Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Umelá inteligencia

Zadanie 2

Problém obchodného cestujúceho

Sližik Ján

Vyučujúci: Mgr. Lukáš Kohútka, PhD. Dátum: 7.11.2022

Obsah

1. Úvod....3
2. Genetický algoritmus....3
2.1. Algoritmy výberu rodičov....3
2.2. Kríženie....4
2.3. Mutácia....4
3. Zakázané prehľadávanie....4
4. Reprezentácia údajov....4
4.1. Genetický algoritmus....4
4.2. Zakázané prehľadávanie....5
4.3. Obmedzenie riešenia....5
5. Testovanie....5
5.1. Testovanie Genetického algoritmu....5
5.2. Porovnanie Ruletového výberu a Turnajového výberu....8
5.3. Testovanie Zakázaného prehľadávania....9
5.4. Porovnanie rôznych veľkostí tabu listu....11
6. Zhodnotenie....12
7. Možná optimalizácia....12
Návod na spustenie....12
Zdroje....12

1. Úvod

Cieľom tohto zadania je za využitia metaheuristických metód genetického algoritmu a zakázaného prehľadávania prísť na čo najlepšie možné riešenie známeho problému obchodného cestujúceho. Obchodný cestujúci sa snaží navštíviť všetky mestá na svojej ceste iba raz s tým, že prejde čo najkratšiu cestu. Keďže počet možných ciest je úmerný všetkým permutáciám miest, náročnosť tohto problému, keby sme ho chceli riešiť brute force je pre x rovné počtu bodov O(x!). Pre tento problém neexistuje žiadna konečná podmienka, teda metaheuristické metódy sú oklieštené iba počiatočnými nastaveniami užívateľa a môžu, no aj nemusia nájsť najoptimálnejšie riešenie, no podstatné je, že sú oveľa efektívnejšie ako brute force.

2. Genetický algoritmus

Tento algoritmus inšpirovaný prírodou využíva náhodu a prirodzenú selekciu na to, aby sa dopracoval k čo najlepšiemu jedincovi. V prípade obchodného cestujúceho je jedincom permutácia bodov reprezentujúca cestu. Fitnes funkcia každého jedinca je 1/dĺžka cesty, takže čím kratšia cesta tým silnejší jedinec. Prvá generácia sa vytvorí z čisto náhodných jedincov do každej ďalšej sa posúva istý počet najlepších jedincov, náhodných jedincov a zvyšok detí rodičov z predošlej generácie. Počet jedincov v každej populácií je rovnaký a vždy keď sa nejaký jedinec posúva do ďalšej generácie má v mojej implementácií 10% šancu, že sa zmutuje. Tento proces sa opakuje dokým nie je dosiahnutý zadaný počet generácií. Na to, aby sa zabezpečilo, že sa nepríde o najlepšieho jedinca z dôsledku mutácie pamätám si separátne jeho kópiu, tým, že sa mutujú aj najlepší jedinci je šanca, že sa nájde ešte lepšia cesta.

2.1. Algoritmy výberu rodičov

Dva spôsoby výberu rodičov, ktoré som sa rozhodol implementovať sú ruletový a turnajový výber, súčasťou mojej dokumentácie je aj ich porovnanie.

Turnajový výber zoberie vždy 10% náhodných jedincov z predošlej populácie a z nich vyberie najlepšieho jedinca, ktorý sa stane prvým rodičom. Proces sa opakuje rovnako pre druhého rodiča.

Ruletový výber pridelí každému jedincovi pravdepodobnosť úmernú jeho fitness, takže silnejší jedinci majú vyššiu šancu na to, aby sa stali rodičmi, no existuje stále šanca, aby sa aj slabší jedinci stali rodičmi. Spôsob akým dosiahnem ruletový výber je, že pred tým ako začnem pridávať deti vypočítam súčet fitness predošlej generácie a pri každom volaní hľadania rodiča ruletovým výberom si vygenerujem náhodné číslo pomocou random.uniform <0,max), prechádzam jedincami predošlej generácie a ak súčet je vyšší ako výber, tak returnem jedinca na danom indexe.

2.2. Kríženie

Spôsob pre ktorý som sa rozhodol zabezpečuje, aby sa body v ceste nového jedinca nachádzali iba raz. Najskôr vyjmem náhodnú časť z prvého rodiča a zvyšok doplním takým spôsobom, že prechádzam druhým jedincom a ak sa bod ešte nenachádza v ceste nového jedinca pridám ho do cesty dieťaťa.

2.3. Mutácia

Mutáciu aplikujem na všetkých jedincov okrem náhodných. Napriek tomu, že by sa mohlo javiť, že jej pravdepodobnosť je nízka, iba 10%, môže mať veľmi pozitívny efekt. Spôsob ako mutujem je, že vygenerujem náhodný index z cesty vymením ho s nasledujúcim.

