Primerjava hevrističnih algoritmov za trgovskega potnika

Janez Brest[1], Štefan Brest[1], Janez Žerovnik[2,3]
[1]Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Inštitut za računalništvo
Smetanova 17, 2000 Maribor, Slovenija
[2]Univerza v Mariboru, Fakulteta za strojništvo
Smetanova 17, 2000 Maribor, Slovenija
[3]IMFM, Jadranska 19, 1111 Ljubljana, Slovenija
janez.brest@uni-mb.si

Performance Comparison of Heuristic Algorithms for the Traveling Salesman Problem

The traveling salesman problem (TSP) is one of the most studied problems in combinatorial optimization. The TSP is simply stated, has practical applications, and is a representative of a large class of important scientific and engineering problems. There is given a list of n cities and distances between them; in which order should the cities be visited so that the tour is as short as possible? Each city should be visited once before returning to the starting point.

In this paper, the perfomance comparison of heuristic algorithms RAI, OR-opt, and RAI+OR is outlined.

1 Uvod

V članku opisujemo reševanje naloge trgovskega potnika, ki sodi v skupino zelo zahtevnih NP-težkih problemov [9, 1]. Če velja P≠NP[1], potem za NP-težke probleme ne obstajajo polinomski algoritmi.

Matematično problem TSP (*ang. traveling salesman problem*) preprosto formuliramo takole: V uteženem polnem grafu je treba poiskati minimalni Hamiltonov cikel. Hamiltonov cikel je sprehod v grafu, na katerem je vsaka točka, razen prve in zadnje, natanko enkrat. Utež sprehoda je vsota uteži njegovih povezav (osnovne pojme teorije grafov glej npr. [13, 12]).

Članek [2] govori o tem, da z uporabo hevrističnih algoritmov lahko poiščemo rešitev trgovskega potnika, ki je blizu optimalne rešitve (ali je celo optimalna), in je njihova časovna zahtevnost precej manjša v primerjavi z algoritmi, ki poiščejo optimalno rešitev trgovskega potnika.

V [4, 3] je opisan hevristični algoritem RAI za reševanje trgovskega potnika, ki se je posebej dobro obnesel na nesimetričnih primerkih.

[1]P≠NP je verjetno najbolj znan odprti problem teoretičnega računalništva (glej [6]). Prevladuje mnenje, da velja P≠NP.

Moderne hevristične algoritme za nesimetrični problem trgovskega potnika lahko razvrstimo v tri razrede [5, 8, 7]: hevristike, ki temeljijo na gradnji cikla (na primer algoritma Nearest Neighbor, Greedy); algoritme lokalnega iskanja, ki temeljijo na razporejanju delov cikla (na primer algoritem Kanellakis-Papadimitriou), in algoritme, ki temeljijo na širitvi cikla v smislu minimalnega oboda (variante algoritma, ki jih je predlagal Zhang).

Nadaljevanje prispevka je organizirano takole: drugi razdelek vsebuje kratek opis problema. V 3. razdelku predstavimo hevristične algoritme, med katerimi želimo posebej izpostaviti algoritem RAI+OR. Rezultati, ki smo jih dobili s pomočjo hevrističnih algoritmov na optimizacijskih primerkih, so prikazani v 4. razdelku. Sledi še sklepni, 5. razdelek, kjer podamo smernice za nadaljnje raziskave.

2 Opis problema

Definicija: Dan je graf z n vozlišči. Za vsak par vozlišč c_i in c_j je podana razdalja $d(c_i, c_j)$, ki jo krajše označimo z d_{ij}. Cilj pri reševanju TSP je poiskati permutacijo vozlišč, označimo jo s π, ki minimizira:

$$\sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(n)}, c_{\pi(1)})$$

Probleme trgovskega potnika lahko delimo na simetrične in nesimetrične. Pri simetričnih problemih za vsako vozlišče velja $d_{ij} = d_{ji}$, kar pomeni, da je razdalja od vozlišča i do vozlišča j enaka razdalji od j od i. Pri nesimetričnih problemih je lahko $d_{ij} \neq d_{ji}$.

