Ľubomír Dlhý, ID: 102897

EVOLUČNÉ ALGORITMY

Ľubomír Dlhý, ID: 102897

ZADANIE

Obchodný cestujúci má navštíviť viacero miest. V jeho záujme je minimalizovať cestovné náklady a cena prepravy je úmerná dĺžke cesty, snaží sa nájsť najkratšiu možnú cestu tak, aby každé mesto navštívil práve raz. Keďže sa nakoniec musí vrátiť do mesta z ktorého vychádza, jeho cesta je uzavretá krivka.

Úloha c) - Problém obchodného cestujúceho

Je daných aspoň 20 miest (20 – 40) a každé má určené súradnice ako celé čísla X a Y. Tieto súradnice sú náhodne vygenerované. (Rozmer mapy môže byť napríklad 200 * 200 km.) Cena cesty medzi dvoma mestami zodpovedá Euklidovej vzdialenosti –vypočíta sa pomocou Pytagorovej vety. Celková dĺžka trasy je daná nejakou permutáciou (poradím) miest. Cieľom je nájsť takú permutáciu (poradie), ktorá bude mať celkovú vzdialenosť čo najmenšiu. Výstupom je poradie miest a dĺžka zodpovedajúcej cesty.

RIEŠENIE

REPREZENTÁCIA ÚDAJOV

Mesto je reprezentované súradnicami x a y. Všetky mestá sú v poli. Vzdialenosti jednotlivých miest sa nachádzajú v matici, kde prvý index označuje prvé mesto a druhý index druhé mesto. Jednotlivé poradia miest sú reprezentované poľom indexov miest. Celková vzdialenosť sa vypočíta ako súčet vzdialeností susedných miest.

Na riešenie problému som použil 3 rôzne algoritmy (2 na lokálne prehľadávanie a 1 na genetické programovanie):

- **TABU SEARCH**
- **SIMULOVANÉ ŽÍHANIE**
- **GENETICKÝ ALGORITMUS**

Ľubomír Dlhý, ID: 102897

TABU SEARCH

Tabu Search je algoritmus lokálneho prehľadávania s optimalizáciou, ktorá sa snaží zamedziť uviaznutiu v lokálnom extréme. Používa zoznam zakázaných permutácií (tabu list), do ktorého sú vkladaní najlepší nasledovníci. Tým sa zamedzí cyklenie medzi uzlami.

REPREZENTÁCIA ÚDAJOV

Mestá sú reprezentované dvojicou x,y (súradnicami). Všetky mestá sú uložené v poli. Sekvencia miest je daná poľom indexov jednotlivých miest.
Tabu list je implementovaný poľom, v ktorom sa nachádzajú sekvencie miest. Veľkosť tabu listu neuvádzam ako konštantu, ale odvíja sa od počtu miest a koeficientu z dôvodu vyhovujúceho fungovania aj pri väčších počtoch miest.

NASTAVITEĽNÉ PARAMETRE

- **Počet iterácií** - počet iterácií algoritmu
- **Koeficient veľkosti tabu listu** - veľkosť tabu listu je určená vzorcom **|T| = k*N**, kde N je počet miest a k je koeficient veľkosti tabu listu

ALGORITMUS

1. Vytvor počiatočnú permutáciu miest, ohodnoť ju a nastav ju ako najlepšie riešenie
2. Vygeneruj nasledovníkov (permutácie vzniknuté prehadzovaním poradím miest) a ohodnoť ich
3. Vyber takého najlepšieho nasledovníka, ktorý sa nenachádza v tabu liste a má najnižšiu vzdialenosť miest
4. Ak je vybraný najlepší nasledovník lepší ako globálne najlepšie riešenie, nastav najlepšie riešenie na tohto nasledovníka, inak pokračuj
5. Vlož najlepšieho nasledovníka do tabu zoznamu
6. Ak tabu zoznam prekročil veľkosť určenú parametrom, vymaž najstarší prvok, inak pokračuj
7. Zväčši počet iterácií o 1
8. Ak počet iterácií dosiahol limit určený parametrom, ukonči program, inak choď na krok č. 2

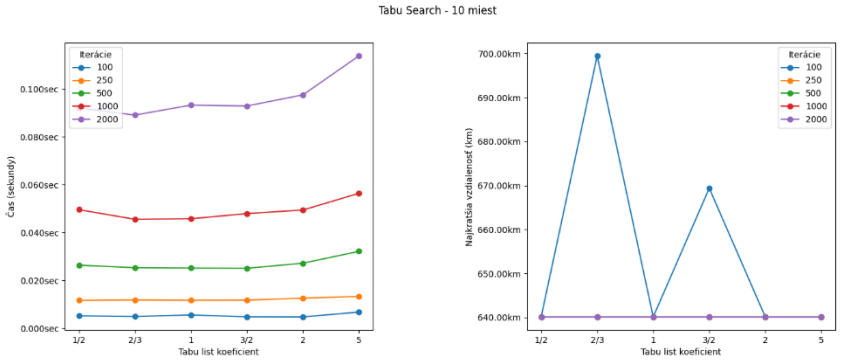
GENEROVANIE NASLEDOVNÍKOV

Množinu nasledovníkov generujem tak, že pre každé mesto vytvorím takú permutáciu, v ktorej je vymenené toto mesto a ďalšie náhodne vybrané mesto. Takto sa v ideálnom prípade vytvorí množina N nových permutácií, kde N je počet miest.

