Úvod do umělé inteligence (NAIL120)

10. cvičení

Jirka Fink

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

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

Letni semestr 2021/22

Poslední změna 26. dubna 2022

6. domácí úkol: Bellman update

Zadání (zkráceno)

- Na Marsu přistane robot, který se má dostat na základnu
- Přistání není 100 % úspěšné, takže
  - nepřistál přímo na základně, ale musí k ní dojet
  - poškodili se motory a robot často jede jinam než řídící jednotka zadala
- Naštěstí funguje alespoň lokalizace, takže robot vždy ví, kde se na čtvercové mřížce nachází
- Přesun po povrchu Marsu není ještě bezpečný, ale pro každé políčko je známá pravděpodobnost, že se na něm robot nenávratně zasekne
- Cílem je najít nejbezpečnější cestu na základnu
  - vzhledem k poškozeným motorům nelze předem spočítat optimální cestu
  - proto pro každou pozici spočítáme nejlepší pokyn, který může řídící jednotka zadat

Teorie her

Popis

Teorie her analyzuje široké spektrum konfliktních rozhodovacích situací, kde může docházet ke střetu zájmů.

Teorie her

Popis

Teorie her analyzuje široké spektrum konfliktních rozhodovacích situací, kde může docházet ke střetu zájmů.

Typy her

- Kooperativní/kompetitivní: Můžou hráči spolupracovat a vytvářet koalice?
- Simultálnní/tahové: Hrají hráči současně?
- S úplnou informací: Mají hráči všechny informace?
- Stochastické/deterministické: Ovlivňuje výsledek hry náhodný jev?
- Konečné: Je hra vždy konečná?

Teorie her

Popis

Teorie her analyzuje široké spektrum konfliktních rozhodovacích situací, kde může docházet ke střetu zájmů.

Typy her

- Kooperativní/kompetitivní: Můžou hráči spolupracovat a vytvářet koalice?
- Simultálnní/tahové: Hrají hráči současně?
- S úplnou informací: Mají hráči všechny informace?
- Stochastické/deterministické: Ovlivňuje výsledek hry náhodný jev?
- Konečné: Je hra vždy konečná?

Příklady her

- Šachy, go, poker
- Aukce
- Maticové hry

Příklad maticové hry (Battle of the sexes)

Kluk a holka by chtěli strávit společné odpoledne. Holka by raději šla do kina, kluk na fotbal, ale oba by raději byli spolu.

	holka jde do kina	holka jde na fotbal
kluk jde do kina	(5,6)	(1,1)
kluk jde na fotbal	(2,2)	(6,5)

První číslo udává užitek pro kluka, druhé pro holku.

Triviální hra (NIM)

- Máme hromádku n sirek, které dva hráči postupně odebírají
- Hráč může odebrat 1, 2 nebo 3 sirky
- Vyhrává hráč, který odebere poslední sirku
- Jakou strategii má hráč zvolit?

Pojmy

Triviální hra (NIM)

- Máme hromádku n sirek, které dva hráči postupně odebírají
- Hráč může odebrat 1, 2 nebo 3 sirky
- Vyhrává hráč, který odebere poslední sirku
- Jakou strategii má hráč zvolit?

Pojmy

- Stav: Úplný popis situace hry
- Počáteční a koncový stav
- Pro koncové stavy pravidla určují, kdo vyhrává
- Stav je vyhrávající pro daného hráče, pokud může hrát tak, aby nakonec vyhrál

Pojmy

Triviální hra (NIM)

- Máme hromádku n sirek, které dva hráči postupně odebírají
- Hráč může odebrat 1, 2 nebo 3 sirky
- Vyhrává hráč, který odebere poslední sirku
- Jakou strategii má hráč zvolit?

Pojmy

- Stav: Úplný popis situace hry
- Počáteční a koncový stav
- Pro koncové stavy pravidla určují, kdo vyhrává
- Stav je vyhrávající pro daného hráče, pokud může hrát tak, aby nakonec vyhrál

Hry s užitkem

- Pro každý konečný stav je dáno, jaký užitek má který hráč
- Každý hráč se snaží hrát tak, aby maximalizoval svůj užitek
- Hry s nulových součtem: Součet všech užitků v každém koncovém stavu je nulový

Příklad

NIM 2

- Máme hromádku n sirek, které dva hráči postupně odebírají
- Počet odebraných sirek musí být Fibonacciho číslo (například)
- Vyhrává hráč, který odebere poslední sirku
- Jak rozhodnout, zda je daný stav vyhrávající?

