SEMESTRÁLNA PRÁCA – ÚLOHA OBCHODNÉHO CESTUJÚCEHO - HEURISTKY

Optimalizácia sietí

Benjamín Koša

Žilinská univerzita v Žiline | Fakulta riadenia a informatiky

6.1.2023

Obsah

Zadanie úlohy

Na vybranej testovacej matici vzdialeností riešte úlohu **obchodného cestujúceho** prideleným **heuristickým** algoritmom.

Na skonštruovanie východzej prípustnej trasy využite duálnu heuristiku podľa Vášho zadania:

1. **Algoritmus zväčšovania o najbližší uzol**
   Algoritmus vychádza zo základnej neprípustnej trasy i_1 - i_2 - i_3 - i_1, ktorú v každom kroku zväčší vsunutím spracovávaného uzla medzi dva už zaradené uzly, ktoré nasledujú po sebe v súčasnej trase.
   Spracovávaný uzol sa vyberie z množiny doposiaľ nezaradených uzlov ako uzol, ktorý je najmenej vzdialený od už zaradených uzlov v trase (podľa ***súčtového kritéria*** – súčet vzdialeností kandidáta na spracovanie od každého uzla zaradeného v trase). Miesto, na ktoré je spracovávaný uzol zaradený, je určené tak, aby sa súčasná trasa zväčšila o čo najmenej.
   Základnú neprípustnú trasu i_1 - i_2 - i_3 - i_1 určíme tak, že i_1 bude prvý uzol v zozname v zadanej sieti, i_2 bude doposiaľ nezaradený uzol, ktorý je najviac vzdialený od uzla i_1 a i_3 bude doposiaľ nezaradený uzol, ktorý je najviac vzdialený od uzla i_2.

Takto nájdenú východziu trasu obchodného cestujúceho **zlepšite pomocou metaheuristiky Simulated Annealing** s určeným spôsobom nájdenie okolia aktuálneho riešenia (podľa zadania):

C. **Zmena polohy vrchola v trase** (výmena reťazcov dĺžky 1 typu neprázdny-prázdny)

Pre metaheuristiku Simulated Annealing nastavte parametre takto: teplotu nastavte na počiatočnú hodnotu $t_{max} = 10000$, maximálny počet preskúmaných prechodov od prechodu k súčasnému riešeniu $u = 40$, maximálny počet preskúmaných prechodov od poslednej zmeny teploty $q = 50$. *(Popis algoritmu je v knihe „Optimalizace na dopravních sítích“, str.95-96)*

Popis spôsobu riešenia

Riešenie spočíva z dvoch väčších častí, prvá časť je vytvorenie základnej cesty a v druhej časti sa základnú cestu snažíme vylepšiť pomocou algoritmu simulated annealing.

Tvorenie základnej cesty

Vytvoríme si pole nezaradených vrcholov, z neho podľa zadania vyberáme vrcholy a vkladáme do vytváranej cesty až kým toto pole neje prázdne.

Simulated annealing SA

Základu cestu dáme ako vstupný parameter algoritmu SA, v algoritme náhodne vyberieme vrchol z riešenia, presunieme ho na iné náhodne miesto. Spočítame zmenu dĺžky, ak nastalo zlepšenie zapíšeme si nové lepšie riešenie, ak nenastalo zlepšenie spočítame si akceptačne kritérium a ak je splnená podmienka prijmeme aj horšie riešenie.
Dôvodom prečo akceptujeme aj horšie riešenie je umožniť algoritmu uniknúť lokálnym minimám a dôkladnejšie preskúmať priestor riešenia. Algoritmus sa tak môže vyhnúť uviaznutiu v subotimálnom riešení.

Popis funkcii

Pre úlohu sme sa rozhodli použiť model funkcionálneho programovania.

```
/**
 * search for base way
 * @return
 */
public static ArrayList<Integer> getBaseWay()
```

```
/**
 * count way extension before node insert
 * @param i
 * @param j
 * @param newNode
 * @return
 */
public static int countExtension(int i, int j, int newNode)
```

```
/**
 * count improvement before change of node position
 * @param bestSolution
 * @param cityIndex
 * @param moveToIndex
 * @return
 */
public static int countImprovement(ArrayList<Integer> bestSolution, int
cityIndex, int moveToIndex)
```

```
/**
 * move node in solution
 * @param solution
 * @param cityIndex
 * @param moveToIndex
 */
public static void moveNode(ArrayList<Integer> solution, int cityIndex, int
moveToIndex)
```

```
/**
 * simulatedAnnealing algorithm
 * @param bestSolution solution
 * @param START_TEMPERATURE
 * @param COOLING_RATE
 * @param NUM_ITERATIONS
 * @param MAX_PASSES
 * @param SEED
 */
public static void simulatedAnnealing(
        ArrayList<Integer> bestSolution,
        final int START_TEMPERATURE,
        final int COOLING_RATE,
        final int NUM_ITERATIONS,
        final int MAX_PASSES,
        final int SEED)
```

```
/**
 * count acceptance probability
 * @param bestDistance
 * @param newDistance
 * @param temperature
 * @return
 */
private static double acceptanceProbability(int bestDistance, int
newDistance, double temperature)
```

```
/**
 * count whole solution distance
 * @param way
 * @return
 */
private static  int countDistance(ArrayList<Integer> way)
```

Zaver

Základná trasa ma dĺžku 3040. Metaheuristika SA ju nedokázala zlepšiť. Skúšali sme parametre zo zadania. Skúšali sme aj znížiť základnú teplotu, zväčšiť mieru znižovania teploty, zväčšiť počet iterácii alebo zväčšiť maximálny počet preskúmaných prechodov. Každú zmenu parametrov sme skúšali na minimálne 100 000 seedoch. V žiadnom prípade sa nám nepodarilo zlepšiť základne riešenie. Dôvodom môže byt že základne riešenie má blízko k optimálnemu riešeniu. Základnú trasu je možné mierne vylepšiť ak upravíme algoritmus SA tak že nebudeme prijímať horšie riešenia.