CAPITOLUL II

INTRODUCERE ÎN OPTIMIZAREA COMBINATORIALĂ

§ 1. Obiectul optimizării combinatoriale

Combinatorica este un domeniu reprezentativ al matematicilor care, în esență, se ocupă cu studiul "aranjamentelor" obiectelor unei mulțimi **finite,** conforme unei **structuri** date. În **problemele combinatoriale,** prioritare sunt chestiunile de **existență** şi de **numărare**:

- **există** un anumit tip particular de aranjament?
- **câte** asemenea aranjamente se pot forma?

Caracteristic problemelor combinatoriale este faptul că numărul acestor aranjamente este **finit**.

Dacă aranjamentele unei probleme combinatoriale pot fi **comparate** pe baza unui criteriu de apreciere apare o **problemă de optimizare combinatorială**:

- care este **cel mai bun** aranjament din punctul de vedere al criteriului respectiv?

Exemplul 1.1 Să considerăm un **graf finit** şi **simplu** G = (V,E) în care V este mulțimea **nodurilor** iar E este mulțimea **muchiilor**.Un **arbore** în graful G este un subgraf H = (V,A) , $A \subset E$, cu aceleaşi noduri ca şi G, care, de sine stătător, este un graf **conex** şi **fără cicluri** (vezi figura 1.1 în care muchiile arborelui au fost îngroşate)

Se pun următoarele chestiuni:

- conține G cel puțin un arbore?
- câți arbori distincți se pot forma?

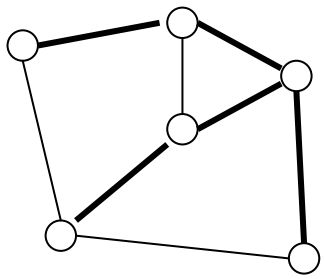


Figura 1.1

În cazul de față "obiectele de lucru" sunt muchiile grafului G. Un "aranjament" al acestor obiecte este o submulțime de muchii $A \subset E$ cu proprietățile:

- orice nod al grafului G este extremitate pentru cel puțin o muchie din A;
- subgraful H = (V,A) este conex şi fără cicluri.

Următorul rezultat stabileşte în ce condiții există "aranjamentele" descrise:

Graful G are arbori dacă şi numai dacă este conex.

Amintim doar în treacăt faptul că există o "formulă" care stabileşte numărul arborilor distincți din G.

Să presupunem mai departe că fiecărei muchii $e \in E$ i s-a asociat un "cost" $c(e) \geq 0$. Definim costul unui arbore $H = (V,A)$ ca fiind suma costurilor muchiilor alcătuitoare:

$$c(H) = \sum_{e \in A} c(e)$$

Rezultă în final următoarea problemă de optimizare combinatorială a cărei rezolvare va fi prezentată în secțiunea 3:

Să se determine în graful G arborele H^* de cost minim.

Exemplul 1.2 Muchiile grafului G din exemplul 1.1 pot fi aranjate şi în alte moduri. Astfel, un **cuplaj** în G este o submulțime $C \subset E$ de muchii două câte două **neadiacente**. Existența acestor noi aranjamente este o chestiune banală întrucât orice muchie din G este un cuplaj. Interesante sunt problemele de optimizare care se pun în acest context:

- să se determine cuplajul C^* cu cele mai multe muchii (**cuplajul maxim**)

sau, în cazul în care muchiile din G sunt "ponderate":

- să se determine **cuplajul** C^* **de pondere maximă**.

(ca mai sus, ponderea unui cuplaj este dată de suma ponderilor muchiilor din cuplaj)

Este foarte important de menționat faptul că problemele de optimizare combinatorială provin în marea lor majoritate din diverse domenii de activitate, în special economică: astfel problema arborelui de cost minim din exemplul 1.1 modelează o largă varietate de situații practice din domeniul rețelelor de comunicații (rețele de calculatoare, de drumuri, televiziune prin cablu etc)

Optimizarea combinatorială are ca obiect **rezolvarea** problemelor specifice adică elaborarea de metode şi algoritmi de găsire efectivă a **celui mai bun "aranjament"** din mulțimea finită a tuturor aranjamentelor posibile.În continuare pentru aranjamentele unei probleme combinatoriale vom folosi termenul de **soluție**.

La prima vedere, s-ar părea că obiectul optimizării combinatoriale este banal: "Ce mare lucru este să găseşti "cel mai bun" element dintr-o mulțime finită?"
În realitate, listarea tuturor soluțiilor unei probleme combinatoriale, în vederea selectării soluției optime, este aproape imposibil de făcut deoarece numărul lor, deşi finit, este de regulă **astronomic** de mare chiar şi în cazul unor probleme de "talie moderată".

Exemplul 1.3 Pentru a înțelege mai bine despre ce este vorba să considerăm o altă problemă tipică de optimizare combinatorială, **problema comisvoiajorului**:

Un comisvoiajor situat în localitatea 0 trebuie să viziteze n localități 1,2,…,n în final întorcându-se de unde a plecat. Cunoscând distanțele dintre localități (sau costurile deplasărilor între localități) el doreşte să găsească traseul cu lungimea totală minimă (sau costul total minim). Este clar că numărul total al traseelor posibile este n! , un traseu fiind perfect determinat de ordinea (permutarea) în care vor fi vizitate cele n localități. O dată fixată această ordine calculul lungimii (costului) traseului corespunzător este o banală operație de adunare a lungimilor (costurilor) tronsoanelor care compun traseul.

În cazul particular a n=20 de localităţi de vizitat, cu un calculator în stare să examineze un miliard de trasee pe secundă, găsirea traseului optim ar dura circa 800 de ani iar dacă, în loc de 20 de localităţi, am avea cu una mai mult, căutarea ar dura în jur de 16800 de ani!!

În lumina acestui exemplu, determinarea **efectivă** a unui anumit element dintr-o mulţime **finită** (dar foarte "numeroasă"…) de posibilităţi este o chestiune **serioasă**: am dori să găsim acest element **în timp util** şi bineînţeles cu un efort de calcul **rezonabil**.
În termeni mai precişi, am dori ca timpul de rezolvare a unei asemenea probleme să depindă **"polinomial"** de dimensiunile problemei.

Pentru unele probleme de optimizare combinatorială acest lucru este posibil; despre aceste probleme vom spune că sunt **uşoare**. Problema arborelui de cost minim sau problema cuplajului maxim (de pondere maximă) ,descrise în exemplele 1.1 şi 1.2, sunt dovedite a fi probleme uşoare.

Pentru marea majoritate a problemelor combinatoriale, găsirea soluţiei optime în timp polinomial nu este încă posibilă, altfel spus metodele "exacte" **cunoscute** necesită un timp de rulare care creşte **exponenţial** o dată cu creşterea dimensiunilor problemei rezolvate. Mai mult există bănuiala că, datorită structurii lor interne, nici nu va fi posibilă vreodată elaborarea unor metode exacte de rezolvare cu "comportament polinomial"!
Din clasa acestor probleme, socotite **grele,** face parte şi problema comisvoiajorlui din exemplul 1.3.

Din cele de mai sus nu trebuie înţeles că problemele "grele" sunt inabordabile. În dimensiune "relativ moderată" aceste probleme pot fi soluţionate "exact" în timp rezonabil. Folosind metode de căutare din ce în ce mai sofisticate – care exploatează eficient structura combinatorială internă – au putut fi rezolvate cu succes şi probleme de dimensiuni apreciabile.Totuşi, nu se poate spune că aceste metode, oricât de sofisticate ar fi ele, sunt în stare "să facă faţă" oricărei probleme de acelaşi fel şi mai mult, pot avea comportamente radical diferite pe probleme asemănătoare şi de aceeaşi talie.

Având în vedere aceste observaţii, de mare importanţă, mai cu seamă practică, sunt metodele şi algoritmii care determină "în timp polinomial" o soluţie "acceptabilă"(suboptimală). Pentru aceste procedee, numite generic **euristici**, aprecierea eficacităţii lor va fi dată de "distanţa" dintre soluţia suboptimală şi cea optimă veritabilă! Am putea aprecia că o anumită euristică este "bună" dacă valoarea suboptimală a criteriului de performanţă ar fi, de exemplu, de cel puţin 90% din valoarea optimă!
Câteva modalităţi de rezolvare a problemei comisvoiajorului sunt date în secţiunea 4.

Cele mai multe probleme de optimizare combinatorială sunt modelate cu ajutorul teoriei grafurilor; exemplul1.1 este edificator în acest sens. Mai toate pot fi descrise alternativ prin programe liniare cu variabile întregi, în special bivalente, de unde strânsa legătură dintre optimizarea combinatorială şi programarea în numere întregi.

Exemplul 1.4 Reluăm problema cuplajului maxim din exemplul 1.2. Ataşăm fiecărei muchii $e \in E$ o variabilă booleană x_e cu semnificaţia:

$$x_e = \begin{cases} 1 & \text{daca muchia } e \in E \text{ va face parte din cuplajul maxim;} \\ 0 & \text{in caz contrar} \end{cases}$$

Pentru fiecare nod $v \in V$, fie E(v) mulţimea muchiilor din G care au o extremitate în v. Problema cuplajului maxim este perfect descrisă de programul bivalent:

$$\begin{cases} (\max) f = \sum_{e \in E} x_e \\ \sum_{e \in E(v)} x_e \le 1 \quad (\forall) v \in V \\ x_e \in \{0,1\} \end{cases}$$

§ 2. Domenii de aplicare ale optimizării combinatoriale

Între problemele de optimizare combinatorială se găsesc:

- problema drumului de valoare minimă între două noduri ale unui graf;
- problema fluxului maxim;
- problema de afectare.

deja studiate în cursul "Bazele Cercetării Operaționale" [10]. Ele intră, alături de problema arborelui de cost minim în categoria problemelor uşoare.

Lista problemelor grele este impresionantă şi se îmbogăţeşte continuu. În afara problemei comisvoiajorului mai putem aminti următoarele probleme mai reprezentative:

2.1 Problema croirii În multe domenii cum ar fi industria mobilei, a sticlei, a hârtiei, a confecțiilor, industria metalurgică apar frecvent probleme de debitare (sau croire) a unor repere din suporți mai mari. Importanța economică a acestor probleme rezidă în dependența directă a reducerii consumului de materiale de utilizarea unor scheme eficiente de croire.

De obicei, clasificarea problemelor de croire se face după dimensiunile relevante ale reperelor: avem probleme unidimensionale, bidimensionale sau tridimensionale. În croirea unidimensională de care ne vom ocupa mai mult, reperele se diferențiază printr-o singură dimensiune chiar dacă ele au mai multe. Cazul tipic este ecela al debitării unor bare de difrite lungimi din bare mai mari.Iată şi o situație concretă: o firmă produce un carton special în rulouri având o anumită lățime. Clienții solicită rulouri având aceeaşi lungime ca şi ruloul standard dar de lățimi mai mici. În acest caz reperele ca şi suporții sunt bidimensionali dar relevantă este o singură dimensiune – lățimea.

Aspectul combinatorial al unei probleme de croire este dat de modalitățile de tăiere ale unui suport în repere, modalități numite **rețete de croire**.Chestiunea de optimizare constă în a determina de câte ori una sau alta dintre rețetele de croire trebuie folosită pentru producerea unor cantități specificate de repere astfel încât numărul total de suporți consumați să fie minim.

Problema de croire unidimensională va f studiată mai în amănunt în secțiunea 6 din capitolul III.

2.2 Problema generală a ordonanțării Elementele primare sunt:

- o mulțime de utilaje;
- o mulțime de repere (joburi) de realizat (îndeplinit).

Realizarea unui reper necesită efectuarea unei secvențe de operații pe (unele din) utilajele date.Ordinea de parcurgere a utilajelor poate fi aceeaşi pentru toate reperele sau poate diferi de la un reper la altul.Fiecare operație necesită un anume timp, presupus cunoscut, şi două operații distincte,

executabile pe un acelaşi utilaj, nu pot fi programate simultan.Uneori pentru unele repere sunt impuse termene de livrare.

În aceste ipoteze,cum trebuie organizată lansarea reperelor în execuție astfel încât:

- timpul total de inactivitate al utilajelor să fie minim

sau

- durata de execuție a tuturor reperelor să fie minimă

sau

- numărul reperelor al căror termen de livrare este depăşit să fie cât mai mic etc.

