Unitatea de învăţare 6

INTRODUCERE ÎN OPTIMIZAREA COMBINATORIALĂ
Problema comisvoiajorului (Traveling Salesman Problem ≡ TSP)

Cuprins

Obiectivul unităţii de învăţare 6

Problema comisvoiajorului a fost déjà introdusă în unitatea de învăţare 4 ca exemplu reprezentativ de problemă de optimizare combinatorială "grea". Tot acolo au fost examinate câteva posibilităţi de modelare cu ajutorul grafurilor sau a programării în numere întregi. TSP are numeroase aplicaţii practice; unele au fost déjà semnalate, altele vor fi date în continuare. În ciuda simplităţii sale, „se cere determinarea celui mai scurt traseu de vizitare a unor puncte cu întoarcerea în locul de plecare", rezolvarea problemei comisvoiajorului rămâne o veritabilă provocare mai cu seamă în cazul în care numărul punctelor este "mare".

Obiectivul unităţii de învăţare 6 este de a trece în revistă câteva metode de rezolvare – în general suboptimală – a TSP cu ilustrările numerice de rigoare.

6.1 Formularea problemei. Clasificare. Aplicaţii

Exemplul 1 O companie petrolieră are mai multe platforme de foraj marin. Periodic, un elicopter cu baza pe uscat, vizitează platformele în vederea efectuării unor controale de rutină după care se întoarce la bază. În ce ordine vor fi vizitate platformele pentru ca distanţa totală parcursă să fie minimă?

Exemplul de mai sus constituie una din numeroasele concretizări ale **problemei comisvoiajorului** (abreviat **TSP** ≡ **T**raveling **S**alesman **P**roblem):

Un comisvoiajor în localitatea 0 doreşte să viziteze localităţile 1,2,…,*n* la sfârşitul călătoriei el întorcându-se de unde a plecat. Nu există nici o restricţie în ceeace priveşte deplasarea de la un oraş la altul. Pentru oricare două oraşe diferite $i \neq j$ se cunoaşte costul c_{ij} al deplasării comisvoiajorului din *i* în *j* (în alte contexte c_{ij} poate reprezenta distanţa dintre oraşele *i* şi *j* sau timpul necesar efectuării deplasării din *i* în *j*) În ce ordine vor fi vizitate localităţile 1,2,…,*n* astfel încât să se minimizeze costul total al călătoriei (sau distanţa totală parcursă sau durata totală a deplasării).

O problemă a comisvoiajorului se numeşte **simetrică** dacă:

$$c_{ij} = c_{ji} \text{ pentru orice } i \neq j$$

altminteri ea se zice **asimetrică**.

Între problemele simetrice, o clasă importantă este formată din acelea în care "costurile" c_{ij} verifică **inegalitatea triunghiului**:

$$c_{ij} + c_{jk} \geq c_{ik} \text{ pentru orice tripletă de indici diferiţi } i, j, k.$$

Aceste probleme se numesc **euclidiene.**

De exemplu, o problemă care implică deplasări **în linie dreaptă** este o problemă euclidiană. Iată un exemplu de situaţie practică care nu implică "deplasări" şi care conduce totuşi la o TSP, în general asimetrică.

Exemplul 2 Stabilirea ordinii de lansare în execuţie a unor repere. Pe un utilaj se execută mai multe repere (operaţii). Fiecare reper (operaţie) are o durată de execuţie cunoscută. De multe ori se întâmplă ca trecerea de la executarea unui reper la executarea altuia să necesite un anumit timp de pregătire al utilajului mai mic sau mai mare funcţie de caracteristicile celor două repere dar şi de ordinea în care acestea sunt procesate. De exemplu, într-o situaţie de croire, trecerea de la utilizarea (repetată) a unei reţete de debitare la o altă reţetă implică o "întrerupere" a activităţii, necesară repoziţionării cuţitelor de tăiere.

În acest context, se pune problema stabilirii ordinii de lansare în execuţie a reperelor astfel încât suma timpilor de pregătire să fie minimă.

6.2 Problema firmei de curierat rapid

Problema firmei de curierat rapid (**D**ial **a** **R**ide **P**roblem, abreviat **DARP**) are următoarea formulare generală:

Sunt date o reţea şi o listă de comenzi de transport. În fiecare comandă se specifică sursa şi destinaţia transportului care trebuie să fie noduri ale reţelei.Transporturile sunt efectuate de un server care pleacă dintr-un anumit nod al reţelei şi care, după satisfacerea tuturor comenzilor, se întoarce în punctul de plecare. Capacitatea serverului este limitată în sensul că el nu poate transporta mai mult de un anumit număr de comenzi simultan.Toate deplasările serverului se fac pe muchiile reţelei. Unele muchii pot fi parcurse în ambele sensuri altele doar într-un singur sens, prestabilit. Se presupun cunoscute lungimile tuturor muchiilor din reţea.
Problema de optimizare constă în a stabili cum vor fi transportate comenzile pentru ca distanţa totală parcursă de server să fie minimă.

În unele situaţii, o comandă poate prevedea şi o fereastră de timp în care are loc fie preluarea comenzii de la sursă fie predarea acesteia la destinaţie.

Problema are numeroase aplicaţii:

- în activitatea firmelor de curierat rapid sau taximetrie;
- în transportul persoanelor cu handicap locomotor;
- în transportul pe verticală de mărfuri sau persoane cu ajutorul ascensorului;
- în deplasarea pe orizontală a unor obiecte cu ajutorul unui pod rulant, macara sau braţ mecanic.

DARP este o generalizare a problemei comisvoiajorului. Într-adevăr, orice TSP se identifică cu o DARP în care:

- comenzile corespund oraşelor de vizitat;
- sursa şi destinaţia fiecărei comenzi coincid cu oraşul corespunzător comenzii.

În consecinţă, **DARP este o problemă de optimizare combinatorială grea.**

În continuare vom studia DARP în cel mai simplu caz, în care:

-serverul nu poate transporta o dată mai mult de o comandă;
- nu există ferestre de timp.

şi vom arăta că această problemă particulară este echivalentă cu o problemă a comisvoiajorului, asimetrică.

Vom folosi notaţiile:

- G ≡ reţeaua de transport cu nodurile 0,1,…,*n*, nodul 0 fiind punctual de plecare şi întoarcere al serverului;

- $d_{ij} \equiv$ distanţa de la nodul *i* la nodul *j* măsurată pe drumul cel mai scurt existent în reţea între *i* şi *j*. După cum se ştie, determinarea celor mai scurte drumuri între nodurile unui graf este o problemă de optimizare (combinatorială) **uşoară**, rezolvabilă de exemplu cu algoritmul lui FLOYD. Întrucât este posibil ca unele muchii să fie parcurse doar într-un sens, se poate întâmpla ca $d_{ij} \neq d_{ji}$ pentru anumite perechi de noduri.
- $R = \{r_1, r_2, ..., r_m\} \equiv$ lista comenzilor de transport cu $r_i = (s_i, t_i)$ unde s_i este sursa comenzii r_i iar t_i este destinaţia ei. s_i şi t_i sunt noduri ale grafului G.

Definim un **traseu complet** al serverului ca fiind o succesiune de noduri şi arce **permise** din graful G:

$$\lambda : 0 \equiv i_0 \to i_1 \to i_2 \to \cdots \to i_{p-1} \to i_p \equiv 0 \qquad (1)$$

care începe şi sfârşeşte în nodul 0 şi are proprietatea:

Pentru fiecare comandă $r_i = (s_i, t_i)$ nodurile s_i , t_i se găsesc printre nodurile din (1), nodul t_i apărând **în urma** nodului s_i.

Deoarece serverul nu poate transporta mai mult de o comandă o dată, orice traseu complet este o **reuniune de drumuri** de forma:

$$\lambda \equiv \mu_1 \cup \lambda_1 \cup \mu_2 \cup \lambda_2 \cup \quad \cdots \cup \mu_m \cup \lambda_m \cup \mu_{m+1} \qquad (2)$$

unde:

$\mu_1 \equiv$ porţiunea din λ de la nodul 0 la sursa primei comenzi transportate;
$\lambda_1 \equiv$ porţiunea din λ de la sursa la destinaţia primei comenzi transportate;
$\mu_2 \equiv$ porţiunea din λ de la destinaţia primei comenzi la sursa celei de a doua comenzi transportate ş.a.m.d.
…
$\mu_{m+1} \equiv$ porţiunea din λ pe care serverul o parcurge de la destinaţia ultimei comenzi transportate la locul de parcare 0.

