IZU Guide

Jedná se neoficiálního průvodce předmětem Základy umělé inteligence.

Toto není jediný materiál pro studium ke zkoušce, doporučujeme jej kombinovat s přednáškami.

Dokument může obsahovat pravopisné chyby, za které se předem omlouváme a budeme rádi, za případné upozornění.

Obsah

1 Prohledávací algorimty

1.1 BFS – Breadth First Search

- Je úplný a optimální
- Časová i paměťová náročnost je exponenciální O(b^{d+1}), proto se nepoužívá
- **Popis:**
  1. Sestroj frontu `OPEN` (bude obsahovat všechny uzly určené k expanzi) a umísti do ní počáteční uzel
  2. Je-li fronta `OPEN` prázdná, pak úloha nemá řešení a ukonči proto prohledávání jako neúspěšné
  3. Vyber z čela fronty `OPEN` první uzel
  4. Je-li vybraný uzel uzlem cílovým, ukonči prohledávání jako úspěšné a vrať cestu od kořenového uzlu k uzlu cílovému (vrací se posloupnost stavů, nebo operátorů)
  5. Vybraný uzel expanduj, všechny jeho bezprostřední následníky umísti do fronty `OPEN` a vrať se na bod 2
- **Možné modifikace:**
  1. Přidání fronty `CLOSED`
  2. Výběr těch, kteří ještě nejsou v `OPEN`
  3. Výběr těch, kteří nejsou předky
  4. Testování a generování

1.2 BS – Bidirectional Search

- Je úplný a optimální
- Náročnost je O(2*$b^{(d/2)}$)
- **Popis:**
  1. Sestroj fronty `OPEN1` a `OPEN2` (budou obsahovat všechny uzly určené k expanzi) a seznamy `CLOSED1` a `CLOSED2` (budou obsahovat všechny expandované uzly). Do fronty `OPEN1` umísti počáteční uzel a do fronty `OPEN2` cílový uzel
  2. Je-li fronta `OPEN1` prázdná, pak úloha nemá řešení, a proto ukonči prohledávání jako neúspěšně
  3. Vyber z čela fronty `OPEN1` první uzel a umísti tento uzel do seznamu `CLOSED1`
  4. Vybraný uzel expanduj. Pokud některý bezprostřední následník je prvkem fronty `OPEN2` (*"můstek"*), ukonči prohledávání jako úspěšné a vyznač cestu od počátečního uzlu k uzlu cílovému (v `CLOSED1` jsou uzly od počátečního stavu k můstku, v `CLOSED2` jsou uzly od můstku k cílovému stavu), jinak ulož tohoto následníka do fronty `OPEN1`
  5. Vyber z čela fronty `OPEN2` první uzel a umísti tento uzel do seznamu `CLOSED2`
  6. Vybraný uzel expanduj. Pokud některý bezprostřední následník je prvkem fronty `OPEN1` (*"můstek"*), ukonči prohledávání jako úspěšné a vyznač cestu od počátečního uzlu k uzlu cílovému (v `CLOSED1` jsou uzly od počátečního stavu k můstku, v `CLOSED2` jsou uzly od můstku k cílovému stavu), jinak ulož tohoto následníka do fronty `OPEN2` a jdi na bod 2

1.3 DFS – Depth First Search

- Není úplný ani optimální
- Časová složitost je exponenciální O(b^m), paměťová složitost je lineární O(b * m)
- **Popis:**
  1. Sestroj zásobník `OPEN` (bude obsahovat všechny uzly určené k expanzi) a umísti do ní počáteční uzel
  2. Je-li zásobník `OPEN` prázdný, pak úloha nemá řešení a ukonči proto prohledávání jako neúspěšné
  3. Vyber z vrcholu zásobníku `OPEN` první uzel
  4. Je-li vybraný uzel uzlem cílovým, ukonči prohledávání jako úspěšné a vrať cestu od kořenového uzlu k uzlu cílovému (vrací se posloupnost stavů, nebo operátorů)
  5. Vybraný uzel expanduj, všechny jeho bezprostřední následníky umísti do zásobníku `OPEN` a vrať se na bod 2
- **Modifikace:** Neexpanduj už expandované(ty v `OPEN`) a také předky. Tato modifikace je úplná ale není optimální

1.4 DLS – Depth Limited Search

- Není úplný ani optimální
- **Popis:**
  1. Sestroj zásobník `OPEN` (bude obsahovat všechny uzly určené k expanzi) a umísti do ní počáteční uzel s označením hloubky(0)
  2. Je-li zásobník `OPEN` prázdný, pak úloha nemá řešení a ukonči proto prohledávání jako neúspěšné
  3. Vyber z vrcholu zásobníku `OPEN` první uzel
  4. Je-li vybraný uzel uzlem cílovým, ukonči prohledávání jako úspěšné a vrať cestu od kořenového uzlu k uzlu cílovému (vrací se posloupnost stavů, nebo operátorů)
  5. Pokud je hloubka vybraného uzlu menší než zadaná maximální hloubka, tak tento uzel expanduj, všem jeho bezprostředním následníkům přiřaď hloubku o jedničku větší než je hloubka expandovaného uzlu a umísti je do zásobníku `OPEN` a vrať se na bod 2