Există o mare varietate de tipuri de probleme de ordonanțare care se înscriu în acest enunț general. Modelarea lor - prin programare în numere întregi sau prin grafuri - este în general dificilă, ca să nu mai vorbim de rezolvarea lor "exactă", imposibil de realizat în timp rezonabil, chiar şi atunci când numărul reperelor şi al utilajelor este relativ mic. Ca urmare, au fost elaborate euristici de rezolvare suboptimală, marea lor diversitate fiind explicată prin aceea că ele încearcă să exploateze particularitățile structurii acestor probleme, particularități care pot să difere radical de la o problemă la alta.

Unele cazuri particulare vor fi studiate în secțiunea 5 a acestui capitol.

2.3 Problema alegerii traseelor O firmă trebuie să satisfacă un număr de cereri de aprovizionare într-o anumită perioadă. Livrările se fac de la un anumit depozit şi sunt operate cu mai multe de mijloace de transport, fiecare cu o capacitate limitată. Fiecare vehicul pleacă încărcat de la depozit, vizitează unul sau mai mulți clienți, cărora le livrează comenzile făcute după care, complet descărcat, se întoarce la depozit pentru a fi eventual reîncărcat. Problema care se pune în acest context este aceea de a stabili traseele de vizitare ale clienților astfel încât distanța totală parcursă de vehicule să fie minimă.

O posibilitate de rezolvare suboptimală este descrisă în secțiunea 6 din acest capitol.

2.4 Problema orariilor Într-o universitate trebuie programată activitatea de învățământ pe următorul semestru. Se cunosc:

- numărul cursurilor care trebuie ținute;
- numărul claselor disponibile;
- profesorii abilitați să țină cursurile.

Cum trebuie făcută repartizarea profesorilor pe cursuri şi a cursurilor pe clase astfel încât:

- două cursuri diferite să nu fie programate în aceeaşi clasă şi în acelaşi timp;
- cursurile să fie atribuite profesorilor de specialitate;
- doi profesori cu aceeaşi specialitate să nu fie repartizați pe acelaşi curs.

O asemenea repartizare complexă se numeşte **orar** şi până aici s-a formulat doar o problemă de **existență.** "Optimizarea" elaborării unui orar este o chestiune delicată existând multe criterii de apreciere care nu întotdeauna sunt concordante. Din punctul de vedere al studenților un orar este "bun" dacă minimizează totalul deplasărilor zilnice de la o clasă la alta. Atât pentru studenți cât şi pentru profesori contează numărul "ferestrelor" adică al intervalelor de timp dintre două activități zilnice consecutive care ar trebui să fie cât mai mic, etc.

2.5 O problemă de comunicații prin satelit Zilnic, un mare volum de informații TV, radio sau telefon se transmite prin sateliți. Utilizarea sateliților a devenit imperios necesară pentru

asigurarea transmiterii informațiilor la mare distanță. În mare, un satelit de comunicații este un "releu": el preia semnalul unei stații terestre de emisie şi îl retransmite altei stații, de recepție.Dispozitivul de la bordul satelitului care realizează legătura dintre cele două stații se numeşte **transponder** sau **conector**; sateliții moderni au până la 24 de conectoare. Una din tehnologiile avansate de operare a sistemelor de comunicații prin satelit este SS/TDMA (satellite switched time division multiple access) ; în cadrul acesteia fiecare conector de la bordul satelitului este atribuit - pentru un interval de timp - unei perechi de zone terestre (stații de emisie - recepție). Un set complet de atribuiri se va numi în continuare **mod de conectare**. Atribuirile ce compun un mod de conectare se efectuează simultan de la bordul satelitului. O secvență de moduri de conectare care permite realizarea unui program de legături pe o perioadă de timp se numeşte **structură de conectare.**

O problemă importantă care apare în legătură cu tehnologia SS/TDMA este **planificarea eficientă** în timp a momentelor de realizare ale atribuirilor ce compun o structură de conectare.

Trecem acum la formalizarea problemei. "Cererea" corespunzătoare unei structuri de conectare poate fi reprezentată printr-o matrice:

$$D = [d_{ij}] \quad 1 \leq i,j \leq n$$

numită **matrice de trafic** unde:

$n \equiv$ numărul total de zone terestre între care trebuie realizate legăturile;

$d_{ij} \equiv$ "lungimea" trenului de date necesar a fi transmis din zona i în zona j, lungime exprimată de regulă prin timpul necesar transmiterii trenului de date respectiv.

Transmiterea acestor date implică o programare în timp a conectorilor de pe satelit pe diferite perechi de zone. Aceasta înseamnă **descompunerea** matricii de trafic D într-un număr de matrici de aceeaşi dimensiune cu ea, numite **matrici de conectare:**

$$D^k = [d_{ij}^k] \quad k = 1,2,\ldots,q$$

cu următoarele proprietăți:

- pe fiecare rând sau coloană din D^k există cel mult un element nenul (altfel spus, în D^k există cel mult n elemente nenule, oricare două nesituate pe acelaşi rând sau coloană. Aceasta revine la a spune că fiecare conector este atribuit la cel mult o pereche de zone;
- suma celor q matrici de conectare este egală cu matricea de trafic dată:

$$D = \sum_{k=1}^{q} D^k$$

Rezultă imediat că descompunerea matricii D în matrici cu proprietățile de mai sus este posibilă numai dacă $q \geq n$.

Criteriul de performanță ce trebuie avut în vedere în primul rând este transmiterea datelor între diferitele zone în timpul cel mai scurt. Este clar că timpul necesar transmiterii datelor corespunzătoare unui mod de conectare este dat de cea mai mare componentă a matricii de conectare aferente, astfel că timpul total asociat descompunerii ($D^1,D^2,\ldots,D^q$) este:

$$T = \sum_{k=1}^{q} \max_{i,j} d_{ij}^{k}$$

Cu aceste pregătiri se obține următoarea problemă de optimizare combinatorială:

Să se determine o descompunere $(D^1, D^2, ..., D^q)$ a matricii de trafic D în matrici de conectare astfel încât timpul total de transmisie T să fie minim.

Se observă că dacă q > n anumite trenuri de date d_{ij} din D vor fi fragmentate şi transmise pe parcursul mai multor moduri de conectare. Impunând condiția ca fiecare tren de date să fie transmis într-un singur mod de conectare rezultă problema:

Să se descompună matricea de trafic D în n matrici de conectare $D^1, D^2, ..., D^n$ astfel încât timpul total de transmisie T să fie minim.

§ 3. Problema arborelui de cost minim

Dată în secțiunea 1 ca un prim exemplu de problemă de optimizare combinatorială, problema arborelui de cost minim are numeroase aplicații practice. Contextul general este următorul: un număr de "puncte" trebuie conectate între ele pentru facilitarea transmiterii unui anumit serviciu (telefon, TV, calculator) Între puncte există "legături potențiale" a căror realizare implică un anumit cost. Problema care apare este aceea de a vedea ce legături vor fi efectiv realizate stfel încât:

- oricare două puncte să fie conectate – direct sau indirect – în vederea utilizării serviciului;
- suma costurilor legăturilor realizate să fie minimă.

Exemplul 3.1 [6] Centrul regional de calculatoare X trebuie să instaleze un număr de linii speciale de comunicații care să lege cinci utilizatori la un nou computer. Deoarece instalarea rețelei este costisitoare conducerea centrului dorește ca lungimea totală a liniilor instalate să fie cât mai mică. Deşi computerul central poate fi conectat direct cu toți utilizatorii `ar` fi mai economic să fie instalate linii directe doar către unii, ceilalți intrând în rețea prin legarea lor la utilizatorii deja conectați.Rețeaua legăturilor posibile este vizualizată prin graful din fig 3.1. Valorile numerice înscrise pe muchii reprezintă distanțe în km.

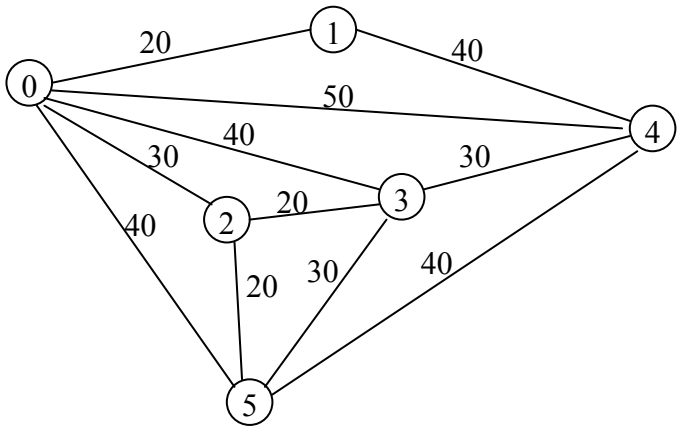


Figura 3.1

Exemplul 3.2 Prefectura judeţului X şi-a fixat ca obiectiv modernizarea reţelei drumurilor care leagă între ele localităţile 1,2,…,8 reţea vizualizată în fig. 3.2. Pe fiecare muchie este înscrisă o valoare numerică reprezentând costul modernizării tronsonului respectiv (în milioane lei). Din cauza bugetului limitat, prefectura doreşte ca într-o primă etapă să modernizeze numai unele drumuri astfel încât:

- între oricare două localităţi să se poată circula pe tronsoane modernizate;
- costul întregii operaţii de modernizare parţială să fie minim.

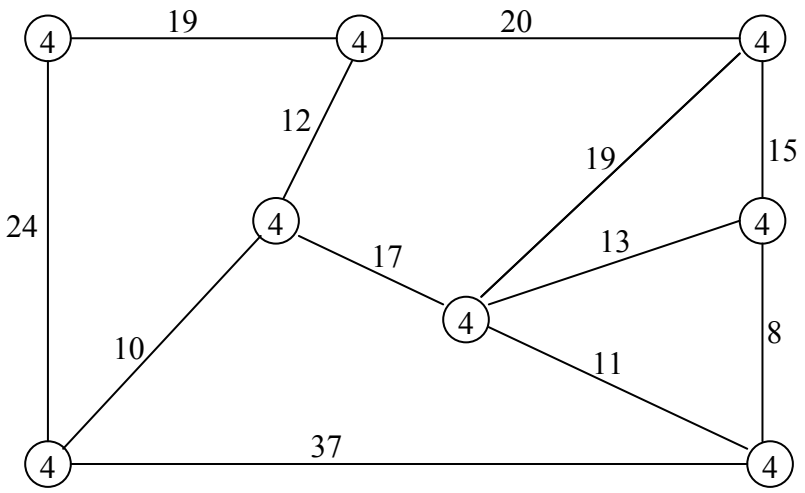


Figura 3.2

Rezolvarea problemelor de acest tip se face cu ajutorul unui algoritm foarte simplu, datorat lui **Kruskal**:

- selectarea muchiilor care vor forma arborele căutat se va face în ordinea crescătoare a costurilor;
- presupunând că muchiile e_1 , e_2 ,…, e_k au fost deja selectate, următoarea muchie e_{k+1} va fi astfel aleasă încât să nu formeze ciclu cu (unele din) muchiile e_1 , e_2 ,…, e_k.

Este clar că procedura se opreşte după un număr finit de paşi. Dacă numărul total de muchii selectate este n – 1, n fiind numărul nodurilor din graful original, s-a găsit un arbore de cost minim; altminteri, graful dat nu este conex!

Exemplul 3.3 Vom aplica algoritmul lui Kruskal în graful conex din fig. 3.1. Există trei muchii cu costul minim 20;ele nu formează un ciclu astfel că ele pot fi selectate în orice ordine, de exemplu: {0,1} , {2,3} , {2,5}. Mai departe a fost aleasă muchia {0,3} urmată de {3,4}; muchia {3,5}a fost respinsă deoarece forma un ciclu cu muchiile {2,3} şi {2,5} deja selectate! Procedura s-a oprit deoarece graful are şase noduri şi au fost alese cinci muchii.Arborele cu costul minim 120 este reprezentat în fig.3.3

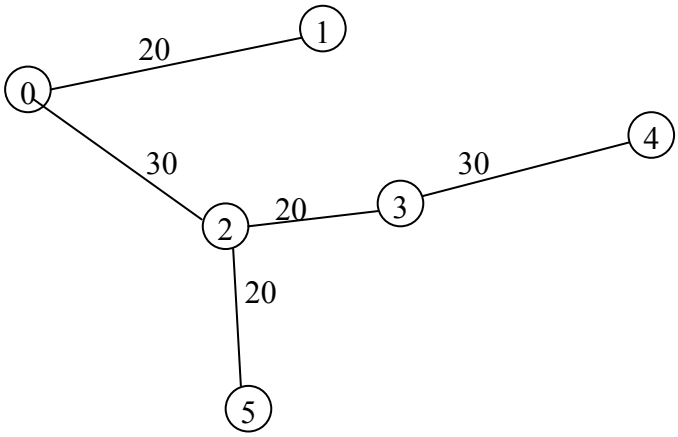


Figura 3.3

§ 4. Problema comisvoiajorului

Un *comisvoiajor* are de vizitat un număr de *oraşe*, la sfârşitul voiajului el întorcându-se în localitatea de plecare. Deplasarea dintr-o localitate în alta presupune anumite *cheltuieli* (sau *distanțe* de parcurs sau *durate* de mers).