3. Zakázané prehľadávanie

Tento metaheuristický algoritmus je vylepšením známeho horolezeckého algoritmu. Horolezecký algoritmus využíva seba optimalizáciu tým, že prehľadá všetkých susedov terajšieho riešenia a vyberie si najlepšieho nasledovníka, tento algoritmus je veľmi efektívny pokiaľ prehľadávaná krivka môže iba stúpať, avšak pre problém obchodného cestujúceho je možné dostať sa do lokálneho maxima v ktorom sa horolezecký algoritmus zacyklí. Na to aby sa nezacyklil existuje takzvaný tabu list do ktorého si vždy uloží lokálne maximum, tj. cestu jedinca ktorého susedia majú iba nižšie ohodnotenie ako on. Pokiaľ sa maximálna veľkosť tabu listu prekročí, tak sa vyhodí prvý prvok v liste a na koniec pridá nový. Na to aby tento algoritmus fungoval efektívne je veľmi dôležité správne určiť maximálnu veľkosť tabu listu. Pokiaľ je veľkosť tabu listu príliš malá zacyklí sa a pokiaľ je príliš veľká je možné, že si všetky možné pohyby zakáže a nebude môcť pokračovať, alebo počítanie riešenia sa príliš natiahne.

Spôsob akým prehľadávam všetkých možných susedov jedinca je veľmi jednoduchý, prechádzam jeho cestou a vymieňam vždy terajší index so všetkými nasledujúcim. Pôvodne som vymieňal iba vedľajšie indexy, no taký spôsob prehľadávania susedov oveľa neefektívnejší. Následne zosortenými susedmi vždy prechádzam a pokiaľ sa sused nenachádza v tabu stane sa terajším prvkom.

4. Reprezentácia údajov

4.1. Genetický algoritmus

Počas behu programu si držím v pamäti vždy momentálnu populáciu a najlepšieho jedinca. Na to aby som vedel poskytnúť analýzu si pamätám aj list fitness najlepšieho jedinca v populácií, čo ale nie je bežnou súčasťou tohto algoritmu. Pokiaľ sa užívateľ rozhodne pre veľkú populáciu pamäťové nároky programu budú väčšie, no algoritmus má šancu, že sa k dobrému riešeniu dopracuje v menej iteráciách. Naopak pokiaľ je jedincov v populácií menej bude program

potrebovať viacej iterácií. Na to aby tento algoritmus fungoval správne je veľmi dôležité si posielať vždy do ďalšej generácie minimálne top 10% najlepších jedincov.

4.2. Zakázané prehľadávanie

Tento algoritmus má vo všeobecnosti oveľa lepšie pamäťové nároky ako Genetický algoritmus, keďže si pamätá iba najlepšieho jedinca, terajšieho jedinca a jeho potenciálnych susedov. Čisto kvôli analýze si udržiavam v pamäti aj vývoj najlepšieho prvku a list hodnôt fitnes jedincov pre každú iteráciu.