POROVNANIE PARAMETROV

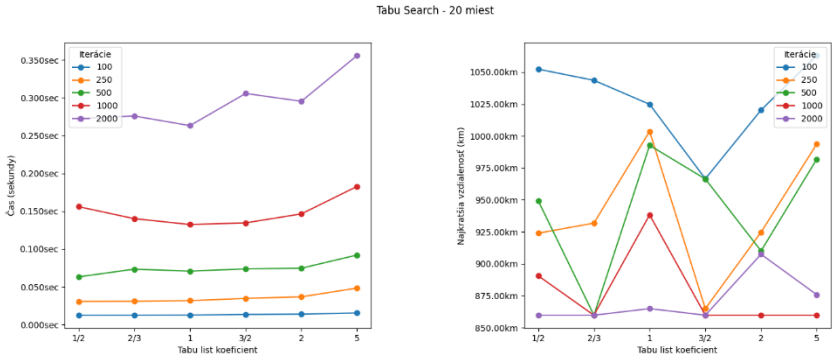
Určenie správnych parametrov je dôležité pre optimálne fungovanie algoritmu a zabráneniu uviaznutiu v lokálnom minime, resp. pri prípadnom cyklení medzi niektorými lokálnymi minimami.
Testoval som viacero hodnôt počtu iterácií (100, 250, 500, 1000, 2000) a koeficientu veľkosti tabu listu (1/2, 2/3, 1, 3/2, 2, 5). V testoch som zaznamenával a porovnával čas trvania programu a výsledné riešenie (cesta medzi mestami). Každý testovací scenár je vykonaný 10-krát a výsledok je určený ako priemer zo všetkých parciálnych výsledkov.

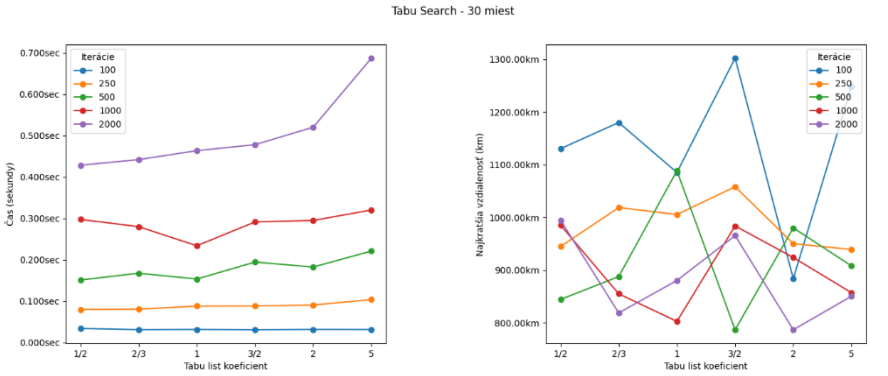
Ľubomír Dlhý, ID: 102897



Pri 10 mestách sú riešenia takmer rovnaké – iba pri 100 iteráciách nedokázal algoritmus vždy nájsť optimálne riešenie. Dĺžka trvania sa mení hlavne s rastúcim počtom iterácií a nie kvôli veľkosti tabu zoznamu.

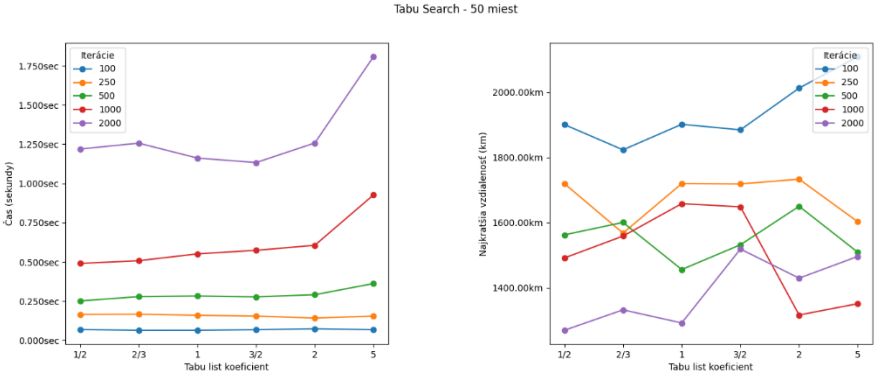
Pri 20 mestách sa riešenia značne líšia. Najlepšie výsledky dosiahli 2000 a 1000 iterácií pri všetkých veľkostiach tabu listu. 250 a 500 iterácií stále dokážu nájsť celkom dobré riešenie, dokonca niekedy to optimálne. Iba 100 iterácií neukazuje vhodné riešenie v žiadnej situácii.





