ALGORITMI ZA REŠEVANJE SPLOŠNEGA PROBLEMA TRGOVSKEGA POTNIKA

Vladimir Batagelj
Oddelek za Matematiko, Univerza v Ljubljani

Povzetek

Problem trgovskega potnika (PTP) spada med osnovne probleme kombinatorične optimizacije. V članku je podanih nekaj osnovnih dejstev o PTP in pregled pomembnejšh algoritmičnih pristopov k njegovemu reševanju.

Abstract

The Traveling Salesman Problem (TSP) is one among the basic problems of combinatorial optimization. In the paper some basic facts about TSP are given, followed by an overview of the main algorithmic approaches for solving TSP.

■■■

Članek je nastal na osnovi gradiva za 2. seminar Izbrana poglavja iz računalništva na FNT, Oddelek za matematiko in mehaniko, Ljubljana v decembru 1992. Glavni cilj sestavka je bralca seznaniti s problemom trgovskega potnika (PTP) in s pomembnejšimi pristopi k njegovemu reševanju. Pri tem se ne spuščamo v podrobnosti, temveč bralca, ki bi želel izvedeti več, napotimo na ustrezne vire. Prav tako so iz pregleda izpuščeni algoritmi za posamezne posebne vrste PTP (na primer Evklidski) [10].

1. OSNOVE

1.1 Optimizacijske naloge

Naj bo na množici Φ dana funkcija

$$P : \Phi \rightarrow IR^*$$

kjer je $IR^* = IR \cup \{+\infty, -\infty\}$. Množici Φ bomo rekli *množica dopustnih rešitev*, funkciji P pa *namenska* ali *kriterijska* funkcija.

Pogosto pri določitvi množice dopustnih rešitev Φ izhajamo iz širše *množice rešitev* Ω, ki jo je enostavneje opisati. Množico dopustnih rešitev Φ tedaj sestavljajo tiste rešitve $X \in \Omega$, ki zadoščajo predikatu *dopustnosti* ali *omejitvam* $\Phi(X)$

$$\Phi = (\Omega, \Phi) = \{X \in \Omega : \Phi(X)\}$$

Postavimo

$$Min(\Phi,P) = \{X \in \Omega : \forall Y \in \Phi : P(Y) \geq P(X)\}$$

Vsi elementi množice Min(Φ,P), če je ta neprazna, imajo isto vrednost kriterijske funkcije P. Označimo jo z min(Φ,P) in jo razširimo na primer, ko je množica Min(Φ,P) prazna, s predpisom:

$$\min(\Phi,P) = \begin{cases} \inf_{X \in \Phi} P(X) & \Phi \neq \emptyset \\ \infty & \Phi = \emptyset \end{cases}$$

Tedaj lahko zapišemo tudi

$$Min(\Phi,P) = \{X \in \Phi : P(X) = \min(\Phi,P)\}$$

Na podoben način lahko vpeljemo tudi Max(Φ,P) in max(Φ,P); ali pa z uporabo zvez:

Ext.1 $\quad Max(\Phi,P) = Min(\Phi,-P)$
Ext.2 $\quad \max(\Phi,P) = -\min(\Phi,-P)$

Ti dve zvezi nam omogočata, da se omejimo samo na Min in min.

Optimizacijsko nalogo ali nalogo (matematičnega) programiranja dobimo, če dodamo dvojici (Φ,P), glede na značilnosti naloge, vsaj eno od naslednjih, ali njim podobnih, zahtev:

P1. določi Min(Φ,P)
P2. določi $X \in Min(\Phi,P)$
P3. določi min(Φ,P)

P4. določi zaporedje ($X_i : X_i \in \Phi$, $i \in IN$), tako da velja $\lim_{i\to\infty} P(X_i) = \min(\Phi,P)$

P5. določi $X \in \Phi$, tako da bo (z dano verjetnostjo) razlika $P(X) - \min(\Phi,P)$ zadosti majhna.

P6. določi $X \in \Phi$.

Tako zastavljene naloge bi pravzaprav morali označiti (Φ,P,min), kajti za vsako obstaja tudi naloga (Φ,P,max), ki se od nje razlikuje le po tem, da so v pogojih minimumi zamenjani z maksimumi.

