Úvod do umělé inteligence (NAIL120)

4. cvičení

Jirka Fink

https://ktiml.mff.cuni.cz/~fink/

Katedra teoretické informatiky a matematické logiky
Matematicko-fyzikální fakulta
Univerzita Karlova v Praze

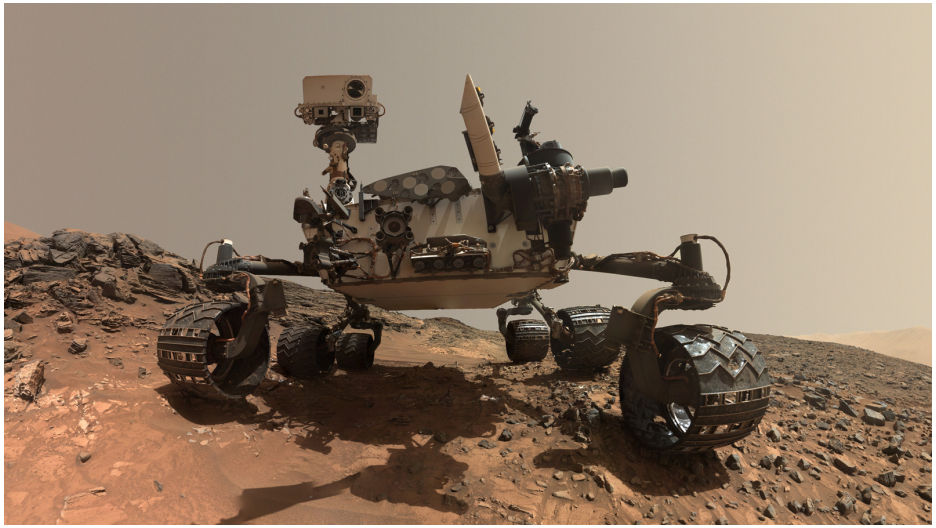
Letní semestr 2025/26

Poslední změna 16. března 2026

Příklady obecných postupů k řešení algoritmických úloh

- Lineární programování
- Konvexní optimalizace
- Constraint satisfaction programming (splňování podmínek)
- SAT (splnitelnost logických formulí)
- Automatické plánování

Automatické plánování



Základní idea

Najít posloupnost akcí, která postupně mění stav z počátečního do cílového.

- Stav je popsán konečnou množinou proměnných
  - Základní řešiče mají jen logické proměnné, pokročilé řešiče mají i číselné proměnné

Základní idea

Najít posloupnost akcí, která postupně mění stav z počátečního do cílového.

- Stav je popsán konečnou množinou proměnných
  - Základní řešiče mají jen logické proměnné, pokročilé řešiče mají i číselné proměnné
- Počáteční stav: Ohodnocení všech proměnných na začátku plánu

Základní idea

Najít posloupnost akcí, která postupně mění stav z počátečního do cílového.

- Stav je popsán konečnou množinou proměnných
  - Základní řešiče mají jen logické proměnné, pokročilé řešiče mají i číselné proměnné
- Počáteční stav: Ohodnocení všech proměnných na začátku plánu
- Cílový stav: Logické nebo aritmetické podmínky na hledaný stav

Základní idea

Najít posloupnost akcí, která postupně mění stav z počátečního do cílového.

- Stav je popsán konečnou množinou proměnných
  - Základní řešiče mají jen logické proměnné, pokročilé řešiče mají i číselné proměnné
- Počáteční stav: Ohodnocení všech proměnných na začátku plánu
- Cílový stav: Logické nebo aritmetické podmínky na hledaný stav
- Ohodnocení proměnných se mění pomocí akcí
  - Každá akce má předpoklady, které musí být splněny, aby bylo možné akci provést
  - Každá akce má efekty měnící hodnoty proměnných
  - Parametry, aby akce mohli být obecné

Základní idea

Najít posloupnost akcí, která postupně mění stav z počátečního do cílového.