Pri 30 mestách nie je rozdiel medzi 1000, 2000 a 500 iteráciami taký signifikantný. 250 a 100 iterácií neukazujú v žiadnom prípade optimálne riešenie.

Pri 50 mestách má najlepšie výsledky 2000 iterácií. 1000 a 500 iterácií sú približne podobné. 250 a 100 iterácií sa veľmi nepribližujú k globálnemu riešeniu.



Testy s rôznymi počtami miest ukázali, že nevynikajú niektoré hodnoty iterácií a veľkosti tabu listu. 2000 iterácií hľadá suverénne najlepšie riešenie, ale trvá príliš dlho. 1000 a 500 iterácií ukazujú približne podobné výsledky, pričom 500 iterácií je časovo efektívnejšie ako 1000 iterácií. Ostatné počty iterácií neukazujú optimálne riešenie pri zvyšujúcich sa počtoch miest. Veľkosť tabu zoznamu zásadne nevyplýva na riešenie, ale pri veľkých veľkostiach tabu listu ukazuje vyššiu časovú náročnosť. Po zvážení všetkých aspektov som ako **najvhodnejšie** riešenie zvolil **500 iterácií** a **koeficient veľkosti 1** najmä kvôli výbornému pomeru hodnoty vzdialenosti za vykonaný čas.

Ľubomír Dlhý, ID: 102897

SIMULOVANÉ ŽÍHANIE

Simulované žíhanie je algoritmus lokálneho prehľadávania s optimalizáciou, ktorá sa snaží zamedziť uviaznutiu v lokálnom extréme. S určitou pravdepodobnosťou prejde aj do horšieho uzla, čím sa snaží vyvarovať uviaznutiu v lokálnom extréme. Táto pravdepodobnosť sa postupne zmenšuje určitou rýchlosťou. Ak sa nepodarí prejsť do uzla, tak sa program končí a dostávame riešenie.

REPREZENTÁCIA ÚDAJOV

Mestá sú reprezentované dvojicou x,y (súradnicami). Všetky mestá sú uložené v poli. Sekvencia miest je daná poľom indexov jednotlivých miest.
Konečná teplota sa počíta ako 1 – Miera chladnutia

NASTAVITEĽNÉ PARAMETRE

- Počiatočná teplota - počiatočná teplota algoritmu
- Miera chladnutia - rýchlosť, akou sa teplota bude ochladzovať (násobiť teplotu mierou chladnutia)
- Konečná teplota - ak teplota dosiahne konečnú teplotu (alebo menšiu), program skončí, je určená vzorcom 1 – Miera chladnutia

ALGORITMUS

1. Vytvor počiatočnú permutáciu miest, ohodnoť ju a nastav ju ako najlepšie riešenie
2. Vygeneruj náhodného nasledovníka (permutácia vzniknutá vymenením poradím 2 náhodných miest) a ohodnoť ho
3. Ak je nasledovník lepší ako aktuálna permutácia, nastav permutáciu na nasledovníka a pokračuj na bod 5
4. Inak náhodne vygeneruj číslo a skús s pravdepodobnosťou nastaviť horšieho nasledovníka na permutáciu
5. Zníž teplotu pomocou miery chladnutia (vynásob teplotu koeficientom chladnutia)
6. Ak teplota je väčšia ako konečná teplota, tak opakuj od bodu 2, inak ukonči program

GENEROVANIE NASLEDOVNÍKOV

Pri každej iterácii programu je vytvorený jeden nasledovník pomocou výmeny 2 náhodne zvolených miest.

PRAVDEPODOBNOSTNÁ FUNKCIA

Pravdepodobnostná funkcia je daná rovnicou:

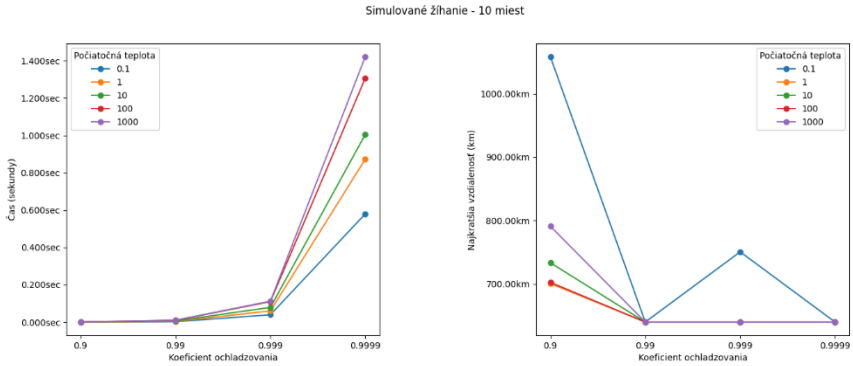
$$p = \begin{cases} 1, & D_n < D_b \\ e^{\frac{D_b - D_n}{T}}, & D_n \geq D_b \end{cases}$$

$D_n - veľkosť\ cesty\ nasledovníka$
$D_b - veľkosť\ cesty\ aktuálnej\ permutácie$
$T - aktuálna\ teplota$

