Reševanje problema trgovskega potnika s k-optimalnim in Lin-Kernighanovim algoritmom

Žan Jernejčič in Ines Šilc

V projektni nalogi bova reševala Problem trgovskega potnika s pomočjo k-optimalnega in Lin-Kernighanovega algoritma.

Problem trgovskega potnika oziroma Travelling salesman problem (krajše TSP) je problem, kjer imamo podanih n mest in razdalje med vsemi (za vsak par mest imamo torej podano, koliko sta si oddaljeni). Zanima nas, ali lahko obiščemo vsako mesto in se na koncu vrnemo v prvotno mesto. Če označimo $d_{i,j}$ kot razdaljo mad i-tim in j-tim mestom, iščemo torej:

$$\min_{\pi \in S_n} \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)}$$

kjer je S_n množica vseh permutacij danih n mest.

Naivna rešitev je očitna, pogledamo $(n-1)!$ kombinacij, torej iz vsakega mesta v vsako drugo mesto, si zapišemo vse kombinacije in kakšno razdaljo smo prepotovali, ter izberemo tisto možnost, kjer je bila razdalja najkrajša. Poskusimo še z drugimi rešitvami.

K-optimalni algoritem

K-optimalni algoritem, je lokalni algoritem, kjer iščemo rešitev Problema trgovskega potnika in sorodnih problemov. Algoritem deluje tako, da zmanjša dolžino trenutnega potovanja, dokler ne dosežemo poti, katere dolžine ne moremo izboljšati.

Poznamo dve glavni različici k-opt algoritma, to sta 2-Opt in 3-Opt algoritem. 2-Opt algoritem deluje tako, da odstrani dve vozlišči grafa (dve mesti), tako razdeli pot na dva dela, poišče najboljšo rešitev podproblema, in nato poveže nazaj mesti na najboljši možen način. 3-Opt algoritem odstrani tri povezave ali vozlišča, tako dobimo 3 podomrežja mest. V naslednjem koraku analiziramo najboljšo pot med temi tremi podomrežji. To ponavljamo za druge tri povezave, dokler nismo poskusili vseh možnih poti v danem omrežju.

Lin-Kernighanov algoritem

Glavna razlika med k-optimalnim in Lin-Kernighanovim algoritmom je, da si pri k-optimalnemu na začetku izbremo fiksen k, medtem ko pri Lin-Kernighanovemu na vsakem koraku pogledamo kakšen k bi se nam najbolj splačalo vzeti. Lin-Kernighanov algoritem na vsakem koraku začne s $k = 2$ in k povečuje dokler ne najde zamenjave, ki bi skrajšala trenutni obhod. Če jo najde, potem ponastavi vrednost k nazaj na 2 in začne novo iteracijo. Če algoritem preizkusi vse možne

zamenjave in ne najde izboljšave, potem je dobljena rešitev lokalni minimum.

Na vsakem koraku gradimo množici A in B, ki predstavljata množico povezav, ki jih bomo odstranili iz obhoda $A = \{a_1, \dots a_n\}$ in množico povezav, ki jih bomo dodali $B = \{b_1, \dots, b_n\}$. Množicama na vsakem koraku dodamo para povezav(a_i, b_i). Da poenostavimo delovanje algoritma, morata množici zadoščati nekaterim kriterijem.

- Množici A in B morata biti disjunktni.
- Povezavi a_i iz A in b_i iz B imata skupno vozlišče ter b_i in a_{i+1} tudi. Če je veriga povezav $(a_1, b_1, \dots a_n, b_n)$ sklenjena rečemo, da je menjava zaporedna. Tako dovolimo le zaporedne menjave
- Naj bo povezava $a_i = (t_{2i-1}, t_{2i})$ iz A in $i \geq 3$. Če povežemo vozlišči t_{2i} in t_1, potem dodane in odstranjene povezave tvorijo cikel.
- Naj bo $c(a_i)$ dolžina povezave a_i in naj bo $p_i = c(a_i) - c(b_i)$ dobiček zamenjave a_i z b_i. Potem je vsak delni dobiček $P_i = p_1 + p_2 + \cdots + p_i$ pozitiven.

Ko uvedemo te kriterije omejimo iskalno območje le na povezave, ki bolj verjetno izboljšajo obhod in tako dobimo lokalni minimum.

Načrt dela

V projektu bova najprej definirala vse potrebne funkcije za generiranje grafov in osnovnih operacij nad njimi, nato pa bova izpeljala oba algoritma na čim več različnih grafih in primerjala rezultate in zahtevnost obeh algoritmov.

Viri:

- Johnson, D. S., & McGeoch, L. A. (1997). *The traveling salesman problem: A case study in local optimization. Local search in combinatorial optimization*, 1(1), 215-310.