1.5 UCS – Uniform Cost Search

- Je úplný a optimální
- **Bez closed:**
  1. Sestroj seznam `OPEN` a umísti do něj počáteční uzel včetně jeho (nulového) ohodnocení
  2. Je-li seznam `OPEN` prázdný, pak úloha nemá řešení a ukonči proto prohledávání jako neúspěšné
  3. Vyber ze seznamu `OPEN` uzel s nejnižším ohodnocením
  4. Je-li vybraný uzel cílový, ukonči prohledávání jako úspěšné a vrať cestu od kořenového uzlu k uzlu cílovému
  5. Vybraný uzel expanduj, všechny jeho bezprostřední následníky včetně jejich ohodnocení umísti do seznamu `OPEN` a vrať se na bod 2
- **S closed:**
  1. Sestrojte dva prázdné seznamy `OPEN`(bude obsahovat uzly určené k expanzi) a `CLOSED`(bude obsahovat seznam expandovaných uzlů). Do seznamu `OPEN` umísti počáteční uzel včetně jeho ohodnocení
  2. Je-li seznam `OPEN` prázdný, pak úloha nemá řešení a ukonči proto prohledávání jako neúspěšné
  3. Vyber ze seznamu `OPEN` uzel s nejnižším ohodnocením
  4. Je-li uzel cílový, ukonči prohledáváni jako úspešné a vraťte cestu od koreňového uzlu k uzlu cílovému(vrací se posloupnost stavů nebo operátorů)
  5. Vybraný uzel expandujte a jeho bezprostředný následníky, kteří nejsou ve seznamu `CLOSED` umístěte do seznamu `OPEN`(včetne jejich ohodnocení). Expandovaný uzel umístěte do seznamu `CLOSED`. Z uzlů, které v seznamu `OPEN` vyskytují vícekrát ponechte, uzel s nejlepším ohodnocením, ostatní ze seznamu OPEN vyškrtněte a vraťte se na bod 2

1.6 A*

- Je úplný a optimální
- Zná cilový stav a využívá téhle informace k tomu, aby lépe odhadl cestu k němu
- Jako heuristiku používá spodní odhad k realné ceně ⇒ odhad, který je realné ceně cesty nejblíž
- **Popis:**
  1. Sestroj seznam `OPEN` (bude obsahovat všechny uzly určené k expanzi) a umísti do něj počáteční uzel včetně jeho ohodnocení
  2. Je-li seznam `OPEN` prázdný, pak úloha nemá řešení a ukonči proto prohledávání jako neúspěšné
  3. Vyber ze seznamu `OPEN` uzel s nejlepším(nejnižším) ohodnocením
  4. Je-li vybraný uzel uzlem cílovým, ukonči prohledávání jako úspěšné a vrať cestu od kořenového uzlu k uzlu cílovému (vrací se posloupnost stavů, nebo operátorů)
  5. Vybraný uzel expanduj, všechny jeho bezprostřední následníky, kteří nejsou jeho předky, umísti do seznamu `OPEN`, a to včetně jejich ohodnocení. Z uzlů, které se v seznamu `OPEN` vyskytují vícekrát, ponech pouze uzel s nejlepším ohodnocením, ostatní ze seznamu `OPEN` vyškrtni, a vrať se na bod 2

1.7 GS – Greedy Search

- Je úplný ale není optimální
- Ohodnocuje uzly pouze heuristickou funkcí ⇒ odhadovanou cenou z daného uzlu do uzlu cílového ⇒ k expanzi vybírá uzel, který má toto hodnocení nejnižší
- Dobrá heuristika může časovou náročnost výrazně redukovat
- Pokud se do seznamu `OPEN` ukládají všichni bezprostřední následníci expandovaného uzlu ⇒ i jeho předci(v bodu 5 se pak vypustí kontrola *"kteří nejsou jeho předky"*) pak **GS není uplný!**
- **Popis:**
  1. Sestroj seznam `OPEN` (bude obsahovat všechny uzly určené k expanzi) a umísti do něj počáteční uzel včetně jeho ohodnocení
  2. Je-li seznam `OPEN` prázdný, pak úloha nemá řešení a ukonči proto prohledávání jako neúspěšné
  3. Vyber ze seznamu `OPEN` uzel s nejlepším(nejnižším) ohodnocením
  4. Je-li vybraný uzel uzlem cílovým, ukonči prohledávání jako úspěšné a vrať cestu od kořenového uzlu k uzlu cílovému (vrací se posloupnost stavů, nebo operátorů)
  5. Vybraný uzel expanduj, všechny jeho bezprostřední následníky, kteří nejsou jeho předky, umísti do seznamu `OPEN`, a to včetně jejich ohodnocení. Z uzlů, které se v seznamu `OPEN` vyskytují vícekrát, ponech pouze uzel s nejlepším ohodnocením, ostatní ze seznamu `OPEN` vyškrtni, a vrať se na bod 2

1.8 Backtracking

- Není úplný ani optimální
- Má extremně nízkou paměťovou náročnost
- **Popis:**
  1. Sestroj zásobník `OPEN` (bude obsahovat uzly určené k expanzi) a umísti do něj počáteční uzel
  2. Je-li zásobník `OPEN` prázdný, pak úloha nemá řešení a ukonči proto prohledávání jako neúspěšné
  3. Jde-li na uzel na vršku zásobníku aplikovat první/další operátor, tak tento operátor aplikuj a pokračuj bodem 4, v opačném případě odstraň testovaný uzel z vrcholu zásobníku a vrať se na bod 2
  4. Je-li nový vygenerovaný uzel, tj. uzel vzniklý aplikací operátoru na uzel na vršku zásobníku, uzlem cílovým, ukonči prohledávání jako úspěšné a vrať cestu od kořenového uzlu k uzlu cílovému (vrací se posloupnost stavů, nebo operátorů). Jinak ulož nový uzel na vršek zásobníku a vrať se na bod 2