POROVNANIE PARAMETROV

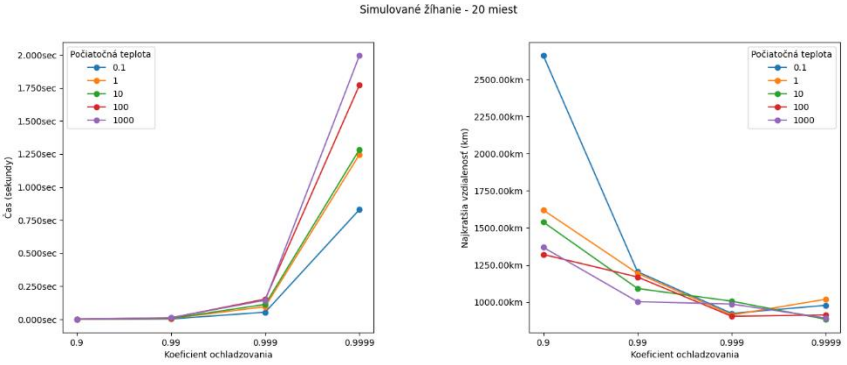
Určenie správnych parametrov je dôležité pre optimálne fungovanie algoritmu, zabránenie uviaznutiu v lokálnom minime a vyvarovanie sa príliš nízkej časovej efektivite.
Testoval som viacero hodnôt počiatočnej teploty (0.1, 1, 10, 100, 1000) a mieru ochladzovania (0.9, 0.99, 0.999, 0.9999). V testoch som zaznamenával a porovnával čas trvania programu a výsledné riešenie (vzdialenosť cesty).
Každý testovací scenár je vykonaný 10-krát a výsledok je určený ako priemer zo všetkých parciálnych výsledkov.

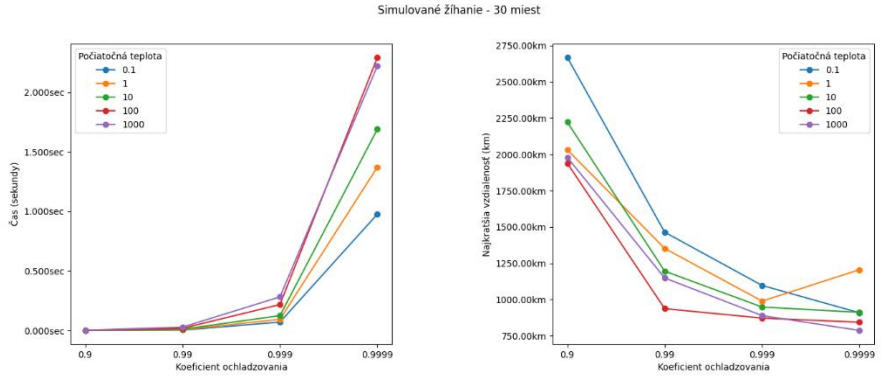
Ľubomír Dlhý, ID: 102897



Pri 10 mestách sa ukazuje veľká časová neefektívnosť pri miere ochladzovania 0,9999 a pri miere ochladzovania 0,9 sú riešenia málo presné.

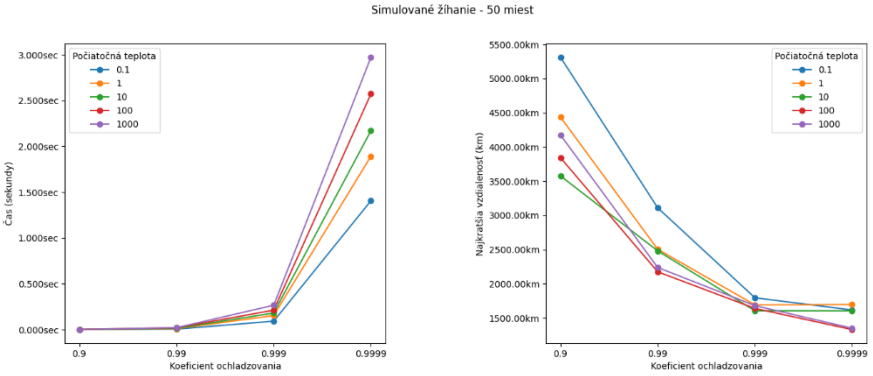
Pri 20 mestách sa potvrdzuje vysoká časová záťaž pri poslednej miere ochladzovania. Nájdené cesty sú lepšie pri menších mierach ochladzovania. Teplota mierne vplýva na riešenie – väčšia teplota spôsobuje optimálnejšie riešenie.





Pri 30 mestách je rovnaký scenár ako pri predošlom teste, iba rozdiel medzi teplotami je viac citeľný. Väčšia teplota nájde lepšie riešenie takmer vo všetkých prípadoch.

Pri 50 mestách sa opäť ukazuje podobný trend ako pri predošlých testoch. Väčšia teplota nájde lepšie riešenie a väčšia miera ochladzovania je časovo náročnejšia, no nájde vhodnejšie riešenie.



