TSP

- Co to je
- Reprezentace
    - Adjacency
        - City **j** na pozici **i** <==> existuje hrana z **i** do **j**
        - Neintuitivní
        - Nefungují klasické crossovery
        - Funguje se schématy
    - Normální
        - Nefungují klasické crossovery, viz následující seznam
        - První idea každého člověka
    - Reálná čísla
- Crossover
    - PMX
        - https://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/PMXCrossoverOperator.aspx
    - CX
        - https://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/CycleCrossoverOperator.aspx
    - ER
        - https://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/EdgeRecombinationCrossoverOperator.aspx
    - OX
        - https://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/Order1CrossoverOperator.aspx
- Initialization
    - Nearest neighbor
        - Hledáme k poslednímu městu
    - Edge insertion
        - Jako nearest neighbor, ale pro všechny města
- Mutation
    - Náhodně prohodit dvě města
    - 2-opt
    - Inverze
        - 12**34567**89 ->12**7654**389
    - Prohodit dvě podcesty
        - 12**34**56**78**9 -> 12**78**56**34**9

Michigan vs Pittsburgh (Evolutionary machine learning)

- Learning of rules = classifiers system
- Cílem je naučit se skupinu pravidel, která korektně klasifikuje data
- Dvě školy genetických algoritmů, aplikujeme je na evoluční machine learning

- Jak má být pravidlový systém reprezentován
- Learning dělíme na
    - Classification
        - Speciální případ supervised learning - máme odpovědi
    - Reinforcement learning
        - Obvykle agent v nějakém prostředí a okolí odesílá nějaká data (sensory na robotovy) a dává nám nějaké rewards (robot se dostal na nějaké místo v prostředí), ale tyto odměny nejsou okamžité (po aplikaci jednoho pravidla)
- Výhoda genetických algoritmů je, že obvykle není velký problém je změnit z klasifikace na reinforcement learning (a naopak), obvykle je potřeba nějak změnit vyhodnocování fitness funkce
- Machine learning
    - Chceme získat nějaká data a z nich vytvořit model
- Rule = pravidlo
    - Speciální podmnožina modelu
    - Uvažujeme tvar IF □ THEN □
- Obě dva přístupy jsou použitelné na klasifikaci (Michigan a Pittsburgh)
- Michigan je lepší na reinforcement learning
- Dneska se na klasifikace používá hlavě Pittsburgh
- Klasifikační systémy postavené na pravidlech mohou mít horší výsledky než modely postavené na neuronových sítích, ty ale není možné tak dobře rozumět, proč tak fungují
- Další problém např. Deep learningu je, že potřebuje **obrovské** množství dat oproti ostatním (klasickým pravidlovým) způsobům
- **Michigan**
    - University of Michigan
    - John Holland zde pracoval
    - Založeno na jeho původní práci (Simple genetic algorithm)
    - Individuál = 1 pravidlo
    - Máme populaci pravidel = kompletní klasifikační systém
    - Jde použít v reinforcement learning i supervised learning
    - Supervised learning je lehčí (máme správné hodnoty)
    - LCS (learning classifier system)
        - 80’ Holland
        - Pouze binární řetězce a hvězdička (znamená 0, nebo 1)
        - Příklad s ideálním partnerem
            - Intelligent - 1
            - Hair color - *
            - Woman - 1
            - Likes music - 1
        - Každý jedinec má jednu hodnotu z reálných čísel často zvanou strength nebo weight - použití jako fitness nebo váha pravidla
        - Části: Environment, populace pravidel, mechanismus selekce akce, mechanismus distribuce odměny, GA

