UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Finančna matematika – 1. stopnja

Katarina Brilej, Sara Kovačič

Uporaba metahevristike GRASP na problemu potujočega trgovca

Projekt OR pri predmetu Finančni praktikum

Mentor: prof. dr. Riste Škrekovski

Ljubljana, 2019

1. Uvod

Metahevristika je algoritemski način reševanja kombinatoričnega optimizacijskega problema, pri katerem na začetku izberemo množico kandidatov za rešitev, in jo iterativno izboljšujemo (glede na neko vnaprej izbrano funkcijo zaželenosti), ter po dovolj korakih vrnemo najboljši element iz te množice. Metahevristike torej vrnejo približne rešitve, a veliko hitreje kot eksaktni postopki. V projektu bova na problem potujočega trgovca implementirali metahevristiko GRASP (*greedy randomized adaptive search procedure*). Problem potujočega trgovca bova rešili tudi kot celoštevilski linearni program in primerjali rešitve. Generirali bova nekaj zanimivih grafov in na njih preizkusili algoritem. Rezultate bova primerjail tudi z rezultati iz spleta in rezultati skupine 7, ki bo na problem potujočega trgovca implementirala genetski algoritem.

2. Problem potujočega trgovca

Problem potujočega trgovca ("travelling salesman problem"/TSP) se glasi:

- **Formulacija v vsakdanjem jeziku:** danih je n mest in razdalja med poljubnim parom mest (od mesta do mesta lahko potujemo po zgolj eni poti). Najdi najkrajšo (najcenejšo) pot, ki se začne in konča v istem mestu ter obišče vsako mesto natanko enkrat.
- **Formulacija v matematičnem jeziku**: v (neusmerjenem enostavnem) polnem grafu K_n z uteženimi povezavami (pozitivne vrednosti) najdi najkrajši cikel, ki vsebuje vsa vozlišča. Ciklom, ki vsebujejo vsa vozlišča grafa, pravimo Hamiltonovi cikli.

2.1. **Celoštevilski linearni program in primerjava z GRASP.**

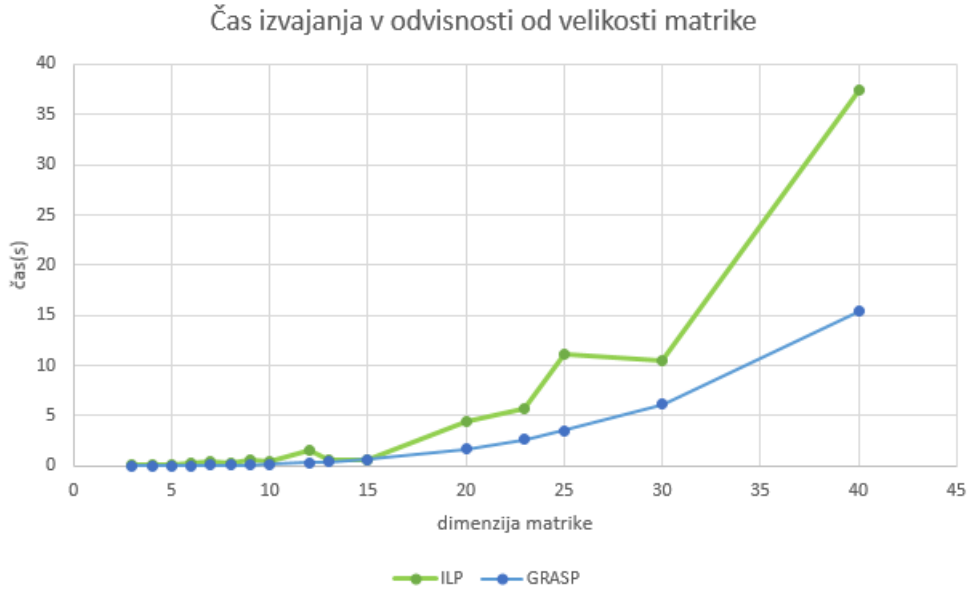
Problem potujočega trgovca lahko predstavimo kot *celoštevilski linearni program.* Označimo mesta s števili $1, \ldots, n$. Strošek (ali razdalja) potovanja iz mesta i v mesto j je $c_{i,j}$, ($1 \leq i, j \leq n$). Minimiziramo strošek potovanja. Definiramo:

$$X_{i,j} := \begin{cases} 1\ ; & \text{potnik gre iz mesta } i \text{ v mesto } j, \\ 0\ ; & \text{sicer,} \end{cases}$$

y_i ... katero po vrsti obiščemo mesto i

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} x_{i,j}\cdot c_{i,j} \\
\text{p. p.} \quad & \sum_{i=1}^{n} x_{i,j} = 1, \text{ za vsak } j \\
& \sum_{j=1}^{n} x_{i,j} = 1, \text{ za vsak } i \\
& x_{i,j} \in \{0,1\}, \text{ za vsak } i \text{ in vsak } j \\
& y_i \in \{1,\ldots,n\}; \text{za vsak } i \\
& y_i + 1 - n + n\cdot x_{i,j} \leq y_j; \text{za vsak } i \text{ in vsak } j > 1
\end{aligned}
$$

Problem potujočega trgovca se da v pythonu elegantno zapisati kot celoštevilski linearni program s pomočjo knjižnice PuLP. PuLP za reševanje problema izbere enega od vgrajenih algoritmov. V našem primeru je PuLP za reševanje izbral PULP_CBC_CMD. S pomočjo knjižnice PuLP sva napisali funckijo, ki sprejme matriko cen povezav, izpiše minimalno ceno potovanja in vrne urejen seznam obiskanih mest. Z merjenjem časa reševanja določenih primerov različnih velikosti ugotovimo, da napisana funkcija *ilp* porabi več časa za reševanje, kot metahevristika $GRASP$.



3. Grasp

GRASP (*greedy randomized adaptive search procedure*) je metahevristika, ki sestoji iz dveh faz: *greedy randomized construction* in *local search*. V prvi fazi na pameten način (odvisno od problema) izberemo izmed vseh možnih rešitev CL (candidate list) množico začetnih približkov RCL (restricted candidates list). To storimo deloma deterministično in deloma stohastično, da zagotovimo, da so začetni približki obetavni, a dovolj razpršeni po celotni množici CL, da bo druga faza pregledala čimvečji del CL. V drugi fazi za vsako izmed teh rešitev$s \in RCL$ pregledamo elemente $s' \in CL$ v njeni okolici (kaj je okolica je od problema in načina reševanja odvisno). Če najdemo boljšo rešitev s', jo dodamo v RCL ter s odstranimo. To

ponavljamo dokler zaustavitveni pogoj (npr. št. iteracij, zahtevana natančnost) ni izpolnjen.

Literatura

[1] Travelling salesman problem
http://en.wikipedia.org/wiki/Travelling_salesman_problem