1.9 BestFS – Best First Search(prohledávání od nejlepšího)

- **Popis:**
  1. Sestrojte seznam `OPEN` (bude obsahovat všechny uzly určené k expanzi) a umístěte do něj počáteční uzel včetně jeho ohodnocení
  2. Je-li seznam `OPEN` prázdný, pak úloha nemá řešení, a ukončete proto prohledávání jako neúspěšné
  3. Vyberte ze seznamu `OPEN` uzel s nejlepším(nejnižším) ohodnocením
  4. Je-li vybraný uzel uzlem cílovým, ukončete prohledávání jako úspěšné a vraťte cestu od kořenového uzlu k uzlu cílovému
  5. Vybraný uzel expandujte. Všechny jeho bezprostřední následníky, kteří nejsou jeho předky, umístěte do seznamu `OPEN` včetně jejich ohodnocení. Z uzlů, které se v seznamu `OPEN` vyskytují vícekrát, ponechte pouze uzel s nejlepším ohodnocením, ostatní ze seznamu `OPEN` vyškrtněte a vraťte se na bod 2

1.10 IDS – Postupné zanořování do hloubky

- Je úplný i optimální
- Nelze-li stanovit hloubku řešení a nemáme-li k dispozici dostatek paměti pro použití metody BFS užijeme IDS
- Princip této metody spočívá v opakovaném použití metody DLS s postupným zvyšováním hloubky prohledávání
- **Popis:**

1. Nastav hloubku prohledávání na hodnotu 1
2. Použij metodu `DLS`. Skončí-li tato metoda úspěchem (nalezením cesty), skonči také úspěchem a vrať nalezenou cestu
3. Dosáhla-li hloubka prohledávání maximální hodnotu, skonči neúspěchem. Jinak inkrementuj hloubku prohledávání a vrať se na bod 2

1.11 Hill-climbing – Lezení do kopce

- Není úplný ani optimální
- Používá k ohodnocení uzlu, podobně jako metoda Greedy search, pouze heuristiku h(n) ⇒ funkce, která ohodnocuje *"kvalitu řešení"* spíše než vzdálenost od cílového řešení ⇒ pak čím lepší řešení ohodnocovaný uzel představuje, tím vyšší je mu přiřazena hodnota a algoritmus vybírá k expanzi uzel s nejvyšším ohodnocením
- **Popis:**
  1. Vytvoř uzel *Current* totožný s počátečním uzlem (včetně jeho ohodnocení)
  2. Expanduj uzel *Current*, ohodnoť jeho bezprostřední následníky a vyber z nich nejlépe ohodnoceného (nazvěme jej *Next*)
  3. Je-li ohodnocení uzlu *Current* lepší než ohodnocení uzlu *Next*, ukonči řešení a vrať jako výsledek uzel *Current*
  4. Ulož uzel *Next* do uzlu *Current* a vrať se na bod 2

1.12 Simulovaného žíhání

- Pro reálné časové možnosti není zaručena ani úplnost ani optimálnost.
- Má extrémně malou prostorovou složitost
- Stochastická metoda
- Cílem je překonání lokálních extrémů vedoucích k častým neúspěchům metody Hill- climbing
- **Popis:**
  1. Vytvoř tabulku s předpisem pro klesání teploty v závislosti na kroku výpočtu.
  2. Vytvoř uzel *Current* totožný s počátečním uzlem (včetně jeho ohodnocení). Nastav krok výpočtu na nulu ($k = 0$)
  3. Z tabulky zjisti aktuální teplotu $T (T = f(k))$. Je-li tato teplota nulová ($T = 0$) ukonči řešení a vrať jako výsledek uzel *Current*
  4. Expanduj uzel *Current* a z jeho bezprostředních následníků vyber náhodně jednoho z nich (nazvěme jej *Next*)
  5. Vypočítej rozdíl ohodnocení uzlů *Current* a *Next* ⇒ $\Delta E = \text{value}(Next) - \text{value}(Current)$.
  6. Jestliže $\Delta E > 0$, tak ulož uzel *Next* do uzlu *Current*, jinak ulož uzel *Next* do uzlu *Current* s pravděpodobností $e^{\Delta E/T}$
  7. Inkrementuj krok výpočtu k a vrať se na bod 3

1.13 AO – AND/OR

- Je základním neinformovaným algoritmem
- **Popis:**
  1. Sestroj dva prázdné seznamy `OPEN` a `CLOSED`. Do seznamu `OPEN` ulož počáteční uzel (problém)
  2. Vyjmi uzel zleva ze seznamu `OPEN` a označ jej jako uzel X
     a) Je-li uzel (problém) X řešitelný, přenes informaci o jeho řešitelnosti jeho předchůdcům. Je-li řešitelný počáteční problém, ukonči řešení jako úspěšné ⇒ vytvoř a vrať relevantní část AND/OR grafu
     b) Není-li uzel (problém) X řešitelný a nelze-li jej rozložit na podproblémy, přenes informaci o jeho ne řešitelnosti jeho předchůdcům. Není-li řešitelný počáteční problém, ukonči řešení jako neúspěšné
     c) Expanduj X (rozlož X na podproblémy) a všechny jeho následníky ulož do `OPEN`
  3. Ulož X do `CLOSED`
  4. Je-li seznam `OPEN` prázdný, ukonči řešení jako neúspěšné, jinak se vrať na bod 2

2 Hraní her

2.1 Jednoduché hry

- Za jednoduché hry budeme považovat hry, u kterých lze v reálném čase prohledat celý jejich AND/OR graf
- K řešení takových her lze použít algoritmus AO s tím, že v případě řešitelnosti není nutné vracet celou část AND/OR grafu, ale pouze tah hráče A, který vede k jeho výhře