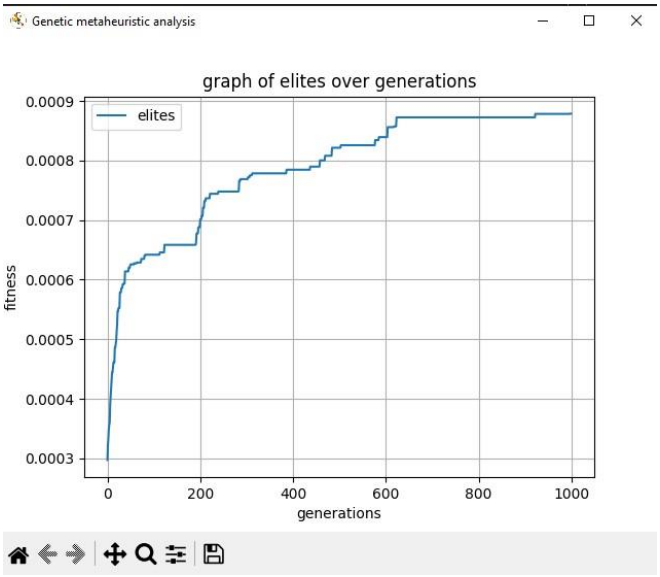
4.3. Obmedzenie riešenia

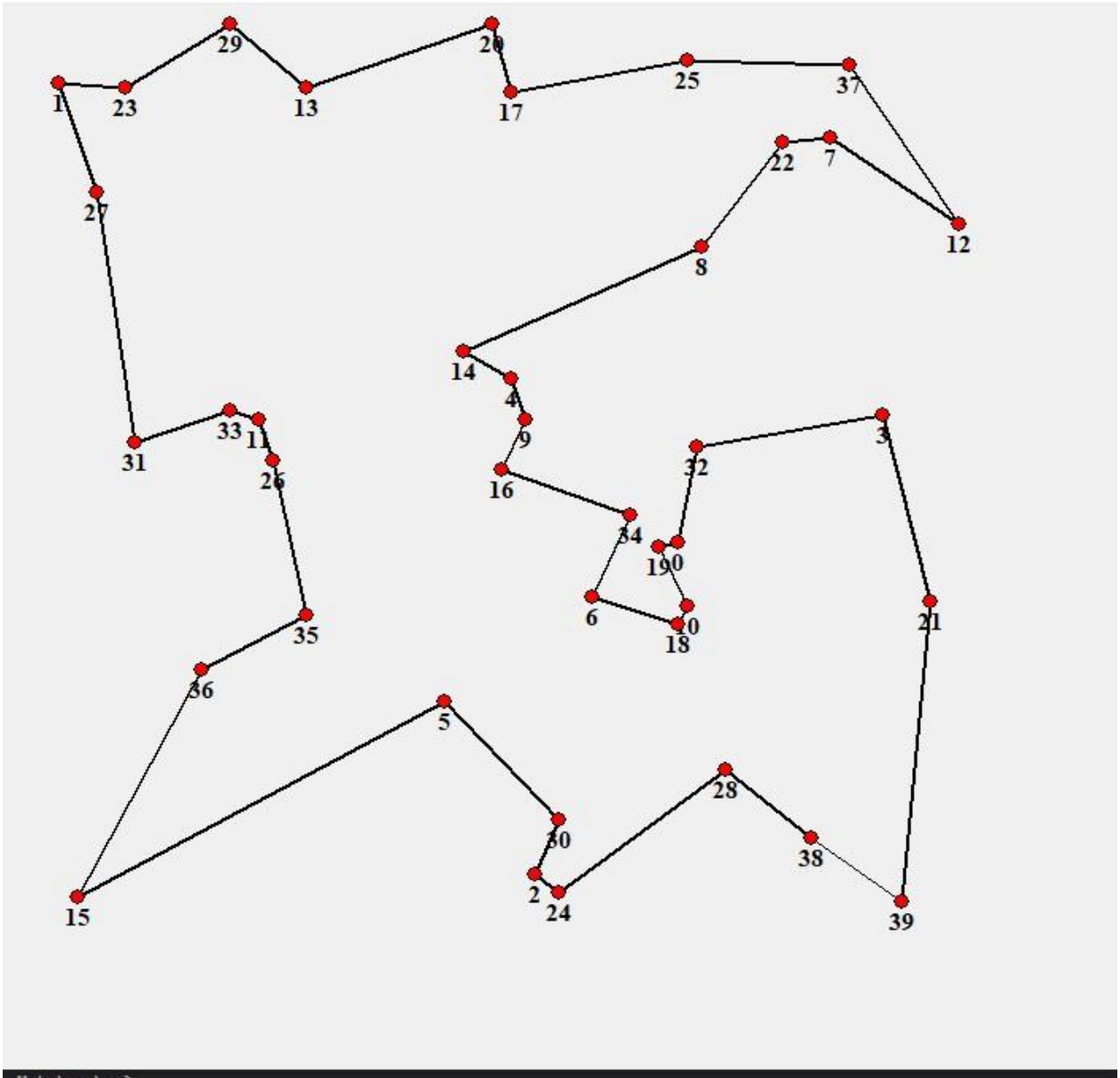
Moje riešenie je optimalizované pre rozmer mapy 200 x 200, každý bod má iba 2D súradnice a predpokladám, že miest bude podľa zadania najviac 40, takže sa mi oplatí na začiatku vypočítať všetky vzdialenosti dvojíc bodov a uložiť si ich v pamäti do hashovacej tabuľky, zisťovanie veľkosti cesty je potom oveľa rýchlejšie ako počítanie jej vzdialenosti vždy nanovo pri obidvoch implementovaných algoritmoch.