- Stav je popsán konečnou množinou proměnných
  - Základní řešiče mají jen logické proměnné, pokročilé řešiče mají i číselné proměnné
- Počáteční stav: Ohodnocení všech proměnných na začátku plánu
- Cílový stav: Logické nebo aritmetické podmínky na hledaný stav
- Ohodnocení proměnných se mění pomocí akcí
  - Každá akce má předpoklady, které musí být splněny, aby bylo možné akci provést
  - Každá akce má efekty měnící hodnoty proměnných
  - Parametry, aby akce mohli být obecné

Planning Domain Definition Language (PDDL)

- PDDL je jazyk, který se snaží standardizovat plánování v UI
- Popis modelu je rozdělen do dvou souborů
  - Problém: Seznam objektů, počáteční a cílový stav
  - Doména: Seznam proměnných a akcí

Příklad: Přesun míčů mezi místnostmi

Problem

```
(define (problem balls)
  (:domain balls)
  (:objects room1 room2 ball1)
  (:init (room room1)
         (room room2)
         (ball ball1)
         (at ball1 room1))
  (:goal (at ball1 room2)))
```

Příklad: Přesun míčů mezi místnostmi

Problem

```
(define (problem balls)
  (:domain balls)
  (:objects room1 room2 ball1)
  (:init (room room1)
         (room room2)
         (ball ball1)
         (at ball1 room1))
  (:goal (at ball1 room2)))
```

Domain

```
(define (domain balls)
  (:predicates (room ?r) (ball ?b) (at ?b ?r))
  (:action move
    :parameters (?b ?from ?to)
    :precondition (and (ball ?b)
                       (room ?from)
                       (room ?to)
                       (at ?b ?from))
    :effect (and (at ?b ?to)
                 (not (at ?b ?from)))))
```

Příklad: Přesun míčů mezi místnostmi

Problem

```
(define (problem balls)
  (:domain balls)
  (:objects room1 room2 ball1)
  (:init (room room1)
         (room room2)
         (ball ball1)
         (at ball1 room1))
  (:goal (at ball1 room2)))
```

Domain

```
(define (domain balls)
  (:predicates (room ?r) (ball ?b) (at ?b ?r))
  (:action move
    :parameters (?b ?from ?to)
    :precondition (and (ball ?b)
                       (room ?from)
                       (room ?to)
                       (at ?b ?from))
    :effect (and (at ?b ?to)
                 (not (at ?b ?from)))))
```

Nalezený plán

```
(move ball1 room1 room2)
```

Špatný popis úlohy můžeme vést k nereálnému výsledku

Problem

```
(define (problem balls)
  (:domain balls)
  (:objects room1 room2 room3 room4 ball1)
  (:init (room room1) (room room2) (room room3) (room room4)
         (ball ball1)
         (at ball1 room1))
  (:goal (and (at ball1 room2)
              (at ball1 room3)
              (at ball1 room4))))
```

Špatný popis úlohy můžeme vést k nereálnému výsledku

Problem

```
(define (problem balls)
  (:domain balls)
  (:objects room1 room2 room3 room4 ball1)
  (:init (room room1) (room room2) (room room3) (room room4)
         (ball ball1)
         (at ball1 room1))
  (:goal (and (at ball1 room2)
              (at ball1 room3)
              (at ball1 room4))))
```

Domain

```
(define (domain balls)
  (:predicates (room ?r) (ball ?b) (at ?b ?r))
  (:action move
    :parameters (?b ?from ?to)
    :precondition (at ?b ?from)
    :effect (at ?b ?to)))
```

Špatný popis úlohy můžeme vést k nereálnému výsledku

Problem

```
(define (problem balls)
  (:domain balls)
  (:objects room1 room2 room3 room4 ball1)
  (:init (room room1) (room room2) (room room3) (room room4)
         (ball ball1)
         (at ball1 room1))
  (:goal (and (at ball1 room2)
              (at ball1 room3)
              (at ball1 room4))))
```

Domain

```
(define (domain balls)
  (:predicates (room ?r) (ball ?b) (at ?b ?r))
  (:action move
    :parameters (?b ?from ?to)
    :precondition (at ?b ?from)
    :effect (at ?b ?to)))
```

Nalezený plán

```
(move ball1 room1 room4)
(move ball1 room1 room3)
(move ball1 room1 room2)
```

Příklad: Strip Problem

Popis problému

Máme robota, který má dvě chapadla, do kterých umí uchopit míč, položit jej na zem a přesouvat se mezi místnostmi.

```
(define (problem strips-gripper2)
   (:domain gripper-strips)
   (:objects rooma roomb ball1 ball2 left right)
   (:init (room rooma)
          (room roomb)
          (ball ball1)
          (ball ball2)
          (gripper left)
          (gripper right)
          (at-robby rooma)
          (free left)
          (free right)
          (at ball1 rooma)
          (at ball2 rooma))
   (:goal (at ball1 roomb)))
```