Z vykonaných testov sme zistili, že miera ochladzovania a počiatočná teplota priamo ovplyvňujú výsledné riešenie. **Najlepšou mierou ochladzovania** bola najmenšia hodnota, ale keďže bola veľmi časovo neefektívna, tak ako optimálnu hodnotu považujem **0,999**. Teplota v priamej úmernosti ovplyvňovala optimálnosť riešenia. Avšak, pri posledných dvoch hodnotách (100 a 1000) nebol až tak signifikantný rozdiel, preto som vybral za najlepšiu hodnotu **počiatočnú teplotu 100**.

Ľubomír Dlhý, ID: 102897

GENETICKÝ ALGORITMUS

Genetický algoritmus je evolučný algoritmus, ktorý postupným vývojom konverguje k extrému. Aby sme zamedzili konvergencii k lokálnemu extrému a približovali sa ku globálnemu riešeniu, treba optimálne nastaviť kríženie, mutáciu a výber rodičov.

REPREZENTÁCIA ÚDAJOV

Mestá sú reprezentované dvojicou x,y (súradnicami). Všetky mestá sú uložené v poli. Sekvencia miest je daná poľom indexov jednotlivých miest.
Chromozóm je jedna sekvencia (permutácia) miest. Gény sú mestá (indexy miest). Populácia je pole chromozómov.

NASTAVITEĽNÉ PARAMETRE

- Veľkosť populácie
- Maximálny počet generácií — Predvolená hodnota je 300
- Miera mutácie — Pravdepodobnosť mutácie
- Výber rodičov — 1 - Tournament(2), 2 - Tournament(3), 3 - Roulette

ALGORITMUS

1. Vytvor počiatočnú populáciu náhodným generovaním chromozómov (permutácií) a ohodnoť každý chromozóm v populácii
2. Vyber metódou selekcie 2 rodičov
3. Skríž rodičov metódou kríženia a vytvor tak 2 potomkov
4. S pravdepodobnosťou mutácie urob mutáciu na chromozómoch
5. Vlož oboch potomkov do novej populácie
6. Ak nová populácia nie je naplnená potrebným počtom chromozómov, choď na krok 2, inak pokračuj
7. Ohodnoť chromozómy v populácii
8. Zvýš počet generácií o 1
9. Ak počet generácií prevýšil maximálny počet generácií, vyber najlepší chromozóm z populácie a ukonči, inak choď na krok 2

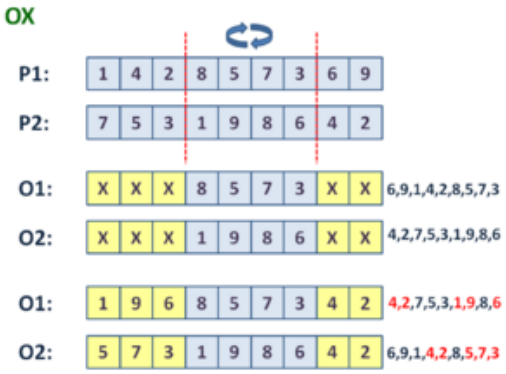
FITNESS FUNKCIA

Fitness funkcia je súčet vzdialeností medzi susednými mestami, teda veľkosť cesty.

KRÍŽENIE

Keďže chromozómy sú vlastne permutácie, kríženie musí dodržať podmienku toho, že jednotlivé gény sa budú v chromozóme nachádzať vždy iba raz. Existuje niekoľko metód kríženia, ktoré spĺňajú túto podmienku. Ja som použil Order Crossover (OX) – kríženie poradia.

Metóda vyberie 2 body, kde sa rodičia rozrežú. Takto dostaneme 3 časti. Prostrednú časť nechávame u potomka. Potom sa vytvorí pre každého rodiča permutácia, ktorá začína od druhého bodu rezu. Z tejto sekvencie sa odstránia prvky z prostrednej časti opačného potomka. Následne vzniknutá sekvencia sa doplní do potomka od druhého rezného bodu.



MUTÁCIA

Mutácia je iba jednoduchá výmena dvoch náhodne zvolených miest v danom chromozóme.

SELEKCIA

Používam 3 metódy selekcie: Turnaj (k=2), Turnaj (k=3), Ruleta

Ľubomír Dlhý, ID: 102897

V turnaji sa vyberie *k* náhodných prvkov z populácie a vyberie sa najlepší z nich (s najmenšou fitnes hodnotou)

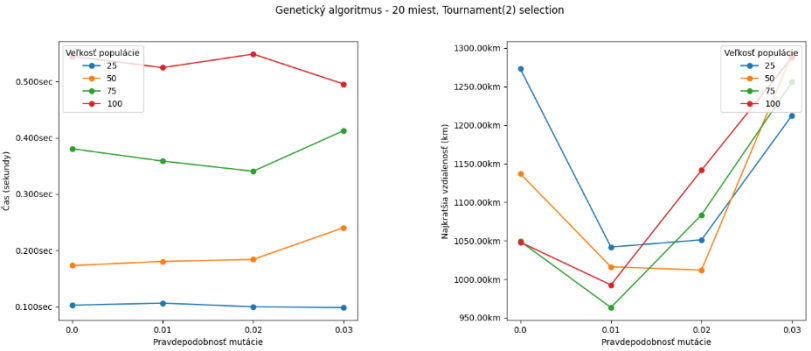
V rulete sa vygeneruje náhodné číslo v intervale [0, suma upravených fitness hodnôt). Postupne sa pripočítavajú upravené fitnes hodnoty chromozómov až pokiaľ táto suma neprevýši náhodne vygenerované číslo. Prvok, pri ktorom sa suma prevýši je vybraný. Fitnes hodnoty sa musia upraviť, keďže hľadáme globálne minimum a nie maximum. Upravené hodnoty sa rátajú nasledovne:

