An algorithm for the Rural Postman Problem

Article · January 1981

CITATIONS
70

READS
1,199

4 authors:

Nicos Christofides
Imperial College London
104 PUBLICATIONS **7,857** CITATIONS
SEE PROFILE

Vicente Campos
University of Valencia
36 PUBLICATIONS **1,230** CITATIONS
SEE PROFILE

Angel Corberán
University of Valencia
90 PUBLICATIONS **1,728** CITATIONS
SEE PROFILE

E. Mota
University of Valencia
13 PUBLICATIONS **416** CITATIONS
SEE PROFILE

IC.OR.81.5

AN ALGORITHM FOR THE RURAL POSTMAN PROBLEM

MARCH 1981

NICOS CHRISTOFIDES *

VICENTE CAMPOS **

ANGEL CORBERAN **

ENRIQUE MOTA **

Department of Management Science,
* Imperial College, London. SW7 2BX, England
** University of Valencia. Spain.

ABSTRACT

The rural postman problem (RPP) is a general case of the Chinese postman problem where a subset of the set of links of a given graph is "required" to be traversed at minimum cost. If this subset does not form a connected graph, but forms a number of disconnected components the problem is NP-complete, and is also a generalization of the travelling salesman problem. In this paper we present a branch and bound algorithm for the exact solution of the RPP based on bounds computed from Lagrangean relaxation and on the fathoming of some of the tree nodes by the solution of 1-matching problems.

Computational results are given for graphs up to 84 vertices, 180 links, 74 "required" links.

1. INTRODUCTION

Consider a connected non directed graph G=(N,A), where N is a set of n vertices and A is a set of m links. Let $c_\ell \geq 0$ be the cost of link ℓ. The Chinese postman problem (CPP) is to find the shortest circuit such that each link is traversed at least once.

The rural postman problem(RPP) is a more general case.Given a subset of "required"links $A_R \subseteq A$ ($A_R \neq \phi$) the problem is to find a minimum cost circuit such that each link of A_R is traversed at least once.The cost of such a circuit is the sum of the costs of its links. If $A_R=A$ we have the CPP ,already solved by Edmonds and Johnson(1973).

The RPP is a combinatorial optimization problem with some real life applications, e.g. delivery of post, garbage collection, inspection of distributed systems (gas pipelines, electric power distribution networks). It is a generalization of the travelling salesman problem (TSP) in the sense that every TSP can be converted to an RPP as follows. Each city i of the TSP is replaced by a required link (i', i'')of the RPP of zero cost. Each link (i,j) of the TSP is replaced by a link (i', j'') in the RPP with the same cost but not required. The non required links in the solutionto the RPP then form a solution to the TSP.

If in the RPP the starting vertex of the " postman " is specified, a similar transformation to the one above for this initial vertex converts the problem to the general form defined earlier.

The purpose of this paper is to present an exact algorithm for this problem, with particular emphasis on its computational performance. The algorithm is based on the development of bounds derived from Lagrangean relaxations and the embedding of these into a tree search procedure. Computational results are given for graphs up to 84 vertices, 180 links and 74 required links.

2. PROBLEM TRANSFORMATIONS

Before formulating the problem we will transform the original graph G in order to simplify the problem structure and formulation.

Let $G_R = (N_R, A_R)$ be the graph induced by A_R, where N_R is the subset of vertices in N which are incident with at least one link of A_R. From G_R we construct the complete graph G'_C as follows: A link between every pair of vertices of N_R is added to G_R, with cost equal to the cost of the shortest path between them, (calculated in the original graph G). The resulting complete graph we call $G'_C = (N_R, A_R \cup A'_S)$, where A'_S is the set of the added links. Note that in G'_C there is an additional link in parallel with every link in A_R. Figure 1(a) shows the graph G where the set of links A_R is shown with solid lines. Figure 1(b) shows the resulting graph G'_C.

It is worthwhile to note here that a circuit in G'_C such that each link of A_R is traversed at least once produces a solution circuit of the RPP in G, after substituing all the shortest paths used for their corresponding links. Thus, any feasible solution of the RPP in G'_C produces a feasible solution in the original graph G and it is easy to show that every optimal solution of the RPP in G'_C produces an optimal solution of the RPP in G, with the same cost.