5. Testovanie

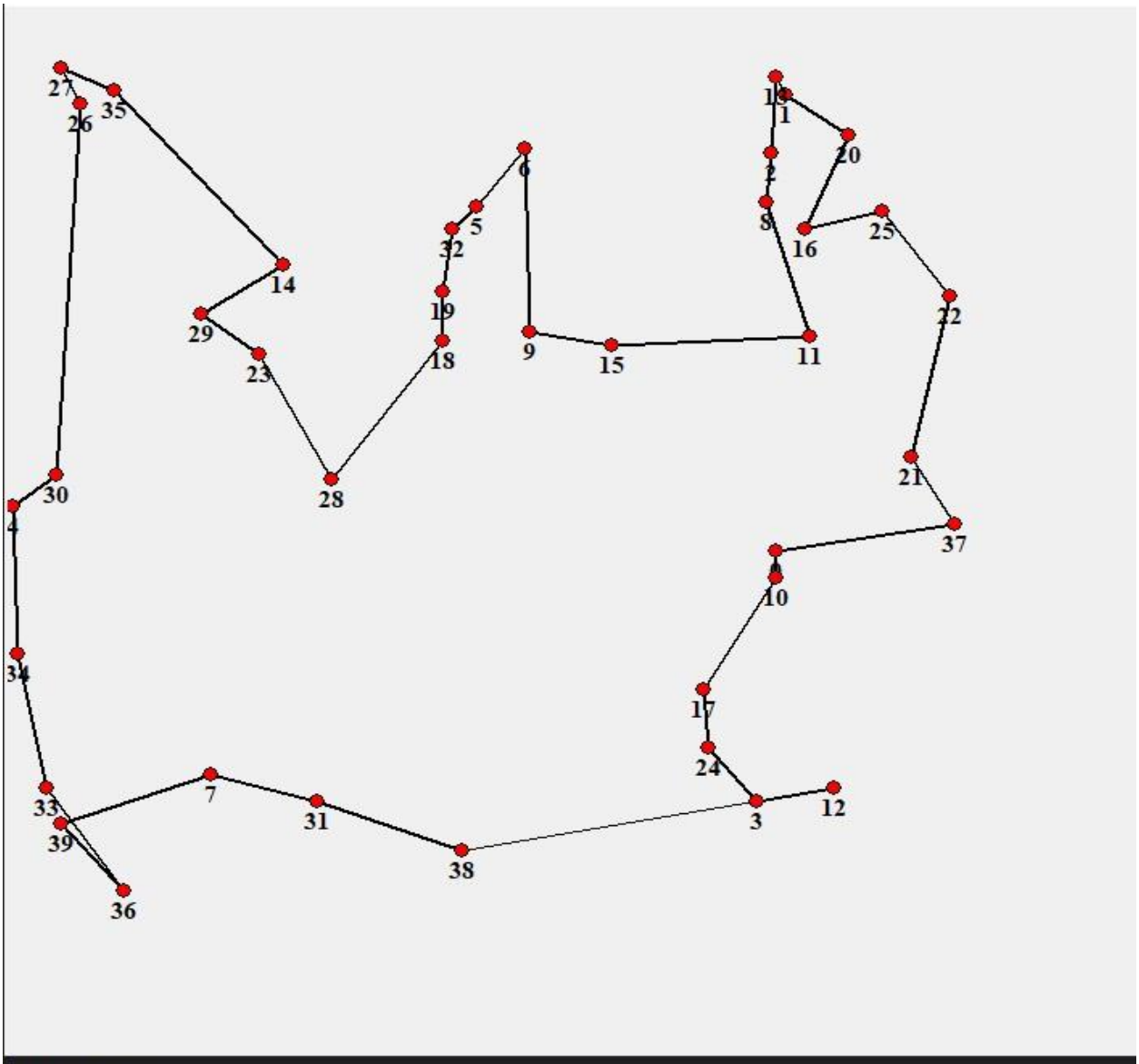
5.1. Testovanie Genetického algoritmu

Môj program poskytuje jednoduchý všeobecný test na genetický algoritmus. Užívateľ má možnosť vybrať si počet náhodne vygenerovaných bodov a pokiaľ nechce pokračovať s predvolenými parametrami nastaviť si ich po svojom, ako napríklad veľkosť populácie, počet generácií, elít, náhodných jedincov, môže si vybrať aj či chce aplikovať mutácie, turnajový, či ruletový výber. S dobrými základnými parametrami sa mi podarilo nájsť suboptimálnu cestu pre 40 bodov oba spôsoby výberu rodičov. Tento spôsob testovania poskytuje aj jednoduchú analýzu formou grafu, ktorá zobrazuje vývoj najlepšieho jedinca v priebehu generácií.






```
Mutate y\n ?
y
Tournament or Roulette t\r ?
t
Do you wish to change basic options y\n ?
Individuals in a population 100, Number of generations 1000, Elites 10, Randoms 0
n
number of points: 40
Time 8.359312057495117s
Distance 1119.8934045095123
```