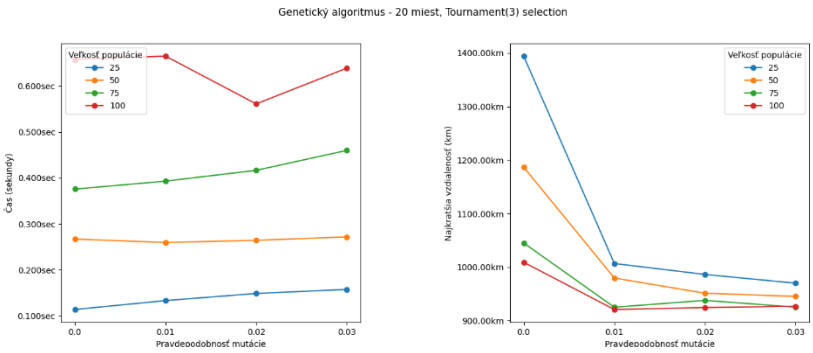
F' = maximálna fitness hodnota v populácii + 1 – fitness hodnota chromozómu

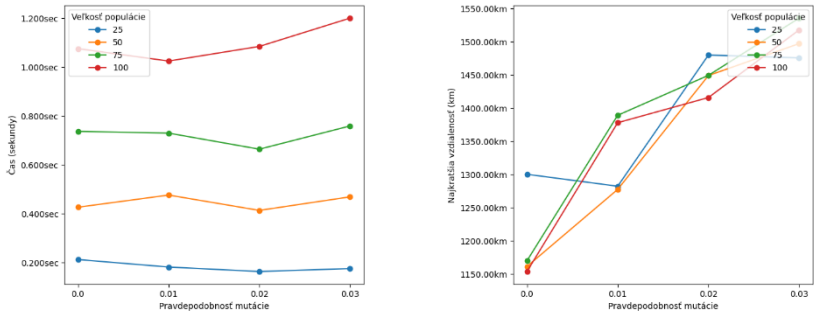
To spôsobí, že chromozóm s najmenšou fitnes hodnotou bude mať najväčšiu časť v rulete.