Este posibil ca unele dintre drumurile "în gol" $\mu_1, \mu_2, ..., \mu_{m+1}$ să se reducă la un nod şi astfel să aibe lungimea zero! Aceasta se întâmplă atunci când destinaţia unei comenzi coincide cu sursa următoarei comenzi transportate.
Este clar că, pentru minimizarea distanţei totale parcurse:

- serverul va transporta fiecare comandă $r_i = (s_i, t_i)$ pe drumul **cel mai scurt** de la s_i la t_i;
- de la destinaţia unei comenzi transportate la sursa următoarei comenzi, serverul va trebui să se deplaseze pe drumul **cel mai scurt**!

În consecinţă, în structura unui traseu complet dată în (2) putem presupune că $\lambda_1, \lambda_2, ..., \lambda_m$ sunt cele mai scurte drumuri pe care se pot transporta comenzile şi ca urmare suma

$d(\lambda_1)+d(\lambda_2)+\cdots+d(\lambda_m)$ a lungimilor acestor drumuri poate fi considerată o **constantă**, **independentă de ordinea** în care vor fi transportate comenzile.

Tot aşa, putem presupune că drumurile "în gol" $\mu_1,\mu_2,....,\mu_{m+1}$ sunt cele mai scurte drumuri posibile, însă structura lor şi de aici suma lungimilor lor depinde **esenţial de ordinea** în care comenzile sunt transportate!

În concluzie, minimizarea distanţei totale parcurse de server este echivalentă cu determinarea ordinii în care vor fi transportate comenzile astfel încât **suma lungimilor drumurilor "în gol" să fie minimă**.

Aceasta este o problemă a comisvoiajorului, asimetrică, în care:

- "oraşele" sunt comenzile $r_0,r_1,....,r_m$ unde $r_0=(0,0)$ este o comandă fictivă cu sursa şi destinaţia concentrate în nodul 0;
- pentru oricare două oraşe ≡ comenzi diferite $r=(s,t)$ şi $r'=(s',t')$ "costul $c_{r,r'}$ al deplasării de la *r* la *r'* " este lungimea celui mai scurt drum de la destinaţia *t* a comenzii *r* la sursa *s*' a comenzii următoare *r*'.

6.3 Un algoritm de tip B&B pentru TSP în cazul asimetric

Reamintim că problema comisvoiajorului a fost modelată ca o problemă de afectare:

$$\begin{cases} (\min) z=\sum\limits_{i=0}^{n}\sum\limits_{j=0}^{n} c_{ij}x_{ij} \\ \sum\limits_{j=0}^{n} x_{ij}=1 \qquad i=0,1,..,n \\ \sum\limits_{i=0}^{n} x_{ij}=1 \qquad j=0,1,..,n \\ x_{ij}\in\{0,1\} \end{cases} \tag{1}$$

cu restricţii „speciale", menite să elimine apariţia subtraseelor, variabilele bivalente x_{ij} având semnificaţia:

$$x_{ij}=\begin{cases} 1 \text{ daca comisvoiajorul merge direct din orasul } i \text{ in orasul } j \text{ si } i\neq j \\ 0 \text{ in caz contrar sau daca } i=j \end{cases}$$

unde $i,j=0,1,...,n$

(vezi unitatea de învăţare 4, secţiunea 4.2)

6.3.1 Algoritmul lui Eastman

Următorul algoritm, datorat lui EASTMAN (1958), determină traseul optim al unei probleme **asimetrice** a comisvoiajorului prin rezolvarea unei liste de probleme de **afectare** care diferă de problema (1) prin unele costuri c_{ij} schimbate în $+\infty$.

Algoritmul utilizează:

- locaţie x_{CMB} care reţine **C**el **M**ai **B**un traseu găsit pe parcurs;
- locaţie z_{CMB} care păstrează valoarea traseului depus în x_{CMB};
- listă $\mathcal{L}$ de probleme de afectare, rezolvabile cu algoritmul ungar.

Start Iniţializăm:

$$x_{CMB} = \varnothing \qquad z_{CMB} = +\infty \qquad \mathcal{L} = \{\text{problema de afectare (1)}\}$$

(Locaţia x_{CMB} se poate iniţializa şi cu un traseu fie cunoscut fie determinat printr-o euristică oarecare. Valoarea traseului se înregistrează în z_{CMB})

Conţinutul unei iteraţii:

Pasul 1 Dacă lista $\mathcal{L}$ este vidă, **Stop**: traseul de valoare minimă se găseşte în x_{CMB}. În caz contrar selectăm **prima** problemă de afectare din listă. Problema selectată se şterge din $\mathcal{L}$. Se trece la:

Pasul 2 Se rezolvă problema de afectare selectată (cu algoritmul ungar). Dacă valoarea optimă a funcţiei obiectiv este $\geq z_{CMB}$ se revine la pasul 1. Dacă valoarea optimă a funcţiei obiectiv este $< z_{CMB}$ se trece la:

Pasul 3 Examinăm soluţia optimă a problemei de afectare rezolvate la pasul 2. Dacă această soluţie este un **traseu complet** se actualizează x_{CMB} şi z_{CMB} cu traseul găsit, respectiv cu valoarea acestuia, după care se revine la pasul 1. Dacă soluţia examinată este o **reuniune de subtrasee disjuncte** se trece la:

Pasul 4 Se alege subtraseul **cu cele mai puţine arce**. Pentru fiecare arc (i , j) al subtraseului ales se adaugă, **în capul listei** $\mathcal{L}$, problema de afectare rezultată din problema rezolvată punând $c_{ij} = +\infty$ (deci la lista $\mathcal{L}$ se vor adăuga atâtea probleme, câte arce are subtraseul ales!). Se revine la pasul 1.

Observaţii: 1) Raţiunea pasului 4 este următoarea: dacă $i \to j \to k \to \cdots \to i$ este subtraseul ales, este clar că traseul optim nu va conţine unul din arcele $i \to j$ sau $j \to k$ sau … ,ce compun subtraseul.Altfel spus , în soluţia optimă a TSP vom avea $x_{ij} = 0$ sau $x_{jk} = 0$ sau …În consecinţă, traseul optim va fi căutat traseele în care $x_{ij} = 0$ apoi cele în care $x_{jk} = 0$ etc. Or, restrângerea la traseele în care $x_{ij} = 0$ se face punând $c_{ij} = +\infty$!

Bineînţeles că putem considera orice subtraseu din soluţia problemei rezolvate la pasul 2. Alegerea celui "mai mic" se face în ideea de a nu încărca "excesiv" lista $\mathcal{L}$!

2) În termenii generali ai principiului Branch & Bound, **ramificarea** are loc la pasul 4 iar **mărginirea** la pasul 2 (nu „strică" revederea secţiunii 3.2 din unitatea de învăţare 3!!)

3) Algoritmul este aplicabil problemelor **asimetrice** de dimensiune **moderată.**Punctul slab al procedurii îl constituie **impredictibilitatea** dimensiunii listei $\mathcal{L}$ ca şi a dinamicii acestei liste!
În principiu, algoritmul poate rezolva exact orice TSP însă aplicarea lui la cazul simetric nu este recomandată fiind foarte greoaie!

Exemplu numeric Se dă problema asimetrică a comisvoiajorului cu cinci localităţi 0, 1,...,4 definită de următoarea matrice a costurilor de deplasare c_{ij}:

Tabelul 6.1

	0	1	2	3	4
0	∞	10	25	25	10
1	1	∞	10	15	2
2	8	9	∞	20	10
3	14	10	24	∞	15
4	10	8	25	27	∞

0 este punctul de plecare şi întoarcere al comisvoiajorului. După cum se vede costul deplasării între oraşele 0 şi 4 nu depinde de sens: $c_{04} = c_{40} = 10$. În schimb, deplasarea de la 0 la 1 este mai „scumpă" decât deplasarea în sens invers: $c_{01} = 10 > 1 = c_{10}$. Se pune problema determinării unui traseu de vizitare a localităţilor cu plecarea şi întoarcerea în 0, care să treacă o singură dată prin 1, 2, 3 şi 4 şi care să coste cât mai puţin. Pentru aceasta vom aplica algoritmul descris mai sus.

