Úvod do umělé inteligence (NAIL120)

2. 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 4. března 2026

Přístupy k řešení algoritmických úloh

Hledání specializovaných algoritmů na konkrétní úlohy

Přístupy k řešení algoritmických úloh

Hledání specializovaných algoritmů na konkrétní úlohy

- Dijkstrův algoritmus na hledání nejkratší cesty
- Borůvkův algoritmus na hledání minimální kostry
- Aho-Corasic algoritmus na vyhledávání v textu
- Dinicův algoritmus na hledání maximálního toku
- Strassenův algoritmus na násobení matic

Přístupy k řešení algoritmických úloh

Hledání specializovaných algoritmů na konkrétní úlohy

- Dijkstrův algoritmus na hledání nejkratší cesty
- Borůvkův algoritmus na hledání minimální kostry
- Aho-Corasic algoritmus na vyhledávání v textu
- Dinicův algoritmus na hledání maximálního toku
- Strassenův algoritmus na násobení matic

Nebylo by lepší mít obecný algoritmus na řešení podobných úloh?

Přístupy k řešení algoritmických úloh

Hledání specializovaných algoritmů na konkrétní úlohy

- Dijkstrův algoritmus na hledání nejkratší cesty
- Borůvkův algoritmus na hledání minimální kostry
- Aho-Corasic algoritmus na vyhledávání v textu
- Dinicův algoritmus na hledání maximálního toku
- Strassenův algoritmus na násobení matic

Nebylo by lepší mít obecný algoritmus na řešení podobných úloh?

Je možné na každou úlohu najít algoritmus?

Přístupy k řešení algoritmických úloh

Hledání specializovaných algoritmů na konkrétní úlohy

- Dijkstrův algoritmus na hledání nejkratší cesty
- Borůvkův algoritmus na hledání minimální kostry
- Aho-Corasic algoritmus na vyhledávání v textu
- Dinicův algoritmus na hledání maximálního toku
- Strassenův algoritmus na násobení matic

Nebylo by lepší mít obecný algoritmus na řešení podobných úloh?

Je možné na každou úlohu najít algoritmus?

Neexistuje algoritmus, který rozhodne, zda daný program zastaví.

Obecnější postupy řešení algoritmických úloh

Jak zkonstruovat postup na řešení obecnější třídy úloh?

- Zvolit rozumnou třídu úloh
- Vymyslet zápis úloh této třídy
- Najít obecný algoritmus na řešení těchto úloh

Obecnější postupy řešení algoritmických úloh

Jak zkonstruovat postup na řešení obecnější třídy úloh?

- Zvolit rozumnou třídu úloh
- Vymyslet zápis úloh této třídy
- Najít obecný algoritmus na řešení těchto úloh

Příklady tříd úloh

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

Constraint satisfaction programming (CSP)

Popis úlohy pomocí CSP

- Konečná množina proměnných
- Každá proměnná má danou konečnou množinu hodnot, kterých může nabývat
- Množina podmínek na ohodnocení proměnných (typy podmínek závisí na řešiči)
- Řešiče CSP hledají ohodnocení proměnných splňující všechny podmínky
- Příklady řešičů: Python-constraint, IBM ILOG CP, Minion

Constraint satisfaction programming (CSP)