It is possible to simplify graph G'_C further and obtain a graph $G_C = (N_R, A_R \cup A_S)$ having the same RPP solutions by eliminating
(a) All links $(i,j) \in A'_S$ for which $c_{ij} = c_{ik} + c_{kj}$ for some k, and
(b) One of two links in parallel if they both have the same cost.

Figure 1(c) shows the graph G_C corresponding to graph G'_C in figure 1(b).

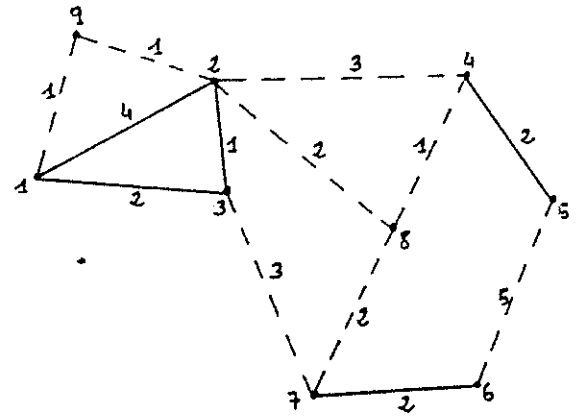


Figure 1.a

The original graph G=(N,A)

——— required link

– – – – non-required link

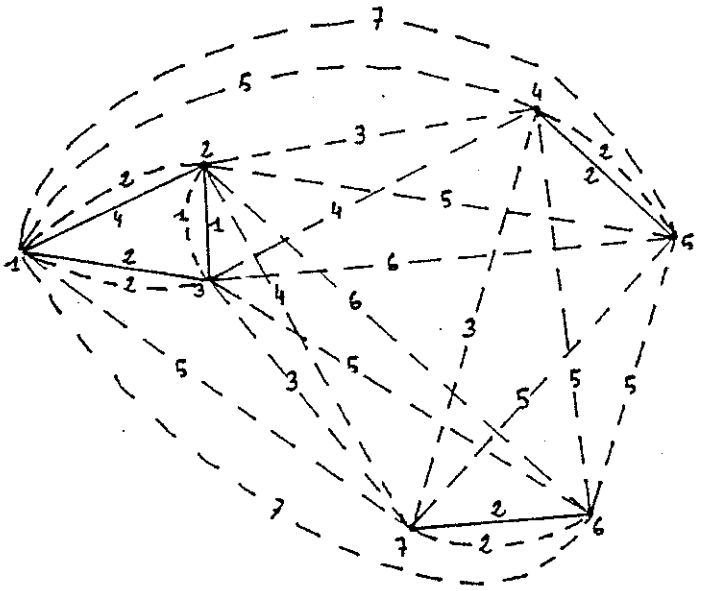


Figure 1.b

The graph $G_C' = (N_R, A_R \cup A_S')$

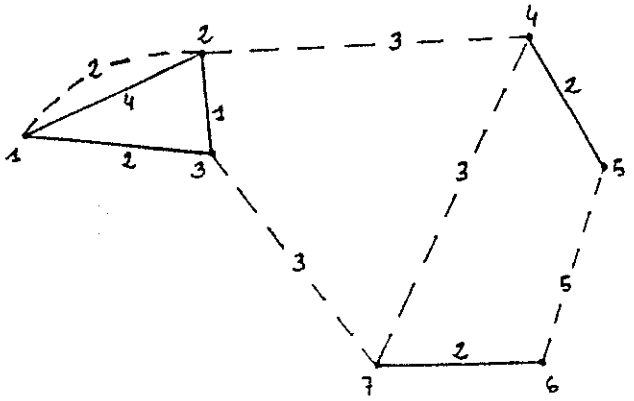


Figure 1.c

The simplified graph $G_C = (N_R, A_R \cup A_S)$

Figure 1. Graph transformations

3. FORMULATION

G_c is a connected non directed graph. Let $\{a_{i\ell}\}$ be the incidence matrix of the graph, $i \in N_R$, $\ell \in A_R \cup A_S$, i.e:

$$a_{i\ell} = \begin{cases} 1 & \text{if link } \ell \text{ is incident at vertex } i \\ 0 & \text{otherwise} \end{cases}$$

We define x_ℓ as the number of times that link $\ell \in A_R$ is repeated in an optimal circuit of the RPP. Then $1+x_\ell$ will be the number of times that link ℓ has been passed through in an optimal circuit. Let y_ℓ, $\ell \in A_S$, be the number of times that link $\ell \in A_S$ appears in an optimal circuit of the RPP.