In ce ordine trebuiesc vizitate oraşele astfel încât costul total al deplasării (sau lungimea traseului sau durata călătoriei) să fie minim?

Astfel enunțată, problema comisvoiajorului (notată pe scurt TSP de la **T**ravelling **S**alesman **P**roblem) este una din cele mai grele probleme de optimizare combinatorială. Pe lângă importanța teoretică, ea are numeroase aplicații practice. De exemplu, stabilirea celor mai "economice" trasee turistice într-un mare oraş sau zonă istorică (din punctul de vedere al costurilor sau distanțelor) este o probemă a comisvoiajorului. De asemenea, stabilirea ordinii în care un număr de operații (joburi) vor fi executate pe un utilaj, astfel încât suma timpilor de pregătire ai utilajului să fie cât mai mică, este o problemă de acelaşi tip.

4.1 Modelul matematic al problemei comisvoiajorului

Să notăm cu 0,1,...,n oraşele problemei, 0 fiind oraşul din care comisvoiajorul pleacă şi în care revine la terminarea călătoriei. Fie c_{ij} *costul* deplasării din localitatea i în localitatea $j \neq i$; punem $c_{ij} = \infty$ dacă nu există legătură directă între i şi j sau în cazul i = j. Un *traseu* va fi descris cu ajutorul variabilelor *bivalente* :

$$x_{ij} = \begin{cases} 1 \text{daca traseul trece direct de la orasul i la orasul j, } j \neq i; \\ 0 \text{ in caz contrar sau daca } i = j; \end{cases}$$

Urmează a fi minimizată funcția:

$$f = \sum_{i,j=0}^{n} c_{ij} x_{ij} \tag{4.1.1}$$

Din orice oraş i, comisvoiajorul trebuie să se îndrepte către o altă localitate, astfel că:

$$\sum_{j=0}^{n} x_{ij} = 1 \qquad i = 0,1,\ldots,n \tag{4.1.2}$$

In orice oraş ar ajunge, comisvoiajorul vine dintr-o localitate anterior vizitată, aşa că:

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad j = 0,1,\ldots,n \tag{4.1.3}$$

Observăm că ansamblul (4.1.1) - (4.1.3) constituie modelul matematic (PA) al unei probleme generale de afectare. O examinare mai atentă a problemei arată că restricțiile (4.1.2) şi (4.1.3) nu definesc numai trasee ce trec prin toate oraşele (o singură dată!) ci şi "reuniuni" de circuite disjuncte mai mici! Astfel, într-o problemă cu 7 oraşe 0,1,...,6 ansamblul:

$$x_{03} = x_{35} = x_{50} = x_{21} = x_{14} = x_{46} = x_{67} = x_{72} = 1 \quad , \quad x_{ij} = 0 \quad \text{în rest}$$

satisface restricțiile (4.1.2) şi (4.1.3) dar nu constituie un traseu complet aşa cum se vede din figura 4.1

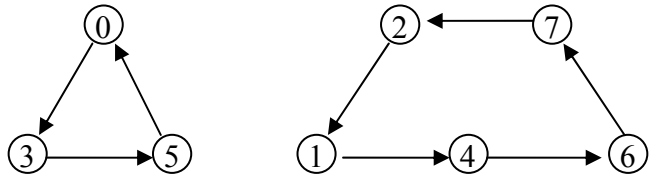


Fig. 4.1

Pentru evitarea ecestui neajuns asociem oraşelor 1,2,...,n , diferite de localitatea de plecare şi sosire 0, variabilele $y_1,y_2,\ldots,y_n$ şi introducem inegalitățile:

$$y_i - y_j + n \cdot x_{ij} \le n - 1 \qquad i , j = 1,2,\ldots,n \; ; \; i \ne j \tag{4.1.4}$$

In principiu noile variabile pot lua orice valoare reală. Vom arăta că dacă $x = (x_{ij})$ determină un traseu complet atunci există valori numerice $y_1,y_2,\ldots,y_n$ care împreună cu x_{ij} satisfac relațiile (4.1.4). Intr-adevăr, definim y_i ca fiind numărul de ordine al localității i în cadrul traseului determinat de x. Pentru $i \ne j$ două situații sunt posibile:

- localitatea j nu urmează imediat după localitatea i. Atunci $x_{ij} = 0$ şi (4.1.4) devine $y_i - y_j \le n - 1$ ceeace este adevărat având în vedere semnificația acordată valorilor $y_1,y_2,\ldots,y_n$.
- localitatea j urmează imediat după localitatea i. Atunci $y_j = y_i + 1$, $x_{ij} = 1$ şi (4.1.4) se verifică cu egalitate.

Să presupunem acum că $z = (x_{ij})$ defineşte o reuniune de "subtrasee" disjuncte. Alegem unul care nu trece prin localitatea 0 şi fie k numărul deplasărilor cel compun. Presupunând că ar exista valori numerice $y_1,y_2,\ldots,y_n$ care să satisfacă (4.1.4) însumăm pe acelea corespunzătoare deplasărilor de la un oraş la altul din subtraseul ales. Obținem $n \cdot k \le (n - 1) \cdot k$ ceeace este evident fals.

In concluzie, ansamblul (4.1.1) - (4.1.4) constituie modelul "corect" al problemei comisvoiajorului. Este vorba de un *program mixt* ce utilizează atât variabile întregi (bivalente) cât şi variabile care pot lua orice valoare. In continuarea unei observații anterioare, putem spune că *problema comisvoiajorului* (TSP) este o *problemă de afectare* (PA) (ansamblul (4.1.1) - (4.1.3)) cu restricțiile *speciale* (4.1.4).

In considerațiile de mai sus costurile c_{ij} nu sunt "*simetrice*" adică este posibil ca $c_{ij} \ne c_{ji}$. *Problema simetrică a comisvoiajorului* se caracterizează prin $c_{ij} = c_{ji}$; aceasta se întâmplă dacă , de exemplu, c_{ij} sunt distanțe sau timpi de parcurgere.

Un caz particular al problemei simetrice este aşa numita *problemă euclidiană a comisvoiajorului* în care distanţele c_{ij} satisfac inegalitatea triunghiului :

$$c_{ik} \le c_{ij} + c_{jk} \qquad (\forall)\ i,j,k.$$

4.2 Un algoritm exact de rezolvare a problemei comisvoiajorului

Următorul algoritm, de tip B&B, reduce rezolvarea TSP la rezolvarea unei *liste de probleme de afectare*. El s-a dovedit suficient de robust pentru probleme *asimetrice* cu un număr *moderat* de oraşe. Principalul dezavantaj îl reprezintă faptul că numărul problemelor de afectare de rezolvat este impredictibil şi poate fi imens în cazul situaţiilr reale.Algoritmul este datorat lui Eastman. Ca orice procedură de tip B&B, el utilizează:

- o "locaţie" x_{CMB} care va reţine "cel mai bun" traseu găsit de algoritm până la un moment dat;
- o variabilă z_{CMB} care reţine costul traseului memorat în x_{CMB};
- o listă `L` de probleme de afectare asemănătoare problemei (PA) şi care diferă de problema (PA) iniţială prin anumite costuri c_{ij} schimbate în ∞;

La start:

- x_{CMB} poate fi locaţia "vidă" în care caz $z_{CMB} = \infty$ sau reţine un traseu arbitrar ales sau generat printr-o metodă euristică, x_{CMB} fiind iniţializat cu costul aferent acestui traseu;
- lista `L` cuprinde numai problema (PA) iniţială, determinată de (4.1.1) - (4.1.3);

Pasul 1. Dacă lista `L` este vidă **Stop**. Altminteri, selectează o problemă din lista `L` . Problema aleasă se şterge din `L` . Se trece la:

Pasul 2. Se rezolvă problema selectată. Dacă valoarea funcţiei obiectiv este $\ge z_{CMB}$ se revine la pasul 1. Altminteri, se trece la:

Pasul 3. Dacă soluţia optimă este un traseu complet se actualizează x_{CMB} şi z_{CMB} după care se revine la pasul 1. Altminteri se trece la:

Pasul 4. (Soluţia optimă nu este un traseu ci o reuniune de "subtrasee mai mici"). Din soluţia optimă se selectează circuitul cu cel mai mic număr de arce. Pentru fiecare arc (i,j) din circuitul ales (pentru care $x_{ij} = 1$) se adaugă la lista `L` problema obţinută din cea rezolvată punând $c_{ij} = \infty$. Se revine la pasul 1.

Observaţii: 1) Raţiunea pasului 4 este clară: dacă $i \to j \to k \to ... \to l \to i$ este circuitul selectat este clar că în traseul optim vom avea:

$$x_{ij} = 0 \text{ sau } x_{jk} = 0 \text{ sau } ... \text{ sau } x_{li} = 0.$$

In consecinţă, vom căuta traseul optimal printre cele în care $x_{ij} = 0$ şi dacă nu este acolo printre cele în care $x_{jk} = 0$ ş.a.m.d. Limitarea la traseele în care $x_{ij} = 0$ se face efectuând schimbarea $c_{ij} = \infty$.

2) La pasul 4, în lista `L` vor apare atâtea probleme câte arce are circuitul selectat; de aceea se alege circuitul cu cel mai mic număr de arce!

In raport cu termenii generali ai unei metode B&B, *ramificarea* are loc la pasul 4 în timp ce *mărginirea* se efectuează la pasul 2.

Exemplul 4.2.1 Se consideră problema asimetrică a comisvoiajorului cu cinci oraşe, definită de următoarea matrice a costurilor:

Oraşe	**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	∞

Tabelul 4.1

Start. Se pleacă cu traseul $0 \to 1 \to 2 \to 3 \to 4 \to 0$, care se reţine în x_{CMB} şi al cărui cost este $z_{CMB} = 10 + 10 + 20 + 15 + 10 = 65$. Lista `L` a problemelor de afectare ce urmează a fi rezolvate cuprinde numai problema iniţială (PA).

Iteraţia 1. Se şterge (PA) din lista `L` şi se rezolvă. Se obţine soluţia optimă:

$$x_{04} = x_{40} = x_{12} = x_{23} = x_{31} = 1 \quad , \quad x_{ij} = 0 \text{ în rest}$$

ce corespunde reuniunii următoarelor subtrasee:

$$0 \to 4 \to 0 \quad \text{şi} \quad 1 \to 2 \to 3 \to 1$$

Selectăm subtraseul $0 \to 4 \to 0$ şi adăugăm la `L` problemele PA_1 şi PA_2 derivate din PA înlocuind c_{04} respectiv c_{40} cu ∞ .

Iteraţia 2. Selectăm problema PA_1 şi o rezolvăm. Lista `L` va cuprinde mai departe numai problema PA_2. Pentru PA_1 se găseşte soluţia optimă:

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

care corespunde traseului complet $0 \to 3 \to 1 \to 2 \to 4 \to 0$. Costul său este $65 = z_{CMB}$. Nu am găsit deci un traseu mai bun decât cel pe care-l avem în x_{CMB}.

Iteraţia 3. Rezolvăm problema PA_2, singura rămasă în `L` .Se obţine soluţia optimă:

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

care corespunde traseului $0 \to 4 \to 1 \to 2 \to 3 \to 0$. Costul noului traseu este $62 < z_{CMB}$,drept care îl reţinem în x_{CMB} şi actualizăm $z_{CMB} = 62$.

Deoarece lista `L` este acum vidă traseul reţinut în x_{CMB} este optim.

4.3 Metode euristice de rezolvare aproximativă a problemei comisvoiajorului

De obicei, soluţia optimă a unei TSP este foarte greu de găsit (dacă nu chiar imposibil de găsit) atunci când numărul oraşelor de vizitat este mare (peste 50). Pentru determinarea unei soluţii măcar acceptabile au fost elaborate mai multe procedee euristice. Aceste procedee merită atenţie din două puncte de vedere:

- se poate da un "*certificat de garanţie*" pentru soluţia obţinută în sensul că este posibilă evaluarea "*depărtării*" maxime de soluţia *optimă;*
- soluţia aproximativă este găsită cu un *efort de calcul relativ moderat* şi într-un *timp rezonabil*, aceasta însemnând că cei doi parametri depind *polinomial* de dimensiunea problemei (adică numărul de oraşe);

In multe situații practice, procedeele euristice au condus chiar la soluția optimă dar acest lucru a fost confirmat numai prin aplicarea unui algoritm exact.