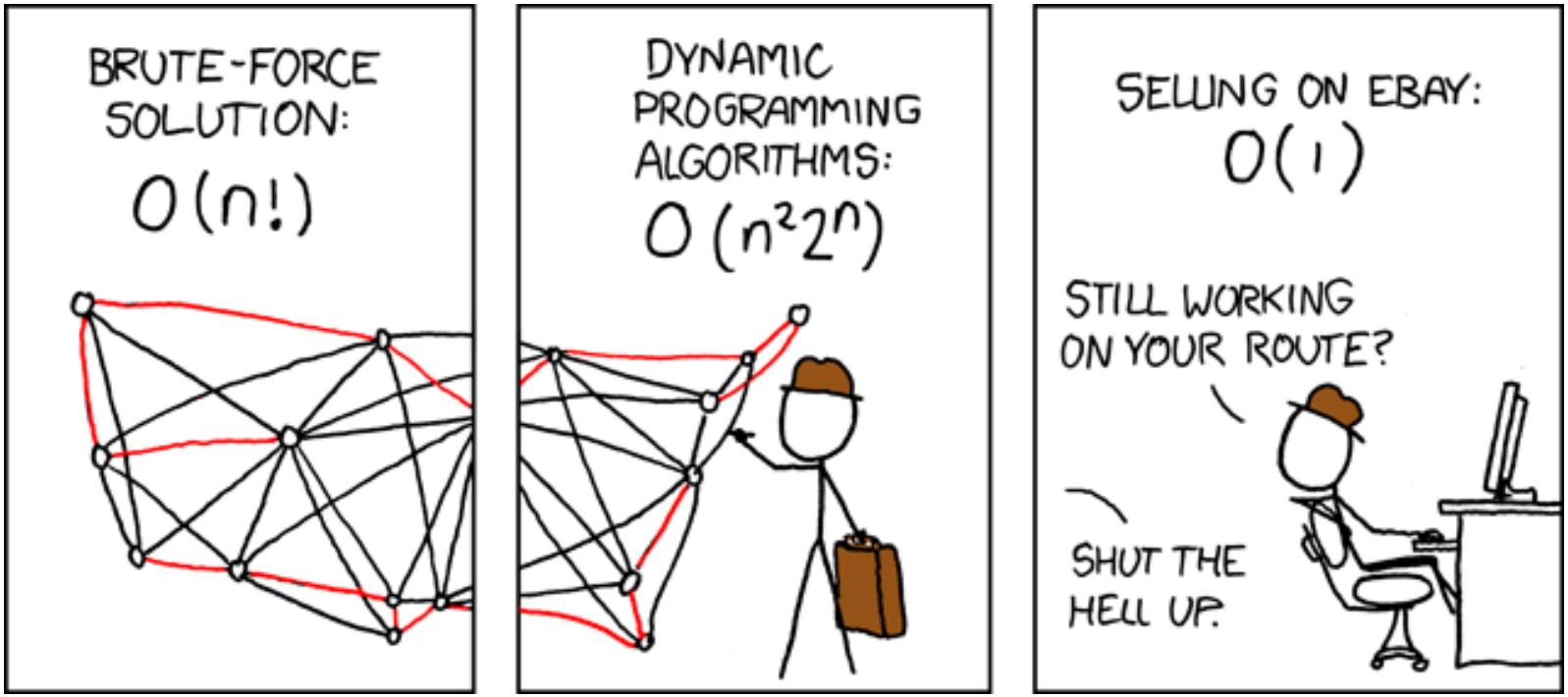
Příklad: Strip Domain 1/2

```
(define (domain gripper-strips)
  (:predicates (room ?r) (ball ?b) (gripper ?g) (at-robby ?r)
               (at ?b ?r) (free ?g) (carry ?o ?g))
  (:action move
   :parameters (?from ?to)
   :precondition (and (room ?from)
                      (room ?to)
                      (at-robby ?from))
   :effect (and (at-robby ?to)
                (not (at-robby ?from))))
```