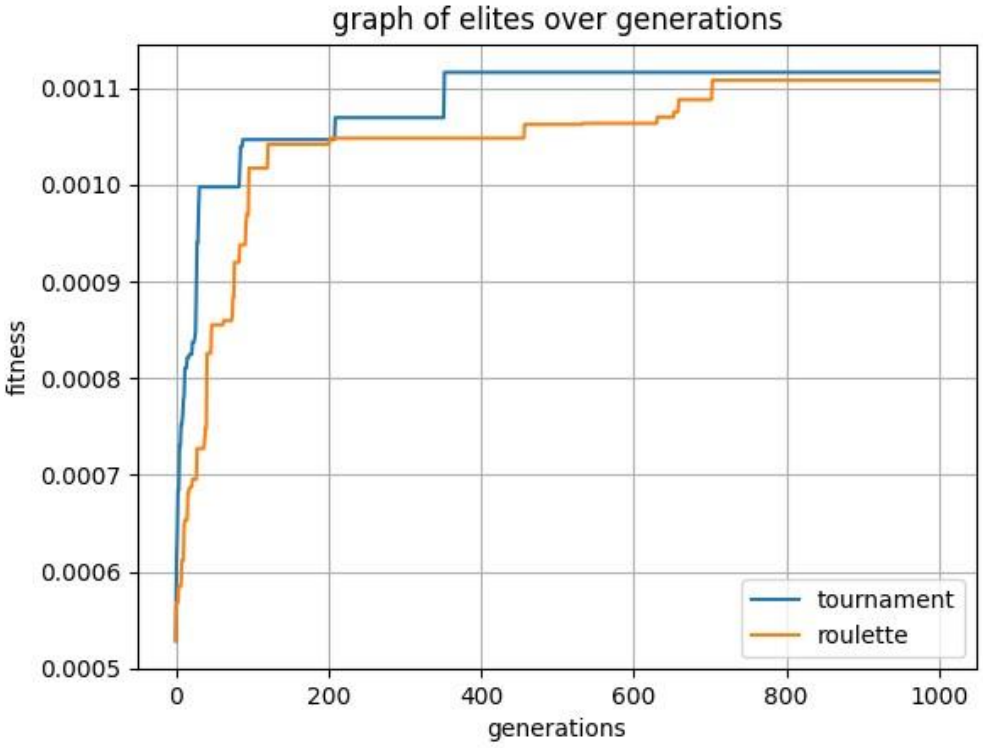



```
Mutate y\n ?
y
Tournament or Roulette t\r ?
r
Do you wish to change basic options y\n ?
Individuals in a population 100, Number of generations 1000, Elites 10, Randoms 0
n
number of points: 40
Time 7.789004325866699s
Distance 1013.4895319669016
```

5.2. Porovnanie Ruletového výberu a Turnajového výberu

Na to, aby podmienky obidvoch algoritmov boli rovnaké rozhodol som sa pre porovnanie použiť súradnice 20 bodov, ktoré máme poskytnuté v zadaní, taktiež si ukladám prvú náhodne vygenerovanú populáciu a teda obidva spôsoby výberu rodičov začínajú na rovnakom bode. Keďže som si vedomý ako veľmi záleží na implementácií týchto algoritmov, aj parametre ako počet populácií, jedincov, či šancu mutácie majú presne rovnakú.

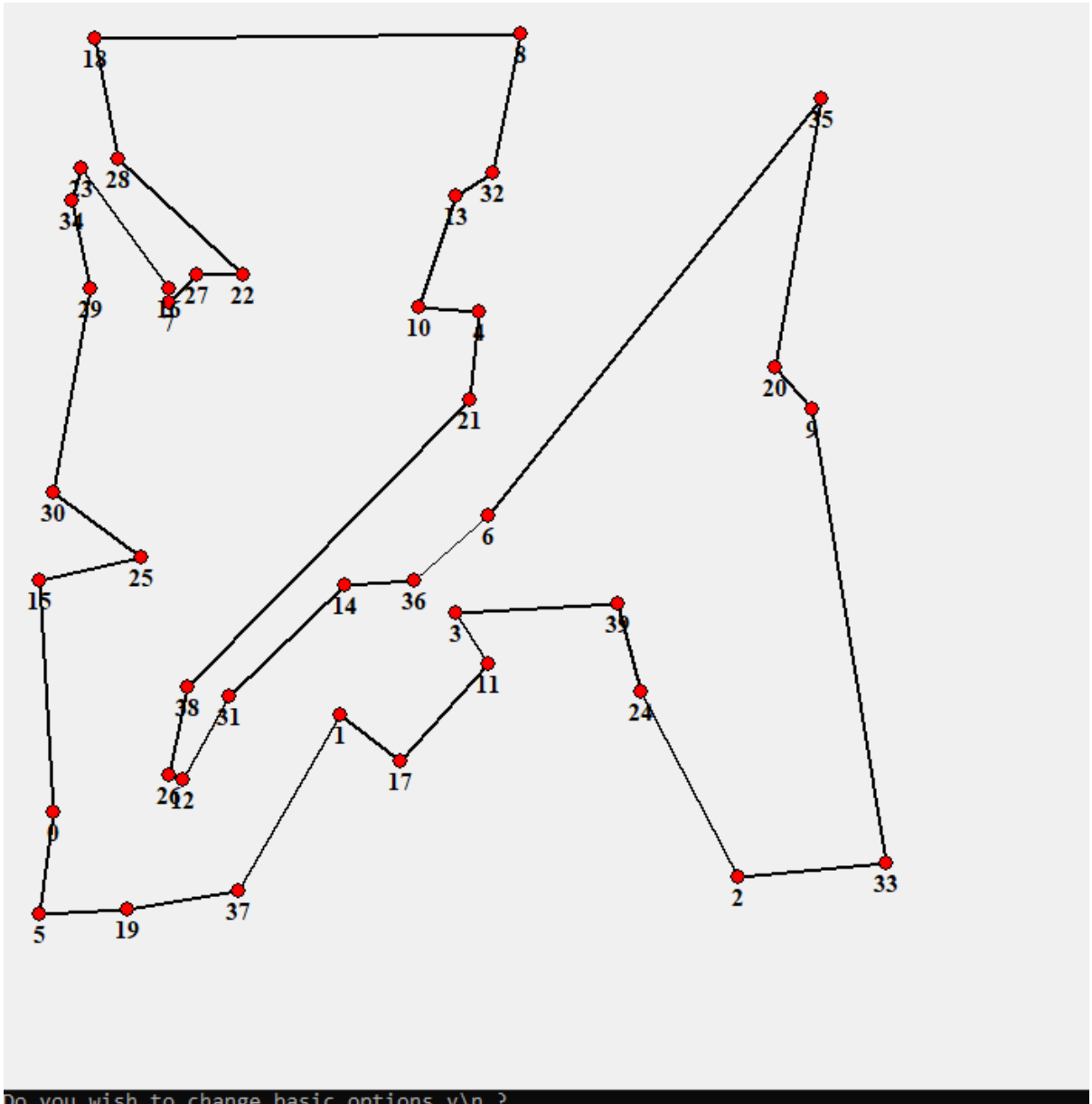
Z viacerých spustení sa mi podarilo dôjsť k takému záveru, že moja implementácia Tournament má vyššiu šancu nájsť lepšie riešenie ako Roulette, ale trvá o čosi dlhšie. Napriek týmto zanedbateľným rozdielom sa mi pomocou oboch podarilo niekoľkokrát nájsť najoptimálnejšie riešenie pre tieto body, takže dalo by sa skonštatovať, že obe implementácie sú pre tento problém efektívne.



