successors*

# Tree Search Algorithm

## Formulation



```
function TREE-SEARCH(problem, fringe) returns a solution, or failure
  fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node ← REMOVE-FRONT(fringe)
    if GOAL-TEST(problem, STATE(node)) then return node
    fringe ← INSERTALL(EXPAND(node, problem), fringe)
```

BFS

Insert at the end

DFS

Insert at the beginning

$s \leftarrow$  a new NODE

PARENT-NODE[ $s$ ] ← *node*; ACTION[ $s$ ] ← *action*; STATE[ $s$ ] ← *result*

PATH-COST[ $s$ ] ← PATH-COST[*node*] + STEP-COST(*node*, *action*,  $s$ )

DEPTH[ $s$ ] ← DEPTH[*node*] + 1

add  $s$  to *successors*

**return** *successors*

# Searching the State Space

## Algorithms

---



- Breadth first search **BFS**
- Depth first search **DFS**
- Depth limited search (DFS with search limit  $l$ )
- Iterative deepening search (Iteratively increase  $l$ )

# BFS/DFS Exercises