In principiu, o metodă euristică construieşte o soluție *prin tatonare, din aproape în aproape, făcând la fiecare pas, cea mai bună alegere posibilă.* Din nefericire, această "schemă" *nu* conduce, de regulă, la *cea mai bună alegere globală.*

In continuare, vom prezenta mai multe euristici pentru rezolvarea *problemei euclidiene a comisvoiajorului*, adică a problemei în care c_{ij} sunt *distanțe* ce satisfac:

- condiția de simetrie: $c_{ij} = c_{ji}$:
- inegalitatea triunghiului $c_{ik} \leq c_{ij} + c_{jk}$;

Euristica E_1 (mergi la cel mai apropiat vecin)

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

Exemplul 4.3.1 Să considerăm cele 13 orașe din rețeaua laticială din figura 4.2; distanțele dintre orașe sunt date în tabelul 4.2.

Aplicând euristica descrisă se obține traseul din figura 4.3 cu lungimea 5099.

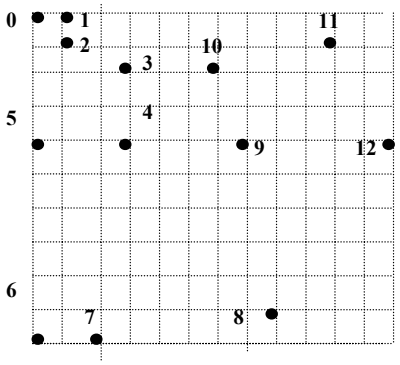


Figura 4.2

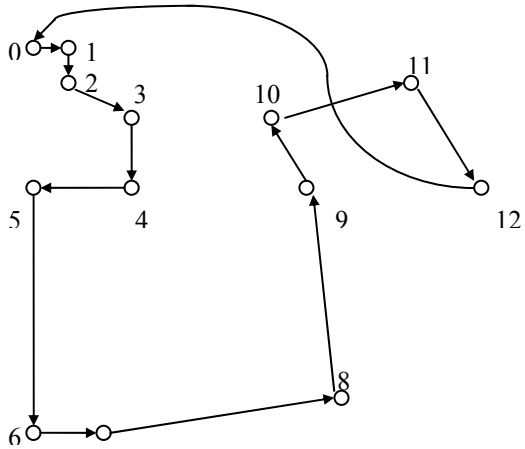


Figura 4.3

Localitatea	0	1	2	3	4	5	6	7	8	9	10	11	12
0	∞	100	141	360	500	400	1000	1019	1204	806	632	1004	1204
1		∞	100	282	447	412	1004	1004	1140	721	538	905	1170
2			∞	223	360	316	905	905	1063	670	509	900	1140
3				∞	200	360	854	806	860	447	300	707	921
4					∞	300	670	608	707	400	360	761	900
5						∞	600	632	943	700	632	1044	1200
6							∞	200	806	921	1000	1345	1341
7								∞	608	781	894	1204	1166
8									∞	509	728	824	640
9										∞	223	424	500
10											∞	412	632
11												∞	360
12													∞

Tabelul 4.2

Euristica E_2 (Ajustare locală)

- se pleacă de la un traseu complet, determinat fie "la întâmplare" fie printr-o altă euristică;
- se caută două arce (u,v) şi (x,y) ale traseului care se încrucişează - ca în figura 4.4 a) - şi se compară distanţele $c_{uv} + c_{xy}$ cu $c_{ux} + c_{vy}$. Dacă $c_{uv} + c_{xy} > c_{ux} + c_{vy}$ se înlocuiesc arcele (u,v) şi (x,y) cu (u,x) şi (v,y) obţinându-se un traseu de lungime mai mică (vezi figura 4.4 b));
- procedura se opreşte în momentul în care nu mai există situaţii similare cu cea descrisă;

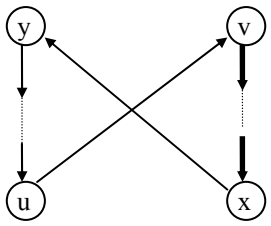


Figura 4.4 a)

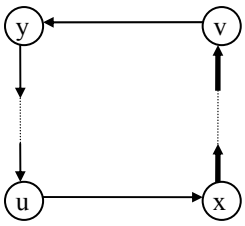
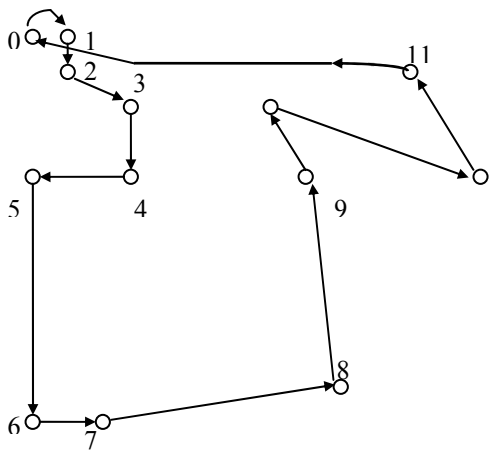


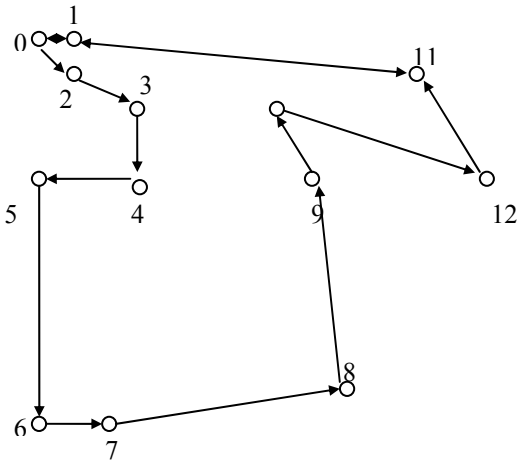
Figura 4.4 b)

Exemplul 4.3.2 Pentru ilustrare să considerăm traseul obţinut din euristica E_1. In figura 4.3 este vizibilă încrucişarea arcelor (10,11) şi (12,0). Deoarece:

$$c_{10,11} + c_{12,0} = 424 + 1264 = 1688 > 1636 = 632 + 1004 = c_{10,12} + c_{11,0}$$

un traseu mai scurt se obţine folosind arcele (10,12) şi (11,0) aşa cum se arată în figura 4.5 a).





a) b)

Figura 4.5

Lungimea noului traseu este: 5099 - (1688 - 1636) = 5047.

Rezolvând analog încrucişarea nou apărută a arcelor (1,2) şi (11,0) - vezi figura 4.5 a) - în care:

$$c_{1,2} + c_{11,0} = 100 + 1004 = 1104 > 1046 = 141 + 905 = c_{0,2} + c_{11,1}$$

se obține traseul din figura 4.5 b) cu lungimea: 5047 - (1104 - 1046) = 4989.
Deoarece în noul traseu nu mai apar încrucişări euristica E_2 se opreşte.

Următoarele procedee sunt "mai performante" dar necesită şi un efort de calcul "mai mare".

Euristica E_3

- Se determină un *arbore* H *de lungime minimă* (aceasta este o problemă "uşoară" rezolvabilă de exemplu cu algoritmul lui Kruskal, vezi seţiunea 3);
- fiecare muchie a arborelui H şe înlocuieşte cu *două muchii* de aceeaşi lungime. Se obţine un *multigraf* H' în care fiecare nod are grad *par*. Conform teoremei lui Euler în H' există un *ciclu eulerian*; cu alte cuvinte, este posibil să se viziteze oraşele astfel încât fiecare muchie a multigrafului H' să fie parcursă *o singură dată*.

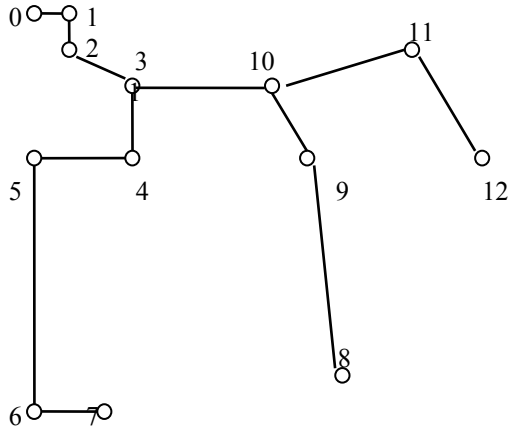
Fie:

$$0 \to x \to y \to \ldots \to u \to v \to 0 \qquad (4.3.1)$$

succesiunea în care muchiile multigrafului H' au fost parcurse. Lungimea acestui "traseu" este de două ori mai mare decât lungimea arborelui minimal H. In această secvenţă, unul sau mai multe oraşe apar de mai multe ori. Vom proceda atunci la următoarea *operaţie de scurtcircuitare*:

- în succesiunea (4.3.1) se determină prima tripletă de oraşe succesiv vizitate

$$i \to j \to k$$



cu proprietatea că oraşul j "din mijloc" *mai apare ulterior în succesiune*. Se înlocuieşte secvenţa i $\to$ j $\to$ k cu arcul i $\to$ k. Prin această scurtcircuitare:

- fiecare oraş continuă să fie vizitat cel puţin o dată;
- noul traseu este mai scurt pentru că $c_{ij} + c_{jk} \geq c_{ik}$;

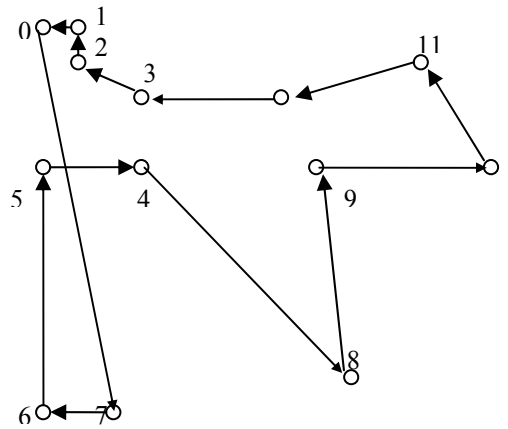
Operaţia de scurtcircuitare se repetă până când în secvenţa (4.3.1) "actualizată" fiecare oraş, cu excepţia localităţii 0 apare o singură dată.

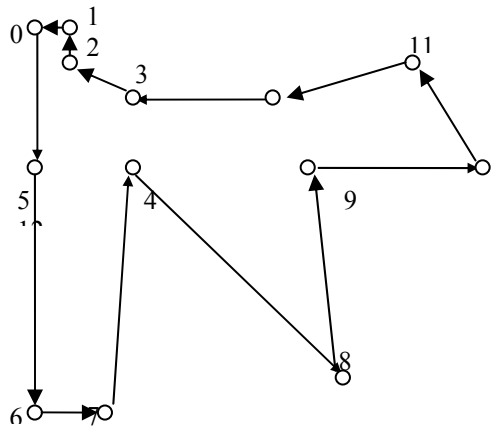
Se poate demonstra că lungimea traseului obţinut prin această euristică este cel mult de două ori mai mare decât lungimea traseului optim (în practică lucrurile stau însă mai bine...)

Figura 4.6

Exemplul 4.3.3 In figura 4.6 este vizualizat un arbore minimal H pentru graful din figura 4.2. Un ciclu eulerian în multigraful H', dedus din H prin "dublarea" muchiilor acestuia, arată astfel:

$0 \to 1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 6 \to 5 \to 4 \to 3 \to 10 \to 9 \to 8 \to 9 \to 10 \to 11 \to 12 \to 11 \to 10 \to 3 \to 2 \to 1 \to 0$





În succesiunea $0 \to 1 \to 2$ de lungime $c_{01} + c_{12} = 100 + 100 = 200$ oraşul 1 mai este vizitat "în final". Conform celor de mai sus, vom înlocui această succesiune cu arcul direct $0 \to 2$ de lungime $141 < 200$. Rezultă traseul:

$0 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 6 \to 5 \to 4 \to 3 \to 10 \to 9 \to 8 \to 9 \to 10 \to 11 \to 12 \to 11 \to 10 \to 3 \to 2 \to 1 \to 0$

Mai departe succesiunea $0 \to 2 \to 3$ de lungime $141 + 223 = 364$ se înlocuieşte cu arcul direct $0 \to 3$ cu lungimea 360, pentru că prin oraşul 2 se mai trece o dată!
Continuând procesul de scurtcircuitare se obține în final traseul complet:

$$0 \to 7 \to 6 \to 5 \to 4 \to 8 \to 9 \to 12 \to 11 \to 10 \to 3 \to 2 \to 1 \to 0$$

vizualizat în figura 4.7; lungimea sa este 5330. Rezolvând "încrucişarea" arcelor (0,7) şi (5,4) se obține traseul mai scurt din figura 4.8 cu lungimea 5019.