POROVNANIE PARAMETROV

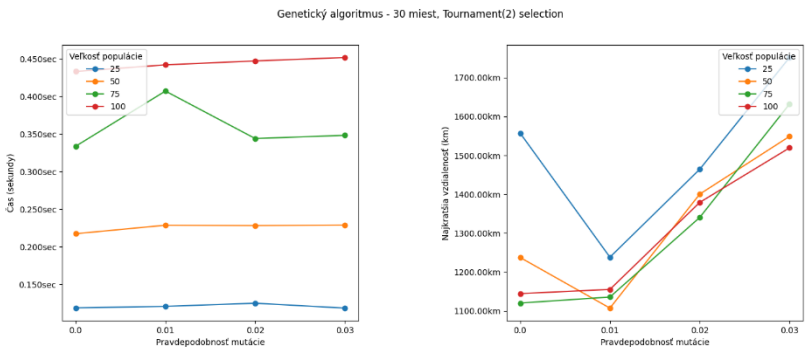
20 miest

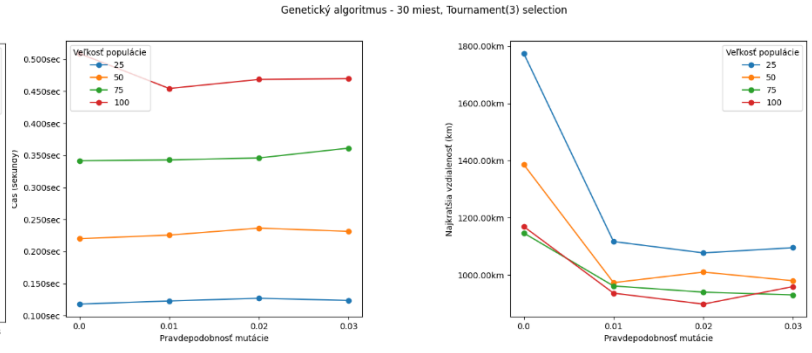


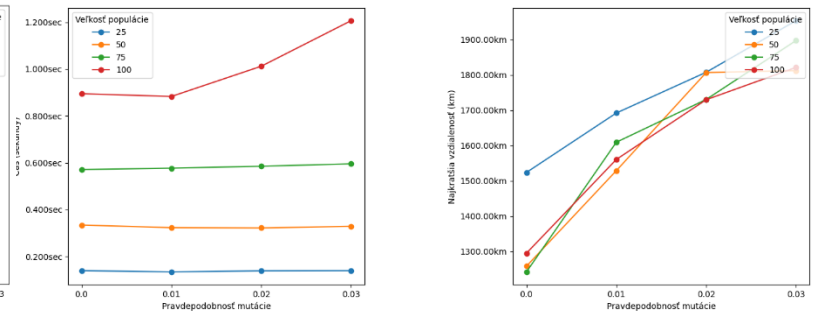




30 miest







Pre 20 miest nadobúda najlepšie hodnoty selekcia Tournament(3) s veľkosťou populácie 100 a 75 s pravdepodobnosťou mutácie 1%.

Tournament(2) je druhá najlepšia metóda výberu, veľkosť 75 a 100 prevažujú a mutácia 1% dosahuje najlepšie výsledky.

Ruleta je najhoršia. So zväčšujúcou sa mutáciou sa významne zhoršuje riešenie.

Pre 30 miest je najlepšia selekcia Tournament(3), veľkosť populácie 100 / 75 / 50 a pravdepodobnosť mutácie 1%.

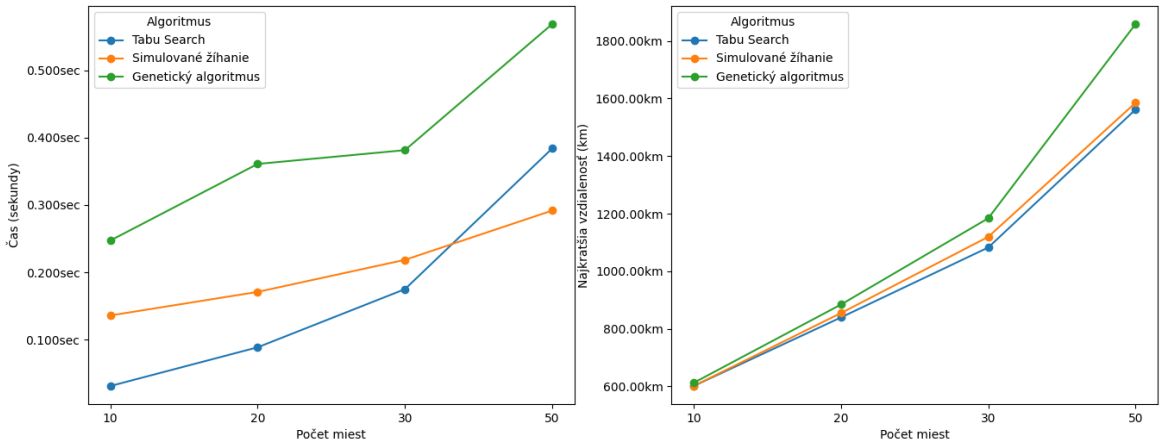
Tournament(2) je druhá najlepšia, kde sú opäť najlepšie 3 najväčšie veľkosti populácie a pravdepodobnosť mutácie 0% alebo 1%.

Ruleta je vo výsledkoch najhoršia a s rastúcou mutáciu sa zhoršuje.

Najlepšie výsledky dosahuje **Tournament(3), mutácia 1%** a **veľkosť populácie 50** vzhľadom na časovú náročnosť.

Ľubomír Dlhý, ID: 102897

POROVNANIE ALGORITMOV



Porovnanie algoritmov

Po testovaní jednotlivých algoritmov a nájdení optimálnych nastavení z hľadiska riešenia i časovej efektivity som urobil porovnanie algoritmov. Algoritmy boli testované na rovnakých mapách a na vzorke 1000 testov.
Tabu Search nachádza najlepšie riešenia, no má najprudší časový nárast a dá sa očakávať, že so zvyšujúcim počtom miest sa bude výrazne spomalovať. Je to spôsobené najmä tým, že prehľadávanie nasledovníkov je lineárne závislé od počtu miest. Druhé najlepšie je Simulované žíhanie s veľmi tesným rozdielom. Simulované žíhanie nemá veľmi prudký časový rast v závislosti od počtu miest. Najhoršie výsledky podal genetický algoritmus najmä pri veľkom počte miest (50), taktiež je aj najviac časovo náročné.

POUŽÍVATEĽSKÉ ROZHRANIE

Po spustení programu sa v konzoli otvorí interaktívne menu, kde máme možnosť použiť buď predvolenú mapu zo zadania alebo náhodnú mapu. Taktiež máme možnosť nastaviť vlastné parametre alebo použiť prednastavené.

```
 Urcte akciu:
 (1) Vzorova mapa s predvolenymi parametrami
 (2) Vzorova mapa s uzivatelom urcenymi parametrami
 (3) Nahodna mapa s predvolenymi parametrami
 (4) Nahodna mapa s uzivatelom urcenymi parametrami
```