The RPP can now be formulated as follows:

Let $C_1, C_2, \ldots, C_k$ be the connected components of the graph induced by A_R in G_C. We will use C_i to indicate the set of vertices of component i. The family of the C_i, $i=1,\ldots,k$ will be denoted by F. If $V \subset F$ is a subfamily of F we will use $N(V)$ to be the set of all vertices in the elements of V i.e. $N(V) = \bigcup_{C_i \in V} C_i$.

$$\text{Min} \sum_{\ell \in A_R} c_\ell x_\ell + \sum_{\ell \in A_S} c_\ell y_\ell + \sum_{\ell \in A_R} c_\ell \qquad (3.0)$$

$$\text{s.t:} \sum_{\ell \in A_R} a_{i\ell} (1+x_\ell) + \sum_{\ell \in A_S} a_{i\ell} y_\ell \equiv 0 \ (\text{mod.}2) \quad \forall i \in N_R \qquad (3.1)$$

$$\sum_{\ell \in K_t} y_\ell \geq 1 \qquad \forall K_t = \{(i,j) \in A_S \mid i \in N(V_t),\ j \in N(\overline{V_t}),\ V_t \subset F\} \qquad (3.2)$$

$$\begin{aligned} &x_\ell \geq 0, \text{ integer } \forall \ell \in A_R \\ &y_\ell \geq 0, \text{ integer } \forall \ell \in A_S \end{aligned} \qquad (3.3)$$

K_t will be referred to as a link cut-set of G_C. Constraints (3.1) ensure that in the solution the degree of every vertex is even. Constraints (3.2) assures that the optimal circuit will connect all the connected components.

We introduce new integer variables w_i , $i \in N_R$, in order to change the constraints (3.1) into the equations (3.1a) and (3.4):

$$\sum_{\ell \in A_R} a_{i\ell} (1+x_\ell) + \sum_{\ell \in A_S} a_{i\ell} y_\ell - 2w_i = 0 \quad \forall i \in N_R \tag{3.1a}$$

$$w_i \geq 0 \quad \text{integer} \quad \forall i \in N_R \tag{3.4}$$

The following propositions give bounds for the value of each variable of the problem in any optimal solution of the RPP.

PROPOSITION 1

The variables associated with an optimal solution of the RPP are bounded as follows:

$$\begin{aligned} &0 \leq x_\ell \leq 1 \quad \forall \ell \in A_R \quad \text{and integer} \\ &0 \leq y_\ell \leq 2 \quad \forall \ell \in A_S \quad \text{and integer} \end{aligned} \tag{3.3a}$$

Proof:

This is quite evident, because otherwise x_ℓ and y_ℓ could be reduced by two, and since $c_\ell \geq 0$, the value of the objective function would decrease or remain the same.

PROPOSITION 2

For the variables w_i , $i \in N_R$, we have:

$$d(i) \leq w_i \leq d_{G_C}(i) \quad , \; i \in N_R \text{ , integer} \tag{3.4a}$$

where:

$$d(i) = \begin{cases} \frac{1}{2} \; d_{G_R}(i) & \text{if vertex } i \text{ has even degree in } G_R . \\ \frac{1}{2} \; \{1+d_{G_R}(i)\} & \text{otherwise.} \end{cases}$$

$d_G(i)$ will be referred to, in general, as the degree of vertex i in G.

Proof:

Considering constraints (3.1a) and (3.3a) we have:

(a) $$w_i \geq \frac{1}{2} \sum_{\ell \in A_R} a_{i\ell} = \frac{1}{2} \; d_{G_R}(i)$$

However, considering that w_i is integer we have:

$$w_i \geq \frac{1}{2} \; d_{G_R}(i) \text{ , if } d_{G_R}(i) \text{ is even}$$

and $$w_i \geq 1 + \frac{1}{2} \; d_{G_R}(i), \text{ if } d_{G_R}(i) \text{ is odd.}$$

(b) $$w_i \leq \frac{1}{2} \left[\sum_{\ell \in A_R} 2a_{i\ell} + \sum_{\ell \in A_S} 2a_{i\ell} \right] = \sum_{\forall \ell} a_{i\ell} = d_{G_C}(i)$$