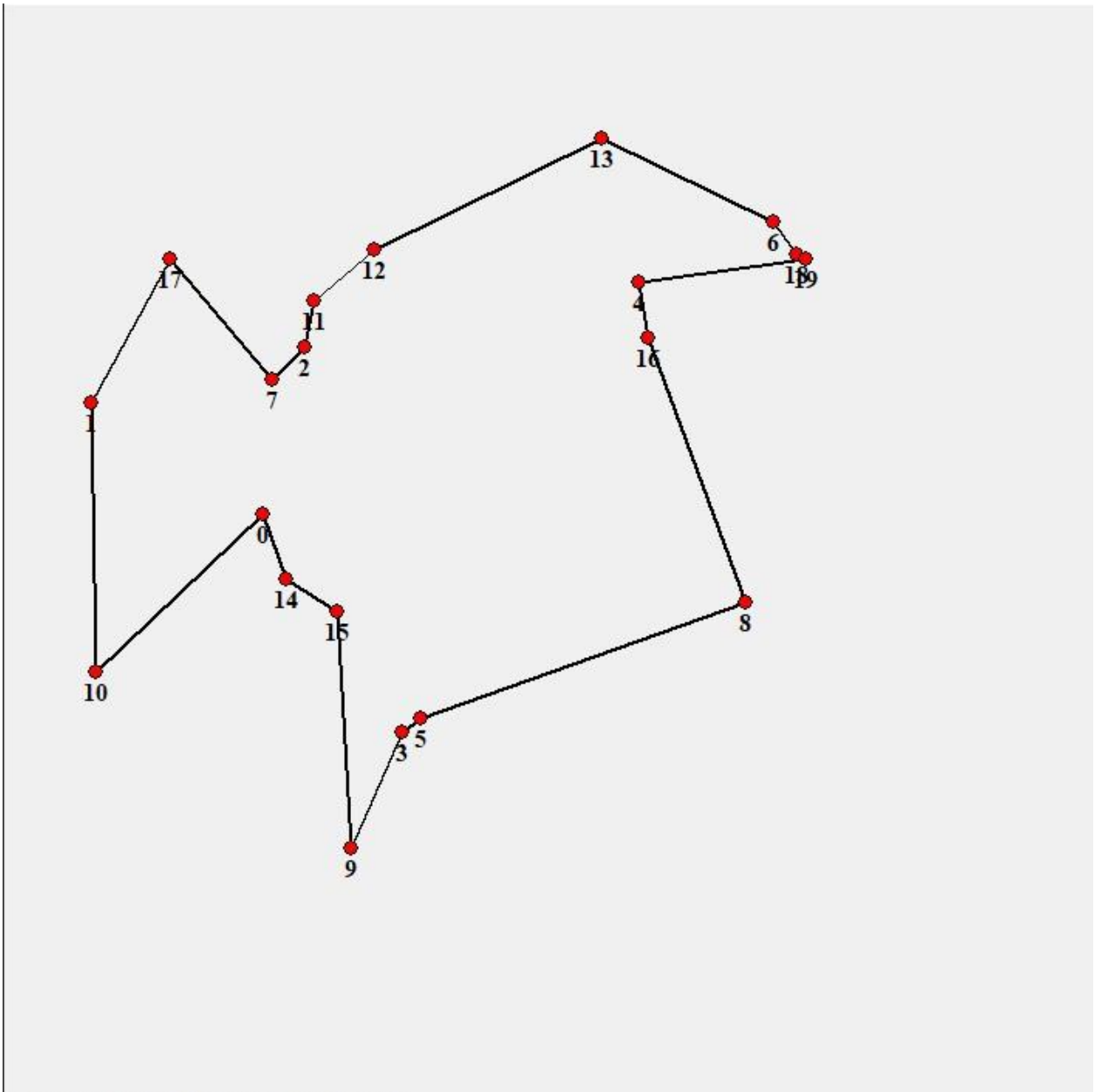

```
Tournament
Time 4.1968142986297615
Distance 895.7063250228808
Best Route [0, 2, 5, 9, 7, 3, 1, 6, 10, 13, 16, 19, 12, 15, 18, 17, 14, 11, 8, 4]
Roulette
Time 3.0331482887268066s
Distance 902.3833357313246
Best Route [10, 13, 16, 19, 12, 15, 18, 17, 14, 11, 8, 4, 0, 5, 2, 3, 1, 6, 7, 9]
```