Poseben primer simetričnega trgovskega potnika so evklidski problemi trgovskega potnika, kjer so vozlišča grafa razporejena v dvodimenzionalnem (tridimezionalnem) prostoru, razdalje med njimi pa so definirane z evklidsko razdaljo. Problem trgovskega potnika si lahko predstavljamo takole: Vozlišča grafa so kraji, ki jih mora trgovski potnik obiskati, povezave pa poti, po katerih hodi

(v današnjem času bi raje rekli, da se vozi) iz kraja v kraj. Povezave med kraji so različno dolge in imajo še dodatne omejitve (slabo vozna cesta, gost promet itd.). Vsaki povezavi priredimo utež, ki pomeni ceno, potrebno, da trgovski potnik opravi to pot. Trgovski potnik mora obiskati vsak kraj natanko enkrat in se vrniti v izhodiščni kraj ter pri tem narediti najkrajšo pot oziroma stroški potovanja morajo biti minimalni.

Omenili smo že, da sodi problem trgovskega potnika v skupino NP težkih problemov (ne moremo jih rešiti v polinomskem času). Naivno iskanje optimalne rešitve zahteva pregled vseh mogočih Hamiltonovih ciklov danega grafa. Vseh Hamiltonovih ciklov v polnem grafu je $(n-1)!$, kjer je n število vozlišč grafa. Zato naivni pristop odpade že pri zelo majhnih n.

Ker je problem TSP NP-težak, uporabljamo hevristične algoritme [11, 9], ki imajo manjše časovne zahtevnosti, s katerimi pa ne moremo vedno poiskati optimalne rešitve. Optimalni rešitvi se bolj ali manj približamo.

3 Hevristični algoritem RAI+OR

V tem poglavju opišemo hevristični algoritem, ki smo ga razvili na način, da smo združili dva že znana algoritma. Prvi je naš algoritem RAI [4, 3], drugi pa je v literaturi znan algoritem OR-opt [9], ki ga bomo na kratko opisali na koncu tega poglavja.

Tako dobljen nov hevristični algoritem, poimenujmo ga RAI+OR, želimo uporabiti za reševanje naloge predvsem nesimetričnega trgovskega potnika. Predstavljeni algoritem je preprost in ima polinomsko časovno zahtevnost. Ideja algoritma je naslednja: iz cikla, ki je trenutna rešitev, odstranimo množico vozlišč in jih nato ponovno vstavimo v cikel glede na optimizacijsko kriterijsko funkcijo.

Algoritem RAI+OR:

1. Začetni cikel naj sestoji iz poljubnega vozlišča in zanke.
2. Naključno izberi vozlišče, ki ga še ni v ciklu.
3. Izbrano vozlišče vstavi med dve sosednji vozlišči[2] v ciklu na najcenejši način. Če cikel ne vsebuje vseh vozlišč, pojdi na korak 2.
4. *Nad dobljenim polnim ciklom izvedi algoritem OR-opt.*
5. Pomni trenutno rešitev, poimenujmo jo S.
6. Korake od 7 do 13 ponovi P-krat.
7. Naključno izberi i in j ($i, j \in N = \{1, \ldots, n\}$, $1 \leq i \leq j \leq n$).
8. Iz S odstrani del poti, ki se začne z i in konča z j, in poveži vozlišče $i-1$ z vozliščem $j+1$.
9. *Nad dobljenim ciklom, ki ne vsebuje v prejšni točki odstranjene poti, izvedi algoritem OR-opt.*
10. Naključno izberi vozlišče na odstranjeni poti.
11. Izbrano vozlišče vstavi med dve sosednji vozlišči v ciklu na najcenejši način. Če cikel ne vsebuje vseh vozlišč, pojdi na korak 10.
12. *Nad dobljenim polnim ciklom izvedi algoritem OR-opt.*
13. Trenutno novo dobljeno rešitev primerjaj z rešitvijo S in boljšo obdrži.

Prvih pet korakov zgradi začetno rešitev. Lokalna optimizacija se izvaja v glavni zanki, ki zajema korake od 7 do 13. Iz cikla, ki pomeni trenutno rešitev, odstranimo nekaj sosednjih vozlišč, izvedemo optimizacijo OR-opt, nato odstranjena vozlišča drugo za drugo ponovno vstavimo v cikel na najcenejši način. Ko cikel vsebuje vsa vozlišča, izvedemo še optimizacijo OR-opt.

Algoritem RAI+OR smo dobili tako, da smo algoritmu RAI [4, 3] dodali točke 4, 9 in 12, ki so napisane poševno.

Predvsem zaradi časovne zahtevnosti, smo izbrali fiksno število korakov lokalne optimizacije ($P = n^2$, kjer je n število vozlišč grafa).

V vsaki ponovitvi je število odstranjenih in ponovno vstavljenih vozlišč največ n in pri vsakem vstavljanju vozlišča v cikel je mogočih največ n različnih mest vstavitve, torej primerjav. Od tod izvira ideja za n^2 ponovitev optimizacijske zanke.