Thus, the RPP can be formulated as follows:

$$
P_R \left\{
\begin{array}{ll}
\text{Min} \sum_{\ell \in A_R} c_\ell x_\ell + \sum_{\ell \in A_S} c_\ell y_\ell + \sum_{\ell \in A_R} c_\ell & (3.0) \\
\text{s.t:} & \\
\sum_{\ell \in A_R} a_{i\ell} (1 + x_\ell) + \sum_{\ell \in A_S} a_{i\ell} y_\ell - 2 w_i = 0 \quad \forall i \in N_R & (3.1a) \\
\sum_{\ell \in K_t} y_\ell \geq 1 \quad \forall K_t = \{(i,j) \in A_S \mid i \in N(V_t),\ j \in N(\overline{V_t}),\ V_t \subset F\} & (3.2) \\
0 \leq x_\ell \leq 1 \qquad \forall \ell \in A_R\ ,\ \text{integer} & (3.3a) \\
0 \leq y_\ell \leq 2 \qquad \forall \ell \in A_S\ ,\ \text{integer} & \\
d(i) \leq w_i \leq d_{G_C}(i) \quad ,\ i \in N_R\ ,\ \text{integer} & (3.4a)
\end{array}
\right.
$$

4. AN ALGORITHM FOR RPP

We develop an exact solution procedure for the RPP based on branch and bound where, at each node of the tree lower bounds are computed by solving a related problem which is a relaxation of P_R. As for all branch and bound methods the quality of the computed bounds has a much greater influence on the effectiveness of the algorithm than any branching rules that may be used to generate the subproblems during the search. In view of this we will concentrate our attention on the problems of calculating bounds.

4.1 Bounds from the shortest spanning tree (SST)

We will develop this bound in much the same way as the corresponding bound was developed for the TSP by Held and Karp [4,5], and Christofides [2]. It was noted in the introduction, that the RPP is a generalization of the TSP and some additional considerations are necessary here.

Consider problem P_R and the relaxed problem $P_R(u)$ obtained from P_R by relaxing -in a Lagrangean fashion- constraints (3.1a) using multipliers u_i, $i \in N_R$. Problem $P_R(u)$ becomes:

$$P_R(u) \begin{cases} \text{Min} \sum_{\ell \in A_R} (c_\ell - u_{i_\ell} - u_{j_\ell})\, x_\ell + \sum_{\ell \in A_S} (c_\ell - u_{i_\ell} - u_{j_\ell})\, y_\ell \\ \qquad + \sum_{i \in N_R} u_i \Big(2w_i - \sum_{\ell \in A_R} a_{i\ell} \Big) + \sum_{\ell \in A_R} c_\ell \\ \text{s.t: constraints (3.2), (3.3a) and (3.4a)} \end{cases}$$

where i_ℓ and j_ℓ are the terminal vertices of link ℓ.

For a given set of multipliers u, the solution to $P_R(u)$ is:

Set: $x_\ell = 1$, if $c_\ell - u_{i_\ell} - u_{j_\ell} \leq 0$

$x_\ell = 0$, otherwise

Set: $w_i = d_{G_C}(i)$, if $u_i \leq 0$

$w_i = d(i)$, otherwise.

The problem in y_ℓ defined by the second term in the objective of $P_R(u)$, constraints (3.2) and the second of the constraints in (3.3a), is the problem of finding the SST in a "condensed" graph $\tilde{G}_C = (\tilde{N}, \tilde{A})$, where a vertex i of $\tilde{G}_C$ corresponds to a connected component C_i of G_R. A link (i,j) of $\tilde{G}_C$ exists iff there exists some link (i',j') in G_C with $i' \in C_i$ and $j' \in C_j$. The cost of link (i,j) in $\tilde{G}_C$ is :

$$\tilde{c}_{ij} = \min_{\substack{i' \in C_i \\ j' \in C_j}} \{ c_{i'j'} \}$$

Thus, the optimal values of y_ℓ for problem $P_R(u)$ can be obtained by solving the SST in the graph $\tilde{G}_C$; setting $y_\ell = 1$ for the links in the SST having positive costs $\tilde{c}_{ij}$, and setting $y_\ell = 2$ for all links having negative costs (whether in the SST or not).