Optimizacijski nalogi (Φ,P,min) in (Ψ,Q,min) sta *enakovredni*, če iz rešitev ene dobimo rešitve druge in obratno. Relacijo enakovrednosti označimo z ≅. Zaradi zvez Ext.1 in Ext.2 je naloga (Φ,P,max) enakovredna nalogi (Φ, -P,min). Zato se lahko v nadaljnem omejimo samo na naloge minimizacije.

Množica optimizacijskih nalog sestavlja *optimizacijski problem*. Običajno združimo v optimizacijski problem med seboj podobne naloge, ki imajo enako obliko in se razlikujejo le po podatkih. Nalogi, ki pripada optimizacijskemu problemu, pravimo tudi *primerek* problema.

1.1.1. Naloga o prirejanju

n ljudi mora opraviti n opravil. Stroški, ki jih imamo, če človek i opravi opravilo j naj bodo a_{ij}. Ljudem moramo prirediti vsakemu po eno opravilo, tako da bodo skupni stroški najmanjši, pri čemer pa morajo biti opravljena vsa opravila. Zastavljeni problem lahko prevedemo na optimizacijsko nalogo (S_n, P, min), kjer je S_n množica vseh permutacij števil od 1 do n in

$$P(\pi) = \sum_{i=1}^{n} a_{i,\pi(i)}$$

V linearni algebri ponavadi permutacijo π podamo z dvojiško matriko X, določeno s predpisom

$$xij = \begin{cases} 1 & j = \pi(i) \\ 0 & \text{sicer} \end{cases}$$

nalogo o prirejanju pa oblečemo takole: (Φ, P, min), kjer je I = 1..n ,

$$\Phi = \{X \in \{0,1\}^{n^2} : \forall i \in I : \sum_{j\in I} x_{ij} = 1, \forall j \in I : \sum_{i\in I} x_{ij} = 1\}$$

in

$$P(X) = \sum_{(i,j)\in I\times I} a_{ij}x_{ij}$$

Pogoji v opisu množice Φ zagotavljajo, da je X matrika neke permutacije. Iz tega zapisa naloge o prirejanju vidimo, da sodi med naloge (celoštevilskega) linearnega programiranja.

1.1.2. Naloga o trgovskem potniku

Trgovec se je namenil, da bo obšel n mest tako, da se bo v vsakem mudil samo enkrat. Znane so razdalje d_{ij} med posameznimi mesti. Kako naj trgovec potuje, da bo opravil čim krajšo pot?

Pri formalizaciji problema nadomestimo zemljevid z grafom povezanosti mest G = (V,E). Točke tega grafa so mesta, povezave pa predstavljajo prometne zveze med mesti. Vsaki povezavi je pripisana še pripadajoča razdalja.

Vsakemu trgovčevemu potovanju ustreza Hamiltonov cikel po grafu G. Tega lahko popišemo z urejeno n-terko točk π, sestavljeno po pravilu

j = π(i) - točka j je v ciklu naslednik točke i

Če naj π popisuje Hamiltonov cikel, mora biti ciklična permutacija. Tako dobimo nalogo (Γ_n, P, min), kjer je Γ_n množica cikličnih permutacij števil od 1 do n in

$$P(\pi) = \sum_{i\in I} d_{i,\pi(i)}$$

Najbrž ni potrebno posebej opozarjati na podobnost problema o trgovskem potniku s problemom o prirejanju. Žal je podobnost le površinska - za problem o prirejanju obstajajo učinkoviti algoritmi; problem trgovskega potnika pa je NP-težek, kar med drugim pomeni, da imajo vsi znani (točni) algoritmi zanj eksponentno zahtevnost. Je pa ta podobnost s pridom izkoriščena v nekaterih algoritmih za reševanje problema trgovskega potnika.

1.2. Grafi in problem trgovskega potnika

Naj bo G = (V,A) končen enostaven usmerjen graf z množico točk V in množico povezav $A \subseteq V \times V$.

Končno zaporedje točk $\gamma = (v_0, v_1, v_2, \ldots, v_d)$, ki zadošča pogoju

$$\forall i \in 1..d : (v_{i-1}, v_i) \in A$$

imenujemo *sprehod po* G; d je dolžina sprehoda γ. Če je $v_0 = v_d$, je sprehod *sklenjen* ali *obhod*. Sprehod je *osnoven* ali *pot*, če gre skozi posamezno točko grafa največ enkrat. Osnovnemu obhodu rečemo krajše *cikel*. Sprehod je *poln*, če gre skozi vse točke grafa. Polnemu ciklu pravimo tudi *Hamiltonov cikel*.