Start

Iniţializăm: $x_{CMB} = \varnothing$ $z_{CMB} = +\infty$ $\mathcal{L} = \{\text{PA}\}$

unde PA este problema de afectare (1) cu costurile c_{ij} din tabelul 6.1

Iteraţia 1

1 Selectăm problema PA şi o ştergem din listă: $\mathcal{L} = \varnothing$;
2 Rezolvăm problema selectată; se obţine soluţia optimă:

$$x_{04} = x_{31} = x_{12} = x_{23} = x_{31} = 1 \quad , \quad x_{ij} = 0 \text{ în rest.}$$

care corespunde subtraseelor disjuncte $0 \to 4 \to 0$ şi $1 \to 2 \to 3 \to 1$.

- Selectăm subtraseul $0 \to 4 \to 0$ şi adăugăm la lista $\mathcal{L}$ problemele de afectare PA_1 şi PA_2 derivate din PA prin schimbarea $c_{04} = +\infty \quad c_{40} = +\infty$:

$$\mathcal{L} = \{\text{PA}_1 , \text{PA}_2\}$$

Iteraţia 2

- Selectăm problema PA_1 şi actualizăm lista : $\mathcal{L} = \{PA_2\}$.
- Rezolvarea problemei PA_1 conduce la soluţia

$$x_{03} = x_{31} = x_{12} = x_{24} = x_{40} = 1 \quad , \quad x_{ij} = 0 \text{ în rest}$$

care este un traseu complet cu valoarea 65. Actualizăm:

$$x_{CMB} = \{0 \to 3 \to 1 \to 2 \to 4 \to 0\} \qquad z_{CMB} = 65$$

Iteraţia 3

- Selectăm problema PA_2. Lista $\mathcal{L}$ devine vidă.
- PA_2 are soluţia optimă:

$$x_{04} = x_{41} = x_{12} = x_{23} = x_{30} = 1 \quad , \quad x_{ij} = 0 \text{ în rest}$$

care este un traseu complet cu valoarea 62. Actualizăm:

$$x_{CMB} = \{0 \to 4 \to 1 \to 2 \to 3 \to 0\} \qquad z_{CMB} = 62$$

Iteraţia 4

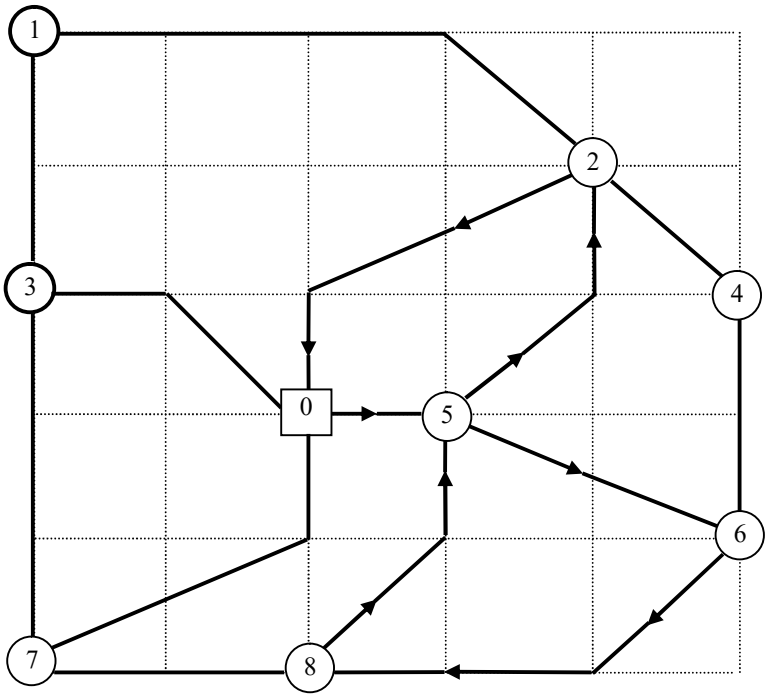
Deoarece lista $\mathcal{L}$ este vidă algoritmul se opreşte. Traseul cu cel mai mic cost este reţinut în x_{CMB}

6.3.2 Aplicaţie la problema firmei de curierat rapid

Se consideră reţeaua stradală din figura 6.1 Unele muchii pot fi circulate în ambele sensuri, altele numai în sensul indicat. Un server, localizat în punctul 0 are de transportat cinci comenzi ale căror surse şi destinaţii sunt date în tabelul 6.2 Se pune problema determinării traseului de lungime minimă care începe şi sfârşeşte în nodul 0 şi pe care serverul trebuie să-l parcurgă în vederea transportării celor cinci comenzi de la surse la destinaţii.

Tabelul 6

Comanda	Sursa	Destinaţia
r_1	1	6
r_2	2	7
r_3	6	3
r_4	4	1
r_5	8	2

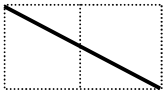


Latura patratului caroiajului măsoară 1 km = 10 hm

diagonala

măsoară 1,5 km =15 hm

diagonala



măsoară 2,3 km =23 hm

Figura 6.1

Tabelul 6.3

	0	1	2	3	4	5	6	7	8
0	*	**45** 0,3,1	**35** 0,5,2	**25** 0,3	**50** 0,5,2,4	**10** 0,5	**33** 0,5,6	**33** 0,7	**53** 0,7,8
1	**45** 1,3,0	*	**45** 1,2	**20** 1,3	**60** 1,2,4	**55** 1,3,0,5	**78** 1,3,0,5,6	**50** 1,3,7	**70** 1,7,8
2	**33** 2,0	**45** 2,1	*	**58** 2,0,3	**15** 2,4	**43** 2,0,5	**35** 2,4,6	**66** 2,0,7	**70** 2,4,6,8
3	**25** 3,0	**20** 3,1	**60** 3,0,5,2	*	**75** 3,0,5,2,4	**35** 3,0,5	**58** 3,0,5,6	**30** 3,7	**50** 3,7,8
4	**48** 4,2,0	**60** 4,2,1	**15** 4,2	**73** 4,2,0,3	*	**58** 4,2,0,5	**20** 4,6	**75** 4,6,8,7	**55** 4,6,8
5	**58** 5,2,0	**70** 5,2,1	**25** 5,2	**83** 5,2,0,3	**40** 5,2,4	*	**23** 5,6	**78** 5,6,8,7	**58** 5,6,8
6	**68** 6,4,2,0	**80** 6,4,2,1	**35** 6,4,2	**85** 6,8,7,3	**20** 6,4	**60** 6,8,5	*	**55** 6,8,7	**35** 6,8
7	**33** 7,0	**50** 7,3,1	**68** 7,0,5,2	**30** 7,3	**83** 7,0,5,2,4	**43** 7,0,5	**66** 7,0,5,6	*	**20** 7,8
8	**53** 8,7,0	**70** 8,7,3,1	**50** 8,5,2	**50** 8,7,3	**65** 8,5,2,4	**25** 8,5	**48** 8,5,6	**20** 8,7	*

- Începem prin a determina – cu o procedură specializată cum ar fi algoritmul lui FLOYD – drumurile de lungime minimă dintre nodurile reţelei. Vezi tabelul 6.3 unde, în celula (*i*, *j*), apare atât lungimea drumului minimal de la *i* la *j* cât şi nodurile prin care trece acesta.

De exemplu cel mai scurt drum de la nodul 3 la nodul 4 are lungimea de 75 hm = 7,5 km şi trece prin nodurile 3, 0, 5, 2 şi 4. Datorită faptului că unele muchii au sens unic de parcurgere, drumul „invers" cel mai scurt de la nodul 4 la nodul 3 are lungimea 73 hm = 7,3 km şi trece prin nodurile 4,2,0 şi 3.