- Bucking brigate algorithm (patří k LCS)
  - Původní práce Hollanda
  - Chceme použít s reinforcement learning
  - Strength pravidla by měla mít komplexnější způsob definice
  - Příklad s robotem:
    - Robot provádí sekvenci akcí iniciovanou pravidly
    - Ri -> Rj -> Rk -> R_lucky
    - R_lucky bude pravidlo po jehož provedení robot dostane "odměnu" za to, že dosáhl nějaké požadované akce
    - Celou odměnu dostane pravidlo R_lucky, ačkoliv to není fér, protože celá sekvence akcí předtím způsobila, že robot udělal co udělal
    - Bucking brigate algorithm redistribuje odměnu do předchozích pravidel
    - Podmínka pro aplikaci pravidla je, že se okousne trochu z jeho strength za spuštění jeho akce, takto se akumulují jakoby penízky a když dojde k odměně, tak ji redistribujujeme podle ukousnuté strength
    - I gave up some of my strength, but I will receive more in return
  - Málo používané, je obtížné správně zvolit právě tu správnou hodnotu investice ze strength a správnou hodnotu odměnu kterou pak zaplatit zpátky
  - Dobrá myšlenka, v praxi obtížně implementované, dnes opuštěný koncept
  - Jsou tam messages a rules
  - GA operuje nad rules
  - Komplikované i implementovat
- ZCS (Zero classifier system)
  - Nová generace lidí v 90' (Wilson)
  - Zjednodušení (žádné messages a bucket brigade)
  - Nejprve úspěšné
  - Bez bucking brigate, ale historii si nějak pamatovat budeme
  - Příklad
    - Jsem neaplikovatelné pravidlo -> nic se nestane
    - Jsem aplikovatelný -> ale vím, že nejsem správný, nedostanu se aplivací tohoto pravidla do požadovaného stavu, tak snížíme strength (WUT jak to poznám)
    - Každé pravidlo musí dát nějaký strength (malé množství)
    - Zvýšíme strength pravidel, které správné odpověděli, **ale pouze těch v předchozím kroku** (použiji naakumulované penízky)
  - Nějak jestli jsem pochopil, tak neřešíme komplexní historii, ale pouze jeden dva stavy dozadu
  - COVER operátor (tohle je důležitá součást ZCS)
    - Je těžký mít pořád pravidla, která jsou aplikovatelná na každou situaci

                - Řeší právě tuto situaci, nemám pravidlo na aplikování v dané situaci
                - Aplikací COVER operátoru vytvoříme nové pravidlo (podle vstupu, na které nebylo aplikovatelné žádné pravidlo)
                - Přidáme toto nové pravidlo do populace
            - Charakterizace
                - Mnohem jednodušší mechanismus, který si pamatuje pouze několik předchozích kroků
                - Odměňuje aktuální a několik posledních
                - COVER operátor
    - XCS (Extended classifier systems)
        - Vylepšená verze ZCS (také od Wilsona)
        - State-of-the-art pro learning klasifikátory
        - Spousta různých popravených variant se dnes používá
        - ZCS fungovali stále dobře, ale měli několik nevýhod
            - Odměny fungují dobře pro předchozí krok, ale ne úplně dobře v dlouhodobém měřítku
            - Vyhrávající pravidla jsou obvykle ta, co jsou více obecně aplikovatelná v různých situacích (oproti specializovaných pravidlům, ačkoliv ty mohou být stejně důležité)
        - Předchozí systémy byly všechny strength based
            - Když bylo víc pravidel na výběr, vybíralo se podle strength
        - Strength is good for reward and výběr akce ale vývoj evoluce by měl být zaměřený na nejspolehlivější pravidla, které dávají přesnější predikce
        - Nová pravidla z úspešných pravidel
        - Používá GA na Action set (kandidáti na úspešná pravidla)
- **Pittsburgh**
    - Specializované na klasifikaci
    - První generaci studentů Johna Hollanda
    - Tohle je šílené
    - Individuál = množina pravidel = klasifikační systém
    - Definujeme, že třeba potřebujeme 5 pravidel na ovládání robota, budeme tedy mít takového individuála a vylepšovat ho v rámci nějaké populace, kde všichni individuálové mají 5 pravidel.
    - Z více pravidel na jedince vyplývají i složitější genetické operátory
        - Jak provést crossover na všechny pravidla? (mezi jedinci, nebo uvnitř jedince - je to stále crossover?)
        - Mutace i crossover na více úrovních
    - GIL
        - Jednoduché a prakticky použitelné
        - Specializované na binární klasifikaci (prvek spadá do kategorie 0, nebo 1)
        - Pravidla mají implicitní klasifikační skupinu
        - Typy pravidel
            - Complex

