

Metaheuristiky

Semestrálna práca

Vypracoval: Tomáš Staroň

Študijná skupina: 5ZII11

Obsah

[1. Úvod 3](#_Toc196250437)

[2. Popis aplikácie 3](#_Toc196250438)

[3. Architektúra aplikácie 4](#_Toc196250439)

[3.1 Hlavné komponenty 4](#_Toc196250440)

[3.2 Kľúčové služby 4](#_Toc196250441)

[4. Editor dopravnej siete 4](#_Toc196250442)

[4.1 Práca s uzlami 4](#_Toc196250443)

[4.2 Práca s hranami 5](#_Toc196250444)

[5. Import a export dát 5](#_Toc196250445)

[6. Generovanie náhodnej siete 6](#_Toc196250446)

[7. Clarke-Wright algoritmus 6](#_Toc196250447)

[7.2 Princíp algoritmu 6](#_Toc196250448)

[7.3 Implementácia 6](#_Toc196250449)

[8. Zobrazenie výsledkov 7](#_Toc196250450)

[9. Adresárová štruktúra projektu 8](#_Toc196250451)

[10. Kľúčové algoritmy a údajové štruktúry 8](#_Toc196250452)

[11. Používateľská príručka 9](#_Toc196250453)

[11.1 Vytvorenie dopravnej siete 9](#_Toc196250454)

[11.2 Editácia vlastností 9](#_Toc196250455)

[11.3 Práca so súbormi 10](#_Toc196250456)

[11.4 Generovanie náhodnej siete: 10](#_Toc196250457)

[11.5 Výpočet optimálnych trás 10](#_Toc196250458)

[12. Záver 10](#_Toc196250459)

# Úvod

Semestrálna práca sa zameriava na riešenie úlohy obchodného cestujúceho (TSP) pomocou genetického algoritmu. Implementovaný algoritmus využíva elitnú stratégiu, kríženie typu Uniform Crossover (UX) a selekciu Baker's Stochastic Universal Selection (SUS), ktoré sú vhodné pre permutačné problémy ako TSP. Cieľom je nájsť čo najkratšiu trasu medzi mestami v Košickom kraji efektívne počas evolučného procesu.

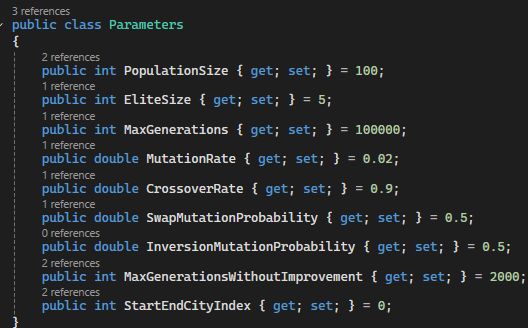
# Implementácia riešenia

V tejto kapitole si prejdeme implementáciu riešenia problému.

## Pomocné triedy

Trieda **Parameters:**

nastavuje parametre genetického algoritmu, ktoré sa používajú počas behu algoritmu.



Obrázok 1Parametre algoritmus v triede Parameters

Trieda **City** – ukladá informácie o danom meste ako je jeho ID, názov a kód.

Trieda **DataLoader –** Načíta vstupné údaje z textových súborov.

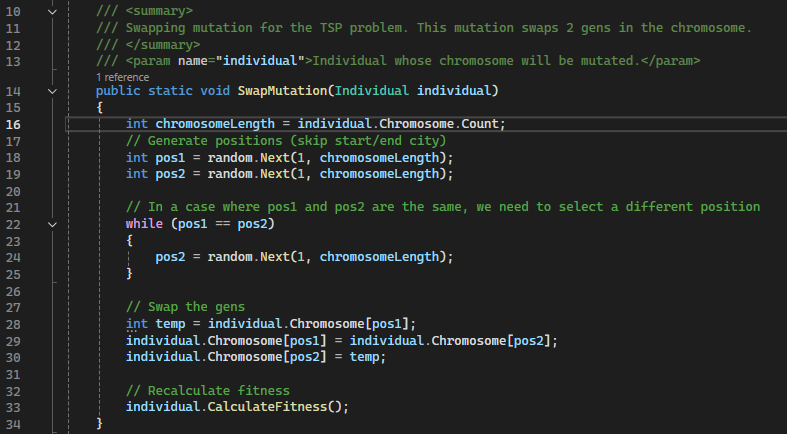
Trieda **Individual –** Ukladá chromozóm jedinca, fitness a taktiež pozná vzdialenosti medzi uzlami siete. Disponuje metódami Clone, CalculateFitness a ToString.

