Problém obchodného cestujúceho.
(Travelling Salesman Problem)

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.

Zadanie:

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.

Riešenia:

Genetický algoritmus

Genetická informácia je reprezentovaná vektorom, ktorý obsahuje index každého mesta v nejakom poradí (nejaká permutácia miest). Keďže hľadáme najkratšiu cestu, je najlepšie vyjadriť fitnes jedinca ako prevrátenú hodnotu dĺžky celej cesty.

Jedincov v prvej generácii inicializujeme náhodne – vyberáme im náhodnú permutáciu miest. Jedincov v generácii by malo byť tiež aspoň 20. Je potrebné implementovať aspoň dve metódy výberu rodičov z populácie.

Kríženie je možné robiť viacerými spôsobmi, ale je potrebné zabezpečiť, aby vektor génov potomka bol znovu permutáciou všetkých miest. Často používaný spôsob je podobný dvojbodovému kríženiu. Z prvého rodiča vyberieme úsek cesty medzi dvoma náhodne zvolenými bodmi kríženia a dáme ho do potomka na rovnaké miesto. Z druhého rodiča potom vyberieme zvyšné mestá v tom poradí, ako sa nachádzajú v druhom rodičovi a zaplníme tým ostatné miesta vo vektore.

Mutácie potomka môžu byť jednoduché – výmena dvoch susedných miest alebo zriedkavejšie používaná výmena dvoch náhodných miest. Tá druhá výmena sa používa zriedkavo, lebo môže rozhodiť blízko optimálne riešenie. Často sa však používa obrátenie úseku – znova sa zvolia dva body a cesta medzi nimi sa obráti. Sú možné aj ďalšie mutácie, ako napríklad posun úseku cesty niekam inam.

Dokumentácia musí obsahovať konkrétne použitý algoritmus, opis konkrétnej reprezentácie génov, inicializácie prvej generácie a presný spôsob tvorby novej generácie. Dôležitou časťou dokumentácie je zhodnotenie vlastností vytvoreného systému a porovnanie dosahovaných výsledkov aspoň pre dva rôzne spôsoby tvorby novej generácie alebo rôzne spôsoby selekcie. Dosiahnuté výsledky (napr. vývoj fitness) je vhodné zobraziť grafom. Dokumentácia by mala tiež obsahovať opis vylepšovania, dolaďovania riešenia.

Zakázané prehľadávanie (tabu search)

Zakázané prehľadávanie patrí do skupiny algoritmov, ktoré využívajú na hľadanie riešenia v priestore možných stavov lokálne vylepšovanie (optimalizáciu). To znamená, že z aktuálneho stavu si vytvorí nasledovníkov a presunie do takého, ktorý má lepšie ohodnotenie (najlepšieho takého). Ak neexistuje nasledovník s lepším ohodnotením (a nenašli sme dostatočne dobré riešenie), tak sme v lokálnom extréme a je potrebné sa z neho dostať. Tento algoritmus si teda vyberie horšieho nasledovníka a zároveň si uloží aktuálny stav do tzv. zoznamu zakázaných stavov (tabu list). Je to nevyhnutné, aby sme sa z toho horšieho nasledovníka znovu nedostali do tohto lokálne extrému a nevytvorili tak nekonečný cyklus. Zoznam zakázaných stavov je pomerne krátky, aby nám netrvala dlho jeho kontrola. Keď sa zaplní a je doň potrebné vložiť nový stav, tak ten najstarší sa zahodí.

Problém je opäť reprezentovaný vektorom, ktorý obsahuje index každého mesta v nejakom poradí (nejaká permutácia miest). Nasledovníci sú vektory, v ktorých je vymenené poradie niektorej dvojice susedných uzlov.

Dôležitým parametrom tohto algoritmu je dĺžka zoznamu zakázaných stavov. Príliš krátky zoznam spôsobí, že algoritmus bude často pendlovať medzi niekoľkými lokálnymi extrémami, príliš dlhý zoznam natiahne čas riešenia, lebo bude dlho trvať kontrola každého stavu, či nie je zakázaný. Je potrebné nájsť jeho vhodnú dĺžku.

Dokumentácia musí obsahovať konkrétne použitý algoritmus, a reprezentáciu údajov. Dôležitou časťou dokumentácie je zhodnotenie vlastností vytvoreného systému a opis závislosti jeho vlastností na dĺžke zoznamu zakázaných stavov. Použite aspoň dva rôzne počty miest (napr. 20 a 30).

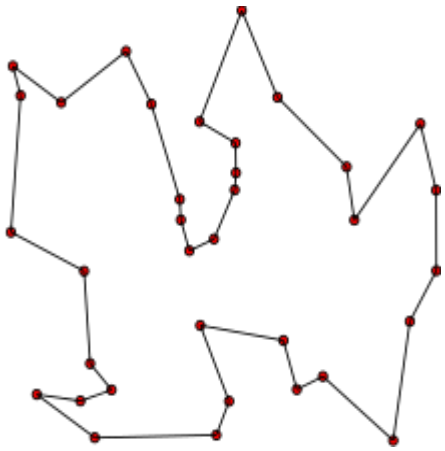
Simulované žíhanie (simulated annealing)

Simulované žíhanie patrí do skupiny algoritmov, ktoré využívajú na hľadanie riešenia v priestore možných stavov lokálne vylepšovanie. Zároveň sa algoritmus snaží zabrániť uviaznutiu v lokálnom extréme. Z aktuálneho uzla si algoritmus klasicky vytvorí nasledovníkov. Potom si jedného vyberie. Ak má zvolený nasledovník lepšie ohodnotenie, tak doň na 100% prejde. Ak má nasledovník horšie ohodnotenie, môže doň prejsť, ale len s pravdepodobnosťou menšou ako 100%. Ak ho odmietne, tak skúša ďalšieho nasledovníka. Ak sa mu nepodarí prejsť do žiadneho z nich, algoritmus končí a aktuálny uzol je riešením. Pre nájdenie globálneho extrému je dôležitý správny rozvrh zmeny pravdepodobnosti výberu horšieho nasledovníka. Pravdepodobnosť je spočiatku relatívne vysoká a postupne klesá k nule.

Problém je opäť reprezentovaný vektorom, ktorý obsahuje index každého mesta v nejakom poradí (nejaká permutácia miest). Nasledovníci sú vektory, v ktorých je vymenené poradie niektorej dvojice susedných uzlov.

Dôležitým parametrom tohto algoritmu je rozvrh zmeny pravdepodobnosti výberu horšieho nasledovníka. Príliš krátky (rýchly) rozvrh spôsobí, že algoritmus nestihne obísť lokálne extrémy, príliš dlhý rozvrh natiahne čas riešenia, lebo bude obiehať okolo optimálneho riešenia. Je potrebné nájsť vhodný rozvrh.

Dokumentácia musí obsahovať konkrétne použitý algoritmus, a reprezentáciu údajov. Dôležitou časťou dokumentácie je zhodnotenie vlastností vytvoreného systému a opis závislosti jeho vlastností na rozvrhu. Použite aspoň dva rôzne počty miest (napr. 20 a 30).



Obr. Príklad trasy medzi náhodne zvolenými mestami

Príklad rozloženia miest (20 miest):

(60, 200), (180, 200), (100, 180), (140, 180), (20, 160), (80, 160), (200, 160), (140, 140), (40, 120), (120, 120), (180, 100), (60, 80), (100, 80), (180, 60), (20, 40), (100, 40), (200, 40), (20, 20), (60, 20), (160, 20)