2.2 Hry s náhodou

- Existuje řada podobných her, které opět hrají dva protihráči, kteří se po jednotlivých tazích hry pravidelně střídají, mají úplnou informaci o stavu hry, hrají čestně a oba si přejí zvítězit
- Na rozdíl od jinych her však při hře používají kostku, resp. kostky, a do hry tak vstupuje neurčitost ⇒ náhoda

2.3 ExpectiMiniMax

1. Nazvěme předaný vstupní uzel uzlem X
2. Je-li uzel X listem (konečným stavem hry, nebo uzlem v maximální hloubce) vrať ohodnocení tohoto uzlu
3. Je-li na tahu hráč A, tak postupně pro všechny jeho možné tahy (bezprostřední následníky uzlu X a hráče B) volej proceduru `ExpectMiniMax` a vrať maximální hodnotu z hodnot `expectimax`. Je-li X kořenovým uzlem vrať i tah, který vede k nejlépe ohodnocenému bezprostřednímu následníkovi
4. Je-li na tahu hráč B, tak postupně pro všechny jeho možné tahy(bezprostřední následníky uzlu X a hráče A) volej proceduru `ExpectMiniMax` a vrať minimální hodnotu z hodnot `expectimin`

2.4 MiniMax

- Volá vždy, když je na tahu hráč A
- Vstupními parametry procedury jsou aktuální stav hry X, maximální hloubka prohledávání a informaci o hráči, který je právěna tahu (A nebo B)
- Procedura vrací přepočítané nejvyšší ohodnocení aktuálního stavu a tah, který k tomuto ohodnocení vede
- **Popis:**
  1. Je-li uzel X listem (konečným stavem hry, nebo uzlem v maximální hloubce) vraťte ohodnocení tohoto uzlu
  2. Je-li na tahu hráč A, tak postupně pro všechny jeho možné tahy (bezprostřední následníky uzlu X, tj. stavy před tahem hráče B) volejte proceduru `MiniMax` a vraťte maximální z navrácených hodnot a tah, který k nejlépe ohodnocenému bezprostřednímu následníkovi vede (použije se pak pouze u kořenového uzlu, tj. u aktuálního stavu hry, před skutečným tahem hráče A)
  3. Je-li na tahu hráč B, tak postupně pro všechny jeho možné tahy (bezprostřední následníky uzlu X, tj. stavy před tahem hráče A) volejte proceduru `MiniMax` a vraťte minimální z navrácených hodnot

2.5 AlfaBeta

- Volá podobně jako procedura MiniMax vždy, když je na tahu hráč A
- Parametry procedury AlfaBeta jsou stejné, jako parametry procedury MiniMax, navíc používá dva parametry, α a β : 1
- **Popis:**
  1. Je-li X dosud neohodnoceným počátečním/kořenovým uzlem, nastavte parametry α a β na hodnoty $\alpha = -\infty$, $\beta = \infty$ (v praxi nastavte hodnoty těchto proměnných na minimální a maximální možnou hodnotu)
  2. Je-li uzel X listem (konečným stavem hry, nebo uzlem v maximální hloubce) ukončete proceduru a vraťte ohodnocení tohoto uzlu
  3. Je-li na tahu hráč B, tak přejděte na bod 4, jinak pokračujte (na tahu je hráč A)
     - Dokud platí nerovnost $\alpha < \beta$, tak postupně pro první/další tah (tj. pro prvního/dalšího bezprostředního následníka uzlu X) volejte proceduru AlfaBeta s aktuálními hodnotami parametrů α a β. Po každém vyšetřeném tahu nastavte hodnotu parametru α na maximum z aktuální a navrácené hodnoty

- Je-li $\alpha \geq \beta$, nebo nemá-li uzel X žádného dalšího bezprostředního následníka, ukončete proceduru, vraťte aktuální hodnotu parametru α a tah, který vede k nejlépe ohodnocenému bezprostřednímu následníkovi (opět se použije pouze při návratu ke kořenovému uzlu ⇒ při návratu k aktuálnímu stavu hry před tahem hráče A pokud má stejné ohodnocení více možných tahů, musí se použít vždy tah, který byl ohodnocen jako první z těchto tahů)

4. Na tahu je hráč B
   - Pokud platí nerovnost $\alpha < \beta$, tak postupně pro první/další tah (tj. pro prvního/dalšího bezprostředního následníka uzlu X) volejte proceduru AlfaBeta s aktuálními hodnotami parametrů α a β. Po každém vyšetřeném tahu nastavte hodnotu parametru β na minimum z aktuální a navrácené hodnoty
   - Je-li $\alpha \geq \beta$, nebo nemá-li uzel X žádného dalšího bezprostředního následníka, ukončete proceduru a vraťte aktuální hodnotu parametru β.

3 Učící algoritmy

3.1 DTW – Dynamic Time Warping

- Metoda potřebuje množinu referenčních slov a pokud možno, každé slovo několikrát v různé podobě
- Slova rozložíme na mikrosegmenty a určíme jejich příznaky
- Posloupnost příznaků postupně porovnáváme s referenčními posloupnostmi a za rozpoznané slovo vezmeme to, které nejvíce odpovídá
- Neporovnáváme ale jen příznaky mikrosegmentů, které mají stejný čas, ale také mikrosegmentů okolních, čímž dosahujeme nezávislosti rozpoznávání na rychlosti mluvení
- To, se kterými mikrosegmenty porovnáváme určují různé strategie