Časovna zahtevnost algoritma za optimizacijo OR-opt je $O(n^3)$. V najslabšem primeru je časova zahtevnost našega algoritma $T_w(n) = O(n^2(n^2 + n^3)) = O(n^5)$.

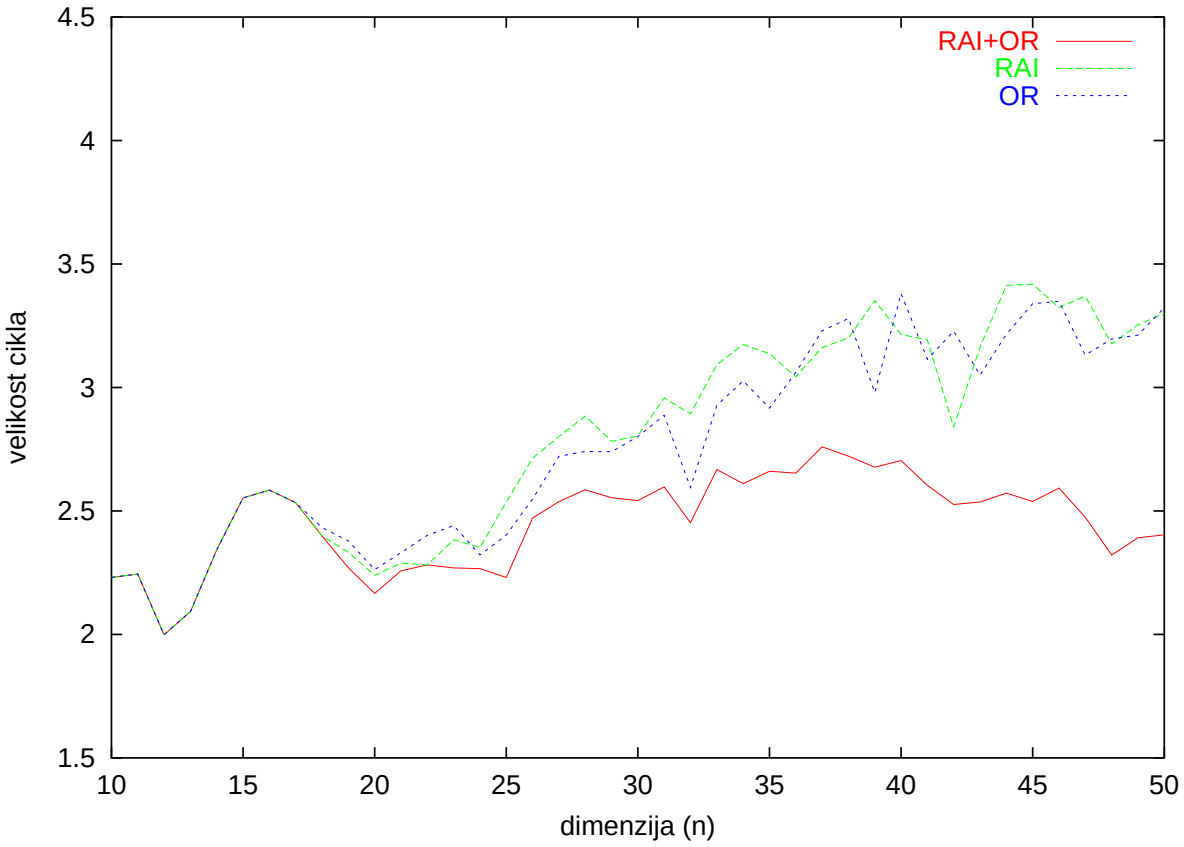
OR-opt [9] (str. 220) je algoritem lokalne optimizacije in deluje takole. Za vsako povezano pot iz s vozlišč (najprej je s enako 3, zatem 2 in na koncu 1), preverimo, ali lahko vstavimo pot med poljubni dve vozlišči, da dobimo boljšo rešitev. Če to lahko storimo, pot vstavimo. Ko smo izčrpali vse poti dolžine 3, nadaljujejo s potmi dolžine 2, ter na koncu še s potmi z enim vozliščem. Ko s takim vstavljenjem ni več izboljšanj rešitve, se algoritem zaključi.

Algoritem OR, ki ga bom v nadaljevanju primejali z algoritmoma RAI+OR in RAI, smo implementirali takole: naključno smo generirali celotno rešitev, nato pa smo izvedli optimizacijo OR-opt in celotno zadevo ponovili $P = n^2$ krat. Med n^2 rešitvami smo poiskali najboljšo dobljeno rešitev.