Če je A = V x V, je graf G *poln*. V polnem grafu ustrezajo Hamiltonovim ciklom natanko vse ciklične permutacije (z začetno točko).

Urejeno trojico G = (V, A, c), kjer je (V, A) graf in $c : A \to IR$ cena povezav, imenujemo *graf z vrednostmi na povezavah*. V nadaljnjem bomo predpostavljali, da je cena nenegativna $c : A \to IR_0^+$.

Cena c zadošča *trikotniški neenakosti*, če zanjo velja

$$\forall x, y, z \in V : c(x,z) + c(z,y) \geq c(x,y)$$

Če ima poleg tega še lastnost $\forall v \in V : c(v,v) = 0$, ji bomo rekli *oddaljenost*. Pozor, oddaljenost ni nujno simetrična $\forall x,y \in V$: $c(x,y) = c(y,x)$. V primeru, ko lahko točke grafa predstavimo kot točke v IR^k in je cena povezave enaka kar Evklidski razdalji med njenima krajiščema, govorimo o *Evklidski* nalogi TP. Pogosto se pri predstavitvi omejimo na ravnino IR^2.

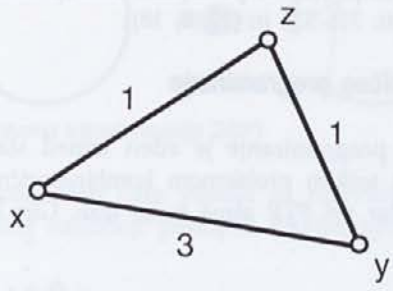
Ceno lahko razširimo s povezav na sprehode s predpisom

$$c(\gamma) = \sum_{i=1}^{d} c(v_{i-1},v_i)$$

Morda pa bi lahko trgovski potnik opravil svoje potovanje ceneje, če bi kakšno od mest obiskal večkrat. To je res v primeru na sliki 1, za katerega velja:

$$c((x,z,y,z,x)) = 4 \quad \text{in} \quad c((x,y,z,x)) = 5.$$

Torej minimalne rešitve naloge TP niso vselej Hamiltonovi cikli, temveč jih moramo iskati med polnimi obhodi.



Slika 1: *Minimalne rešitve naloge TP niso vselej Hamiltonovi cikli*

Naj bo $\Gamma(A)$ množica vseh polnih obhodov grafa G. Tedaj lahko *nalogo trgovskega potnika* zapišemo kot optimizacijsko nalogo

$$TP = (\Gamma(A), c, \min)$$

Množica vseh nalog trgovskega potnika *sestavlja problem trgovskega potnika.*

Dano ceno $\hat{c}$ razširimo na množico V x V s predpisom

$$\hat{c}(u,v) = \begin{cases} c(u,v) & (u,v) \in A \\ \infty & \text{sicer} \end{cases}$$

Tedaj velja

Izrek 1. *Naloga* TP $(\Gamma(A), c, \min)$ *je enakovredna nalogi* $(\Gamma(V \times V), \hat{c}, \min)$.

Množico Hamiltonovih ciklov grafa G označimo s H(A). Nalogo (H(A), c, min) *imenujemo naloga o najcenejšem Hamiltonovem ciklu.*

Čeprav v splošnem rešitev NTP ni vedno Hamiltonov cikel, pa velja:

Izrek 2. *Če v grafu z vrednostmi na povezavah* (V,V x V, c) *velja za ceno* c *trikotniška neenakost, obstaja Hamiltonov obhod* $\gamma \in \text{Min}(\Gamma(V \times V),c)$.

Grafu G = (V,A,c) lahko vselej priredimo nov graf H = (V,V x V,d) takole: Naj bo za u, v ∈ V <u,v> najcenejši sprehod (ali eden izmed njih, če jih je več) iz u v v po G. Tedaj postavimo:

$$d(u,v) = \sum_{(i,j) \in <u,v>} c_{ij}$$

Nova cena d zadošča trikotniški neenakosti.

Izrek 3. *Naloga* TP = $(\Gamma(A), c, \min)$ *po G je enakovredna nalogi* $(H(V \times V), d, \min)$ *najcenejšega Hamiltonovega cikla v H.*

Torej je vsaka naloga TP prevedljiva na enakovredno nalogo iskanja najcenejšega Hamiltonovega cikla v grafu s ceno, ki zadošča trikotniški neenakosti.