- Construim apoi o problemă a comisvoiajorului, asimetrică, echivalentă cu problema dată. „Oraşele" se identifică cu cele cinci comenzi r_1 , ..., r_5 la care se adaugă „comanda fictivă" r_0 concentrată în locul de plecare şi întoarcere a serverului. „Costurile de deplasare" sunt lungimile drumurilor „în gol" de la destinaţia unei comenzi la sursa alteia – vezi tabelul 6.4

Tabelul 6.4

	r_0	r_1	r_2	r_3	r_4	r_5
r_0	∞	45	35	33	50	53
r_1	68	∞	35	0	20	35
r_2	33	50	∞	66	83	20
r_3	25	20	60	∞	75	50
r_4	45	0	45	78	∞	70
r_5	33	45	0	35	15	∞

Exemplu de calcul pentru $c(r_4, r_3) = 78$:

- destinaţia comenzii r_4 este nodul 1;
- sursa comenzii r_3 este nodul 6;
- drumul cel mai scurt de la nodul 1 la nodul 6 este $1 \to 3 \to 0 \to 5 \to 6$ cu lungimea 78 hm = 7,8 km.

- Cu algoritmul lui EASTMAN s-a găsit succesiunea optimă:

$$r_0 \to r_2 \to r_5 \to r_4 \to r_1 \to r_3 \to r_0$$

cu „costul" total minim 95 hm =9,5 km reprezentând lungimea totală a drumurilor „în gol".

Aceasta înseamnă că serverul va pleca din 0 şi va transporta comenzile în ordinea r_2, r_5, r_4, r_1, r_3 după care se va întoarce de unde a plecat.

În notaţia (2) din 6.2 structura traseului optim arată astfel:

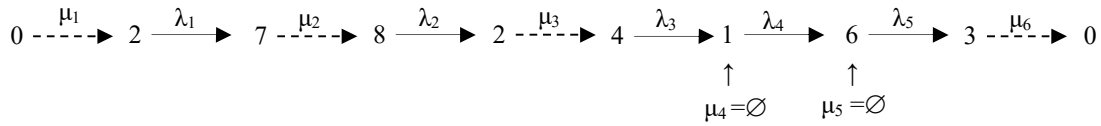


Figura 6.2

sau „în extenso”:

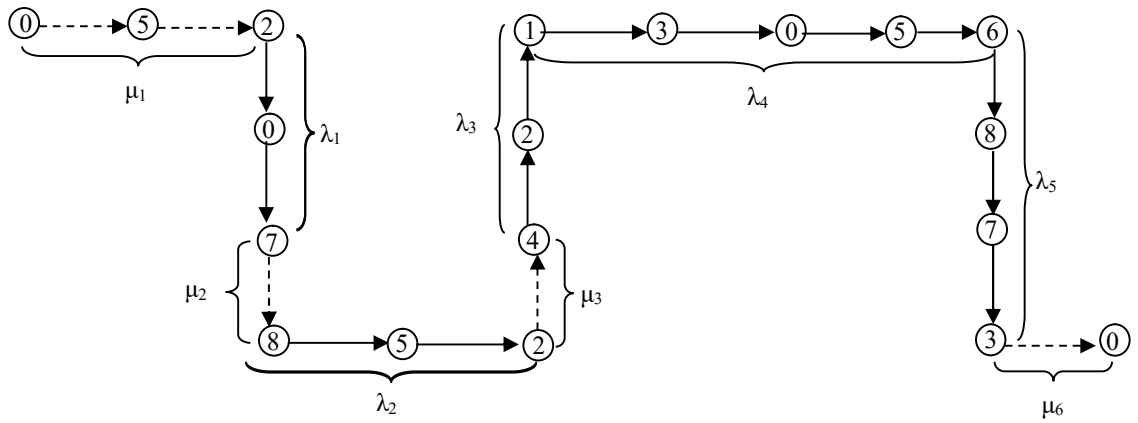


Figura 6.3

Drumurile „în plin” însumează 339 hm = 33,9 km. Întregul traseu optim măsoară:
95 + 339 = 434 hm = 43,4 km. Drumurile în gol reprezintă circa 22% din întregul parcurs.

6.4 Euristici de rezolvare suboptimală a TSP euclidiene

Problema simetrică a comisvoiajorului şi în special cea euclidiană are multe aplicaţii practice şi ca urmare, cunoaşterea unor metode de rezolvare fie şi suboptimală, este importantă. Deoarece complexitatea problemei face ca metodele exacte să fie aplicabile doar problemelor de dimensiune moderată, în continuare vor fi descrise unele metode euristice.

Metodele de rezolvare suboptimală se împart în două categorii:

a) euristici care construiesc efectiv un traseu complet căutând ca acesta să fie „cât mai bun”. Din această categorie amintim:

- euristica: **mergi la cel mai apropiat vecin**;
- euristica: **inserează punctul cel mai depărtat (apropiat)**;
- euristici bazate pe utilizarea arborelui de cost minim ca de exemplu euristica: **dublează muchiile arborelui de cost minim** sau **euristica lui Cristofides.**

b) euristici care ameliorează valoarea unui traseu complet dat sau construit printr-o procedură din prima categorie.

6.4.1 Euristica: mergi la cel mai apropiat vecin

- se pleacă din localitatea 0 către **cea mai apropiată** localitate;
- din ultima localitate **vizitată** se pleacă către **cea mai apropiată** localitate **nevizitată**; în caz că nu mai există localităţi nevizitate se revine la punctul de plecare.

Deşi **local** metoda face **cea mai bună** alegere, deseori în traseul construit apar deplasări „costisitoare" astfel că acesta nu este întotdeauna traseul optim! Euristica este simplă şi, dacă este urmată de o procedură ameliorativă poate conduce la rezultate foarte bune în sensul obţinerii unui traseu foarte apropiat ca valoare de traseul optim. Deoarece un punct poate avea mai mulţi vecini „la fel de apropiaţi" este posibil ca euristica să conducă la mai multe trasee!

Exemplul 1 În figura 6.4 este dată o concretizare a problemei din debutul secţiunii 6.1 Elicopterul are baza de plecare şi întoarcere în punctul 0 iar deplasările între platforme se fac în linie dreaptă. Latura patratului caroiajului măsoară 10 km; distanţele dintre platforme au fost calculate cu teorema lui Pitagora şi sunt date în tabelul 6.5

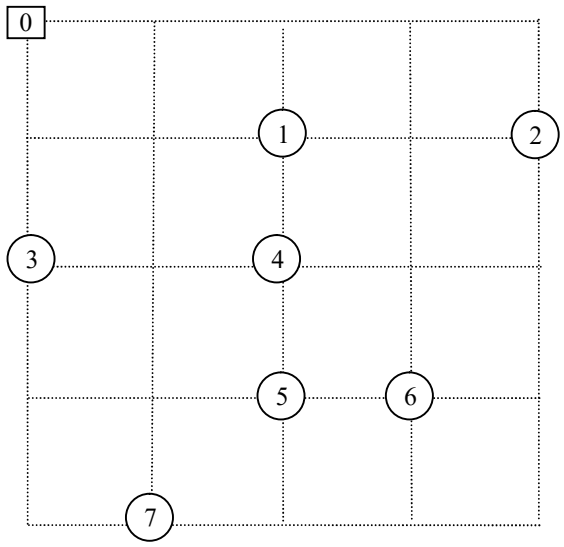


Figura 6.4

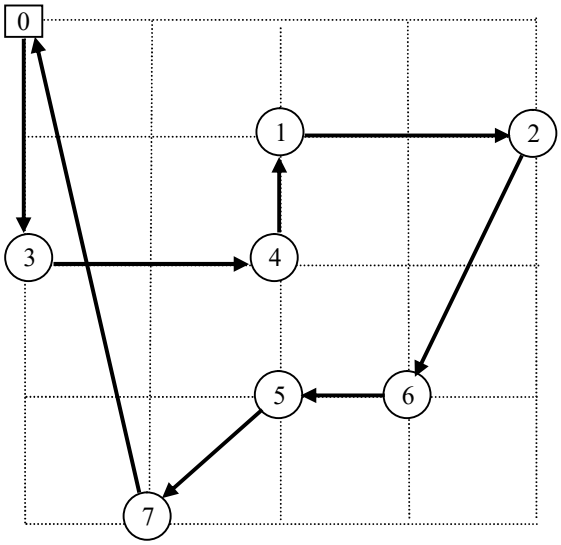
Tabelul 6.5

	1	2	3	4	5	6	7
0	22	41	20	28	36	42	41
1	*	20	22	10	20	22	32
2		*	41	22	28	22	42
3			*	20	22	32	22
4				*	10	14	22
5					*	10	14
6						*	22

Deplasându-ne permanent către cel mai apropiat vecin nevizitat, a rezultat traseul:

$$0 \to 3 \to 4 \to 1 \to 2 \to 6 \to 5 \to 7 \to 0$$

cu lungimea de 157 km, vizualizat în figura 6.5



Cu siguranţă traseul construit nu este cel mai scurt din cauza „încrucişării" arcelor (3,4) şi (7,0). Un traseu mai scurt se va obţine folosind euristica ameliorativă descrisă în următoarea secţiune.