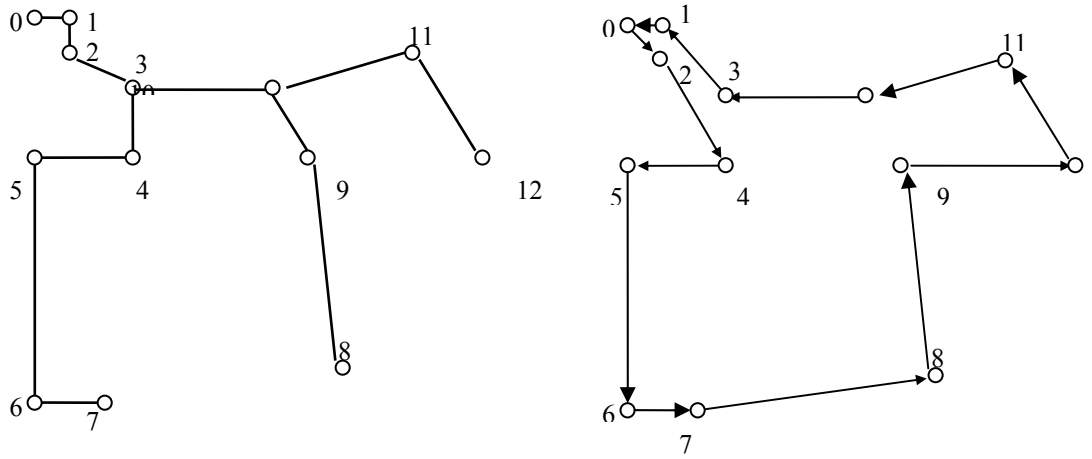
Figura 4.7 Figura 4.8

Menționăm faptul că euristica descrisă poate da rezultate diferite, unele mai bune altele mai puțin bune, funcție de alegerea arborelui minimal H şi a ciclului eulerian în multigraful asociat H'.

Euristica E_4

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 vom determina *cuplajul* C *de pondere minimă,* ponderile muchiilor luate în calcul fiind distanțele dintre extremități.

Fie H' graful rezultat din H prin adăugarea muchiilor cuplajului C, cu mențiunea că dacă între două noduri avem o muchie în H şi o alta în C, graful H' va avea între nodurile respective două muchii.Prin construcție, în H' fiecare nod are grad *par*, astfel că H' are un *ciclu euleri an*. Plecând de aici şi utilizând *tehnica scurtcircuitării* se ajunge la un traseu complet.



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

Exemplul 4.3.4 Reluăm problema euclidiană a comisvoiajorului definită în figura 4.2 şi tabelul 4.2. Un arbore minimal cu lungimea 3527 apare în figura 4.6. Nodurile din H de grad impar sunt 0, 3, 7, 8 şi 12. Prin simpla inspecție (sau utilizând un algoritm de determinare a cuplajului de pondere minimă în subgraful *complet* ale cărui noduri sunt 0, 3, 7, 8 şi 12) se constată că muchiile {0,3} , {7,8} şi {10,12} au cea mai mică pondere totală: $c_{03} + c_{78} + c_{10,12} = 360 + 608 + 632 = 1600$. Adăugând cele trei muchii la arborele H se obține graful H' din figura 4.9. Vizitând oraşele 0,1,...,12 în ordinea:

$$0 \to 1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9 \to 10 \to 11 \to 12 \to 10 \to 3 \to 0$$

toate muchiile grafului H' vor fi parcurse o singură dată. "Scurtcircuitând" secvențele $2 \to 3 \to 4$ şi $9 \to 10 \to 11$ se obține un traseu complet cu lungimea 4853. Invităm cititorul să se convingă prin desen că în acest traseu arcele (9,11) şi (12,10) respectiv (1,2) şi (3,0) se încrucişează. Rezolvând şi aceste încrucişări se obține în final traseul din figura 4.10 cu lungimea 4672.

Ca şi euristica precedentă şi E_4 poate conduce la rezultate diferite, funcție de alegerea arborelui minimal H, al cuplajului C şi al ciclului eulerian în (multi)graful H'.

Figura 4.9 Figura 4.10

§ 5. Introducere în problema ordonanțării

5.1 Problema ordonanțării. Aspecte generale

Considerăm un *proces tehnologic* în care, asupra unor obiecte (materie primă, semifabricate) denumite generic *repere*, se efectuează o serie de transformări cantitative şi/sau calitative în vederea obținerii unor *produse* (acestea pot fi produse finite sau semifabricate mai complexe ce devin repere în alte procese tehnologice). Transformările (prelucrările) pe care le suferă un reper în diferitele faze ale procesului tehnologic se execută pe utilaje specializate. Pentru precizia limbajului vom numi *operație* totalitatea acțiunilor de prelucrare efectuate asupra unui reper pe un anumit utilaj. In principiu, un utilaj poate executa operații aferente mai multor repere dar, la un moment dat, nu poate executa decât o singură operație.

Efectuarea unei operații presupune utilizarea unor *resurse* materiale sau băneşti; avem în vedere în primul rând *durata* operației care se "extrage" din fondul de timp productiv al utilajului pe care se execută operația respectivă.

In contextul descris, *problema ordonanțării reperelor pe utilaje* înseamnă *programarea în timp* a operațiilor necesare obținerii diferitelor produse din nomenclatorul procesului tehnologic studiat, adică stabilirea *ordinii* şi a *termenelor* la care urmează a fi efectuate aceste operații.

In elaborarea unui program concret de lansare în fabricație trebuie să se țină seama de un mare număr de factori cei mai mulți fiind specifici procesului tehnologic studiat. Vom menționa pe cei mai importanți fără pretenția de epuizare a subiectului.

- Cu puține excepții, planificarea nu se referă la repere individualizate ci la *loturi de repere*. În această situație, este posibil ca după ce o anumită operație a fost executată pe o parte a unui lot de repere, utilajul să fie eliberat şi pregătit pentru o operație ce se efectuează pe reperele altui lot, restul reperelor din lotul anterior urmând a fi prelucrat mai târziu. Deoarece în procesul de modelare un lot de repere identice este asimilat cu un singur reper (dar cu o durată de execuție proporțională cu mărimea lotului), situația descrisă mai sus este echivalentă cu întreruperea unei operații şi reluarea ei la un moment ulterior. In consecință, lansarea în execuție a reperelor pe utilaje poate fi concepută cu acceptarea sau neacceptarea ipotezei de *întrerupere a operațiilor.*

- *Ordinea* în care se execută cel puțin unele din operațiile aferente unui reper este esențială în foarte multe cazuri practice. Ca şi în *analiza drumului critic*, condiționările dintre operațiile unui reper pot fi complet descrise cu ajutorul relației de *precedență directă:* operația *a* precede *direct* operația *b* dacă *b* poate fi executată imediat după terminarea operației *a*. Putem vizualiza structura tehnologiei de fabricație a unui reper printr-un *graf orientat* ale cărui noduri sunt operațiile reperului în cauză, arcele reprezentând precedențele directe dintre operații.

Din punctul de vedere al ordinii în care se execută operațiile, problemele de ordonanțare implicând utilizarea a cel puțin două maşini se clasifică în trei categorii:

1) *ordinea* în care se execută operațiile oricărui reper *este neesențială* (**open shop**);
2) toate reperele parcurg utilajele *într-o aceeaşi ordine* (**flow shop**);
3) fiecare reper parcurge utilajele *într-o anumită ordine* definită de tehnologia de fabricație a reperului (**job shop**);

- In practică, pentru orice reper (sau lot de repere) există un *termen de predare* (*livrare*) la care toate operațiile trebuiesc încheiate; în consecință, un program concret de lansare în execuție va fi apreciat ca *admisibil* dacă operațiile fiecărui reper sunt încheiate *înaintea* termenului de predare.

- De cele mai multe ori, pentru a fi siguri că termenele de predare vor fi respectate, planificatorii îşi fixează pentru fiecare reper (sau lot de repere) un termen "intern" de predare care devansează mai mult sau mai puțin, funcție de specificul situației concrete, termenele de livrare. Există şi alte rațiuni care impun asemenea termene, ca de exemplu eliberarea utilajului la o anumită dată în vederea executării altor comenzi. Aceste termene "interne", denumite de obicei *termene impuse*, sunt foarte utile în aprecierea diferitelor programe posibile de lansare în execuție în sensul că un program va fi socotit mai bun decât altul dacă *întârzierile* față de termenele impuse vor fi mai mici fie *pe total* fie *individual*.

- Intr-o situație concretă, toate programele admisibile de ordonanțare raportează termenele de execuție ale operațiilor la un acelaşi *moment de referință:* zero sau o dată calendaristică. Deseori se întâmplă ca unele din utilajele necesare prelucrării reperelor situației să nu fie disponibile la momentul de referință, fiind antrenate în executarea altor comenzi. In raport cu momentul de referință ales de planificator, fiecare utilaj va avea un *termen de eliberare* de la care el devine disponibil (accesibil) şi un program viabil (realist) de ordonanțare trebuie să țină seama de aceste termene.O chestiune similară se pune şi în cazul reperelor care nu întotdeauna sunt disponibile la momentul de referință ci la anumite *termene minime de lansare în execuție.*

- De obicei, pentru a executa o anumită operație, utilajul desemnat trebuie pregătit (reglat). Există cazuri în care timpul de pregătire este apreciabil şi trebuie luat în considerare în elaborarea unui program realist de ordonanțare. Lucrurile se complică foarte mult dacă avem în vedere faptul că timpul de pregătire nu depinde numai de operația ce urmează a fi executată ci şi de operația anterior executată pe acelaşi utilaj.

- Este posibil ca, pentru executarea unei operații, să fie disponibile mai multe utilaje, fie identice, fie neidentice ca performanță. Este clar că, în al doilea caz, durata efectivă de execuție a operației depinde de performanțele utilajului desemnat.

Cu toate că lista factorilor restrictivi ar putea fi continuată, situațiile concrete de ordonanțare implică un număr enorm de programe admisibile. Se impune alegerea unui *criteriu de performanță* care să clasifice aceste programe. Criteriile pot diferi de la o situație la alta şi chiar pentru una şi aceeaşi situație se pot formula criterii *independente* (care să conducă la soluții *diferite*). Vom cita, cu titlu informativ, câteva din criteriile de performanță ce pot fi avute în vedere:

- minimizarea termenului la care toate operațiile de prelucrare ale reperelor sunt încheiate;
- minimizarea sumei întârzierilor față de termenele impuse;
- minimizarea celei mai mari întârzieri față de termenele impuse;
- minimizarea numărului de repere întârziate (față de termenele impuse);
- minimizarea sumei timpilor de pregătire - în cazul mai multor repere şi a unui singur utilaj;

Este clar că din combinarea factorilor mai sus amintiți şi a criteriilor de performanță adecvate rezultă o mare varietate de probleme de ordonanțare. Cercetările în domeniu - unele de ordin exclusiv "bibliografic" - au condus la identificarea a peste 3000 de probleme diferite.

Pentru uşurarea activității de inventariere şi clasificare a problemelor de ordonanțare a fost introdusă eticheta α / β / γ în care:

- α este un *câmp* ce conține informații privind numărul şi caracteristicile *utilajelor* precum şi ordinea de parcurgere a acestora;

- β este un *câmp* ce coține informații privitoare la *operații*: termene impuse, termene de livrare,termene de eliberare ale utilajelor, precedențele dintre operații (dacă există), posibilitatea de a intrerupe operațiile etc;

- γ este un *câmp* care specifică *criteriul de performanță* utilizat;

In marea lor ,majoritate, *problemele de ordonanțare sunt dificile* în sensul că soluția optimă este foarte greu de găsit dacă nu chiar imposibil de găsit în timp rezonabil chiar şi pentru probleme de gabarit "moderat". Acesta este şi motivul (a câta oară?) pentru care în asemenea situații se încearcă o rezolvare "aproximativă" bazată pe metode euristice.