Alpha-beta prořezávání (prunning)

Zpětné prořezávání

Alpha-beta prořezávání (prunning)

Zpětné prořezávání

```
1 winning[0 . . . n] = False
2 for i from 1 to n do
3     for Fibonacci numbers j smaller or equal to i do
4         if not winning[i-j] then
5             winning[i] = True
6             break # Terminate the inner loop
```

Alpha-beta prořezávání (prunning)

Zpětné prořezávání

```
1 winning[0 . . . n] = False
2 for i from 1 to n do
3     for Fibonacci numbers j smaller or equal to i do
4         if not winning[i-j] then
5             winning[i] = True
6             break # Terminate the inner loop
```

Výhody a nevýhody při obecném použití

- Každý stav je vyhodnocen nejvýše jednou
- V netriviálních hrách nám nebude stačit paměť
- Nemusí být jednoduché určit všechny vyhrávající koncové stavy
- Nemusíme umět procházet stavy ve zpětném pořadí

Alpha-beta prořezávání (prunning)

Dopředné prořezávání

Dopředné prořezávání

```
1 Procedure is_winning(n)
2     if n is a Fibonacci number then
3         return True
4     for Fibonacci numbers i smaller than n do
5         if not is_winning(n-i) then
6             return True
7     return False
```

Alpha-beta prořezávání (prunning)

Dopředné prořezávání

```
1 Procedure is_winning(n)
2     if n is a Fibonacci number then
3         return True
4     for Fibonacci numbers i smaller than n do
5         if not is_winning(n-i) then
6             return True
7     return False
```

Výhody a nevýhody při obecném použití

- Vyžaduje málo paměti
- Některé stavy mohou být opakovaně prohledávány
- V netriviálních hrách se výsledku nedožijeme

Alpha-beta prořezávání v asymetrických hrách

Příklad

První hráč odebírá Fibonacciho čísla, druhý mocniny dvojky

Příklad

První hráč odebírá Fibonacciho čísla, druhý mocniny dvojky

```
1  Procedure is_first_player_winning(n)
2      if n is a Fibonacci number then
3          return True
4      for Fibonacci numbers i smaller than n do
5          if not is_second_player_winning(n-i) then
6              return True
7      return False
8  Procedure is_second_player_winning(n)
9      if n is a power of 2 then
10         return True
11     for i powers of 2 smaller than n do
12         if not is_first_player_winning(n-i) then
13             return True
14     return False
```

Otázka

Jak souvisí předchozí algoritmus s přednáškou?

- Na přednášce byl obecných min-max algoritmus
- Na předchozím slajdu žádný min-max není
- Dokonce funkce pro oba hráče jsou stejné (až na vyhodnocování odlišných pravidel)
- Máme min-max algoritmus nebo něco úplně jiného?
- Kde používáme alpha-beta prořezávání?

Otázka

Jak souvisí předchozí algoritmus s přednáškou?

- Na přednášce byl obecných min-max algoritmus
- Na předchozím slajdu žádný min-max není
- Dokonce funkce pro oba hráče jsou stejné (až na vyhodnocování odlišných pravidel)
- Máme min-max algoritmus nebo něco úplně jiného?
- Kde používáme alpha-beta prořezávání?

Obecné převedení minimalizace na maximalizaci

Pro každou funkci $f : M \to \mathbb{R}$, kde M je libovolná množina platí

$$\min_{x \in M} f(x) = -\max_{x \in M} -f(x)$$

$$\max_{x \in M} f(x) = -\min_{x \in M} -f(x).$$

Otázka

Jak souvisí předchozí algoritmus s přednáškou?

- Na přednášce byl obecných min-max algoritmus
- Na předchozím slajdu žádný min-max není
- Dokonce funkce pro oba hráče jsou stejné (až na vyhodnocování odlišných pravidel)
- Máme min-max algoritmus nebo něco úplně jiného?
- Kde používáme alpha-beta prořezávání?

Obecné převedení minimalizace na maximalizaci

Pro každou funkci $f : M \to \mathbb{R}$, kde M je libovolná množina platí

$$\min_{x \in M} f(x) = -\max_{x \in M} -f(x)$$

$$\max_{x \in M} f(x) = -\min_{x \in M} -f(x).$$

Tedy $\max(\min(\max(\min(\max \cdots u(x))) = \max(-\max(-\max(-\max(-\max \cdots u(x)))$

- $u(x)$ značí užitek koncových stavů
- Liché členy maximalizují užitek prvního hráče
- Sudé členy maximalizují užitek druhého hráče

Metody urychlení výpočtu

Metody urychlení výpočtu

Nejprve prozkoumat nadějnější stavy

- Použít heuristiku ohodnocující stavy
- Nejprve prozkoumáme stavy, které vedou na méně možných tahů druhého hráče

Nejprve prozkoumat nadějnější stavy

- Použít heuristiku ohodnocující stavy
- Nejprve prozkoumáme stavy, které vedou na méně možných tahů druhého hráče

Jak jednoduše urychlit tento algoritmus?