Cititorul atent a observat că din nodul 4 se putea merge fie în nodul 1 fie în nodul 5; a fost ales nodul1.Îl invităm să continue aplicarea euristicii mergând de astă dată din 4 în 5.
La acest stadiu al discuţiei, puteţi afirma că „înaintarea" din nodul 4 spre nodul 5 va fi mai „profitabilă" decât cea aleasă?...

Figura 6.5

6.4.2 Euristica: ajustare locală

Este vorba de o euristică din cea de a doua categorie care îmbunătăţeşte un traseu complet dat T.

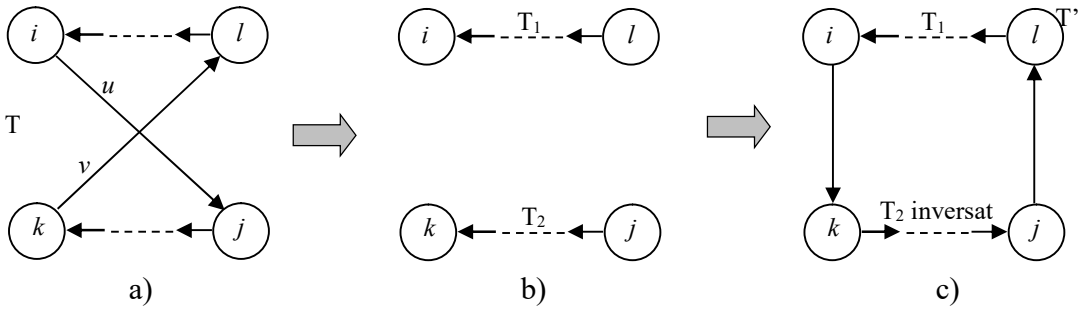


Figura 6.6

- Se examinează **pe rând** toate perechile de arce **neconsecutive** *u*,*v* din T – vezi figura 6.6a).
- Eliminarea arcelor *u* şi *v* „sparge" T în două drumuri disjuncte T_1 şi T_2 – vezi figura 6.6b).
- Aceste drumuri se pot asambla într-un nou traseu T' ≠ T într-un singur mod, indicat în figura 6.6c).

- Dacă:

$$c_{ij} + c_{kl} \leq c_{ik} + c_{jl}$$

noul traseu nu este mai bun ca valoare decât T şi dacă este aşa se trece la examinarea altei perechi de arce neconsecutive din T.

Dacă însă:

$$c_{ij} + c_{kl} > c_{ik} + c_{jl}$$

noul traseu T' are o valoare mai mică decât T. În acest caz, se înlocuieşte T cu T' şi se trece la examinarea perechilor de arce neconsecutive din T'.

Exemplul 2 Aplicăm procedura descrisă traseului din figura 6.5 luând în considerare perechea de arce (3,4) şi (7,0).

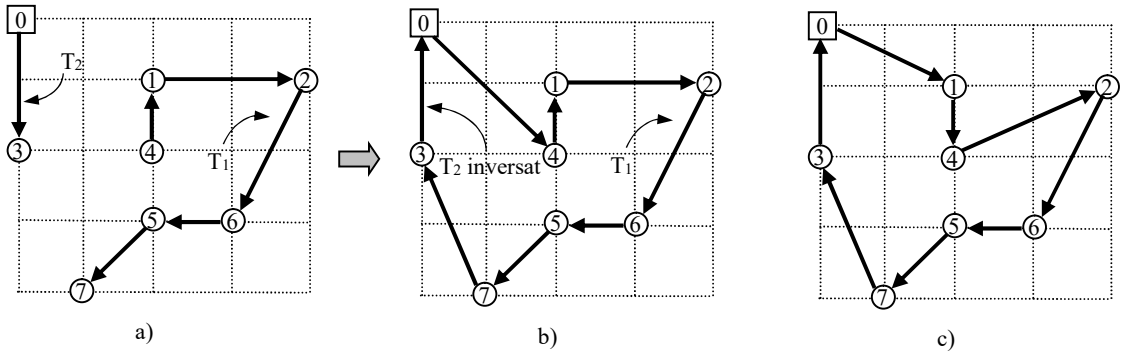


Figura 6.7

Eliminarea arcelor (3,4) şi (7,0) – vezi figura 6.7a) – şi înlocuirea lor cu arcele (0,4) şi (7,3) scurtează distanţa totală de parcurs deoarece:

$$c_{34} + c_{70} = 20 + 41 = 61 > 50 = 28 + 22 = c_{04} + c_{73}$$

Noul traseu, indicat în figura 6.7 b) are lungimea $157 - (61 - 50) = 146$ km.

Atenţie: pot exista şi perechi de arce care nu se încrucişează dar care conduc la micşorarea valorii unui traseu! (de altfel, „încrucişarea" se poate depista numai dacă traseul este vizualizat...)

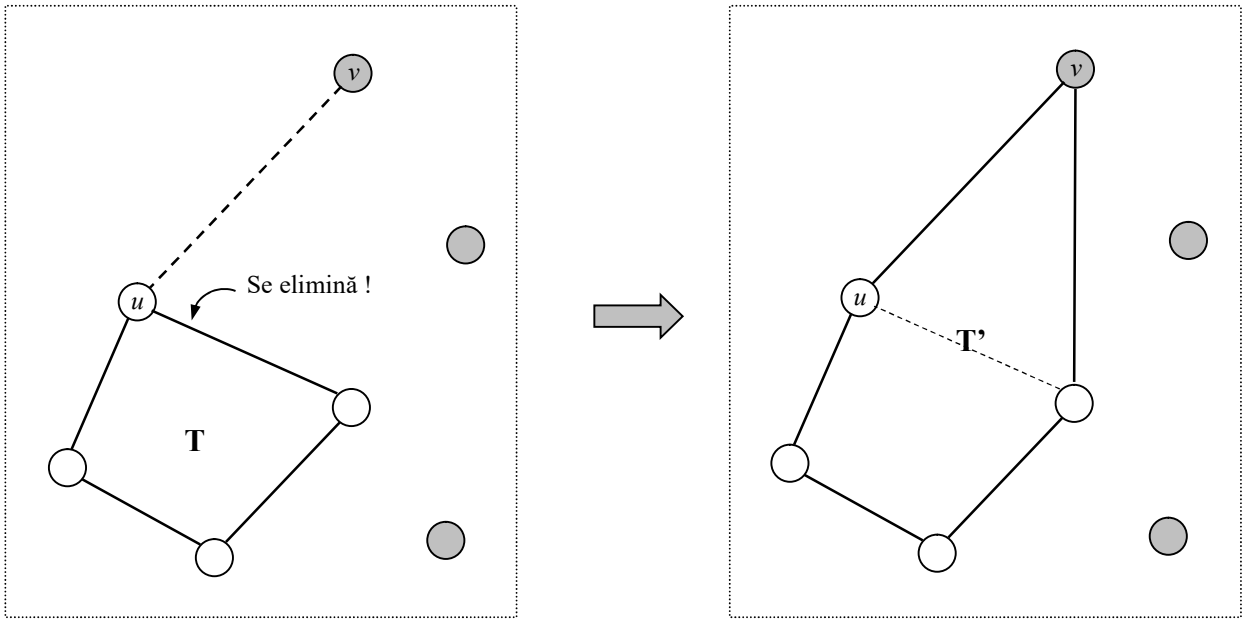
De exemplu, să luăm perechea de arce (0,4) şi (1,2) ale traseului din figura 6.7b). Deoarece:

$$c_{04} + c_{12} = 28 + 20 = 48 > 44 = 22 + 22 = c_{01} + c_{42}$$

prin eliminarea lor se obţine traseul din figura 6.7c) cu lungimea $146 - (48 - 44) = 142$ km. Ajustarea locală continuă cu examinarea altor perechi de arce neconsecutive din ultimul traseu obţinut.