4 Rezultati

V tem poglavju prikažemo dobljene rezultate delovanja algoritmov na grafih, ki smo jih generirali na

[2] Ko cikel iz koraka 1 vsebuje eno vozlišče, izbrano vozlišče vstavimo v cikel (obstaja le ena možnost vstavitve) in rezultat je cikel z dvema vozliščema.



Slika 1: Najboljša rešitev v 100 poskusih za grafe velikost od 10 do 50 vozlišč.

naslednji način. Za dano dimenzijo n smo naključno (enakomerna porazdelitev na $[0, 1]$) generirali povezave polnega (simetričnega) grafa $d \times d$. Dobili smo grafe velikosti od 10 do 50 vozlišč.

Rezultati, ki smo jih dobili pri poskusu, so prikazani na slikah 1 in 2. Za vsak graf smo hevristične algoritme zagnali 100-krat. Na sliki 1 so prikazane najboljše rešitve za grafe velikost od 10 do 50 vozlišč. Opazimo, da za algoritma RAI in OR-opt ne moremo ugotoviti, kateri od njiju daje boljše rezultate. Algoritem RAI+OR pa daje boljše ali enake rezultate kot algoritma RAI in OR. Za grafe, ki imajo več kot 23 vozlišč, pa so rezultati algoritma vedno boljši v primerjavi z rezultati ostalih dveh algoritmov.

Na sliki 2 so prikazane rešitve, ki so dobljene kot povprečje v 100 poskusih, za grafe velikost od 10 do 50 vozlišč. Opazimo, da za algoritma RAI in OR-opt ne moremo ugotoviti, kateri od njiju daje boljše rezultate. Algoritem RAI+OR pa daje boljše rezultate kot algoritma RAI in OR pri vseh grafih.

Izmerili smo tudi porabljen čas osebnem računalniku pri grafu s 50 vozlišči: RAI+OR je potreboval $0.65s$, OR $0.95s$ in RAI $0.03s$.

Pri testnih grafih smo za grafe z majhno dimenzijo izračunali optimalno rešitev s pomočjo algoritma sestopanja, in sicer za dimenzije od 10 do vključno 17. Za graf s 17 vozlišči smo potrebovali približno 14 ur, za ostale, večje grafe pa bi izvajanja algoritma trajalo preveč časa. Za grafe od 10 do 17 vozlišč so vsi algoritmi (izjema je le algoritem RAI, ki ni našel optimalne rešitve pri grafu s 17 vozlišči) našli rešitev, ki je optimalna, kar smo preverili oziroma potrdili z algoritmom sestopanja.

5 Sklep

V članku smo predstavili primerjavo treh hevristični algoritmov RAI+OR, OR, RAI za problem trgovskega potnika. Primerjavo algoritmov smo opravili na simetričnih grafih velikosti od 10 do 50 vozlišč.