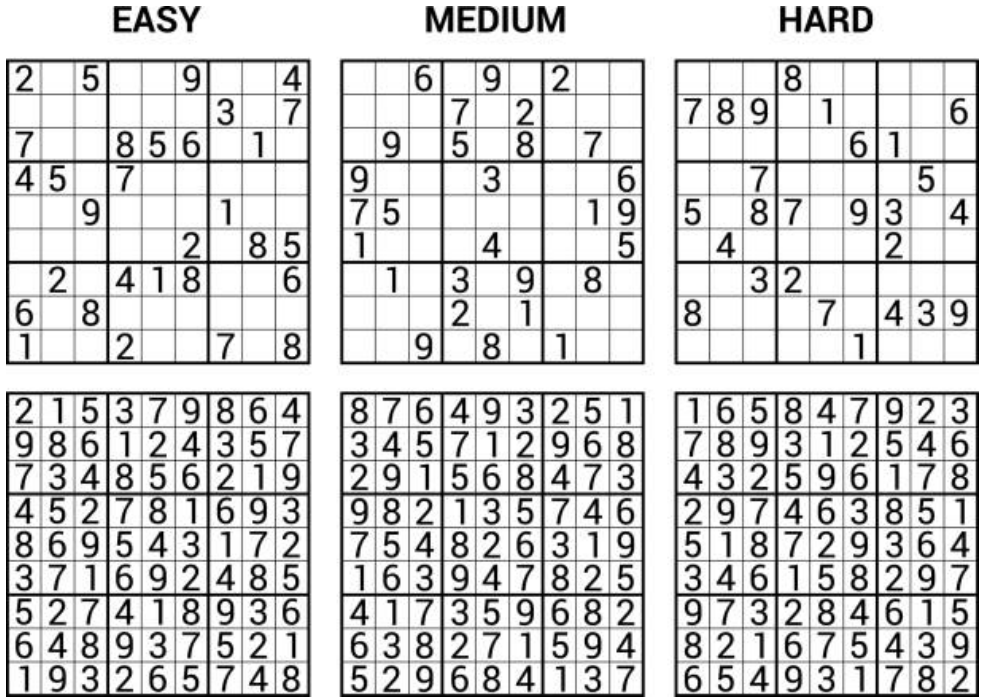
Popis úlohy pomocí CSP

- Konečná množina proměnných
- Každá proměnná má danou konečnou množinu hodnot, kterých může nabývat
- Množina podmínek na ohodnocení proměnných (typy podmínek závisí na řešiči)
- Řešiče CSP hledají ohodnocení proměnných splňující všechny podmínky
- Příklady řešičů: Python-constraint, IBM ILOG CP, Minion

Podmínky podporované řešičem python-constraint

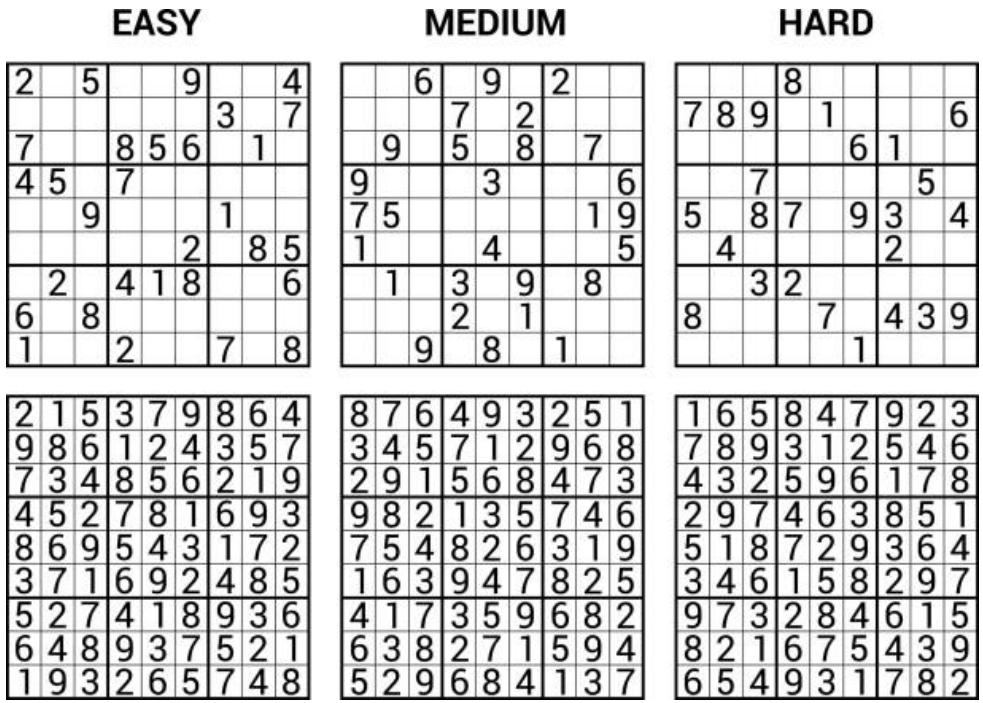
- **AllEqualConstraint**: Ohodnocení proměnných musí být stejné
- **AllDifferentConstraint**: Ohodnocení musí být po dvou různé
- **ExactSumConstraint**: Součet ohodnocení se musí rovnat dané hodnotě
- **MaxSumConstraint**: Součet ohodnocení se musí nejvýše daná hodnota
- **MinSumConstraint**: Součet ohodnocení se musí alespoň daná hodnota
- **InSetConstraint**: Pro danou k-tici proměnných je daná množina k-tic hodnot, které proměnné můžou nabývat
- **NotInSetConstraint**: Zakázaná množina ohodnocení
- **FunctionConstraint**: Podmínka je dána funkcí, která rozhodne, zda dané ohodnocení je přípustné