6.4.3 Euristica: inserează punctul cel mai depărtat

Se consideră un subtraseu T care trece prin unele dintre nodurile 0,1,...,*n*. Printre nodurile nesituate pe traseul T se caută nodul *v*, „cel mai depărtat" de T (reamintim că „distanţa" de la *v* la mulţimea nodurilor subtraseuluiT se defineşte ca fiind **minimul** distanţelor de la *v* la fiecare nod din T). Fie *u* nodul din T „cel mai apropiat" de *v* – vezi figura 6.8a). Se elimină una din muchiile traseului care are o extremitate în *u* şi se conectează extremităţile muchiei eliminate cu nodul *v*. Rezultatul este un nou subtraseu T' cu un nod „mai mare" decât subtraseul T – vezi figura 6.8b).



a) b)

Figura 6.8

De regulă, la start subtraseul T este redus la punctul 0 de plecare şi întoarcere al comisvoiajorului astfel că în *n* paşi rezultă un traseu care trece prin toate punctele de vizitat.
La prima vedere, s-ar părea că inserarea la fiecare iteraţie a „celui mai apropiat" nod nevizitat în locul celui mai depărtat ar fi o idee mai bună. Experimentele numerice arată „în medie" contrariul!!

Exemplul 3 Reluăm problema vizitării platformelor marine din exemplul 1.

- Iniţializăm T = {0}.
- Cele mai depărtate puncte de 0 sunt 2 şi 7 pentru că $c_{02} = c_{07} = 41$ sunt cele mai mari distanţe din tabelul 4.Actualizăm T: $0 \to 2 \to 7 \to 0$.
- Distanţele de la nodurile 1, 3, 4, 5şi 6 la mulţimea {0, 2, 7}a nodurilor subtraseului curent T sunt 20, 20, 22, 14 respectiv 22. Alegând nodul 6, T devine: $0 \to 2 \to 6 \to 7 \to 0$.
- Faţă de noul T cele mai depărtate noduri sunt 1 şi 3.A fost ales şi inserat nodul 3 astfel că T: $0 \to 2 \to 6 \to 7 \to 3 \to 0$.

- Nodurile rămase au fost inserate în ordinea 1, urmat de 4 şi la urmă 5. În final a rezultat traseul complet: T: $0 \to 1 \to 2 \to 6 \to 5 \to 4 \to 7 \to 3 \to 0$ cu lungimea de 148 km.

Observaţie: În mai multe rânduri a fost necesar să facem o alegere a nodului ce urma a fi inserat în subtraseul curent dintr-o mulţime de noduri candidate. Ca urmare procedura poate conduce şi la alte trasee unele posibil mai bune, altele posibil mai proaste decât traseul efectiv construit!

6.4.4 Euristica: dublează muchiile unui arbore minimal

- În graful complet cu nodurile 0,1,...,*n* se determină un **arbore** H de valoare(lungime) minimă. Aceasta este o problemă de optimizare **uşoară** rezolvabilă cu algoritmul lui Kruskal – vezi unitatea de învăţare 5.

- În continuare fiecare muchie din H se înlocuieşte cu două muchii de aceeaşi lungime. Se obţine un multigraf H'.

Prin construcţie, H' este un **graf eulerian** (deoarece gradele tuturor nodurilor sunt pare!) şi ca urmare există posibilitatea traversării **tuturor** muchiilor, **o singură dată,** cu plecarea şi întoarcerea în nodul 0.

- Fie

$$0 \to x \to y \to \cdots\cdots \to u \to v \to 0 \qquad (*)$$

Succesiunea în care toate muchiile multigrafului H' au fost parcurse. În aceeastă secvenţă este posibil ca unul sau mai multe oraşe să apară de mai multe ori. Procedăm la următoarea operaţie de **scurtcircuitare**:

În succesiunea (*) se determină **prima** tripletă $i \to j \to k$ de noduri (oraşe) consecutive în care „mijlocul" j mai apare ulterior în succesiune. Înlocuim secvenţa $i \to j \to k$ cu arcul direct $i \to k$. Prin această operaţie:

- fiecare oraş continuă să fie vizitat măcar o dată;
- noul traseu are o lungime mai mică deoarece $c_{ij} + c_{jk} \geq c_{ik}$ (atenţie, avem în vedere în exclusivitate TSP euclidiene!!)

Scurtcircuitarea se repetă până când, în secvenţa (*) **actualizată,** fiecare oraş apare **o singură dată**, bineînţeles cu excepţia punctului de plecare şi întoarcere 0.

Se poate demonstra că lungimea traseului obţinut prin această euristică este cel mult de două ori mai mare decât lungimea traseului optim (în practică lucrurile stau însă mai bine...)

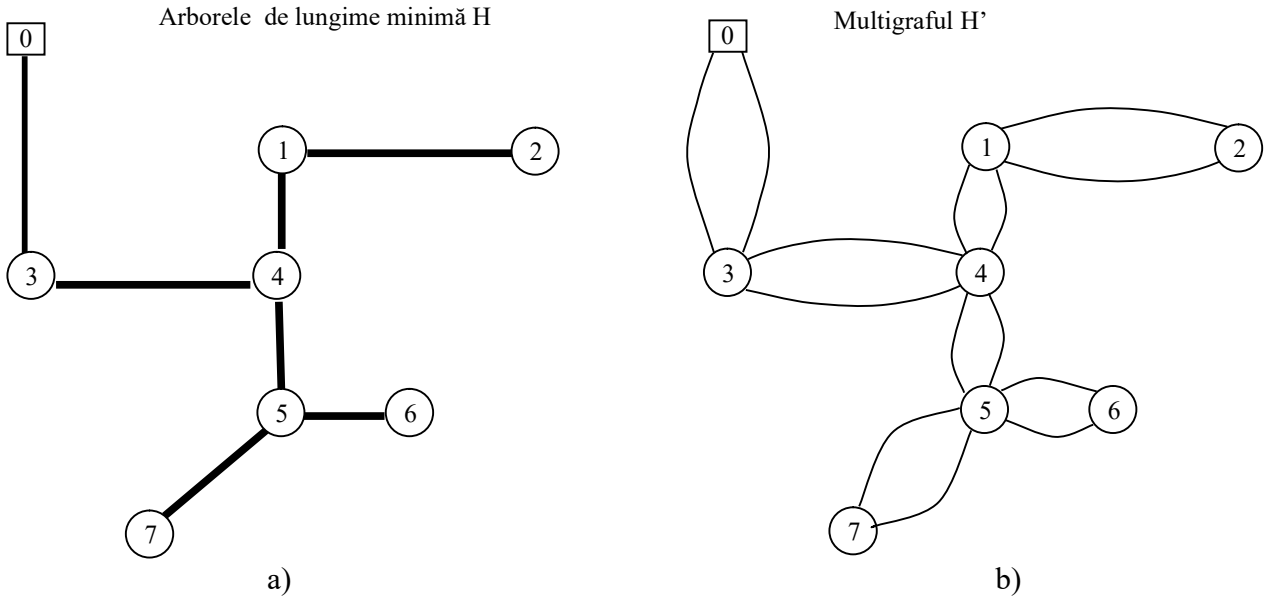


Figura 6.9

Exemplul 4 Aplicăm euristica descrisă problemei din exemplul 1. În figura 6.9a) este vizualizat un arbore H de valoare minimă; în figura 6.9b) apare multigraful H'. Traversarea tuturor muchiilor din H', fiecare o singură dată se poate realiza în mai multe feluri; iată unul din ele:

$$0 \to 3 \to 4 \to 5 \to 6 \to 5 \to 7 \to 5 \to 4 \to 1 \to 2 \to 1 \to 4 \to 3 \to 0$$

Traseul obţinut nu este o soluţie a TSP deoarece unele puncte sunt vizitate de mai multe ori. Eliminarea repetiţiilor prin „scurtcircuitare" este arătată în diagrama din figura 6.10