2. ALGORITMI

Problem trgovskega potnika (potujočega trgovca, kramarja) ima svoje začetke v zabavnih (matematičnih) nalogah [1] kot sta *naloga o požrešnem šahovskem konjičku* (Euler, 1759; Vandermonde, 1771) in *dvajsetiška igra* (Sir William Rowan Hamilton, 1856). Prvi je obravnaval splošni problem obstoja Hamiltonovih ciklov Kirkman 1855.

Pravi problem trgovskega potnika je obravnavan v knjigi nasvetov za trgovske potnike, ki je izšla v Nemčiji leta 1832.

V matematične kroge je PTP zašel v tridesetih letih tega stoletja (Menger, Whitney H., Tucker A.W., Flood M.). Okrog leta 1948 je RAND corporation, kjer je bilo takrat središče operacijskih raziskav, razpisal nagrado za pomemben izrek o PTP.

Leta 1954 so Dantzig, Fulkerson in Johnson objavili prvo rešitev obsežne NTP - 49 mest ZDA; 48 velikih mest v zveznih ameriških državah in Washington D.C. Rešitev so določili z nitko na modelu. Optimalnost pa so dokazali z uporabo linernega programiranja, pri katerem so z *metodo odsekov* (dodatne omejitve) zagotovili cikličnost in celoštevilskost rešitve. Zadostovalo je 25 dodatnih omejitev. Pri pripravi rešitve so uporabili tudi osnovne zamisli *metode razveji in omeji.* Uporabo odsekov pri reševanju nalog celoštevilskega linearnega programiranja je naprej razvijal Gomory.

V tistih časih, začetek petdesetih, so se pojavili prvi pomembnejši rezultati v kombinatorični optimizaciji (linearno programiranje, prirejanja, pretoki, razporejanje poslov, ...). Zato je trdovratnost PTP privlačila posebno pozornost.

Leta 1963 so Little, Murty, Sweeney in Karel objavili postopek razveji in omeji za reševanje PTP in prvi uporabili termin *branch & bound*. Postopka razveji in omeji sta za reševanje PTP predlagala že leta 1958 tudi Eastman in Croes. Leta 1970 sta Held in Karp predložila za ocenjevanje mej uporabo *Lagrangeovske relaksacije*.

Mesto PTP so v svojih člankih o zahtevnosti problemov razjasnili Cook (1971), Karp (1972) in Levin (1973). PTP je NP-težek; prav tako tudi večina podproblemov (npr. Evklidski). Ti rezultati nas vodijo v dve smeri:

- za točno reševanje PTP se najbolje obnesejo izpeljanke metode razveji in omeji. Pri tem je prava umetnost razvoj čim boljših ocen mej. Leta 1980 sta Crowder in Padberg rešila NTP na 318 točkah, ki je bila do 1987, najobsežnejša naloga z dokazano optimalno rešitvijo. Leta 1987 sta Padberg in Rinaldi objavila optimalne rešitve nalog velikosti 532, 1002 in 2392; leta 1988 pa Grötschel in Holland 666 in 1000 [13, 7].
- če je naloga preobsežna ali imamo omejene vire, se moramo zadovoljiti s približnimi rešitvami. Do teh lahko pridemo z raznimi *hevristikami*, kakršna je na primer *najbližji sosed* (Menger, 1930), ali pa s *postopki lokalne optimizacije*.

Poglejmo si posamezne pristope:

2.1 Prebor vseh možnosti

Najpreprostejši točen postopek je prebor vseh možnosti - vseh (n-1)! cikličnih permutacij.

```
CONST nmax = 20;
TYPE  vector = ARRAY [1..nmax] OF integer;
      matrix = ARRAY [1..nmax, 1..nmax] OF integer;
.....
VAR   q, route : vector;
      w : matrix;
.....
PROCEDURE perm(k:integer);
  VAR i, t: integer;
BEGIN
  IF k<=2 THEN BEGIN
     c := c+1; ts := w[q[n],q[1]]; write(lst,c:8,' > ');
     FOR i := 1 TO n DO write(lst,q[i]:3);
     FOR i:=1 TO n-1 DO ts := ts + w[q[i],q[i+1]];
     writeln(lst,ts:8);
     IF ts < bts THEN BEGIN bts := ts; route := q END;
  END ELSE BEGIN
     perm(k-1);
     FOR i:= 2 TO k-1 DO BEGIN
       t := q[i]; q[i] := q[k]; q[k] := t;
       perm(k-1);
       t := q[i]; q[i] := q[k]; q[k] := t
     END;
  END;
END {perm};
BEGIN
.....
     FOR i := 1 TO n DO q[i] := i;
     c := 0; bts := maxint;
     perm(n);
.....
END.
```