            - Konjunkce selektorů
            - S1 & S2 & S3
            - Jedno pravidlo
        - Selector
            - Disjunkce proměnné je disjunkce hodnot z domény
            - (C = R) or (C = B)
    - Variable
        - Domain (diskrétní)
        - Values (z této domény)
    - Příklad: color (R, G, B)
        - Domain (0, nebo 1)
    - Individual je množina complexů
    - Jednoduše reprezentovatelné bitmapou → jdou aplikovat standardní genetické operátory
    - Existují operátory na úrovni complexů i selektorů a je jich spousta
    - Jednoduchá mutace na selektoru (přehodím bit)
    - Reduction
        - Specializovaná mutace na selektoru změním 1 → 0
        - Děláme selektor menší
    - Extension
        - Opak reduction
        - 0 → 1

Úvod

- Důležitá inspirace biologií
    - Darwinova teorie přirozeného výběru (Darwinova evoluční teorie)
        - Polovina 19. Stol.
        - Opravdová komplexní teorie jak evoluce funguje, reprodukce je klíč života
        - Lépe adaptovaní jedinci mají větší šanci se reprodukovat (a tak se v populaci více rozšířit své geny)
        - Založené na pozorování, Darwin procestoval svět na lodi
        - Žil ve spoustě prostředí
        - Neměl žádný důkaz (fyzikální, biologický) pro jeho teorii
    - Mendelova genetika
        - Pravidla genetiky (založená na logice a statistických experimentech)
        - V podobné době co Darwin
        - Nepovšimnuté polovinu století
        - Na začátku 20. stol znovu objeveno britskými biology
        - Existuje nějaká základní jednotka dědičnosti → dnes ji říkáme gen
        - Gen je diskrétní věc → máme ho, nebo ne
        - Mendel ale nevěděl, jak je gen reprezentován v přírodě
        - Máme 2 páry genů, který vytvoří relevantní informaci o našem fenotypu
        - Pioneer základů genetiky

        - Nyní ale víme, že to je více komplikované, genotyp a fenotyp není 1:1 mapování
    - DNA
        - Čekali jsme dalších 100 let
        - Základ jak reprezentovat genetický kód v biologických organismem
        - Double helix (dvojitá šroubovice)
        - Vždy trojice nukleotypů
            - 3 písmena, která kódují jeden z 23 aminokyselin (základní stavební struktura všeho v živých organismech)
        - Dnes v molekulární genetice rozumíme, že DNA přes kodon obsahuje informace, potom existuje proces transkripce do RNA (bijektivní funkce)
        - RNA je pak detailní návrh pro výrobu proteinů
        - Takhle máme mapování z genotypu na fenotyp (DNA → RNA → Protein)
        - Neexistuje cesta zpět fenotyp nemůže ovlivnit genotyp
- Existují další teorie jako např. Lamarkismus, který tvrdí, že získané vlastnosti je možné dědit, dnes víme, že toto není pravda, ale pro evoluční algoritmy se to může hodit.
- Altruism vs Darwinism
    - Mnoho kooperace existuje i v přírodě (mimo lidi)
    - Jak se mohlo altruistické chování vyvinout, když darwinismus snižuje fitness altruistických individuí
    - Tímto se zabývá social biology
- EA je meta algoritmus
    - No free lunch theorem (neexistuje jeden nejlepší algoritmus)
    - Vyplatí se vytvářet specifické varianty EA pro různé účely