Trieda **Population –** Ukladá danú populáciu jedincov pre genetický algoritmus. Pamätá si najlepšieho jedinca s najlepším fitnessom. Disponuje metódami GetElite(vráti zoznam najlepších n jedincov podľa fitnessu), SortByFitness, AddIndividual a GetStatistics

## 2.2 Mutácia

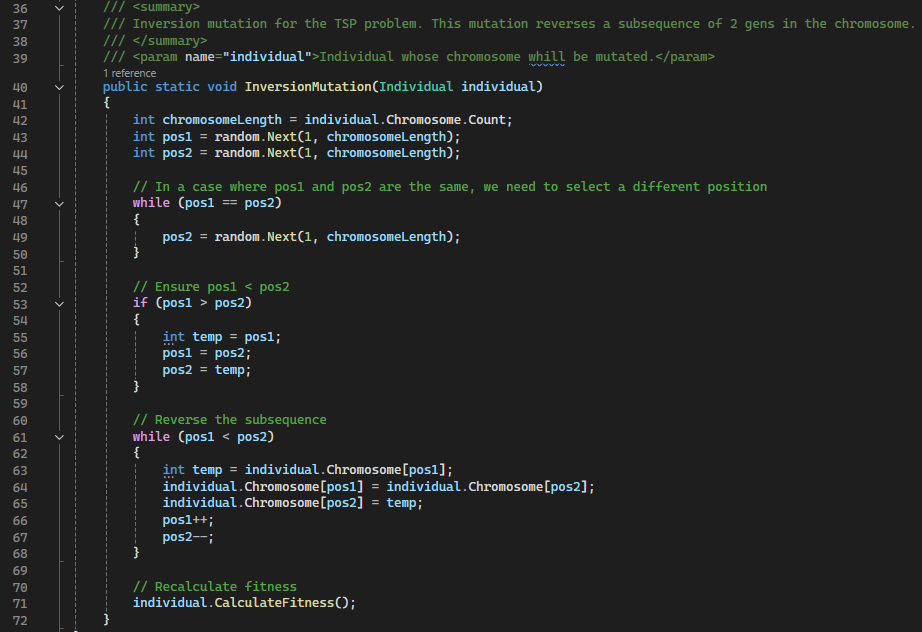
Nakoľko nebolo dané akú mutáciu máme používať v SP, rozhodol som sa použiť 2 typy mutácie.

* Náhodné výmeny 2 navštívených uzlov



Obrázok 2 Mutácia - výmena

* Inverzia poradia navštívených uzlov.



Obrázok 3 Mutácia - inverzia

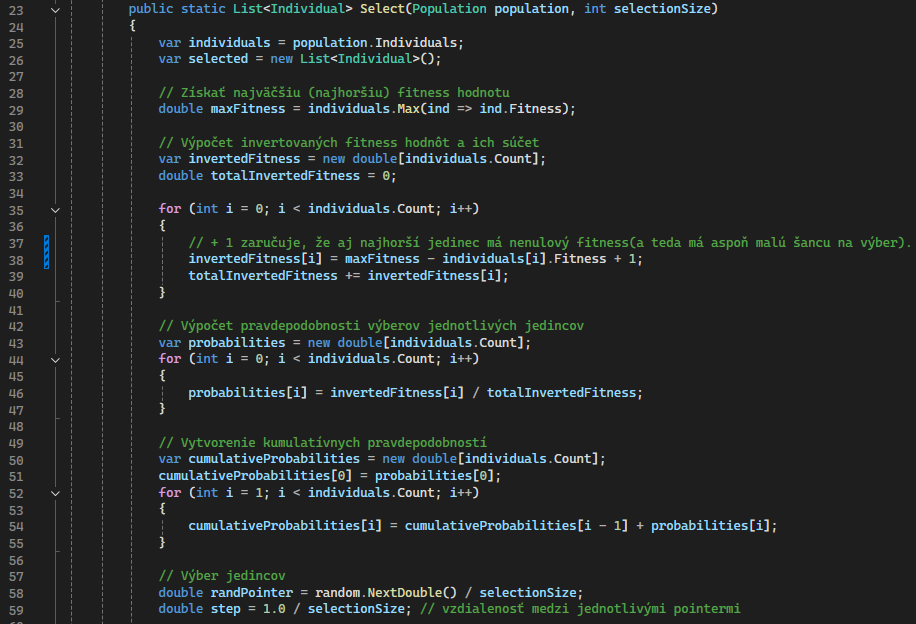
Celý algoritmus mutácií je implementovaný v triede **MutationTSP.**