Příklad: Strip Domain 2/2

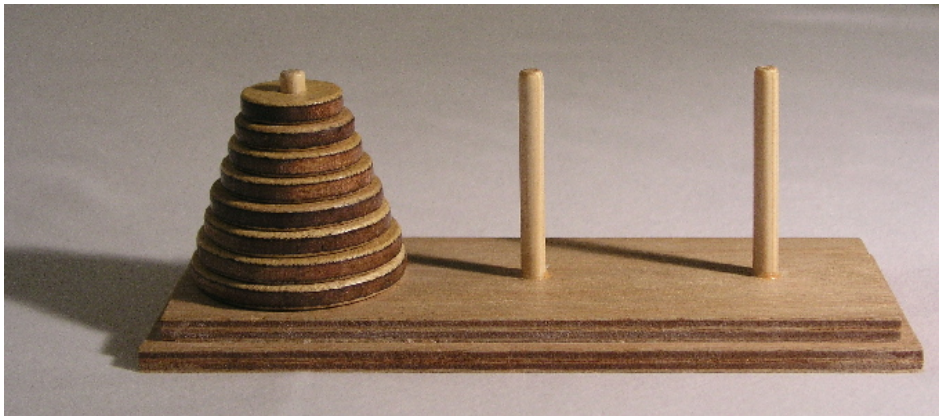
```
(:action pick
 :parameters (?obj ?room ?gripper)
 :precondition (and (ball ?obj)
                    (room ?room)
                    (gripper ?gripper)
                    (at ?obj ?room)
                    (at-robby ?room)
                    (free ?gripper))
 :effect (and (carry ?obj ?gripper)
              (not (at ?obj ?room))
              (not (free ?gripper))))
(:action drop
 :parameters (?obj ?room ?gripper)
 :precondition (and (ball ?obj)
                    (room ?room)
                    (gripper ?gripper)
                    (carry ?obj ?gripper)
                    (at-robby ?room))
 :effect (and (at ?obj ?room)
              (free ?gripper)
              (not (carry ?obj ?gripper)))))
```

Popište hledání Hamiltonovské kružnice pomocí PDDL



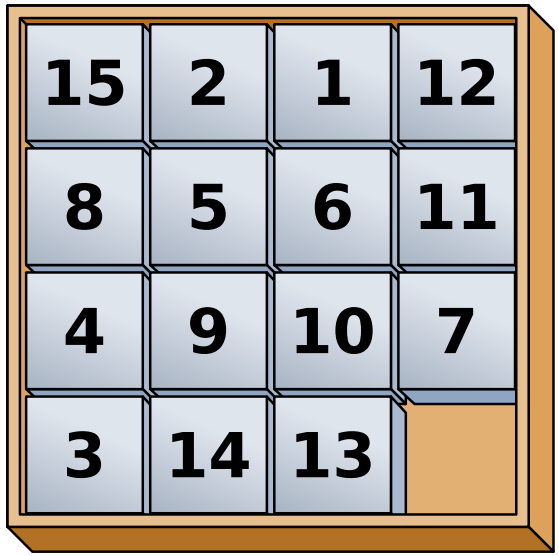
Source: `https://xkcd.com/399/`

Hanojské věže



- Cílem je přemístit všechny kotouče na druhou věž.
- V jednom tahu lze přemístit vrchní kotouč z některé věže a položit jej na jinou věž mající větší kotouč navrchu.
- Popište hledání řešení pomocí PDDL.
- Ilustrativní animace

Popište řešení Loydovy patnácky pomocí PDDL



5. domácí úkol: PDDL

Zadání (zkráceno)

- Máme dány počáteční a cílové pozice krabic, které máme převést pomocí aut
- Každé auto kapacitu jedné krabice
- Napište PDDL doménu obsahující právě tyto akce
  - load(box car place): Naloží krabici do auta.
  - unload(box car place): Vyloží krabici z auta.
  - move(car origin destination): Přesune auto.
- Napište jednu doménu, která vyřeší všech 10 problémů.

5. domácí úkol: PDDL

Zadání (zkráceno)

- Máme dány počáteční a cílové pozice krabic, které máme převést pomocí aut
- Každé auto kapacitu jedné krabice
- Napište PDDL doménu obsahující právě tyto akce
  - load(box car place): Naloží krabici do auta.
  - unload(box car place): Vyloží krabici z auta.
  - move(car origin destination): Přesune auto.
- Napište jednu doménu, která vyřeší všech 10 problémů.

Knihovna pro Python

- pyperplan: Jednoduchý řešič PDDL
  `https://github.com/aibasel/pyperplan`