Exploration (průzkum) vs exploitation (vykořisťování)

- Co by hledací algoritmy měly dělat
- Exploration: Měli bychom trávit čas explorací nových oblastí možných řešení
- Exploitation: Měli bychom využít naše aktuální znalosti a prohledat do hloubky
- Holland původně vymyslel adaptivní plán motivovaný přírodou
    - Explorace a exploitace jsou v rovnováze
    - Pouze explorace → jenom budu hledat a nikdy znalosti nevyužiji
    - Pouze exploitace → velká šance, že skončím v lokálním optimu
- Jak zajistíme rovnováhu?
    - Inspirace z 2-Armed bandit
    - 1-armed
        - Obyčejný gambling mat, s páčkou, kde vyhraješ money, když dostaneš 3x jablíčko třeba
    - 2-armed je zajímavější z matematického hlediska
        - Máme omezené množství coinů
        - Jak identifikovat a využít “arm”, který lépe platí
        - Každé arm má jiné Expected value a varianci…
        - Snažíme se odhadnout statistické hodnoty

        - Řešení: Lepší arm hrajeme exponenciálně vícekrát než druhý arm (zajímavé je, že ten druhý ale stále hrajeme) → náš odhad nemusí být správný
    - K-armed bandit
        - Důležitý problém reinforcement learningu a optimalizace
        - Zajímavé když nevíme žádné statistické údaje
- Holland tedy použil, že genetický algoritmus alokuje více pokusů pro úspěšnější schémata

Evoluční strategie

- Efektivní a populární způsoby jak optimalizovat floating point problémy
- Alternativa k genetickým algoritmům (GA)
- 60' Rechenberg, Schwefel (Německo)
- Měli optimalizační problém s příliš mnoho proměnnými pro analytické řešení
- Experimenty s randomizovanými algoritmů, které hledají řešení
- Nakonec přišli s tím, čemu dnes říkáme evoluční strategie
- Řeší optimalizaci reálných funkcí, pracují s floating point čísly (vektory) (žádné binary jako u Hollanda)
- Individum má dvě části
    - Genetic parameters (numbers - original individual)
        - x1, …, xn
    - Endogenous parameters (ovlivňuje evolution)
        - Můžeme sem dát třeba sigmu, jejíž hodnotu můžeme vyvíjet např. Násobením N(0,1)
        - Tomuto se říká metaevoluce, parametry evoluce procházejí také procesem evoluce
- Nové individum je akceptováno pouze pokud je lepší
- Výběr rodičů se liší oproti GA → zde můžeme mít více rodičů
- Máme ale vždy pouze jednoho potomka (v GA máme 2 obvykle)
- Stále máme spoustu aplikací v deep learningu nebo reinforcement learning
- Máme dvě možnosti na další populaci
    - Buďto vždycky vyměníme starou populaci za novou (jako v GA)
    - Nebo vyrobíme množinu nových a pak vybereme novou populaci ze starých a nových individuí (jako vybereme ty nejlepší)
- Příklad (nejjednodušší evoluční strategie):
    - Minimalizujeme funkci
    - Individum je vektor
    - Populace je jedno individum
    - Vytváříme jednoho potomka
    - Provádíme na něj mutaci (z normálního rozdělení - upravíme nějakou složku vektoru)
    - V tomto případě nemáme crossover
- Jak velká by měla být mutace (krok mutace)?
    - Určuje parametr sigma normálního rozdělení