5.2 Ordonanțarea pe o maşină

Un număr de repere (joburi) codificate 1,2,...,n trebuiesc lansate în execuție *pe un singur utilaj*.

1) Fiecărui reper îi sunt asociate următoarele *date*:

- *durata* p_j;
- *termenul impus* d_j;
- *termenul minim de lansare în execuție* r_j;
- *termenul de livrare* $\overline{d}_j$;
- *timpul de pregătire* p_{ij} *al utilajului pentru trecerea de la jobul i la jobul j;*

Este clar că $r_j + p_j \le d_j \le \overline{d}_j$. Unele dintre aceste date pot fi ignorate într-un caz concret sau altul; în orice caz "obligatorii" sunt duratele şi în cele mai multe cazuri termenele impuse.

2) In ansamblul lor joburile pot fi:

- *independente*, putând fi lansate în fabricație în orice ordine;
- *intercondiționate*;

3) Joburile pot fi executate:

- *cu întrerupere*;
- *fără întrerupere*;

4) Fiecărui job i se ataşează o *variabilă* C_j reprezentând *momentul terminării* execuției sale.
Notă: toate termenele impuse, de livrare, de terminare sunt raportate la un *acelaşi moment de referință* 0 astfel că toate sunt exprimate prin numere reale *nenegative*.

5) *Calitatea* unui program de lansare în execuție a joburilor poate fi evaluată prin mai multe criterii de performanță.

a) de tip *loc îngust*:

- minimizarea *întârzierii maxime*: $\min L_{max} = \max_j \{L_j = C_j - d_j\}$
- minimizarea *costului maxim* datorat întârzierilor: $\min f_{max} = \max_j \{f_j(C_j)\}$

unde $f_j(C_j)$ este o funcție *nedescrescătoare* în variabilă C_j cu $f_j(C_j) = 0$ dacă $C_j \le d_j$;

b) de tip *aditiv*:

- minimizarea *sumei costurilor datorate întârzierilor*: $\min \sum_j f_j(C_j)$;
- minimizarea *sumei ponderate a termenelor de terminare*: $\min \sum_j w_j C_j$;
- minimizarea *sumei întârzierilor efective*: $\min \sum_j \{T_j = \max(0, C_j - d_j)\}$;
- minimizarea *numărului de joburi întârziate*: $\min \sum_j U_j$ unde

$$U_j = \begin{cases} 0 \text{ daca } T_j = 0 \\ 1 \text{ daca } T_j > 0 \end{cases};$$

c) care depind nu numai de caracteristicile individuale ale joburilor dar şi de *ordinea* în care sunt lansate în execuție: $\min_\pi f(\pi)$ unde $\pi = \begin{pmatrix} 1 & 2 & \cdots & n \\ \pi(1) & \pi(2) & \cdots & \pi(n) \end{pmatrix}$;

6) *Codificarea* $\alpha / \beta / \gamma$ va avea, pentru problemele de ordonanțare pe o maşină, următoarea alcătuire:

- $\alpha = 1$;
- indicatorul β va avea trei câmpuri $\beta \equiv$ (— , — , —). In primul câmp se trec datele opționale $r_j, d_j, \bar{d}_j, p_{ij}$ luate în considerare. In al doilea câmp se specifică existența interconditionărilor prin simbolul $\prec$. In ultimul câmp se indică posibilitatea întreruperii execuției unui job prin simbolul P.
- indicatorul γ specifică funcția de optimizat.

7) Se observă uşor că prin combinarea tuturor elementelor amintite mai sus se obține o mare varietate de situații de ordonanțare pe o maşină. Problemele asociate acestor situații se împart în:

- probleme *uşoare*: probleme pentru care se cunosc metode exacte de rezolvare ce determină soluția optimă în timp *polinomial*;
- probleme *grele*: probleme pentru care metodele exacte de rezolvare cunoscute nu sunt polinomiale. Pentru rezolvarea aproximativă a acestor probleme se utilizează diferite *euristici*.
- probleme *nedecise*.

Câteva probleme de ordonanțare "uşoare" ,împreună cu metodele adecvate de rezolvare, vor fi prezentate în continuare.

1) $\mathbf{1 / d_j / L_{max}}$ In această problemă se cunosc *termenele impuse* şi se cere ordonarea reperelor astfel încât să se minimizeze *întârzierea maximă.*

Soluția optimă se obține aplicând regula : *are prioritate jobul cu termenul impus cel mai mic*.

Exemplul 5.2.1 Pentru situaţia cu datele:

Reper	1	2	3	4	5	6	7
Durata p_j	3	4	4	6	9	11	15
Termene impuse d_j	22	28	36	21	15	35	31

Tabelul 5.1

durata de execuţie a tuturor reperelor este 3 + 4 + 4 + 4 + 6 + 9 + 11 + 12 = 52 unităţi de timp indiferent de ordinea în care sunt lansate în prelucrare; ordinea în care se minimizează întârzierea maximă este $5 \to 4 \to 1 \to 2 \to 7 \to 6 \to 3$.

Reper j	5	4	1	2	7	6	3
Moment de terminare C_j	9	15	18	22	37	48	52
Intârziere faţă de termenul impus	-	-	-	-	6	13	16 = L_{max}

Tabelul 5.2

2) **1 / d_j , $\prec$ / L_{max}** Noua problemă diferă de precedenta prin existenţa unor *relaţii de precedenţă* între unele operaţii. Ea se rezolvă foarte simplu prin regula: *dintre reperele încă neprogramate şi fără succesori direcţi se programează "cel mai târziu" reperul cu cel mai mare termen impus.*

3) **1 / - / Σ w_jC_j** In această situaţie, se urmăreşte minimizarea *sumei ponderate a termenelor de terminare*. Reperele vor fi lansate în ordinea în care rapoartele p_j/w_j *formează un şir nedescrescător*.

Exemplul 5.2.2

Reper j	1	2	3	4
Durata p_j	15	28	6	10
Ponderea w_j	3	7	1	5
p_j/w_j	5	4	6	2

$\Rightarrow$ programul optim de lansare $4 \to 2 \to 1 \to 3$

Tabelul 5.3

4) **1 / $\bar{d}_j$ / $\sum C_j$** Faţă de prima problemă, acum se dau *termene de livrare* care, spre deosebire de termenele impuse, trebuiesc respectate. In plus, se urmăreşte minimizarea *sumei termenelor de terminare* ale operaţiilor. Următorul algoritm rezolvă problema indicând, dacă este cazul, incompatibilitatea ei.

Vom nota cu S mulţimea reperelor *neprogramate*.

Start $S = \{1,2,...,n\}$

Pasul 1. Se determină mulţimea: $C = \{k \in S \mid \bar{d}_k \geq \sum_{j \in S} p_j\}$ Dacă C este *vidă* **Stop.** Altminteri:

Pasul 2. Se selectează reperul k* $\in$ C cu *cea mai mare durată:* $p_{k*} = \max_{j \in C} p_j$ şi se programează *cel mai târziu*.

Pasul 3. Se face actualizarea S = S \ {k*} şi se revine la pasul 1.

Deoarece la fiecare iterație se programează un reper, algoritmul se opreşte după n - 1 iterații.

Exemplul 5.2.3 In tabelul 5.4 se dau următoarele date privitoare la opt repere.

Reper	1	2	3	4	5	6	7	8
Durata	10	7	4	8	5	2	9	5
Termene de livrare	35	50	61	59	15	55	19	28

Tabelul 5.4

(aşa cum s-a mai spus, termenele de livrare sunt raportate la momentul de referință 0)

Start S = {1,2,3,4,5,6,7,8}

Iterația 1.

Pasul 1. $\sum_{j \in S} p_j = 50 \Rightarrow$ C = {2,3,4,6};

Pasul 2. $p_{k*} = \max_{j \in C} p_j = 8 = p_4 \Rightarrow$ se programează la sfârşit reperul 4;

Pasul 3. S = {1,2,3,5,6,7,8}.

Iterația 2.

Pasul 1. $\sum_{j \in S} p_j = 50 - 8 = 42 \Rightarrow$ C = {2,3,6};

Pasul 2. $p_{k*} = \max_{j \in C} p_j = 7 = p_2 \Rightarrow$ se programează la sfârşit reperul 2;

Pasul 3. S = {1,3,5,6,7,8}.

Iterația 3.

Pasul 1. $\sum_{j \in S} p_j = 42 - 7 = 35 \Rightarrow$ C = {1,3,6};

Pasul 2. $p_{k*} = \max_{j \in C} p_j = 10 = p_1 \Rightarrow$ se programează la sfârşit reperul 1;

Pasul 3. S = {3,5,6,7,8}.

ş.a.m.d. După şapte iterații se obțin rezultatele sintetizate în următorul tabel:

Reper j	6	5	7	3	8	1	2	4
Durata p_j	2	5	9	4	5	10	7	8
Termen de începere	0	2	7	16	20	25	35	42
Termen de terminare C_j	2	7	16	20	25	35	42	50
Termen de livrare $\overline{d}_j$	55	15	19	61	28	35	50	59
ΣC_j	2	9	25	45	70	105	147	197

Tabelul 5.5

4)Alte exemple de probleme "uşoare" sunt $\mathbf{1 / d_j / \Sigma U_j}$ în care se minimizează *suma joburilor întârziate* sau $\mathbf{1 / d_j / \Sigma w_j C_j}$ în care se minimizează *suma ponderată a termenelor de terminare*.

Problema $\mathbf{1 / r_j , d_j / L_{max}}$ este recunoscută a fi o problemă "dificilă". Dacă se compară noua situaţie cu cea din problema "uşoară" $\mathbf{1 / d_j / L_{max}}$ se constată că "dificultatea" rezolvării se datorează prezenţei *termenelor de disponibilitate* r_j. Aici este interesant de menţionat faptul că dacă este permisă întreruperea operaţiilor, problemele $\mathbf{1 / r_j , d_j, P / L_{max}}$ sau $\mathbf{f_{max}}$ şi chiar $\mathbf{1 / r_j , d_j, \prec, P / L_{max}}$ sau $\mathbf{f_{max}}$ sunt "uşoare".

Deasemenea dificilă este şi problema $\mathbf{1 / d_j / \Sigma T_j}$ în care se minimizează *suma întârzierilor efective*.

Despre problema $\mathbf{1 / r_j, \prec, \bar{d}_j, P / \Sigma C_j}$ nu se poate spune, în prezent, dacă este uşoară sau dificilă.

5.3 Ordonanţarea în flux pe două maşini

Ordonanţarea în flux (**Flow shop**) este o problemă *grea* cu foarte puţine excepţii. Una dintre aceste excepţii este *ordonanţarea pe două maşini*. Problema este următoarea:

Un număr de repere 1,2,...,n sunt prelucrate fiecare întâi pe utilajul U_1 şi apoi pe utilajul U_2. Se cunosc duratele operaţiilor pe fiecare din cele două utilaje. Se cere determinarea ordinii de lansare în execuţie astfel încât *durata realizării tuturor reperelor să fie minimă*.

Pentru rezolvarea problemei se utilizează următorul algoritm, datorat lui **Johnston**.

Pasul 1. Se determină *minimul* duratelor tuturor operaţiilor.

- dacă acest minim reprezintă durate unei operaţii efectuate pe utilajul U_1 atunci reperul corespunzător se programează *la început*, bineînţeles după reperele anterior programate "la început";
- dacă minimul reprezintă durata unei operaţii efectuate pe utilajul U_2 reperul corespunzător se programează *la sfârşit*, dar înaintea celor deja programate "la sfârşit";

Pasul 2. Se elimină din listă reperul programat şi se reia pasul 1.

Observaţie In cazul în care minimul calculat în pasul 1 nu este unic, putem avea trei situaţii:

- minimul este egal cu durata unei operaţii pe primul utilaj dar şi cu durata unei operaţii pe al doilea utilaj. Reperul corespunzător operaţiei de pe *primul utilaj* se programează *la început* iar reperul corespunzător *celeilalte* operaţii, *la sfârşit.*
- minimul este egal cu duratele a două operaţii de pe primul utilaj. Se va plasa *mai în faţă* reperul a cărui operaţie *pe al doilea utilaj durează mai puţin*.
- minimul este egal cu duratele a două operaţii pe al doilea utilaj. Se va plasa *mai la sfârşit* reperul a cărui operaţie pe *primul* utilaj *durează mai puţin*.