3.2 Rozpoznání černobílého obrazu

- **Dekompozice obrazů (barvení běhů):**
  - Barvení běhů začíná bodem uvnitř objektu ⇒ jdeme v určitém směru
  - Pokud narazí na objekt, označí pixel jednou barvou
  - nespojité pixely ⇒ každý jinou barvou dokud se postupným průchodem nespojí ⇒ přebarvení na stejnou barvu ⇒ rozpoznaný jeden objekt
- **Rozpoznání:**
  - Rozpoznává se např. podle těchto příznaků ⇒ plocha (počet px), plocha děr, počty děr, délka hranice atd..
  - Důležité je, že po této fázi porovnávám rozpoznaný objekt s referencčními objekty v DB
  - Pomocí R/S/T různě modifikuji tvar ⇒ při určité míře podobnosti je prohlásím za shodné
- **Zpracování:**
  - Jednobodové transformace
  - Dvoubodové transformace ⇒ součet obrazů, rozdíl obrazů, násobení obrazu maskou, prostorové transformace

3.3 ID3

- Entropie (míra neuspořádanosti) zprávy M s m možnými odpověďmi m_1,m_2,m_n, jejichž pravděpodobnosti výskytu jsou označeny jako$p(m_i)$
- Maximální je pro rovnoměrné rozložení pravděpodobností možných odpovědí
- Minimální(nulová) je pro jedinou, předem známou odpověď
- Vlastnost V, která má k hodnot, rozděluje množinu trénovacích příkladů c do k podmnožin, z nichž každá obsahuje c_i příkladů
- Učení s učitelem, tzn. algoritmus má odezvy na svoje kroky a pozná, jestli *"jde"* spravným nebo špatným směrem
- Je postaven na základě rozhodovacích stromů

- Nejvhodnější atribut vybírá podle informačního zisku, který se da počítat pomocí vzorečků pro entropii a pravděpodobnost
- Pokud se vybere atribut špatný, může dojít k postavení naprosto špatného stromu, který i triviální úlohu velmi zkomplikuje
- **Operace zobecňování:** Nahrazení konstanty proměnnou
- **Operace specializace:** Nahrazení proměnné konstantou
- **Prostor verzí:** množina všech popisů pojmů, která je konzistentní se všemi příklady z trénovací množiny příkladů
- Ukázka pravidel:
  1. Má-li žadatel příjem nižší než 15 tis. Kč, je risk úvěru vysoký
  2. Má-li žadatel příjem mezi 15 až 35 tis. Kč, pak:
     a) Je-li historie splácení jeho úvěrů špatná, je risk úvěru vysoký
     b) Je-li historie splácení jeho úvěrů dobrá, je risk úvěru přiměřený
     c) Je-li historie splácení jeho úvěrů neznámá, pak:
        i. Je-li jeho dluh vysoký, je risk úvěru vysoký.
        ii. Je-li jeho dluh nízký, je risk úvěru přiměřený.
  3. Má-li žadatel příjem vyšší než 35 tis. Kč, pak:
     a) Je-li historie splácení jeho úvěrů špatná, je risk úvěru přiměřený
     b) Je-li historie splácení jeho úvěrů dobrá, je risk úvěru nízký
     c) Je-li historie splácení jeho úvěrů neznámá, je risk úvěru nízký

3.4 ACO – Ant Colony Optimization

- Je inspirována postupem, který používají mravenci při hledání potravy (hledáním nejkratší cesty mezi mraveništěm a zdrojem potravy):
  1. Nejprve náhodně prohledávají blízké okolí mraveniště
  2. Během svého pohybu vypouští na zem chemickou látku, tzv. feromon
  3. Tuto látku cítí a pravděpodobnost výběru jejich dalších cest je dána aktuálními koncentracemi feromonů na začátcích možných dalších cest
  4. Když někdo narazí na zdroj potravy, vyhodnotí její kvalitu i kvantitu a vrací se s částí této potravy do mraveniště
  5. Během zpáteční cesty se množství vypouštěného feromonu tímto mravencem zvyšuje, a to úměrně s kvalitou a množstvím nalezené potravy
  6. Feromonové stopy tak směrují ostatní mravence ke zdrojům potravy, s časem však vyprchávají
- Hlavní rozdíly mezi chováními skutečných mravenců a umělých mravenců v modelech ACO jsou tyto:
  - Skuteční mravenci se pohybují v prostředí asynchronně, pohyb umělých mravenců/agentů je synchronizován.
  - Skuteční mravenci se při návratu do mraveniště řídí feromonovými stopami, umělí mravenci se v každém cyklu vrací do mraveniště po stejných cestách, kterými se v tomto cyklu pohybovali od mraveniště.
  - Skuteční mravenci vypouští feromon neustále, umělí mravenci značkují cestu *"feromonem"* pouze při návratu do mraveniště.
  - Chování skutečných mravenců je založeno na implicitním vyhodnocení cest. To spočívá v tom, že pohyb po kratších cestách trvá mravencům kratší dobu, proto je mohou opakovat častěji a tím se množství feromonu na těchto cestách zvyšuje. Umělí mravenci vyhodnocují cesty explicitně, a to při návratech do mraveniště podle kvality a délky cesty.
  - Skuteční mravenci jsou prakticky slepí, umělí mravenci *"zrak"* mají.
  - Umělí mravenci mají paměť.
  - Prostředí, ve kterém se umělí mravenci pohybují, je diskrétní.