        - Když je sigma velká → děláme exploration
        - Když je malá → exploitation
        - Experimentálně ověřeno, že nejlepší je, když mutace vytvoří lepšího jedince ve 20%
        - Můžeme udělat algoritmus tak, že budeme kontrolovat, jak je mutace úspěšná a podle toho dynamicky měnit sigma parametr
- Selekce (detailněji, něco bylo už výše)
    - Parental
        - totálně random (nevybíráme podle fitness), u GA to bylo ruletou nebo turnajem
    - Environmental
        - Vybereme nejlepšího z populace
        - Potom nová populace jsou buďto nový jedinci (comma strategy)
        - Nebo společně s rodiči (plus strategy) (asi vyberu nejlepší)
        - Comma strategy
            - Noví kandidáti jsou označeni jako L
            - Starý jsou M
            - L > M (musí)
            - Příklad
                - M = 10
                - L = 100 (vygenerujeme kandidáty)
                - Vybereme 10 z nich a uděláme nové M
            - Sklony k zaseknutí v lokálním optimu
            - Doporučné pro domény z reálných čísel
        - Plus strategy
            - L může být jakékoliv, klidně L=1 což je speciální případ (steady state evolution strategy)
            - Doporučené pro diskrétní domény
- Mutace (detailněji)
    - Nejdříve se vždy mutují endogenous parametry (metaevoluce)
    - Pravidlo 1/5
- Crossover (detailněji)
    - Libovolné množství rodičů
    - Uniformní (dominant)
    - Aritmetický (intermediate)

Věta o schématech

- První pokus formalizace genetických algoritmů
- John Holland
- Snaží se popsat teoreticky jak genetický algoritmus funguje
- Staré, ne tak dobré
- Jedna z mála teoriích o genetických algoritmech
- Dnes ale máme lepší teorii a poznatky
- Není příliš obecná

- $o(S)$ … počet pevných pozic schématu (počet 0 a 1)
- $d(S)$ … vzdálenost mezi první a poslední pevnou pozicí
    - Index poslední pevné pozice - index první pevné pozice
- $F(S)$ … průměrná fitness jedinců v populaci, kterým schéma S odpovídá
- Přeformulovaná verze, nepoetická
    - ∀ schémata, která mají malou definující délku $d(S)$, malý řád $o(S)$ a nadprůměrnou fitness $F(S)$ platí, že při běhu GA se exponenciálně množí
        - nadprůměrnou fitness $F(S)$ ~ velká šance, že schéma bude vybráno
        - Malá definující délka $d(S)$ ~ malá šance, že crossover zničí schéma
        - malý řád $o(S)$ ~ malá šance, že mutace zničí schéma
- Budeme se zabývat jak souvisí selection, crossover a mutation se schématy
- Máme nějaká schémata, kterých odpovídají jedinci, kteří jsou využity v GA. Nás bude zajímat které ze schémat budou v další generaci mít nějakého jedince, kterému odpovídají. **Jaké jsou vlastnosti schémat, které se stanou součástí další populace?**
- O tomhle se budeme bavit v kontextu genetických operátorů (selekce, mutace, crossover)
- Jaká je dobrá vlastnost schématu aby bylo vybráno? (selekce)
    - Selekce je podle fitness
    - Pravděpodobnost souvisí s $F(S)$
- Mutace
    - Bavíme se o jedincích v populaci, kteří odpovídají schématu
    - Když mutace chce změnit hodnotu, kde je hvězdička, nic se neděje → schéma přežije
    - Problém je, když se mutace trefí do 0, nebo 1
    - Pravděpodobnost přežití schématu souvisí s $o(S)$
- Crossover
    - Kdy může crossover zničit schéma?
    - Pokud máme hvězdičky na konci nebo začátku, a crossover dělící čára oddělí tyto hvězdičky, tak schéma nebude zničeno
    - Kdekoliv jinde ho ale může zničit
    - Pravděpodobnost přežití schématu souvisí s $d(S)$
- Nyní rozumíme všemu co je potřeba, přesuneme se k důkazu (minimálně nějaký náznak chce)
- Půjdeme od jedné populaci k druhé a budeme se koukat na pravděpodobnost přežití schématu
- $C(S,t)$ … počet individuálů odpovídající schématu S v populaci $P(t)$
- Zajímá nás $C(S,t+1)$
    - Vypočítáme ve třech krocích (selection, crossover, mutation)
- Selekce
    - $F(t) = \Sigma F(u)$ kde $u \in P(t)$ (součet fitness prvků v populaci v čase t)
    - $P_S(v) = \frac{F(v)}{F(t)}$ (pravděpodobnost selekce prvku v)
    - $P_S(S) = \frac{F(S)}{F(t)}$ (pravděpodobnost výběru schématu)