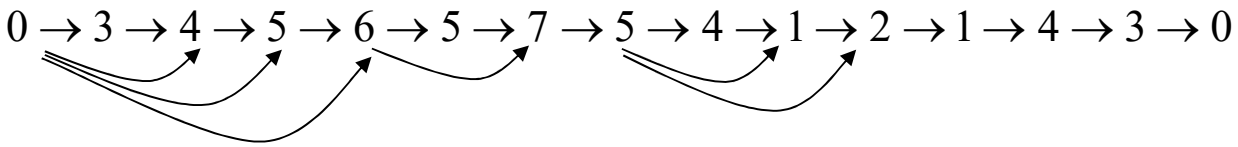


Figura 6.10

A rezultat traseul: $0 \to 6 \to 7 \to 5 \to 2 \to 1 \to 4 \to 2 \to 0$ cu lungimea de 176 km şi vizualizat în figura 6.11a). După aplicarea euristicii de ajustare locală – vezi figurile 6.11 b),c),d) e) se regăseşte traseul din figura 6.7c) care „pare a fi" traseul cel mai scurt. Perechile de arce neconsecutive înlocuite au fost evidenţiate prin linii punctate.

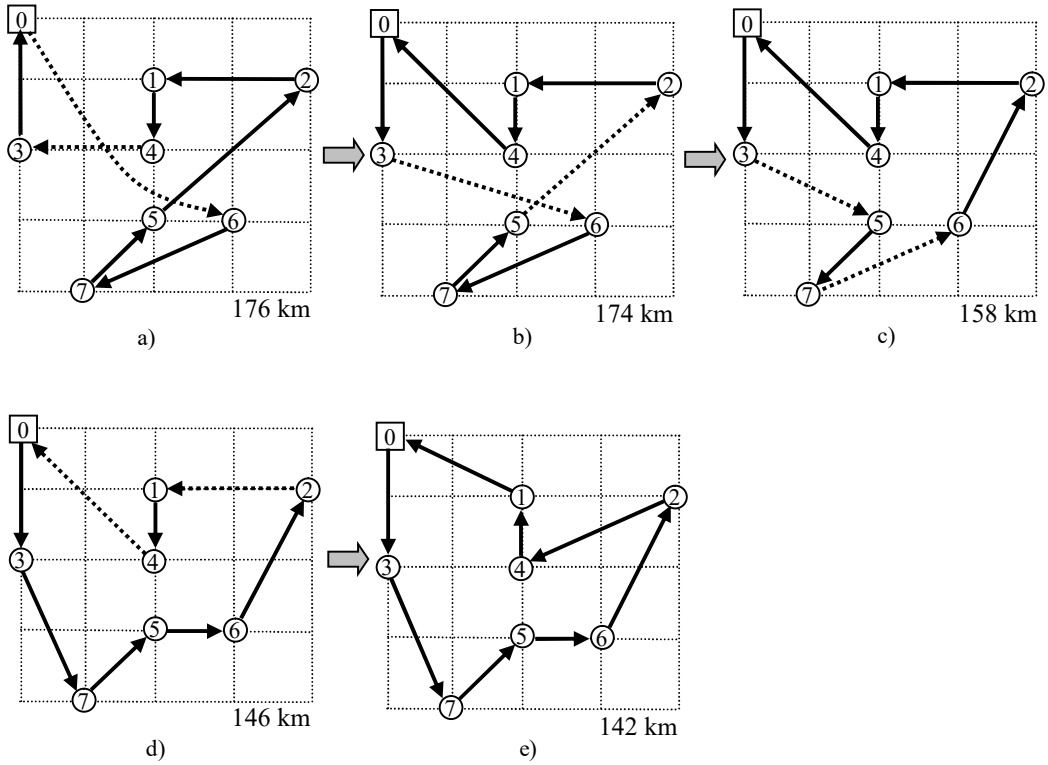


Figura 6.11

6.4.5 Euristica lui Cristofides

Această procedură, datorată lui CRISTOFIDES este o **rafinare** a euristicii precedente.**Deoarece suma gradelor nodurilor din arborele H este pară, numărul nodurilor de grad impar este par!** Folosind numai aceste noduri şi muchiile dintre ele se determină **cuplajul** C de **pondere minimă (**vezi unitatea de învăţare 4!), ponderile muchiilor luate în calcul fiind distanţele dintre extremităţi. Fie H' (multi)graful rezultat din H prin adăugarea muchiilor cuplajului C, cu menţiunea că dacă între două noduri avem o muchie în H şi alta în C, graful H' va avea, între nodurile respective, două muchii! Prin construcţie, în H' fiecare nod are grad **par**, deci H' este un **graf eulerian** şi ca urmare există posibilitatea traversării tuturor muchiilor sale exact o singură dată.Plecând de aici şi folosind **tehnica scurtcircuitării** se ajunge la un traseu complet.

Se poate arăta că lungimea acestui traseu nu depăşeşte 3/2 din lungimea traseului optim.

Exemplul 5 Pentru problema euclidiană a comisvoiajorului din exemplul 1, un arbore de lungime minimă este disponibil în figura 6.9a). Nodurile de grad impar sunt 0,2,6,7 cu gradul 1 şi 4,5 cu gradul 3.Prin simplă inspecţie (sau utilizând un algoritm de determinare a cuplajului de pondere minimă în subgraful complet ale cărui noduri sunt 0,2,4,5,6,7) se constată că muchiile {0,2}, {4,6}, {5,7}au cea mai mică pondere totală: $c_{02} + c_{46} + c_{57} = 69$ .Adăugând aceste muchii la arborele H se obţine multigraful H' din figura 6.12

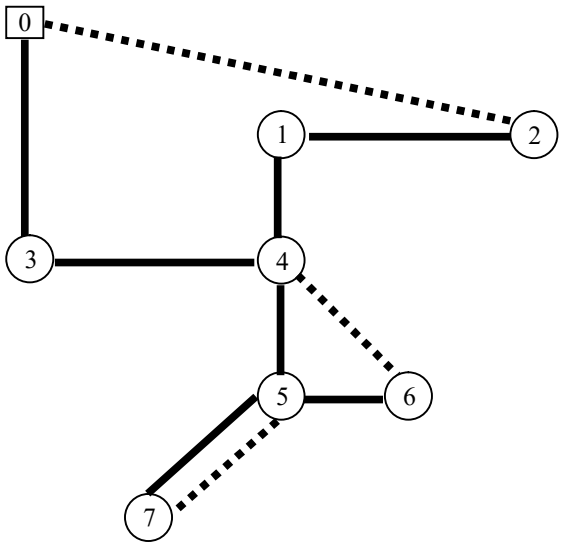


Figura 6.12

O posibilă traversare a tuturor muchiilor o singură dată este dată de succesiunea $0 \to 2 \to 1 \to 4 \to 6 \to 5 \to 7 \to 5 \to 4 \to 3 \to 0$. După scurtcircuitarea secvenţelor $1 \to 4 \to 6$ şi $6 \to 5 \to 7$ resultă traseul $0 \to 2 \to 1 \to 6 \to 7 \to 5 \to 4 \to 3 \to 0$ cu lungimea de 169 km. Invităm cititorul să-l vizualizeze şi să cerceteze dacă poate fi scurtat folosind ajustarea locală.

Probleme propuse

1. Un comisvoiajor situat în localitatea 0 are de vizitat oraşele 1, 2, 3 şi 4.Costurile de deplasare c_{ij} , $0 \le i, j \le 4$, $i \ne j$ de la un oraş la altul sunt date în tabelul 6.6. Se observă că în general $c_{ij} \ne c_{ji}$ altfel spus costul deplasării între două localităţi depinde de sensul de deplasare – avem de a face cu o TSP asimetrică. Să se aplice algoritmul lui Eastman pentru a găsi un traseu de cost minim.

Tabelul 6.6

	0	1	2	3	4
0	.	10	7	9	11
1	8	.	11	12	4
2	9	11	.	13	6
3	6	12	14	.	7
4	10	5	8	8	.