Note that in the presence of constraints (3.1a) the 2 link-connectivity of the solution to P_R, is implied by the remaining constraints of P_R. However, in the relaxed problem $P_R(u)$ the 2-connectivity of the solution $\underline{y}$ to the graph $\tilde{G}_C$ is no longer guaranteed and the explicit constraints:

$$\sum_{\ell \in \tilde{K}_t} y_\ell \geq 2 \qquad \forall \tilde{K}_t = \{ (i,j) \in \tilde{A} \mid i \in S_t,\ j \in \bar{S}_t,\ S_t \subset \tilde{N} \} \qquad (3.5)$$

could be added to the constraints of $P_R(u)$, in order to increase the bound.

In particular, if a constraint t of type (3.5) which is violated, can be detected in the solution $\underline{y}$ to the SST problem defined on $\tilde{G}_C$, this constraint can be relaxed via a Lagrange multiplier λ_t and added to the objective of $P_R(u)$. In practice, it is worthwhile not to iterate in order to find the best values of the multipliers λ_t, but to use some a priori procedure for fixing λ_t. One such procedure is described by Balas and Christofides [1] for the TSP.

The procedure used to derive a bound is therefore:

(i) Find the optimum values of multipliers u so as to maximize the value $V(P_R(u))$-the optimum solution value of problem $P_R(u)$.

(ii) From the values of y_ℓ obtained in (i), identify cuts of type (3.5), compute multipliers for these cuts and append to the objective of $P_R(u)$.

Return to (i) if necessary.

A simple greedy heuristic was used to choose good values of the multipliers u at step (i). Subgradient optimization was also tried but produced only a very slight improvement at a much greater computational cost.

4.2 An upper bound

A heuristic was used to obtain an upper bound to the RPP solution for use in the branch and bound. The heuristic is similar to one suggested for the TSP .

Let T be the solution to the SST of graph $\tilde{G}_C$ and let T^+ be the set of links of G_C corresponding to the links T of $\tilde{G}_C$. Let N_R° be the set of vertices of G_C which have odd degree relative to the links $T^+ \cup A_R$. Let $< N_R^\circ >$ be the graph induced in G_C by the set N_R°. Solve the 1-matching problem on $< N_R^\circ >$. The union of the set of links M in this matching, the set T^+ and the set A_R form a feasible solution to RPP.

The above heuristic solution can be improved in many cases by using the following procedure. Let $(i,j) \in T^+$ and $(j,k) \in M$ be two links which when removed from the set of links $A_R \cup T^+ \cup M$ forming the heuristic solution, and replaced by the single link (i,k), the resulting graph induced by the set of links

$$A_R \cup T^+ \cup M \cup \{(i,k)\} - \{(i,j),(j,k)\}$$

is connected. This resulting graph is then a new feasible heuristic solution and if, in addition, $c_{ik} < c_{ij} + c_{jk}$ this solution is of lower cost.

The performance of this upper bound is given in Table 1 together with the other computational results. In 12 problems out of the total of 24 , this bound was optimal. On average the bound is within 1.1 % of the optimal solution.

4.3 The Tree Search

The branchings in the tree search are made on the links with terminal vertices in different components (these correspond to links of $\tilde{G}_C$). One branch fixes such a link to be in the solution and the other fixes it out of the solution.

Two branching strategies have been tested; the primary objective of both strategies is to choose a set of arcs S during the branching, so as to achieve a connected graph - induced by the set of arcs $A_R \cup S$ - as early as possible.

(i) In this strategy the link chosen for branching at each stage is the one corresponding to the largest cost link in the SST solution to $\tilde{G}_C$ at this stage.

(ii) In this strategy we try to achieve(considering the links in S only) degree two for every vertex of $\tilde{G}_C$. Amongst the possible candidate links, we then choose as a secondary criterion that link which when added to S also satifies constraints (3-1a) for as many vertices of G_C as possible.

Thus, branching strategy (i) is based first on achieving connectivity and then on link costs; whereas branching strategy (ii) is based first on achieving connectivity and then on satisfying as many degree requirements as possible. In computational tests strategy (ii) out-performed strategy (i) by a factor of 8 (as far as computing time is concerned) and the results shown later are given for strategy (ii).

At the stage when k-1 links are fixed by the tree search, the graph G_M containing these links and the required set of links A_R only, is now connected. Let N^O_M be the set of odd degree vertices in G_M and $<N^O_M>$ be the graph induced in G'_C by the set N^O_M. the tree node can now be fathomed completely by solving the 1-matching problem in $<N^O_M>$.