Oba hráči můžou odebírat Fibonacciho čísla

```
1 Procedure is_winning(n)
2     for i = 1 . . . n do
3         if Fibonacci numbers i smaller than n and not is_winning(n-i) then
4             return True
5     return False
```

Metody urychlení výpočtu

Předpočítat si ohodnocení některých stavů

- Předpočítat si stavy blízko koncových pomocí zpětné prohledávání
- Analýzou hry dokázat ohodnocení některých stavů

Metody urychlení výpočtu

Předpočítat si ohodnocení některých stavů

- Předpočítat si stavy blízko koncových pomocí zpětné prohledávání
- Analýzou hry dokázat ohodnocení některých stavů

Příklad

- Ve hře jsou 3 hromádky sirek
- Hráč může z jedné hromádky odebrat libovolný nezáporný počet sirek
- Vyhrává hráč, který odebral poslední sirku

Pro které stavy dokážete rovnou říct výsledek?

Kešování prozkoumaných stavů

- Opakovaným výpočtům zabráníme kešováním výsledků vyhodnocených stavů
- Pamatováním všech výsledků všech stavů rychle dojde paměť

Kešování prozkoumaných stavů

- Opakovaným výpočtům zabráníme kešováním výsledků vyhodnocených stavů
- Pamatováním všech výsledků všech stavů rychle dojde paměť

Postup

- Zvolíme k v závislosti na velikosti paměti
- Ukládáme výsledku vyhodnocení stavů do keše (např. hešovací tabulka)
- Každý stav v cache má čítač, který o 1 zvyšujeme při každém přístupu ke stavu
- Při vložení nového stavu nastavíme čítač na jedna
- Po každých k přístupech do keše
  - Smažeme všechny stavy s hodnotou čítače 1
  - Všem ostatním stavům v keši snížíme hodnotu čítače na polovinu
- Součet stavů v keši je nejvýše $2k$
- Amortizovaná složitost režie keše je $O(1)$ na přístup do keše

Terminologie

- **Dominantní strategie hráče**: Nejlepší strategie pro hráče, ať ostatní udělají cokoliv
- **Nash equilibrium**: Kombinace strategií hráčů, kde se žádnému hráči samostatně nevyplatí svou strategii změnit
- **Pareto optimální**: Kombinace strategií hráčů taková, že neexistuje jiná kombinace strategií hráčů, ve které si nikdo nepohorší a alespoň jeden si polepší

Terminologie

- **Dominantní strategie hráče**: Nejlepší strategie pro hráče, ať ostatní udělají cokoliv
- **Nash equilibrium**: Kombinace strategií hráčů, kde se žádnému hráči samostatně nevyplatí svou strategii změnit
- **Pareto optimální**: Kombinace strategií hráčů taková, že neexistuje jiná kombinace strategií hráčů, ve které si nikdo nepohorší a alespoň jeden si polepší

Vězňovo dilema

	2. zloděj neudá komplice	2. zloděj udá komplice
1. zloděj neudá komplice	(1,1)	(10,0)
1. zloděj udá komplice	(0,10)	(8,8)

Značení (a, b): a je počet let ve vězení pro prvního zloděje a b pro druhého zloděje
Existují dominantní, Nash equilibrium a Pareto optimální strategie?

Battle of the sexes

Kluk a holka by chtěli strávit společné odpoledne. Holka by raději šla do kina, kluk na fotbat, ale oba by raději byli spolu.

	holka jde do kina	holka jde na fotbal
kluk jde do kina	(5,6)	(1,1)
kluk jde na fotbal	(2,2)	(6,5)

První číslo udává užitek pro kluka, druhé pro holku.
Existují dominantní, Nash equilibrium a Pareto optimální strategie?

Matching pennies

- Proti sobě jedou dvě auta.
- Oba řidiči se můžou rozhodnout, zda pojedou v pravém či levém jízdním pruhu.
- Srážka oběma řidičům způsobí mnohem větší škody než když se vyhnou.
- Existují dominantní, Nash equilibrium a Pareto optimální strategie?

Matching pennies

- Proti sobě jedou dvě auta.
- Oba řidiči se můžou rozhodnout, zda pojedou v pravém či levém jízdním pruhu.
- Srážka oběma řidičům způsobí mnohem větší škody než když se vyhnou.
- Existují dominantní, Nash equilibrium a Pareto optimální strategie?
- Najděte další Nash equilibrium ve smíšených strategiích.
- Ve smíšené stragegii si každý hráč zvolí pravděpodobností rozložení možných strategií a chce maximalizovat očekávanou hodnotu užitku.

7. domácí úkol: Teorie her

Zadání (zkráceno)

Pravidla hry

- Máme dva hráče, kteří po jednom odebírají kameny očíslované $1, 2, \ldots, n$.
- Hráč může odebrat libovolný kámen, který je násobek nebo dělitel předchozího odebraného čísla.
- Pokud hráč nemůže odebrat kámen, tak prohrává.

Napište funkci, která dostane seznam zbývajících kamenů a poslední odebraný kámen a rozhodne, zda je daná situace vyhrávající a pokud ano, tak vrátí kámen, který má hráč odebrat v dalším tahu.