5.3. Testovanie Zakázaného prehľadávania

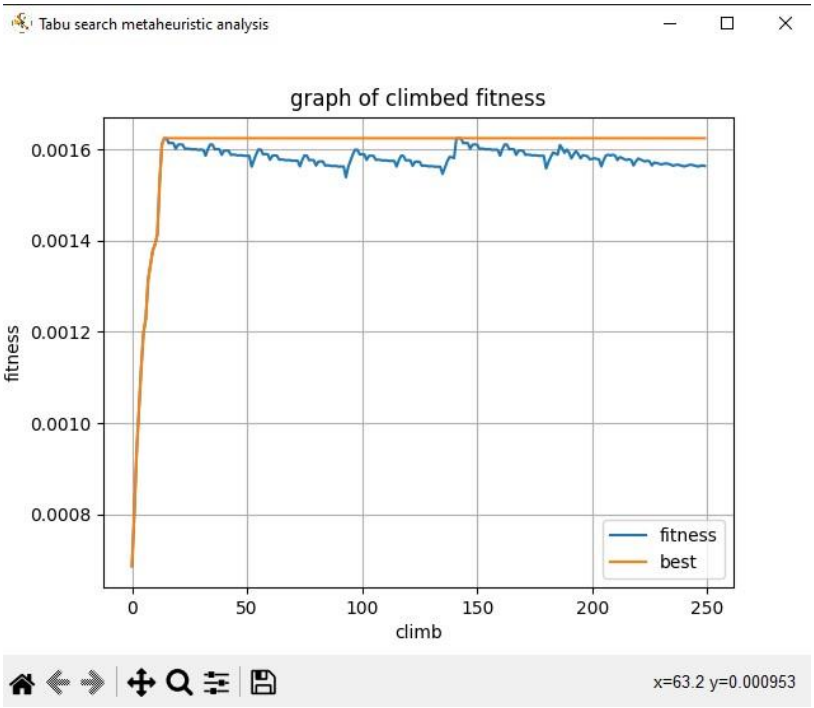
Pri testovaní užívateľ môže zmeniť základné parametre tohto algoritmu pokiaľ nie je spokojný s predvolenými. Môže zmeniť počet iterácií a veľkosť tabu listu. V porovnaní s Genetickým algoritmom je zakázané prehľadávanie schopné nájsť dobré riešenie oveľa rýchlejšie a spoľahlivejšie pre menší počet bodov avšak ak je tých bodov viac veľmi záleží na nastaveniach základných parametrov a časová náročnosť sa prirodzene zväčšuje. Môj základný test poskytuje aj grafové znázornenie vývoja hodnoty fitness momentálneho jedinca a zmeny najlepšej cesty.




```
Do you wish to change basic options y\n ?
Number of climbs 250, Tabu list size 50
n
number of points: 40
Time 34.931981086730965
Distance 1259.9038831722798
```