Constraint satisfaction programming (CSP)



Cvičení: Popište sudoku jako úlohu CSP

Constraint satisfaction programming (CSP)



Cvičení: Popište sudoku jako úlohu CSP

- Proměnné $x_{ij} \in \{1, \ldots, 9\}$ pro $i, j \in \{1, \ldots, 9\}$
- Podmínky $(x_{ij}, x_{i'j'}) \in \{(a, b);\ a, b \in \{1, \ldots, 9\},\ a \neq b\}$ kdykoliv pozice ij a $i'j'$ mají mít různé hodnoty
- Je-li na pozici ij předepsaná hodnota a, tak přidáme podmínku $x_{ij} \in \{a\}$
- Ohodnocení proměnných splňující všechny podmínky odpovídá řešení sudoku a opačně

Perfektní párování

Příklad

- Perfektní párování grafu je podmnožina hran P taková, že každý vrchol má právě jednu incidentní hranu v P
- Popište hledání perfektního párování pomocí CSP

Perfektní párování

Příklad

- Perfektní párování grafu je podmnožina hran P taková, že každý vrchol má právě jednu incidentní hranu v P
- Popište hledání perfektního párování pomocí CSP

CSP

- Proměnné $x_e \in \{0, 1\}$ pro všechny hrany e
- Pro každý vrchol součet ohodnocení incidentních hran musí být právě jedna

Převod na rozhodovací problém

- Chtěli bychom najít řešení $x \in M$ maximalizující funkci $f(x)$
- Ale CSP umí řešit jen rozhodovací problémy

Převod na rozhodovací problém

- Chtěli bychom najít řešení $x \in M$ maximalizující funkci $f(x)$
- Ale CSP umí řešit jen rozhodovací problémy
- Předpokládejme, že umíme pomocí CSP popsat problém nalezení hledání $x \in M$ splňující $f(x) \geq A$ pro libovolné A
- Maximální $x \in M$ najdeme opakováným řešením rozhodnovací úlohy pomocí
  - postupného zmenšování/zvětšování A
  - binárního půlení

Převod na rozhodovací problém

- Chtěli bychom najít řešení $x \in M$ maximalizující funkci $f(x)$
- Ale CSP umí řešit jen rozhodovací problémy
- Předpokládejme, že umíme pomocí CSP popsat problém nalezení hledání $x \in M$ splňující $f(x) \geq A$ pro libovolné A
- Maximální $x \in M$ najdeme opakováným řešením rozhodnovací úlohy pomocí
  - postupného zmenšování/zvětšování A
  - binárního půlení

Plánování on-line schůzek

- Každé schůzky se musí účastnit daná množina lidí
- Schůzky jsou stejně dlouhé a trvají jeden blok
- Všechny schůzky jednoho člověka musí být v různých blocích
- Pomocí CSP najděte plán s minimálním počtem bloků

Příklad

Dosaďte za písmena S, E, N, D, M, O, R, Y cifry 0 až 9 tak, aby

- různým písmenům byla přiřazena různá čísla
- S i M byla různá od 0
- platila rovnost

		S	E	N	D
+		M	O	R	E
	M	O	N	E	Y

Cílem není najít řešení, ale popsat úlohu pomocí CSP.

Příklad

Dosaďte za písmena S, E, N, D, M, O, R, Y cifry 0 až 9 tak, aby

- různým písmenům byla přiřazena různá čísla
- S i M byla různá od 0
- platila rovnost

$$\begin{array}{cccccc} & & S & E & N & D \\ + & & M & O & R & E \\ \hline & M & O & N & E & Y \end{array}$$

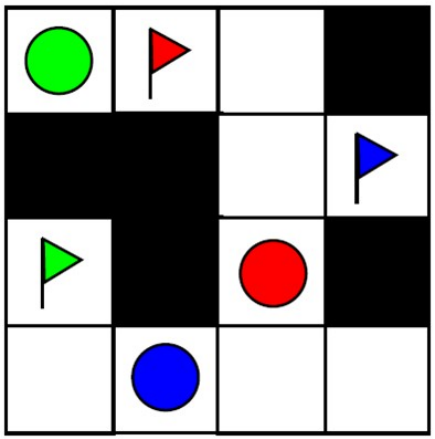
Cílem není najít řešení, ale popsat úlohu pomocí CSP.