Exemplul 5.2.4 Aplicând algoritmul descris problemei cu datele din tabelul 5.6

Reper j	1	2	3	4	5	6
Durata operaţiei pe utilajul U_1	4	8	3	6	7	5
Durata operaţiei pe utilajul U_2	6	3	7	2	8	4

Tabelul 5.6

se obține următorul program optim de lansare: 3 → 1 → 5 → 6 → 2 → 4. Programarea în timp a execuției rezultă din diagrama:

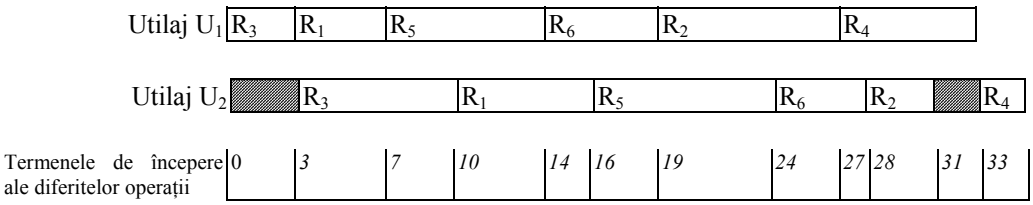


Figura 5.1

Timpul total de prelucrare al tuturor reperelor este de 35 de unități de timp.

Problema generală de ordonanțare **Job shop** este de departe cea mai complicată. Un număr de repere 1,2,...,m se prelucrează pe n utilaje 1,2,...,n fiecare reper parcurgând utilajele într-o *anumită ordine* dictată de tehnologia de fabricație. Se presupun cunoscute duratele operațiilor. Reamintim că un utilaj nu poate procesa la un moment dat decât un singur reper. Nu există restricții privind ordinea în care se execută operațiile de prelucrare ale diferitelor repere pe un acelaşi utilaj. Cum trebuiesc lansate reperele pentru ca timpul total de prelucrare să fie *minim*?

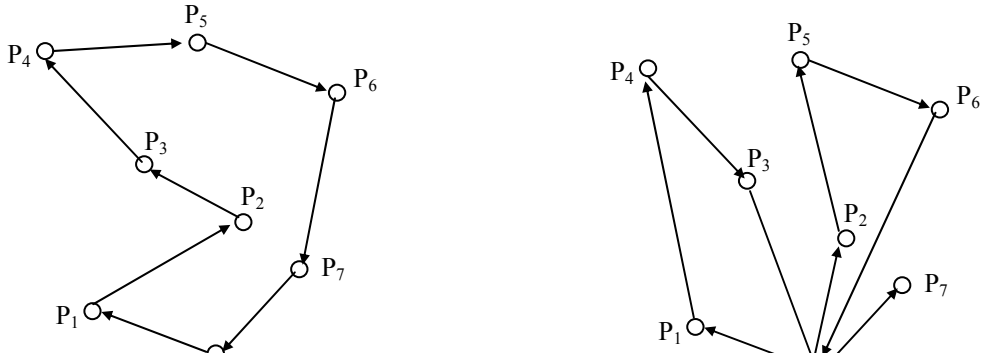
Spațiul limitat nu ne îngăduie să zăbovim asupra acestei probleme care – ea singură – poate constitui subiectul unei cărți de sine stătătoare!

§ 6. Problema stabilirii traseelor de transport

Într-o formă simplificată, problema stabilirii traseelor de transport (pe scurt problema STT) se enunță astfel:

O firmă urmează să livreze un produs mai multor clienți P_1 , P_2 ,..., P_n în cantitățile q_1 , q_2 ,...,q_n. Livrările se fac de la un depozit al firmei, notat P_0 , folosindu-se pentru aceasta un parc de mijloace de transport de diferite capacități.Fie T_1 ,T_2 ,...,T_m tipurile acestor vehicule diferențiate atât prin capacitățile lor c_1 , c_2 ,..., c_m cât şi prin numărul s_1 , s_2 ,..., s_m de unități disponibile pentru efectuarea transporturilor.Prin contract, cererea fiecărui client trebuie onorată la un singur transport. Fiecare vehicol inclus în programul de livrări pleacă de la depozitul P_0 încărcat sau nu la capacitatea maximă, vizitează succesiv un număr de clienți şi după ce se goleşte se întoarce la depozit pentru o eventuală nouă încărcare şi trimitere pe teren. În acest context, se pune problema de a stabili traseele de vizitare ale clienților precum şi tipul si numărul de mijloace de transport necesare satisfacerii tuturor cererilor astfel încât distanța totală parcursă să fie minimă.

Problema STT este una din cele mai complexe probleme de optimizare combinatorială. Ea este o generalizare a problemei comisvoiajorului; Intr-adevăr, dacă ar exista un singur mijloc de transport a cărui capacitate acoperă suma tuturor cererilor clienților atunci STT se reduce la determinarea ordinii în care clienții sunt vizitați astfel încât distanța totală parcursă să fie minimă (vezi fig. 6.1)



Au fost propuse un număr de metode exacte de rezolvare a problemei STT dar eficacitatea lor - pe probleme reale, de dimensiuni mari - este destul de redusă. Mult mai atractive s-au dovedit metodele euristice, capabile să ofere în timp util şi cu un efort de calcul rezonabil soluții acceptabile. Un exemplu de metodă de rezolvare aproximativă este procedura **Clarke - Wright** [ ], descrisă succint în continuare.

1) La început se consideră că fiecare client este aprovizionat direct de la depozit cu un mijloc de transport special detaşat să facă acest lucru. (vezi fig. 6.2)

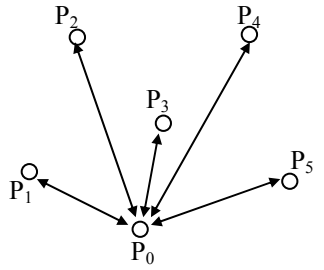


Figura 6.2

Vor fi atâtea trasee câți clienți există. Distanța totală de parcurs va fi:

$D = 2(d_1 + d_2 + ... + d_n)$

unde d_i este distanța de la depozitul P_0 la clientul P_i

2) În continuare se încearcă **concatenarea** acestor trasee în scopul reducerii distanței totale. Să examinăm fig 6.3a):

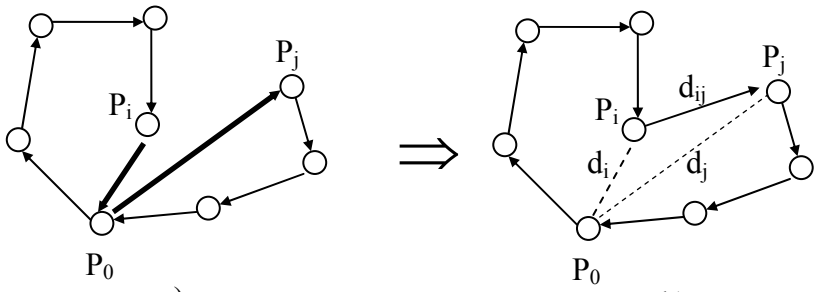


Figura 6.3

în care apar două trasee de lungimiD , D'. În fiecare traseu s-au pus în evidență doi clienți P_i , P_j **vecini** cu depozitul P_0 (relativ la un traseu dat un client se va numi vecin cu P_0 dacă el este fie primul fie ultimul client vizitat; doi clienți se vor zice vecini dacă în traseul dat ei sunt vizitați consecutiv) Concatenarea este ilustrată în fig. 5.3b). Din figură se vede că lungimea traseului rezultat este egală cu:

$$(D - d_i) + d_{ij} + (D' - d_j) = D + D' - (d_i + d_j - d_{ij}) = D + D' - s_{ij} \qquad (6.1)$$

unde:

$$s_{ij} = d_i + d_j - d_{ij}$$

d_{ij} fiind distanța dintre P_i şi P_j. Mărimile s_{ij} , $1 \le i \ne j \le n$ se numesc generic **reduceri**. Este evident că:

$$s_{ij} \ge 0 \text{ şi } s_{ij} = s_{ji}$$

Din (6.1) rezultă atunci că, prin concatenarea a două trasee distanța totală parcursă **se micşorează**.

3) Revenind la fig 6.3a) fie Q şi Q' totalurile cererilor clienților de pe cele două trasee. Cum fiecare traseu este deservit de un singur mijloc de transport este clar că pe cele două trasee sunt fixate două vehicole ale căror capacități de transport - eventual diferite - acoperă cantitățile Q şi Q'. Pe traseul din fig. 6.3b) va trebui transportată cantitatea Q + Q' şi aceasta de către un singur vehicol! În concluzie concatenarea celor două trasee este admisibilă numai dacă se dispune de un mijloc de transport a cărui capacitate acoperă sumă Q + Q'.

4) Deoarece la un moment dat pot exista mai multe concatenări admisibile, are prioritate aceea care produce cea mai mare reducere a distanței totale de parcurs.

5) Procesul se opreşte în momentul în care nu mai există concatenări admisibile.

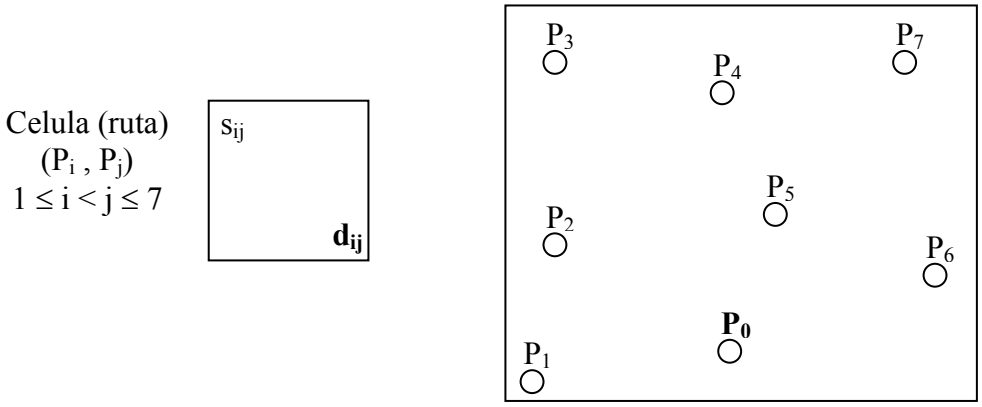


Figura 6.4

Exemplu 6.1 Se consideră rețeaua din fig. 6.4. Cantitățile comandate q_i , distanțele d_i de la depozitul P_0 la clienții P_i distanțele d_{ij} dintre clienți şi reducerile $s_{ij} = d_i + d_j - d_{ij}$ sunt indicate în tabelul 6.1. Firma are disponibile numai vehicole cu capacitatea de 6000 unități.

Client P_i	Cerere q_i	Distanța d_i:P_0 - P_i	Trasee		Cant. circulată pe un traseu	P_1	P_2	P_3	P_4	P_5	P_6	P_7
P_1	1100	316	1	← →	1100	*	316 **224**	247 **608**	88 **640**	1 **539**	0 **632**	50 **849**
P_2	1300	224	2	← →	1300		*	363 **400**	212 **424**	48 **400**	40 **500**	167 **640**
P_3	1200	539	3	← →	1200			*	635 **316**	197 **566**	215 **640**	622 **500**
P_4	1200	412	4	← →	1200				*	320 **316**	367 **361**	771 **224**
P_5	1900	224	5	← →	1900					*	440 **100**	395 **412**
P_6	1800	316	6	←	1800						*	499 **400**

				→								
P_7	1700	583	7	← →	1700							*

Tabelul 6.1

La start se presupune că fiecare client P_i este servit de un singur mijloc de transport care face traseul $P_0 \rightarrow P_i \rightarrow P_0$; vor fi deci şapte trasee însumând $2(d_1 + d_2 + ... + d_7) = 5228$ km şi implicând şapte vehicule (a căror capacitate este folosită parțial...)