- $C(S, t+1) = C(S,t) \cdot n \cdot P_S(S)$
  - Selekci děláme n - krát
  - Pro každou selekci máme spočítanou pravděpodobnost výběru schématu
- Jde to přepsat na $C(S, t+1) = C(S,t) \cdot \frac{F(S)}{F_{avg}(t)}$
  - $F_{avg}(t) = \frac{F(t)}{n}$ (průměrná hodnota fitness v populaci v čase t)
- My jsme ale předpokládali, že schémata mají nadprůměrnou fitness
  - $F(S,t) = F_{avg}(t) + \varepsilon \cdot F_{avg}(t)$ t je celé číslo ≥ 0
  - $C(S, t+1) = C(S,t) \cdot (1+\varepsilon)$
  - $C(S, t+1) = C(S,0) \cdot (1+\varepsilon)^t$
    - Geometrická řada

- Crossover
  - Víme, že schémata mají malou definující délku $d(S)$
  - $P_d(S) = \frac{d(S)}{m-1}$
    - m … délka individua (počet 0 a 1)
    - $m-1$ … počet pozic pro crossover
    - Pravděpodobnost, že schéma bude zničeno
  - $P_s(S) = 1 - \frac{d(S)}{m-1}$
    - Pravděpodobnost, že schémat přežije
  - Nesmíme zapomenout, že obvykle to, že crossover nastane má nějakou pravděpodobnost, tu označíme P_c
  - $P_s(S) \geq 1 - P_c \cdot \frac{d(S)}{m-1}$
  - Dohromady s selekcí máme
    - $C(S, t+1) \geq C(S,t) \cdot \frac{F(S)}{F_{avg}(t)} \cdot (1 - P_c \cdot \frac{d(S)}{m-1})$

- Mutace
  - P_m … bit nepřežije
  - $1 - P_m$ … bit přežije
  - $P_s(S) = (1-P_m)^{o(S)}$
    - Toto je korektní definice, nicméně, pravděpodobnost mutace je nízká
  - $P_s(S) \approx 1 - P_m \cdot o(S)$
  - Zkombinujeme se selekcí a crossoverem
    - $C(S, t+1) \geq C(S,t) \cdot \frac{F(S)}{F_{avg}(t)} \cdot (1 - P_c \cdot \frac{d(S)}{m-1} - P_m \cdot o(S))$
    - Vidíme, že pokud $F(S)$ je velké, $d(S)$ je malé a $o(S)$ je malé, dostaneme růst (exponenciální)

- Problém věty je, že důkaz je postaven na tom, že $C(S,t)$ je vlastně expected value, tedy věta dobře funguje na nekonečných populacích, v praxi to nefunguje na malých populacích
- Hypetéza o stavebních blocích (neplatí, nefunguje)
    - GA hledá optimální řešení problému kombinací schémat, která mají malou definující délku $d(S)$, malý řád $o(S)$ a nadprůměrnou fitness $F(S)$

Další věci

- Evoluční programování
    - 1965 Fogel, Owens a Walsh
    - Evoluce konečných automatů
    - Smazán rozdíl mezi genotypem a fenotypem
    - Důraz na mutace
    - Neexistuje křížení
- Genetické programování
    - 1992 Koza
    - Evoluce jedinců zakódovaných jako LISPovské stromy
    - Použití (nejen) k evoluci programů
- V původní SGA používal Holland ještě jeden genetický operátor, který dneska nepoužíváme – inverze
    - Obrácení části řetězce
    - Neukázala se jako výhodná