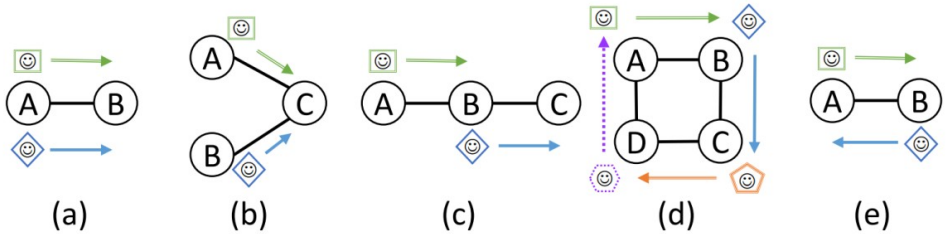
Jednoznačné řešení

9567 + 1085 = 10652

Multi-agent pathfinding

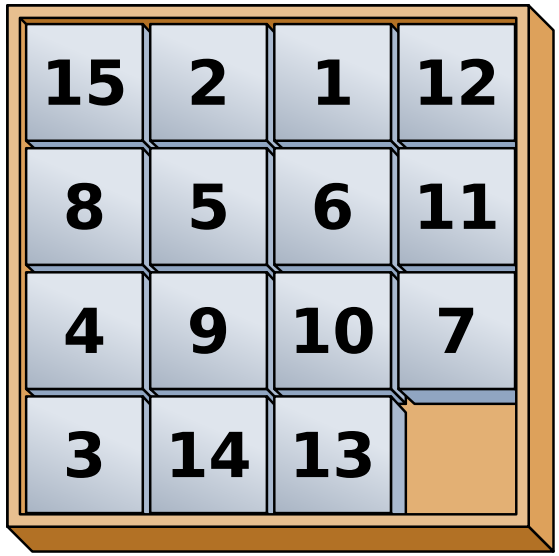
Multi-agent pathfinding





- Každý agent má daný cíl
- Agenti se pohybují simultálně
- Agenti nesmí kolidovat
- Minimalizujeme čas, kdy dorazí poslední

Lze řešení Loydovy patnácky zapsat pomocí CSP?



- Jaké použít proměnné?
- Jaké kombinace ohodnocení proměnných jsou přípustné?

2. domácí úkol: CSP

Zadání (zkráceno)

Pomocí CSP najděte úplné obarvení grafu (total chromatic index)

- Máme obarvit vrcholy i hrany grafu pomocí minimálního počtu barev
- Každé dva sousední vrcholy musí mít různou barvu
- Každé dvě hrany sdílející společný vrchol musí mít různou barvu
- Hrana a její koncové vrcholy musí mít různou barvu

2. domácí úkol: CSP

Zadání (zkráceno)

Pomocí CSP najděte úplné obarvení grafu (total chromatic index)

- Máme obarvit vrcholy i hrany grafu pomocí minimálního počtu barev
- Každé dva sousední vrcholy musí mít různou barvu
- Každé dvě hrany sdílející společný vrchol musí mít různou barvu
- Hrana a její koncové vrcholy musí mít různou barvu

Knihovny pro Python

- networkx: Knihovna pro práci s grafy
  `https://pypi.org/project/networkx/`
- python-constraint: Triviální řešič CSP
  `https://pypi.org/project/python-constraint/`

2. domácí úkol: CSP

Rady

- Přečtěte si pořádně definici úplného barvení a základní pozorování `https://en.wikipedia.org/wiki/Total_coloring`
- Přečtěte si dokumentaci a příklady na CSP `https://github.com/python-constraint/python-constraint/tree/master/examples`
- Ověřte si, že instalujete knihovnu python-constraint a nikoliv jinou
- Přečtěte si zdrojové kódy k domácí úloze
- Buďte trpělivý, v umělé inteligenci nemusí všechny teoreticky správné postupy fungovat v praxi ideálně
- Cílem úkolu je vyzkoušet si jednoduchý CSP řešič na jednoduché úloze