## 2.2 Baker's Stochastic Universal Selection (SUS)

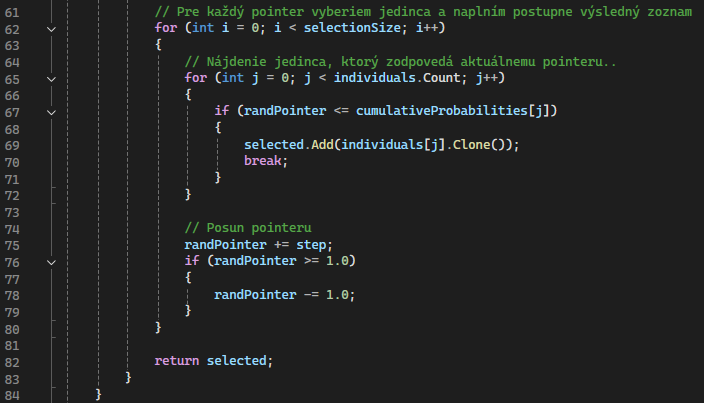
Baker's SUS je selekčný mechanizmus. V TSP je nižšia fitness hodnota lepšia, avšak SUS očakáva, že vyššia hodnota fitness znamená, že daný jedinec je lepší. implementoval som inverziu fitness hodnôt z dôvodov, že platí:

* Zvyšuje selektivitu výberu (lepší jedinci majú oveľa väčšiu šancu na výber)
* Zachovanie poradia jedincov (lepší jedinec má vyšší invertedFitness)
* Zabezpečuje dostatočný rozptyl pravdepodobností
* **Pri nepoužití tohto spôsobu (inverzie) som dostával omnoho horšie výsledky.**

Celý algoritmus je popísaný komentármi priamo v kóde, z tohto dôvodu prikladám snímky obrazovky kde je celý proces popísaný.