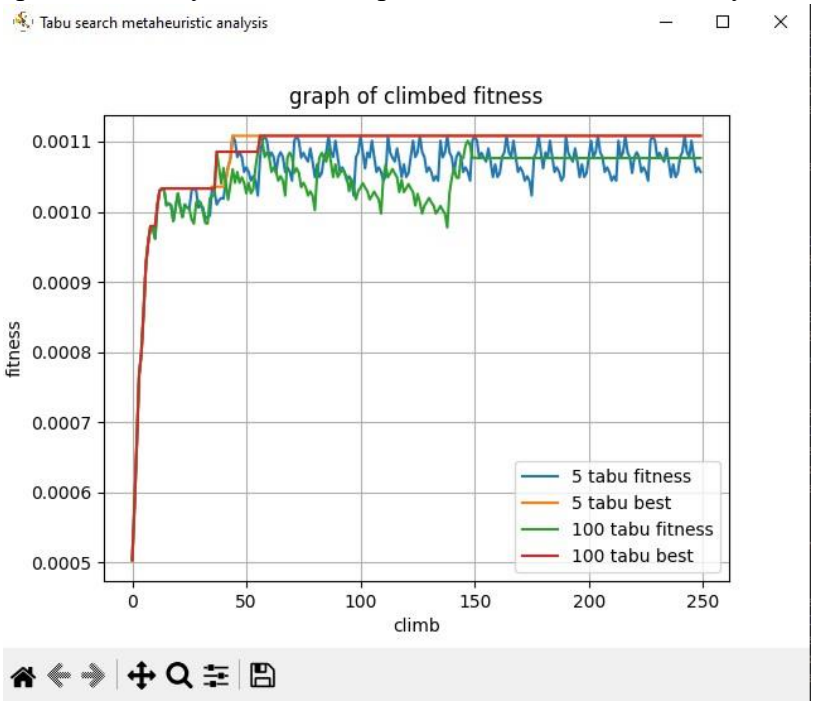



```
Do you wish to change basic options y\n ?
Number of climbs 250, Tabu list size 50
n
number of points: 20
Time 1.814065694808896s
Distance 615.5061276177053
```



5.4. Porovnanie rôznych veľkostí tabu listu

Pri základnom testovaní som si uvedomil akú veľmi dôležitú úlohu hrá veľkosť tabu listu. Preto môj program poskytuje možnosť porovnania v grafe dvoch možných veľkostí tabu listov. Na základe pozorovania je možné povedať, že na to aby tento algoritmus fungoval správne veľkosť tabu listu by sa mala zväčšovať spolu s počtom bodov. Na grafe možno pozorovať, čo sa stane ak je dĺžka tabu listu prikrátka, zacyklenie, alebo pridlhá, zablokovanie všetkých možných posunov.



6. Zhodnotenie

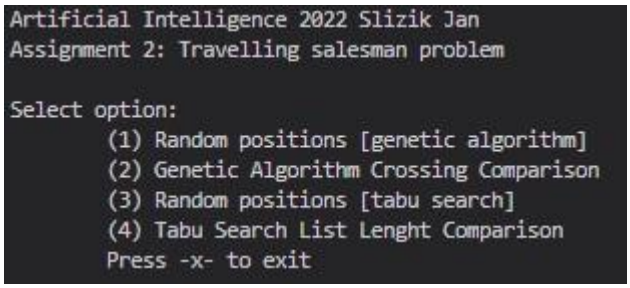
Problém obchodného cestujúceho má mnoho spôsobov riešenia a v tejto práci som sa pokúsil priblížiť fungovanie niektorých z nich.V priebehu testovaní Genetického algoritmu som došiel k záveru, že najlepší pomer tvorby novej generácie je počet elít, 10% populácie, rovnako tak aj náhodných jedincov, 0-10% a zvyšok jedincov je najlepšie keď je výsledkom kríženia, pri všetkých prvkoch sa mi osvedčilo aplikovať 10% mutáciu. Z výberov rodičov sa mi podarilo najviac zoptimalizovať turnajový výber, ktorý má pre mňa vyššiu šancu, že nájde optimálnu cestu ako Ruletový, ktorý má však naopak lepšiu časovú efektivitu. Moja implementácia zakázaného pokiaľ má dobre nastavené základné parametre v rátane veľkosti tabu listu, ktorý musí byť primerane veľký pre daný počet bodov je schopná spoľahlivo nájsť subotimálne riešenie s nízkym využitím pamäte a pokiaľ sa jedná aj o nižší počet bodov vo veľmi rýchlom čase.

7. Možná optimalizácia

Na to aby som uľahčil užívateľovi čo najviac prácu s mojím programom mohol by som vypočítať najefektívnejšie vstupné parametre pre dané počty miest, no toto nebolo predmetom tohto zadania a voľba je preto ponechaná na užívateľa. Taktiež by môj program nemusel byť iba konzolovou aplikáciou a určite by bolo krajšie keby sa jednalo o aplikáciu s GUI, no verím, že moja implementácia je postačujúca.

Návod na spustenie

Aby môj program fungoval správne je odporúčané aby užívateľ skontroloval svoje prostredie, Python 3.10.2. Pre bežného užívateľa poskytujem vo svojom zadaní aj run.bat file obsahujúci python main.py. Všetky vstupy sú ošetrené, takže by sa nemalo stať, že by užívateľovi program len tak spadol. Program je bezpečne ukončený zvolením voľby -x-.



Zdroje

NÁVRAT, P. -- BIELIKOVÁ, M. -- BEŇUŠKOVÁ, Ľ. -- KAPUSTÍK, I. -- UNGER, M. Umelá inteligencia. Bratislava : STU v Bratislave, 2002. 393 s. ISBN 80-227-1645-6.
https://beta.openai.com/playground
https://www.goodrequest.com/blog/algoritmy-umelej-inteligencie-4-geneticke-algoritmy
https://www.goodrequest.com/blog/algoritmy-umelej-inteligencie-tabusearch