Žal je ta postopek uporaben le za zelo majhne naloge. Pri polnem preboru rešitev moramo pregledati (n-1)! permutacij.

n	(n-1)!	n	(n-1)!
10	362880	16	1307674368000
11	3628800	17	20922789888000
12	39916800	18	355687428096000
13	479001600	20	121645100408832000
14	6227020800	25	6204484017332394393600000
15	87178291200	30	8841761993739701954543616000000

Recimo, da program pri polnem preboru (na superračunalniku) pregleda milijon rešitev na sekundo. Tedaj bi pri n= 15 tekel 24.2 ur; pri n = 18 pa že 11.3 let in pri n=30 kar 2.8 10^{17} let.

2.2. Razveji in omeji

Postopek polnega prebora lahko izboljšamo tako, da s kleščenjem slabih vej, omejimo pregledovanje na obetavne veje. To je osnova postopkov razveji in omeji. Glej Kozak [9], str. 316-324 in [12, 8, 18)].

2.3. Dinamično programiranje

Dinamično programiranje je eden izmed standardnih pristopov k težkim problemom kombinatorične optimizacije; vendar pri PTP nima večje teže. Glej Kozak [9], str. 269-271.

2.4. Monte Carlo – naključne permutacije

Pri zelo obsežnih problemih bi lahko dani obhod naključno določili in izračunali njegovo vrednost

```
FOR i := n DOWNTO 2 DO BEGIN
  j := 1 + trunc(i*random);
  t := route[i]; route[i] := route[j]; route[j] := t;
END;
ts := w[route[n],route[1]];
FOR i:=1 TO n-1 DO ts := ts + w[route[i],route[i+1]];
```

To bi velikokrat ponovili in obdržali najboljšo dobljeno rešitev.

Ta zamisel ne vodi do zadovoljivih rezultatov. Uporabili pa jo bomo za določanje naključnih začetnih rešitev pri postopkih lokalne optimizacije.

2.5. Lokalna optimizacija

2.5.1. Globalni in lokalni minimumi

Pogosto lahko v dani množici Ω definiramo relacijo *sosednosti* rešitev $S \subseteq \Omega \times \Omega$, za katero zahtevamo le refleksivnost.

Pri diskretnih optimizacijskih problemih običajno definiramo sosednost z *lokalnimi transformacijami*, ki prevedejo eno rešitev v drugo.

Element $x \in \Phi$ je lokalni minumum glede na S natanko takrat, ko je $x \in \text{Min}(\Phi \cap S(x),P)$; ali drugače povedano, ko

$$\forall \in \Phi \cap S(x) : P(x) \leq P(y)$$

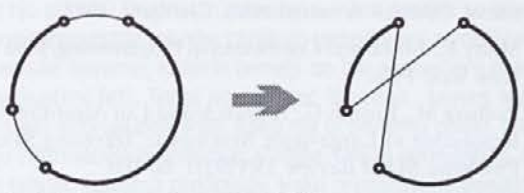
Množico vseh lokalnih minimumov naloge (Φ, P, min) glede na sosednost S označimo LocMin(Φ, P, S).

Zato, da bi poudarili razliko, pravimo elementom množice Min(Φ, P) tudi *globalni minimumi*. Očitno je vsak globalni minimum tudi lokalni. Poleg tega velja še:

Izrek 4. *Za vsako sosednost* S *je*

$$\text{LocMin}(\Phi, P, \Phi \times \Phi) = \text{Min}(\Phi, P) \subseteq \text{LocMin}(\Phi, P, S)$$

in, če je $\emptyset \subset Q \subset S$, *tudi* $\text{LocMin}(\Phi,P,S) \subseteq \text{LocMin}(\Phi,P,Q)$.