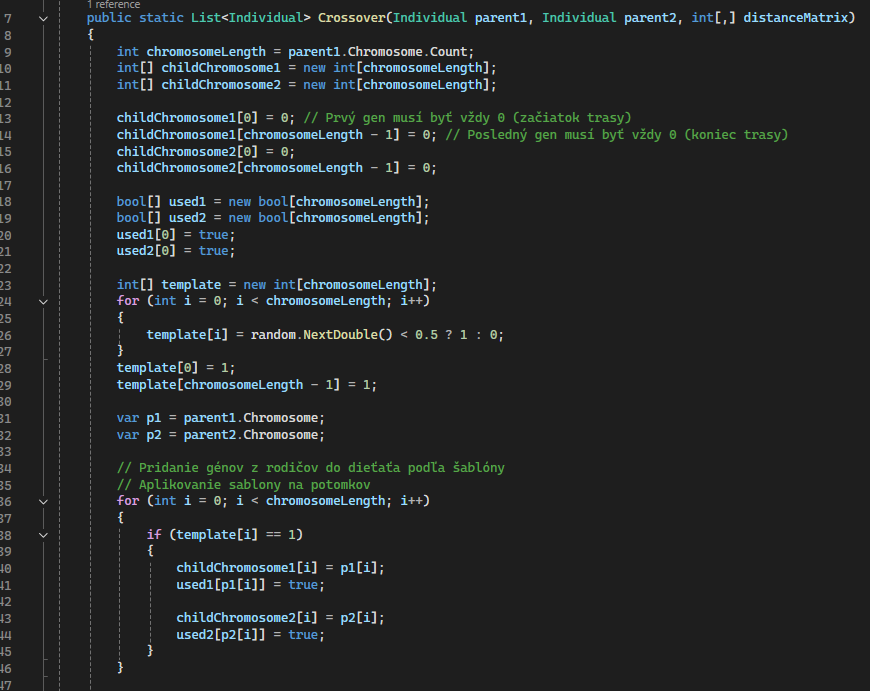
Obrázok 4 Selekcia Baker's SUS 1/2



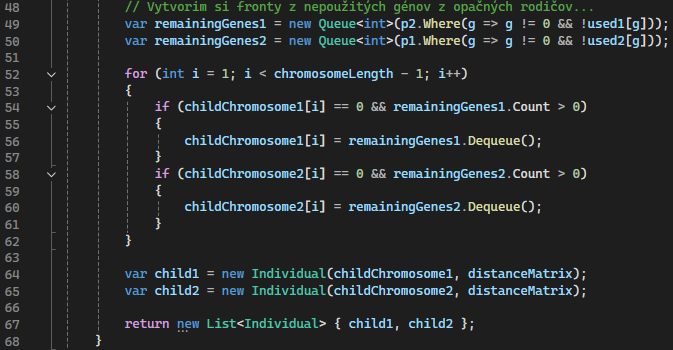
Obrázok 5 Selekcia Baker's SUS 2/2

## 2.2 Uniform Crossover (kríženie)

1. **Inicializácia potomkov** – Prvý a posledný gén v chromozóme je vždy 0, čo reprezentuje fixný začiatok a koniec trasy (napr. v prípade TSP).
2. **Generovanie šablóny (template)** – Pre každý gén sa náhodne rozhodne, či sa vezme od parent1 alebo parent2. Prvý a posledný gén sú však vždy nastavené na stredisko (zachovanie začiatku/konca).
3. **Aplikácia šablóny** – Gén zodpovedajúci hodnote 1 v šablóne sa kopíruje z príslušného rodiča do dieťaťa.
4. **Doplnenie zvyšných génov** – Génové pozície s hodnotou 0 sú doplnené z druhého rodiča, pričom sa zabezpečí, že každý gén (mesto) sa vyskytne len raz.
5. **Vytvorenie nových jedincov** – Na záver sa z hotových chromozómov vytvoria dvaja potomkovia pomocou konštruktora Individual.



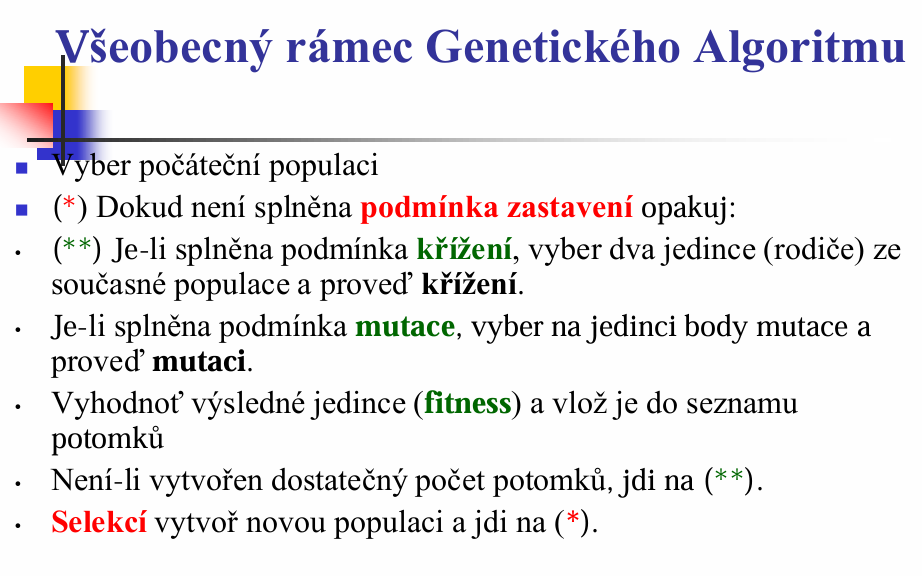
Obrázok 6 Unifrom crossover 1/2



Obrázok 7 Unifrom crossover 2/2

# Hlavný chod programu

Hlavný chod programu zabezpečuje trieda **GeneticAlgorithm**. Navrhnutý bol podľa prednášok, preto prikladám snímok obrazovky kde je popísané všeobecné fungovanie genetického algoritmu.



Obrázok 8 Všeobecný rámec Genetického Algoritmu

# Výsledky