3.5 GA – Genetické algoritmy

- **Jediněc(chromozom):**
  - Jsou tak obvykle reprezentovány jako řetězy znaků, ale obecně jsou možné i jiné jejich reprezentace, například grafy, matice atp.
  - Délky řetězů jsou pak závislé na konkrétních řešených úlohách
  - Kombinace jednotlivých znaků (genů) v chromozomu nese informaci, která se nazývá řešením dané optimalizační úlohy ⇒ a to přestože toto *"řešení"* může mít (a obvykle také má) ke skutečnému,tj. k optimálnímu řešení velmi daleko
- **Populace:**
  - Tvoří jí předem daný počet chromozomů ⇒ tento počet je závislý na řešené úloze, obvykle jde o stovky až tisíce jedinců
  - Jedinci počáteční/výchozí populace se generují náhodně, může se však samozřejmě využít libovolná dostupná heuristika, která umožní generování jedinců s počátečními hodnotami (řešeními) blízkými optimálnímu řešení
- **Hodnocení jedinců a kvalita populace:**
  - Spočívá ve vyhodnocení, jak se řešení představované tímto jedincem liší od správného/optimálního řešení dané úlohy (fitness of solution)
  - Hodnota udává kvalitu řešení chromozomu se nazývá fitness funkce
  - Hodnota fitness funkce se musí zvyšovat s kvalitou řešení ⇒ musí být tím vyšší, čím je řešení představované ohodnocovaným chromozomem bližší správnému řešení
  - Fitness funkce mají výrazný vliv jak na kvalitu řešení, tak i na délku výpočtu, a proto jejich návrhu je nutné věnovat velkou pozornost
  - Kvalita je dána ohodnocením populace ⇒ průměrnou hodnotou hodnot fitness funkcí všech jedinců populace(hodnotu průměrného jedince)
  - Pokud některý jedinec představuje správné/optimální řešení (nelze zjistit u všech úloh!), výpočet samozřejmě okamžitě končí
  - Jinak se pro ukončení výpočtu používá dvou přístupů:
    - kvalita populace delší dobu nezvyšuje
    - bylo dosaženo předem stanoveného maximálního počtu iterací (tj. vytváření nových populací)
  - Řešení v tomto případě pak představuje nejlépe hodnocený chromozom
- **Reprodukce:**
  - **Výběr rodičů:**
    - Provádí na základě ohodnocení všech jedinců populace fitness funkcí, a to buď podle proporcí nebo pořadí
    - Výběr elity k rodičovství se vybere nejlépe ohodnocený jedinc(či jedinci)
    - Turnajový výběr z náhodně vybraných jedinců vítězí (stává se rodičem) nejlépe ohodnocený jedinec
    - Počet se samozřejmě musí rovnat potřebnému počtu rodičů. V turnajové výběru pak dostávají šanci stát se rodiči i hůře ohodnocení jedinci, protože některé turnaje mohou být díky náhodnému výběru *"slaběji"* obsazené
  - **Křížení:**
    - Produkuje nové jedince z informací obsažených v genech rodičů(vzájemná výměna chromozomů)
    - Místa, od kterých změny začínají, se nazývají místy křížení a bývají vybrána náhodně
    - Nejjednodušší křížením je jednobodové křížení ⇒ stanoví se náhodně místo křížení
  - **Mutace:**
    - Spočívá ve změně hodnoty náhodně vybraného genu náhodně vybraného potomka
    - Pravděpodobnost mutace je obvykle velmi nízká
  - **Tvorba nové populace:**
    - **Generační model:** všichni jedinci původní populace jsou v nové populaci nahrazeni potomky
    - **Inkrementační model:** v nové populaci je nahrazen potomkem jediný jedinec původní populace
    - **Modely s překrytím generací:** v nové populaci je nahrazena potomky část jedinců původní populace ⇒ předchozí dva typy modelů jsou extrémními případy tohoto modelu

3.6 PSO – Optimalizace hejnem částic

- Původně navržen pro simulaci pohybů ptačího hejna, původní přístup modifikován ve vícerozměrné vyhledávání až se zjistilo, že algoritmus lze použít k řešení spojitých optimalizačních úloh(hejno při hledání potravy)
- S GA má několik společných rysů:
  1. Pracuje s populacemi jedinců, nazývanými častice, které představují *"řešení"* jako chromozomy v GA
  2. Hodnocení kvality části je prováděno pomocí hodnotících(fitness) funkcí.
  3. Počáteční rozložení částic v prohledávaném prostoru je náhodné
- Každá částice k je reprezentována polohou v n-rozměrném prostoru, vektorem rychlosti a paměti předchozích úspěchů při hledaní
- V průběhů výpočtu se jak poloha, tak rychlost každé částice k mění, a to v závislosti na dosud nejlepší poloze této částice a na dosud nejlepší poloze nalezené dosud všemi m částicemi hejna
- Koecifient setrvačnosti ω se původně bral jako hodnota 1 dnes se používá rozmezí $0.4 - 0.9$
- Koeficienty c_p(kognitivní) a c_g(sociální) jsou váhové a udávají míru důležitosti individuální paměti a sociálního vlivu
  a) Pokud je $(c_p + c_g \leq 4)$ hejno zvolna opisuje spirálu kolem nejlepšího řešení bez garance konvergence
  b) Pokud je $(c_p + c_g > 4)$ je konvergence rychlá a zaručená
  - A proto se oba koeficity volí stejné $\Rightarrow (c_p = c_g = 2.05)$
- **Algoritmus:**
  1. Zvol počet částic m (20 až 100)
  2. Zvol hodnoty koedicientů $\omega, c_p q c_g$
  3. Nastav náhodně počáteční hodnoty pozic a rychlosti částic (x,v)
  4. Nastav pozici x^k částice k jako její dosud nejlepší pozici p^k
  5. Urči nejlepší pozici hejna g, pro které je dáno pozicí nejlépe hodnocené částice $g = p^k$
  6. Nastav index k na první částici($k = 1$)
  7. Urči náhodná čísla r_p a r_g z intervalu $< 0, 1 >$
  8. **Vypočti novou rychlost částice a její novou pozici:**
     - $v^k = \omega v^k + c_p r_p (p^k - x^k) + c_g r_g (g - x^k)$
     - $x^k = x^k + v^k$
  9. Ohodnoť novou polohu částice x_k a pokud je hodnocení lepší než dosavadní nejlepší ohodnocení, pak uprav nejlepší pozici $p^k = x^k$
  10. Pokud je hodnocení částice lepší, než dosud nejlepší pozice hejna, uprav nejlepší pozici hejna $g = p^k$
  11. Inkrementuj index k, a pokud platí $k \leq m$, tak se vrať na bod 7
  12. Pokud se řešení zlepšuje a pokud nebyl překročen zvolený maximální čas výpočtu, tak se vrať na bod 6, jinak řešení ukonči $\Rightarrow$ vrať g což je nejlepší pozice hejna