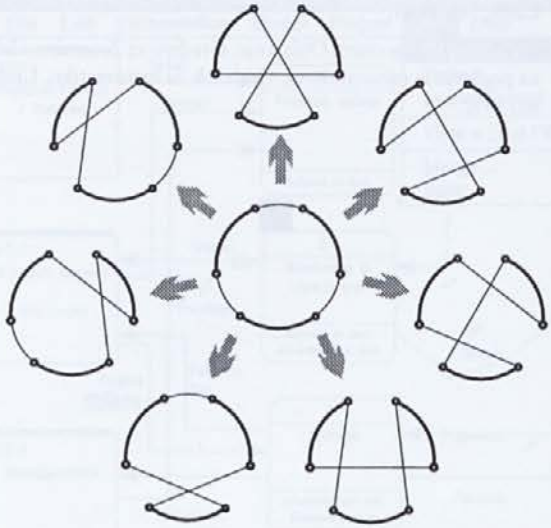


Slika 2. *Lokalna transformacija* 2-OPT

Relacija sosednosti nam ponuja za reševanje optimizacijskih nalog naslednji postopek lokalne optimizacije:

izberi $x \in \Phi$;
while $\exists y \in S(x) \cap \Phi : P(y) < P(x)$ **do** $x := y$;

Če se postopek izteče v končno korakih, konča v lokalnem minimumu.



Slika 3. *Lokalne transformacije 3-OPT*

Pri problemu trgovskega potnika se najpogosteje uporabljajo sosednosti r-OPT, pri katerih so sosednje rešitve določene tako, da iz tekočega cikla odstranimo r povezav in jih nadomestimo z novimi, ki zopet sestavljajo cikel. Na slikah 2 in 3 sta prikazani sosednosti 2-OPT in 3-OPT. Lin je s poskusi pokazal, da je sosednost 3-OPT veliko boljša kot sosednost 2-OPT, učinek sosednosti višjih redov pa ne upravičuje povečane porabe časa [14, 18].

2.5.2. Ohlajanje

Ohlajanje (simulated annealing) je izpeljanka postopka lokalne optimizacije, ki se poskuša s posnemanjem gibanja delcev plina izogniti lokalnim minimumom. Prvi ga je okrog leta 1953 predlagal Metropolis s sodelavci; zelo priljubljen pa je postal v zadnjem desetletju. Postopek lokalne optimizacije z ohlajanjem za PTP najdemo v [15] str. 328-334.

Podrobneje je uporaba ohlajanja pri PTP obdelana v [19].

2.5.3. Prepovedane smeri

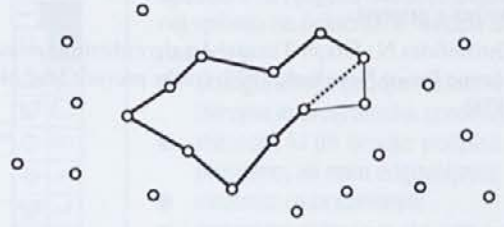
Zadnjih nekaj let se uveljavlja druga izboljšava postopkov lokalne optimizacije - uporaba *prepovedanih smeri* (Tabu search) [6]. Pri teh postopkih se vselej premaknemo v sosednjo dovoljeno rešitev z najmanjšo vrednostjo, pri čemer lahko nekatere premike prepovemo. To nam omogoča, da se skobacamo iz lokalnih minimumov in da upoštevamo nabrano informacijo o prostoru rešitev.

Po poročilih v člankih se pri PTP ti postopki izjemno dobro obnesejo.

2.5.4. Genetski algoritmi

Eden izmed obetajočih pristopov k (približnemu) reševanju kombinatoričnih optimizacijskih problemov so tudi *genetski algoritmi*, pri katerih poskušamo s posnemanjem razvoja (reprodukcija, križanje, mutacije) množice rešitev vzgojiti čim boljše rešitve [4].

2.6. APROKSIMACIJSKI ALGORITMI



Slika 4. *Širjenje delnega cikla*

Hevristični algoritmi pogosto temeljijo na načelu *požrešnosti*. Pri PTP lahko zgradimo take postopke tako, da zaporedoma dodajamo točke delnemu ciklu. Postopki se med seboj ločijo po odločitvah: *katero* točko dodati in *kam*.

Pri zelo velikih nalogah lahko poskušamo priti do dobrih rešitev tudi tako, da v postopkih razveji in omeji obdržimo le najbolj obetavne veje.