Iterația 1 Cea mai mare reducere este $s_{47} = 771$. Concatenăm traseele $P_0 \rightarrow P_4 \rightarrow P_0$ şi $P_0 \rightarrow P_7 \rightarrow P_0$ în traseul $P_0 \rightarrow P_4 \rightarrow P_7 \rightarrow P_0$ obținând o reducere a distanței totale de 771 km. Pe acest traseu va circula un singur mijloc de transport cu încărcătura inițială de 1200 + 1700 = 2900 u. Rămân şase trasee însumând 5228 - 771 = 4457 km. Vezi tabelul 6.2

Iterația 2 Cea mai mare reducere admisibilă - pusă în evidență şi în tabelul6.2 - este $s_{34} = 635$. Concatenăm traseele $P_0 \rightarrow P_3 \rightarrow P_0$ şi $P_0 \rightarrow P_4 \rightarrow P_7 \rightarrow P_0$ în traseul $P_0 \rightarrow P_3 \rightarrow P_4 \rightarrow P_7 \rightarrow P_0$ pe care se va transporta 2900 + 1200 = 4100 u. Numărul de trasee (şi implicit de vehicule folosite) s-a redus la cinci însumând 4457 - 635 = 3822 km. Rezultă situația din tabelul 6.3

Cant. circulată pe un traseu	Clienți	Trasee	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1100	1	(1)→ ←							
1300	2	(2)→ ←							
1200	3	(3)→ ←				635			
2900	4	(4)→							771
1900	5	(5)→ ←							
1800	6	(6)→ ←							
-		7	←						

Tabelul 6.2

Cant. circulată pe un traseu	Clienți	Trasee	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1100	1	(1)→ ←							
1300	2	(2)→ ←							
4100	3	(3)→				635			
-	4								771
1900	5	(4)→ ←							
1800	6	(5)→ ←							499
-	7	←							

Tabelul 6.3

Iterația 3 Următoarea reducere (ca mărime) este s_{37} = 622 ea se exclude deoarece P_3 şi P_7 sunt deja pe acelaşi traseu. Reducerea care va fi luată în considerare este s_{67} = 499 (indicată şi în tabelul 6.3) Se vor concatena traseele $P_0 \rightarrow P_3 \rightarrow P_0$ şi $P_0 \rightarrow P_3 \rightarrow P_4 \rightarrow P_7 \rightarrow P_0$ în traseul $P_0 \rightarrow P_3 \rightarrow P_4 \rightarrow P_7 \rightarrow P_6 \rightarrow P_0$ dealungul căruia se va livra cantitatea 4100 + 1800 = 5900 u. (< 6000 u. ≡ capacitatea unui vehicul) Au rămas patru trasee însumând 3822 - 499 = 3323 km şi se utilizează patru mijloace de transport, unul fiind încărcat aproape la capacitatea maximă. Vezi tabelul 6.4

Iterația 4 Cea mai mare reducere admisibilă este acum s_{12} = 316. Se vor concatena traseele $P_0 \rightarrow P_1 \rightarrow P_0$ şi $P_0 \rightarrow P_2 \rightarrow P_0$ într-unul singur pe care se va transporta cantitatea 1100 + 1300 = 2400 u. Acum sunt numai trei trasee şi trei vehicule iar distanța totală s-a redus la 3323 - 316 = 3007 km. Noua situație este rezumată în tabelul 6.5

Iterația 5 Următoarea reducere admisibilă este s_{25} = 48. Concatenăm traseele $P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_0$ şi $P_0 \rightarrow P_5 \rightarrow P_0$ în traseul $P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_5 \rightarrow P_0$ pe care se va livra cantitatea 2400 + 1900 = 4300 u. Au rămas două trasee fiecare deservit de un vehicul. Distanța totală de parcurs va fi: 3007 - 48 = 2959 km. Este clar că cele două trasee numai pot fi concatenate din cauza capacității limitate de transport a vehiculelor. Vezi tabelul 6.6 Cele două trasee sunt vizualizate în fig.6.5

Cant. circulată pe un traseu	Clienți	Trasee	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1100	1	(1)→ ←		316					
1300	2	(2)→ ←							
5900	3	(3)→				635			
-	4								771
1900	5	(4)→ ←							
-	6	←							499
-	7								

Tabelul 6.4

Cant. circulată pe un traseu	Clienți	Trasee	P_1	P_2	P_3	P_4	P_5	P_6	P_7
2400	1	(1)→		316					
-	2	←					48		
5900	3	(3)→				635			
-	4								771
1900	5	(4)→ ←							
-	6	←							499
-	7								

Tabelul 6.5

Cant. circulată	Clienți	Trase	P_1	P_2	P_3	P_4	P_5	P_6	P_7

∞	10	7	9	11
8	∞	11	12	4
9	11	∞	13	6
6	12	14	∞	7
10	5	8	8	∞

Tabelul 7.1

pe un traseu	i	e							
4300	1	(1)→		316					
-	2						48		
5900	3	(3)→				635			
-	4								771
	5	←							
-	6	←							499
-	7								

Tabelul 6.6

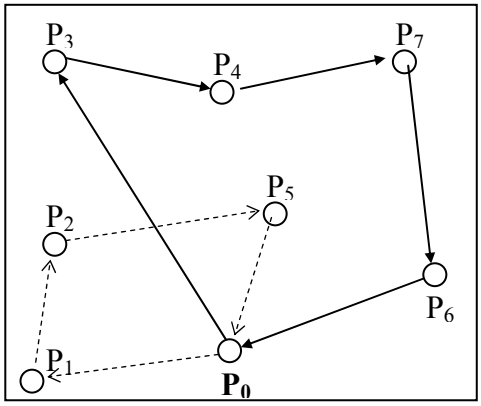


Figura 6.5

§ 7. Întrebări şi probleme

1.Ce este o problemă combinatorială? Dar o problemă de optimizare combinatorială? Dați un exemplu diferit de cele oferite în carte şi discutați-l cu colegii şi cu profesorul dvs.

2.Ce înseamnă că o problemă de optimizare este uşoră (grea)? Dați exemple din fiecare categorie.

3.Rezolvați problema din exemplul 3.2.

4. Un comisvoiajor situat în localitatea 0 are de vizitat oraşele 1,2,3,4. Costurile de deplasare dintr-un oraş în altul sunt date în tabelul 7.1 (Se observă că în general $c_{ij} \neq c_{ji}$, altfel spus costul deplasării între două localități depinde de sensul de deplasare - avem de a face cu o problemă asimetrică !)
Să se aplice algoritmul de tip Branch & Bound al lui Eastman
pentru a gă si un traseu de cost minim.

5. Într-o regiune a țării se afla şapte puncte de interes turistic. Poziţiile relative ale celor şapte puncte sunt date în fig. 7.1.iar distanțele dintre ele sunt date în tabelul alăturat 7.2.

O agenție de turism este interesată în

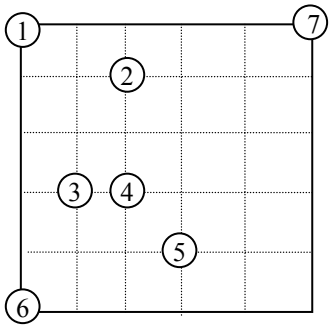


Figura 7.1

	1	2	3	4	5	6	7
1	*	22	32	36	50	50	50
2		*	22	20	36	44	36
3			*	10	22	22	50
4				*	14	28	42
5					*	32	45
6						*	70
7							*

Tabelul 7.2

determinarea celui mai scurt traseu de vizitare a celor şapte obiective.
Rezolvați problema utilizând euristica E_3 şi ajustarea locală E_2.

6.Un comisvoiajor pleacă din localitatea 0 şi trebuie să viziteze localitățile 1,2,...,6 (fiecare o singură dată) la sfârşitul călătoriei urmând a se reîntoarce în punctul de plecare 0 (vezi fig.7.2) În tabelul alăturat 7.3 se dau distanțele între localități.
a) Să se determine un traseu complet folosind euristica E_1: "mergi la cel mai apropiat vecin".
b) Eliminați eventualele încrucişări folosind euristica E_2 de ajustare locală. Ce lungime va avea noul traseu?
c) Determinați un arbore de lungime minimă care să conecteze cele şapte localități după care aplicați euristica E_3 pentru a obține un traseu complet.
d)Reluați punctul c) folosind de această dată euristica E_4.

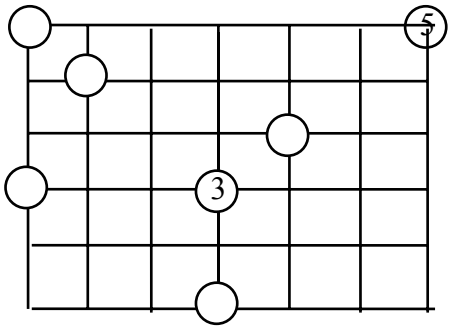


Figura 7.2

loc.	0	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
6							*

Tabelul 7.3

7.Firma X produce pâine pe care o desface către populație prin 12 magazine de specialitate. Fabrica de pâine şi centrele de desfacere sunt localizate în punctele $\mathbf{P_0}$, P_1 ,..., P_{12} indicate în figura 7.3. Pentru ziua următoare cererile sunt date întabelul 7.4:

Centre P_i	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	Total
Cerere (buc.) q_i	2200	300	800	1100	900	700	1400	1600	1600	1800	500	1500	14400

Tabelul 7.4

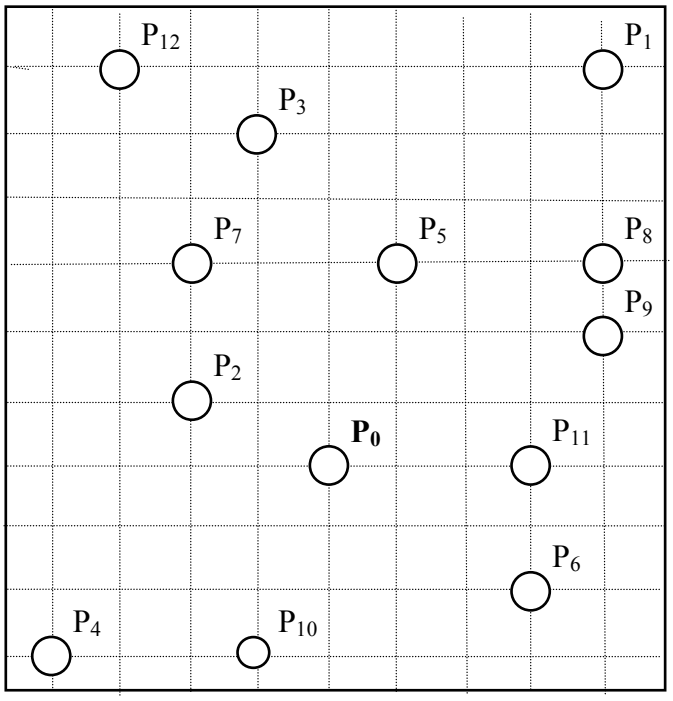


Figura 7.3

Transportul se face cu vehicule cu capacitatea de 4500 buc.
Distanțele d_i de la fabrica $\mathbf{P_0}$ la centrele P_i ca şi distanțele d_{ij} dintre centre de desfacere se vor calcula cu relația lui Pitagora ținând cont de faptul că unitatea de măsură a grilei din fig. 7.3 este de 200m. De exemplu distanța dintre P_1 şi P_7 va fi:

$$d_{17} = \sqrt{6^2 + 3^2} \cdot 200 = \sqrt{45} \cdot 200 = 6{,}7082 \cdot 200 = 1342m$$

a) Să se determine traseele de transport. cantitățile transportate, lungimile traseelor şi distanța totală de parcurs.

b)Care va fi soluția problemei în cazul în care firma are două vehicole cu capacitatea de 4500 buc. şi alte patru cu capacitatea de 2000 buc.?

8. Explicați termenii flow shop, open shop, job shop din problema ordonanțării.

9. Se pune problema lansării în execuție a opt repere 1,2,...,8 pentru care se cunosc:

- duratele p_1 , p_2 ,..., p_8;
- termenele de livrare $\bar{d}_1, \bar{d}_2, \ldots, \bar{d}_8$.(vezi tabelul 7.5)

În ce ordine trebuie lansate în execuție reperele astfel încât:

- termenele de livrare să nu fie depăşite: $C_j \le \bar{d}_j \quad j = 1,\ldots,8$
- suma termenelor de terminare să fie minimă: $\sum_{j=1}^{8} C_j \to min$

Cod reper: j	1	2	3	4	5	6	7	8
Durata : p_j	10	19	14	8	23	17	9	5
Termen de livrare: d_j	25	45	90	100	80	105	50	20

Tabelul 7.5

4. Zece repere se prelucrează în flux pe două utilaje: prima operație se execută pe utilajul A iar a doua pe utilajul B. În tabelul 7.6 se dau duratele de excuție ale celor două operații pentru fiecare reper în parte. În ce ordine vor fi lansate cele zece repere astfel încât durata totală de execuție să fie minimă?

Repere Utilaje	1	2	3	4	5	6	7	8	9	10
A	14	22	50	32	36	41	28	22	30	20
B	30	45	32	58	36	14	41	45	42	58

Tabelul 7.6

b) Aceeaşi chestiune cu datele:

Repere	1	2	3	4	5	6	7
A	11	6	8	15	9	14	13
B	8	11	8	10	12	6	9

Tabelul7.7