3.7 STRIPS

- Program/jazyk určený k řešení úloh
- Odstraňuje problémy monotónní logiky
- Máme operátory a každý z nich má trojici seznamů:
  - **Conditions:** podmínky, kdy smí být operátor použit
  - **Delete:** které predikáty se použitím mají odstranit z databáze
  - **Add:** seznam predikátů, které se použitím musí do databáze přidat
- **Zásobník cílů:** algoritmus končí s prázdným zásobníkem cílů
- **Databáze:** seznam aktuálně platných predikatů

3.8 Učení pojmům (Version-space search)

- Hledá se popis pojmů nebo hypotézy, který má všechny pozitivní příklady a žádný negativní
- Využívá se zobecňování nebo specializace
- Metody prohledávání prostoru verzí:
  - **Specific to general:** od specifického k obecnému
  - **General to specific:** od nejobecnějšího ke specifickému
  - **Candidate eliminations:** spojení obou předchozích principů do jednoho, vybere se jak specifický tak obecný příklad a postupně se zdokonalují, dokud nevýjde stejná věc

3.9 Zpracování řeči

- **Akustická analýza:**
  1. izolovaně pronesená slova
  2. převod spojité řeči na text
- **Lingvistická analýza:**
  1. **Lexikální analýza:** převod textu na jednotlivá slova
  2. **Morfologická analýza:** přiřazení slovního druhu (podstatné jméno/sloveso/..) a gramatické rysy (rod, osoba, číslo, čas,pád) a slovní základ
  3. **Syntaktická analýza:**
  4. **Sémantická analýza:** význam slov, gramatiky/ATN se sémantickými přechody
  5. **Pragmatická analýza:** nejednoznačné věty, ukazovací zájmena, osobní zájmena, příslovečná určení místa a času

3.10 Posilované učení

- Funguje na principu odměn a penalizací
- čím déle tedy algoritmus běží a dostává odměny a penalizace, tím lepe zvládá najít optimalní cestu, protože při každém dalším průchodu je ta správná cesta stále výhodnější a výhodnější
- **Metody:**
  - **Pasivní ADP Learning:**
    - Ohodnocuje stavy podle toho, aby správná cesta vedla po cestě, jejíž ohodnocení se zvyšuje (*"Přihořívá přihořívá hoří"*)
    - Provede řadu náhodných procházek ze startovního místa a zaznamená si kudy šel. Výběr cesty je v tomto kroku náhodný
    - Následně se ze získaných dat zjistí, po kterých polích je dobrá jít a ta se podle toho ohodnotí (například odhadem maximální pravděpodobnosti ML)
    - Při ohodnocování stavu se používá také strategie označené jako Pi. Vysvětlení toho, jak to optimálně zvolit přesahuje rámec IZU
  - **Pasivní TD Learning:**
    - Skoro to samé co ADP, ale místo sbírání dat po několika procházkách, přepočítává data po každé procházce
    - Ohodnocení každého stavu se propočítá vždy, když systém daného stavu dosáhne
  - **Aktivní Q Learning:**
    - Podobná jako TD learning, ale místo ohodnocení přechodu vyhodnotí akci, která se má stát
    - Ačkoliv se to můžou tyto algoritmy zdát podobné, Q learning je aktivní, takže narozdíl od předchozích dvou, kde se pro vyhodnocení využívá předem daná strategie, tady se system sám rozhoduje, kam má jít v každém stavu

3.11 Waltzova metoda

- Metoda popisu scény s mnohostěny, rozdělení uzlů a hran do kategorií
- Způsob, jakým lze získat představu o 3D prostoru, který náš 2D obrazek vyobrazuje
- Žádná stěna nesmí být v singularní poloze ⇒ rovnoběžně se směrem pohledu kamery
- Jednodušší verze metody (scény s trojstrannými mnohostěny):
  - **Hrany:**
    - \+ **konvexní:** tvoří ji dvě stěny svírající z vnitřku tělesa úhel $< 180°$
    - − **konkávní:** tvoří ji dvě stěny svírající z vnitřku tělesa úhel $> 180°$
    - → **obrysová:** tvoří ji dvě stěny, jediná viditelná je po pravé straně šipky
  - **Uzly:**
    - **V:** dvě hrany
    - **W:** tři hrany, jeden z úhlů $> 180°$
    - **Y:** tři hrany, všechny úhly $> 90°$
    - **T:** tři hrany, jeden z úhlů $= 180°$

4 K-means

- **Body:** (1,1);(1,3);(1,4);(3,1);(4,1);(4,4)
- **Středy:** [1,3](třída 1);[1,4](třída 2);[4,1](třída 3)

4.1 1. aproximace

`[1,3]` :(1,1),(1,3) ⇒ [1,2](nový střed)

`[1,4]` :(1,4) ⇒ [1,4](nový střed)

`[4,1]` :(3,1),(4,1),(4,4) ⇒ [3.66,2](nový střed)

- Přiřadíme body k nebližším středům (ke středům s nějmenší vzdálenosti od bodu)
- Určíme nové středy tak, že zprůměrujeme body ležící ve středech. Tím získáme nový střed.