2. Să se aplice algoritmul lui Eastman pentru rezolvarea problemei asimetrice a comisvoiajorului ale cărei costuri sunt date în tabelul 6.7

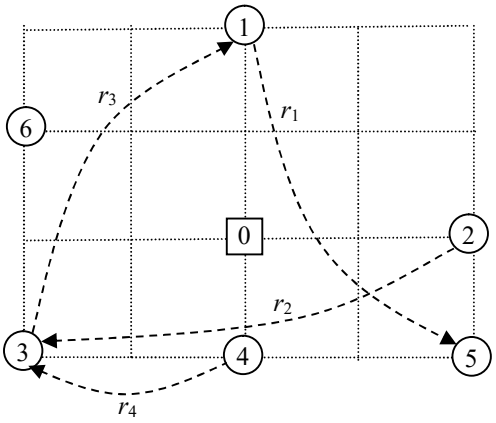
3. Problemele 1 şi 2 sunt „mici” şi cu un program de calculator destul de simplu ele pot fi rezolvate prin enumerarea completă a soluţiilor. Câte trasee complete ar trebui generate în fiecare caz?

Tabelul 6.7

	0	1	2	3	4	5
0	.	18	13	20	22	34
1	10	.	20	17	18	10
2	10	9	.	20	11	20
3	19	27	23	.	6	11
4	24	17	20	7	.	8
5	23	33	18	14	30	.

4. Algoritmul lui Eastman este o procedură de tip B&B. În ce situaţii nu mai are loc ramificarea unui nod?

5. Un server are de transportat patru colete mari dintr-un loc în altul. În figura 6.13 sunt indicate poziţiile locului de parcare al serverului şi ale surselor şi destinaţiilor coletelor. Deplasările se fac numai pe liniile orizontale şi verticale ale caroiajului. Latura patratului caroiajului se va lua ca unitate de lungime.



Comanda	Sursa	Destinaţia
r_1	1	5
r_2	2	3
r_3	3	1
r_4	4	3

Figura 6.13

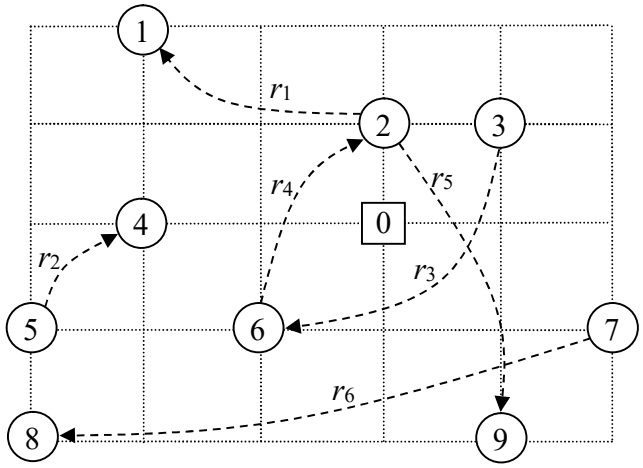
a) În ce ordine vor fi transportate comenzile pentru ca distanţa totală parcursă de server să fie minimă, ştiind că acesta pleacă şi se întoarce în 0 şi nu transportă mai mult de o comandă o dată. Precizaţi distanţa minimă de parcurs şi procentul parcursului „în gol” (se va construi o TSP asimetrică echivalentă care va fi rezolvată cu algoritmul lui Eastman).
b) Pe lângă comenzile indicate, serverul trebuie să treacă şi prin punctul 6, la un service, pentru remedierea unei mici defecţiuni. Cum va arăta traseul minim? (poanta: transformaţi „oprirea în punctul 6” într-o nouă comandă r_5 .

c) Elaboraţi (empiric) un program de transport în ipoteza că serverul poate duce şi două colete o dată!

6. În reţeaua din figura 6.14 nodurile 1, 2,...,9 sunt sursele şi destinaţiile comenzilor de transport r_1 ,r_2,...,r_6. Un server cu locul de parcare în nodul 0 este desemnat să transporte aceste comenzi. Deplasările serverului se fac numai pe liniile orizontale şi verticale ale caroiajului. Latura patratului caroiajului măsoară 1 km.

În ce ordine vor fi efectuate transporturile pentru ca distanţa totală parcursă de server să fie minimă în ipoteza că serverul nu poate transporta mai mult de o comandă odată.

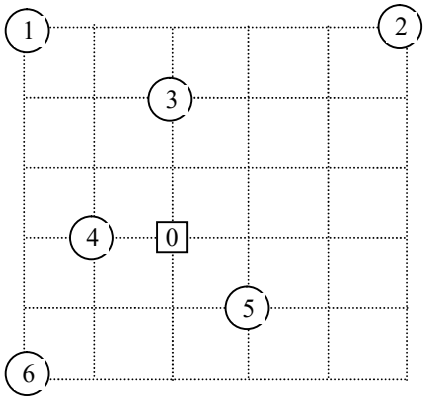
Elaboraţi în manieră empirică un program de lucru ştiind că serverul poate duce două comenzi odată.



Comanda	Sursa	Destinaţia
r_1	2	1
r_2	5	4
r_3	3	6
r_4	6	2
r_5	2	9
r_6	7	8

Figura 6.14

6. Agenţia de turism HAI SĂ HAIDEM situată în localitatea 0 este interesată în determinarea unui traseu de vizitare a şase obiective de interes turistic şi istoric notate 1,2,...,6. Poziţiile relative ale celor şapte puncte sunt indicate în figura 6.15 iar distanţele dintre ele – măsurate în km în linie dreaptă – sunt date în tabelul alăturat.

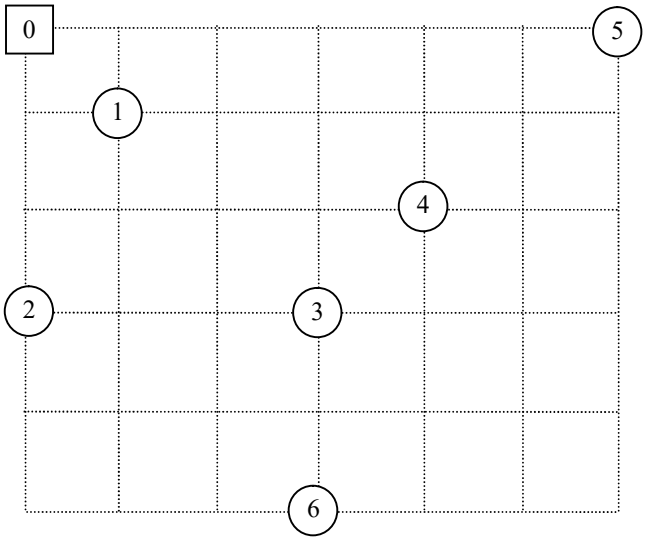


	1	2	3	4	5	6
0	36	42	20	10	14	28
1	*	50	22	32	50	50
2		*	32	50	45	71
3			*	22	32	45
4				*	22	22
5					*	32

Figura 6.15

i) Evident, obiectivul urmărit este determinarea celui mai scurt traseu. Câte soluţii ≡ trasee complete ar fi de examinat în situaţia dată?
ii) Determinaţi un traseu cât mai scurt aplicând **riguros** euristica „mergi la cel mai apropiat vecin”;
iii) Dacă este cazul folosiţi ajustarea locală pentru micşorarea parcursului total;
iv) Reluaţi chestiunea aplicând (riguros!) una sau alta din euristicile explicate în text;
v) Care este cel mai scurt traseu pe care l-aţi găsit?

7. Se consideră problema (euclidiană) a comisvoiajorului cu datele din figura 6.16



	1	2	3	4	5	6
0	14	30	42	45	50	58
1	*	22	28	32	41	45
2		*	30	41	58	36
3			*	14	36	20
4				*	22	32
5					*	54

Figura 6.16

i) Aplicaţi riguros euristica „mergi la cel mai apropiat vecin”. Eliminaţi eventualele încrucişări cu euristica de ajustare locală;
ii) Aplicaţi euristicile „inserează puncul cel mai apropiat” respectiv „cel mai depărtat” – urmate eventual de „ajustarea locală” şi comparaţi rezultatele;
iii) Aplicaţi cele două euristici bazate pe arborele de lungime minimă;
iv) Care este cel mai scurt traseu găsit?