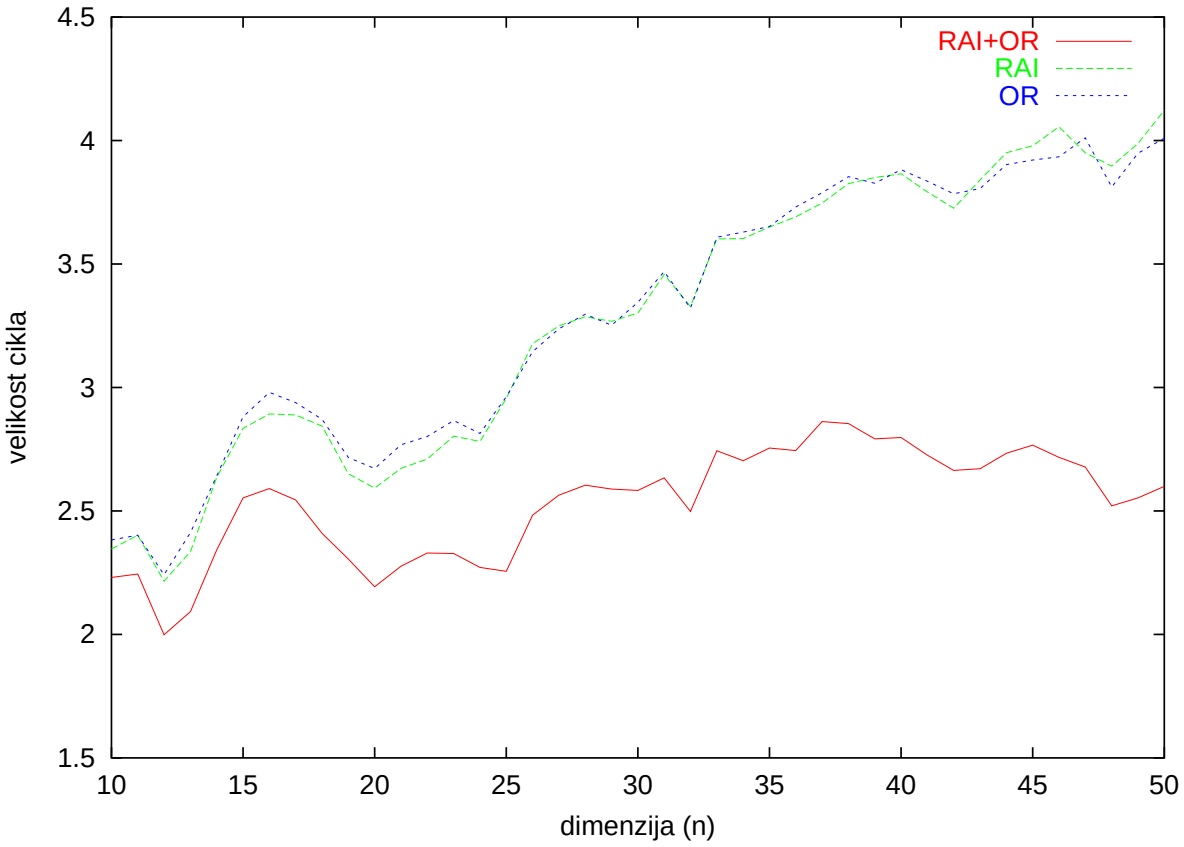
Na podlagi dobljenih rezultatov lahko zaključimo, da hevristični algoritem RAI+OR daje boljše rezultate v primerjavi z ostalima algoritmoma.

Algoritmi so primerni tudi za reševanje nesimetričnih problemov trgovskega potnika.

Nadaljnje raziskave vključujejo primerjavo hevrističnih algoritmov na testnih primerih iz knjižnice TSPLIB [10].

Literatura

[1] V. Batagelj, Algoritmi za reševanje splošnega problema trgovskega potnika, Uporabna Informatika, Leto 1, str. 27–32, 1993.



Slika 2: Povprečje rešitev v 100 poskusih za grafe velikost od 10 do 50 vozlišč.

[2] J. Brest, J. Žerovnik, Aproksimacijski algoritem za nesimetrični problem trgovskega potnika, *Zbornik pete Elektrotehniške in računalniške konference ERK'96*, Zvezek B:15–18, September 1996.

[3] J. Brest and J. Žerovnik, An approximation algorithm for the asymmetric traveling salesman problem. *Ric. oper.*, 28:59–67, 1999.

[4] Janez Brest, Janez Žerovnik in Viljem Žumer. Algoritem za nesimetrični problem trgovskega potnika. *Elektroteh. vestn.*, 70(1/2):40–45, 2003.

[5] J. Cirasella, D. S. Johnson, L. A. McGeoch, and W. Zhang, The asymmetric traveling salesman problem: Algorithms, instance generators, and tests. *Lecture Notes in Computer Science*, 2153:32–59, 2001.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability*, W.H. Freeman and Co., San Francisco 1979.

[7] K. Helsgaun, An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European J. Oper. Res.*, 126 (1), pages 106–130, 2000.

[8] D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovich, Experimental Analysis of Heuristics for the ATSP, in The Traveling Salesman Problem and its Variations, G. Gutin and A. Punnen, Editors, Kluwer Academic Publishers, 2002, Boston, 445–487. http://www.research.att.com/∼dsj/papers.html

[9] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, *The Traveling Salesman Problem*, John Wiley & Sons, New York, 1985.

[10] G. Reinelt, TSPLIB - a traveling salesman problem library, *European Journal of Operations Research*, 52, pp. 125, 1991. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[11] D. J. Rosenkrantz, R. R. Stearns and P. M. Lewis, *An Analysis of Several Heuristics for the Traveling Salesman Problem*, SIAM J. Comput. 6, pp. 563-581, 1977.

[12] R. J. Wilson in J. J. Watkins, Uvod v teorijo grafov, DMFA, Ljubljana, 1997.

[13] J. Žerovnik, Osnove teorije grafov in diskretne optimizacije, Fakulteta za strojništvo, Maribor, 2003.