4.2 2. aproximace

`[1,2]` :(1,1),(1,3) ⇒ [1,2](nový střed)

`[1,4]` :(1,4) ⇒ [1,4](nový střed)

`[3.66,2]` :(3,1),(4,1),(4,4) ⇒ [3.66,2](nový střed)

- Jelikož se středy shodují s předochozími ukončíme výpočet a vrátíme nové středy
- Pokud by se ovšem neshodovali aproximujeme do doby než se budou shodovat či do určité aproximace(musí být zadána)

5 Logika

5.1 Přehled

- **AND:** $A \wedge B$
- **OR:** $A \vee B$
- **Negace:** $\neg A$ nebo $!A$
- **Implikace:** $A \Rightarrow B$
- **Dvojitá implikace:** $A \Rightarrow (B \wedge C)$
- **Ekvivalence:** $A \Leftrightarrow B$
- **Pro všechny platí, že:** $\forall(y)$

- **Existuje:** $\exists(y)$
- **Konstanta:** Většinou začátek abecedy[a,b,c…]
- **Proměnná:** Většinou konec abecedy[…,x,y,z]

5.2 Zjednodušení

- $A[\wedge,\vee]\neg A = T(1)$
- $\neg(A \vee \neg A) = F(0)$
- $A \wedge T = A$
- $A \wedge F = F$
- $A \wedge A = A$
- $A \wedge (A \vee B) = A$
- $A \vee T = T$
- $A \vee F = A$
- $A \vee A = A$
- $A \vee (A \wedge B) = A$
- $\forall y P(y) = P(y)$
- $\forall x \exists y P(x,y) = P(x, f(x))$
- $\exists x \exists y P(x,y) = P(a,b)$

5.3 Příklady – Knihovna

- **V knihovně je vždy Jana, Klára nebo obě:** $(J \vee K) \vee (J \wedge K) \longrightarrow J \vee K$
- **V knihovně je buď Hana nebo Iva, ale ne obě:** $(H \wedge \neg I) \vee (\neg H \wedge I) \longrightarrow (H \vee I) \wedge (\neg H \vee \neg I)$
- **Je-li v knihovně Klára, jsou tam i Gréta a Jana :** $K \Rightarrow (G \wedge J) \longrightarrow (\neg K \vee G) \wedge (\neg K \vee J)$
- **Jana a Iva jsou všude společně (jsou tam obě nebo žádná z nich):** $J \Leftrightarrow I \longrightarrow (\neg J \vee I) \wedge (J \vee \neg I)$
- **Je-li v knihovně Gréta je tam i Hana:** $G \Rightarrow H \longrightarrow \neg G \vee H$
- **Dokažte, že je Iva v knihovně:** $(\neg I)$

5.4 Příklady – Rezolvent

- $\exists(x)\forall(y)(A(x,y,c) \wedge B(x, f(b), y))$
- $\forall(v)\forall(w)\exists(z)(C(z, f(w)) \vee \neg(A(w,z,v)))$
- **Redukování EXIST($\exists(x)$):**
  a) Pokud se před EXIST nic nenachází, smažu ho a danou přoměnnou ⇒ (x) nahradím konstantou, která se ve formuli nevyskytuje ⇒ (a)
  b) Pokud se před EXIST nachází univerzalní kvantifikátor, smažu EXIST a danou přoměnnou ⇒ (z) nahradím funkčním symbolem, který se ve formuli nevyskytuje a obsahuje univerzalní kvantifikátory před EXIST(v přesném pořadí v jakém se před ním nachází)
- $\exists(x)\forall(y)(A(x,y,c) \wedge B(x, f(b), y)) \longrightarrow \forall(y)(A(a,y,c) \wedge B(a, f(b), y))$
- $\forall(v)\forall(w)\exists(z)(C(z, f(w)) \vee \neg(A(w,z,v))) \longrightarrow \forall(v)\forall(w)(C(g(v,w), f(w)) \vee \neg(A(w, g(v,w), v)))$
- **Získání klauzulí[1]:** Oddělíme od sebe operace obsahující AND($A \wedge B$), tak že formulí před AND umístíme jako první a pod ní umístímě zbytek formule za ANDem tím nám vzniknou dvě formule

[1] Klauzule je cokoliv co má všechny kvantifikátory(obecně - Velká písmena) před závorkou a vně jsou literály

- **Klauzule:**
  1. A(a,y,c)
  2. B(a,f(b),y)
  3. $C(g(v,w),f(w)) \vee \neg(A(w,g(v,w),v))$
- **Resolvent a subtituce:**
  a) Pokud klauzule A existuje i jako $NOT(\neg)A$ a má stejný počet paramatrů uděláme rezolventu z 1 a 3 předtím však musíme provést substituci
  b) Substituce: $w \rightarrow a[a/w]; v \rightarrow c[c/v]; y \rightarrow g(c,a)[g(c,a)/y]$
- Výsledek je pouze 1 rezolventa: $C(g(c,a),f(a))$ nepočítá se zde ovšem rezolventa, která nám zbyla($B(a,f(b),y)$). Toto nám nevede k dokázání platnosti
- Platnost dokážeme tak že nám po aplikaci rezolventy zbyte prázdná množina