Ak užívateľ zvolí určiť parametre, tak konzolová aplikácia bude postupne žiadať všetky potrebné parametre.

```
 > Zadajte pocet miest a rozmery mapy:
25 300 300
 ~ Zadajte parametre pre Tabu Search ~
  >> Koeficient velkosti tabu listu: 1
  >> Pocet iteracii: 100
 ~ Zadajte parametre pre Simulovane zihanie ~
  >> Pociatocna teplota: 10
  >> Miera ochladzovania: 0.995
 ~ Zadajte parametre pre Geneticky algoritmus ~
  >> Velkost populacie: 100
  >> Pravdepodobnost mutacie: 0.01
  >> Selekcia 1 - Tournament(2), 2 - Tournament(3), 3 - Roulette: 2
  >> Pocet generacii: 300
```

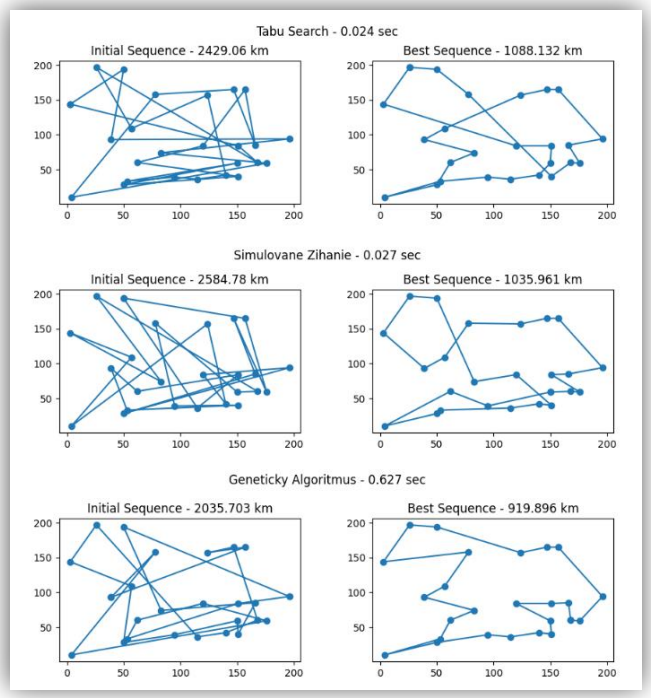
```
 > Chcete vizualizaciu vysledkov na grafe? (Y/N)
Y
 > Chcete animaciu na grafe? (Y/N)
N
```

Na konci sa ešte aplikácia spýta používateľa, či chce riešenie ukázať na grafe, prípadne v animácii.

Ľubomír Dlhý, ID: 102897

Riešenie sa zobrazí v konzoli, kde sú informácie o čase a o vzdialenosti nájdeného riešenia.

```
Vysledky pre [ 25 ] miest:
 > Tabu Search          Trvanie: 0.0240sec, Vzdielenost: 1088.132km
 > Simulovane Zihanie   Trvanie: 0.0270sec, Vzdielenost: 1035.961km
 > Geneticky Algoritmus Trvanie: 0.6270sec, Vzdielenost: 919.896km
```



Ak užívateľ zvolil zobraziť vizualizáciu, tak sa zobrazia grafy, kde sa nachádza prvá permutácia a posledná permutácia spolu s časmi a vzdialenosťami. V prípade animácie sa zobrazí postupný vývoj z prvej sekvencie až po poslednú.

Ľubomír Dlhý, ID: 102897

ZHODNOTENIE

Problém obchodného cestujúceho som vyriešil 3 algoritmami, 2 na lokálne prehľadávanie (Tabu Search, Simulované žihanie) s optimalizáciou a genetickým programovaním (genetický algoritmus). Každému algoritmu som sa snažil nastaviť parametre tak, aby bol pomer dosiahnutého riešenia, t.j. výslednej cesty a potrebného času čo najlepší. Preto som nezvolil najlepšie hodnoty, ktoré som zistil v porovnaní, pretože boli časovo náročnejšie. Pre jednotlivé algoritmy som určil parametre takto:

- Tabu Search koeficient tabu listu – 1, počet iterácií – 500
- Simulované žíhanie počiatočná teplota – 100, miera ochladzovania – 0.999
- Genetický algoritmus veľkosť populácie – 50, miera mutácie – 1%, výber rodičov – Tournament(3)

V porovnaní algoritmov mi vyšlo, že Tabu Search je najlepší čo sa týka riešení aj časovej náročnosti, no má najvýraznejší časový rast, takže pri veľkých počtoch miest sa môže zhoršovať. Simulované žíhanie je výsledkami veľmi tesne za Tabu Search, no časovo je náročnejšie, ale nemá výrazný časový nárast s vzrastajúcim počtom miest. Genetický algoritmus vyšiel z 3 algoritmov najhoršie časovo i riešením, čo by sa však dalo napraviť lepším nastavením parametrov, keďže nastaviteľných parametrov je v tomto prípade najviac.

Program je implementovaný v jazyku Python, verzia 3.8.5 a vyžaduje knižnice:

- random - generovanie náhodných čísel
- time - meranie trvania algoritmov
- math - práca s matematickými funkciami
- matplotlib - grafická vizualizácia na grafoch