Označimo $c^* = \min(H(V \times V), c)$ in s $c(A)$ ceno rešitve, ki jo vrne dani aproksimacijski algoritem A. Zanima nas, ali obstaja taka konstanta α, da velja

$$1 \leq \frac{c(A)}{c^*} \leq \alpha$$

Za algoritem *najbližji sosed* je mogoče pokazati, da velja

$$\alpha = \frac{1}{2}(\lceil \ln n \rceil + 1)$$

Sahni in Gonzales sta leta 1976 pokazala, da je tudi problem aproksimacije za splošni PTP NP-težek.

Dobro aproksimacijo, $\alpha < \frac{3}{2}$, lahko zagotovimo v primeru, ko je cena c razdalja. Kako to storimo je opisano v [11].

3. ZAKLJUČEK

Povzemimo: čeprav je PTP NP-težek problem, ni treba takoj vreči puške v koruzo. Če narava naše naloge ne zahteva točne rešitve, lahko pridemo razmeroma hitro do precej dobrih rešitev, tudi za naloge z več tisoč točkami, z uporabo postopkov lokalne optimizacije ali njihovih izboljšav. Te postopke je razmeroma enostavno sprogramirati. Za naloge TP, pri katerih je zahtevana točna rešitev, je doseg najboljših postopkov okrog 2000 točk. Ti postopki zahtevajo za svoj razvoj veliko znanja in časa. V našem prostoru (vsaj kolikor je avtorju sestavka znano) še niso dostopni.

LITERATURA

1. Batagelj V.: Hamiltonova naloga za grafe. 8. seminar iz matematike: Zanimiva matematika, DMFA SRS, Ljubljana, 1980, 13-25; Presek 11(1983-84)1, 4-16.
2. Batagelj V.: Optimizacijske metode. Zapiski predavanj na FER, skripta v pripravi.
3. Christofides N.: Graph Theory; An algorithmic approach. Academic Press, New York, 1975. (ruski prevod: Mir, Moskva, 1978)
4. Filipič B.: Primerjava genetskih algoritmov na problemu trgovskega potnika. Zbornik XXXV ETAN, Ohrid 1991, 257-264.
5. Gibbons A.: Algorithmic Graph Theory. Cambridge University Press, Cambridge, 1985.
6. Glover F.: Tabu search. ORSA Journal on Computing. Part I: 1(1989)3, 190-206; Part II: 2(1990)1, 4-32.
7. Grötschel M., Holland O.: Solution of large-scale symmetric traveling salesman problems. Mathematical Programming 51(1991), 141-202.
8. Horowitz E., Sahni S.: Fundamentals of Computer Algorithms. Computer Science Press, Rockville, 1978.
9. Kozak J.: Podatkovne strukture in algoritmi. DMFA SRS, Ljubljana, 1986.
10. Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B.: The Traveling Salesman Problem; A guided tour of combinatorial optimization. John Wiley, New York, 1986.
11. Mohar B.: Aproksimacijski postopki za problem trgovskega potnika. Gradivo za 2. seminar: Izbrana poglavja iz računalništva, Oddelek za matematiko, Ljubljana, 1992.
12. Murty K.: Linear and Combinatorial Programming. John Wiley, New York, 1976.
13. Padberg M., Rinaldi G.: A Branch-and-Cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesman Problems. SIAM Review 33(1991)1, 60-100.
14. Papadimitriou C.H., Stieglitz K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs, NJ, 1982.
15. Press W.H., Flannery B.P., Teukolsky S.A., Vetterling W.T.: Numerical Recipes; The art of scientific computing. Cambridge University Press, Cambridge, 1986.
16. Reingold E.M., Nievergelt J., Deo N.: Combinatorial Algorithms; Theory and practice. Prentice-Hall, Englewood Cliffs, NJ, 1977. (ruski prevod: Mir, Moskva, 1980)
17. Sakovič V.A.: Issledovanie operacij. Višejšaja škola, Minsk, 1985.
18. Syslo M.M., Deo N., Kowalik J.S.: Discrete Optimization Algorithms; with pascal programs. Prentice-Hall, Englewood Cliffs, NJ, 1983.
19. Žerovnik J.: Algoritem Ohlajanje. Gradivo za 2. seminar: Izbrana poglavja iz računalništva, Oddelek za matematiko, Ljubljana, 1992.