5. COMPUTATIONAL RESULTS

The algorithm described above has been tested on 24 randomly generated problems varying in size up to 84 vertices , 180 links 74 required links and 8 connected components. Link costs were randomly generated in the range (1-30). The algorithm was coded in FORTRAN and run on the UNIVAC 1100/60 computer. All computational times shown in the tables are UNIVAC 1100/60 seconds.

Table 1 shows the results obtained including the initial upper and lower bounds obtained at the root node of the tree and the optimal solution values.

The graph for an example of a RPP is shown in the Appendix, together with the search tree produced by the algorithm. The example corresponds to the 4th problem in Table 1.

TABLE 1. COMPUTATIONAL RESULTS

1	2	3	4	5	6	7	8	9	10	11
2	25	17	49	24	16	122	122	118	0,689	9
3	20	17	43	22	13	84	84	78	0,612	31
3	28	23	51	24	23	130	130	125	1,357	33
3	20	14	40	14	15	84	83	75	0,463	25
3	11	9	18	7	7	23	23	23	0,159	1
3	9	7	15	5	15	22	21	21	0,157	17
3	10	7	13	4	6	38	38	37	0,125	5
4	17	11	21	7	6	76	76	68	0,219	13
4	22	14	35	12	24	164	163	153	0,615	57
4	31	28	58	26	31	102	102	101	1,817	17
4	15	12	24	10	11	80	80	71	0,474	37
5	36	20	56	16	20	135	129	121	1,58	79
5	25	19	47	17	27	116	112	99	2,61	401
6	40	28	75	31	52	212	209	193	8,12	857
6	71	49	139	67	43	372	366	341	207,61	8923
6	84	50	180	74	108	632	621	592	300,71	14791
6	78	50	184	78	81	480	480	465	174,50	8537
7	24	24	49	20	26	107	102	92	5,14	837
7	72	50	125	63	35	400	398	373	28,24	1209
7	42	31	87	34	57	203	203	193	44,65	4293
7	50	33	73	29	27	280	263	246	4,88	179
7	53	41	110	55	70	411	405	388	82,09	5754
8	40	26	51	19	18	445	445	399	4,65	299
8	28	23	40	16	15	148	148	139	2,58	227

Column. 1. Number of connected components.
" 2. Total number of vertices in the original graph G.
" 3. Number of vertices of graph G_C.
" 4. Total number of links in the original graph G.
" 5. Number of required links.
" 6. Number of non required links in G_C.
" 7. Upper bound.
" 8. Optimal value of the RPP.
" 9. Lower bound.
" 10. Total solution times (computing time for graph transformation and time taken by the branch and bound).
" 11. Nodes of the branch and bound tree.

REFERENCES

(1) E.Balas and N.Christofides: "A restricted lagrangean approach to the travelling salesman problem", Math.Prog.,21,19, (1981).

(2) N.Christofides: "Bounds for the travelling salesman problem". Ops.Res.,20,1044. (1972).

(3) J.Edmonds and E.L.Johnson: "Matching, Euler tours and the chinese postman". Math.Prog.5,88. (1973).

(4) M.Held and R.Karp: "The travelling salesman problem and minimum spanning trees". Ops.Res.,18,1138. (1970).

(5) M.Held and R.Karp: "The travelling salesman problem and minimum spanning trees II". Math.Prog.,1,6. (1971).

APPENDIX

Test example 4:

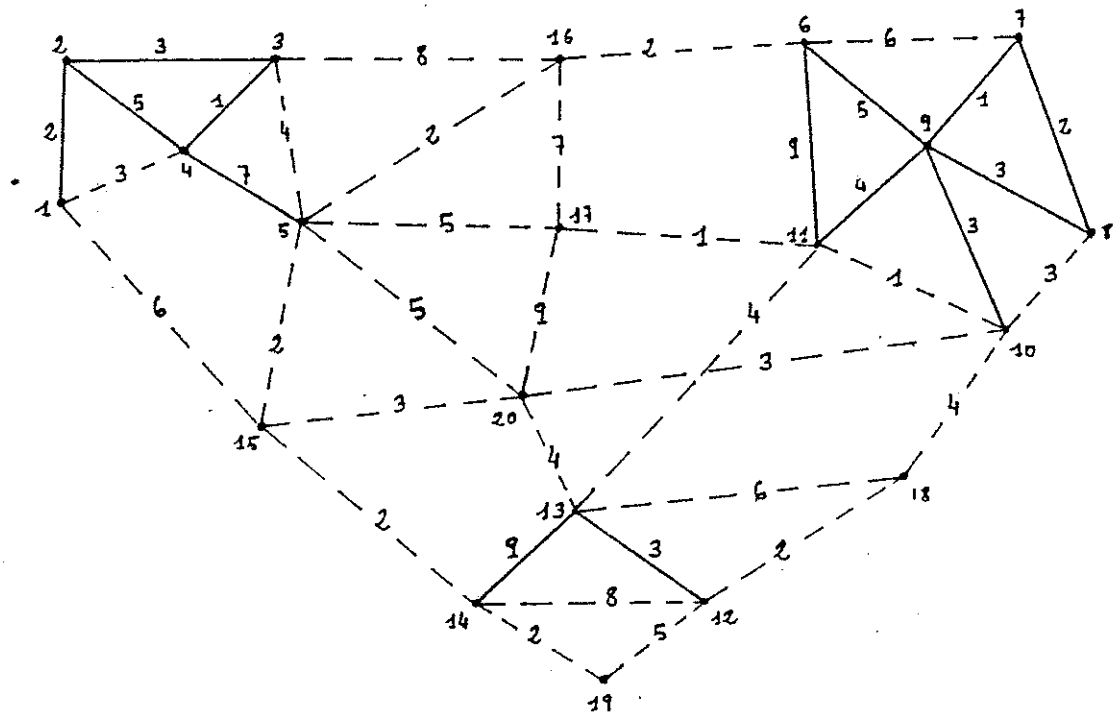


Figure 2. The original graph

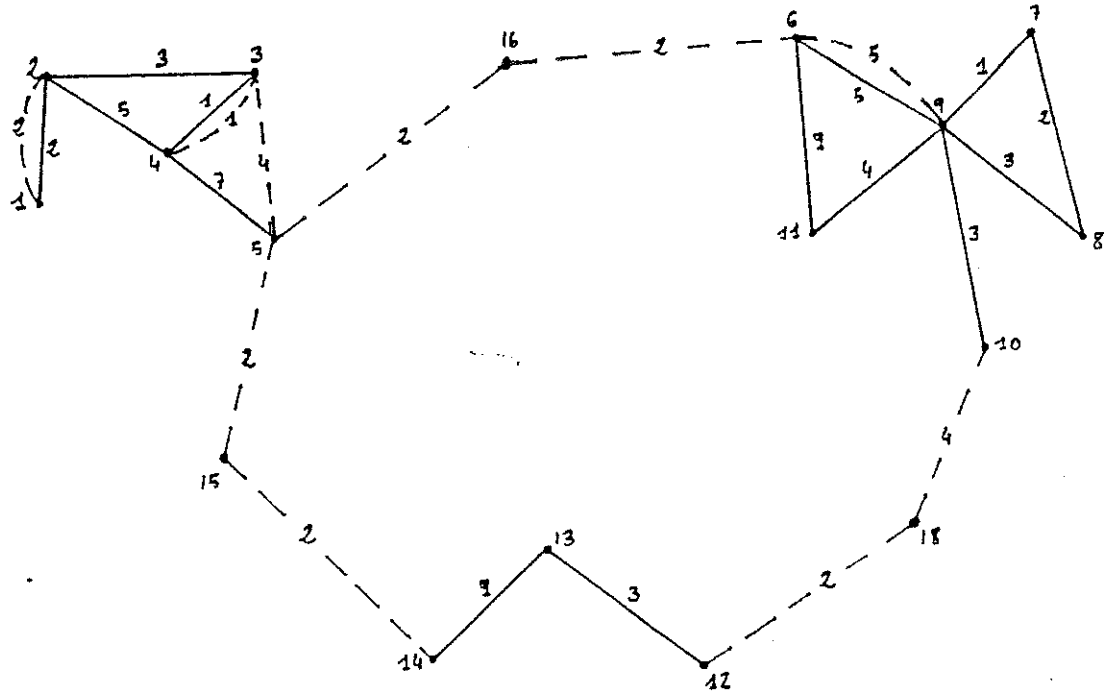
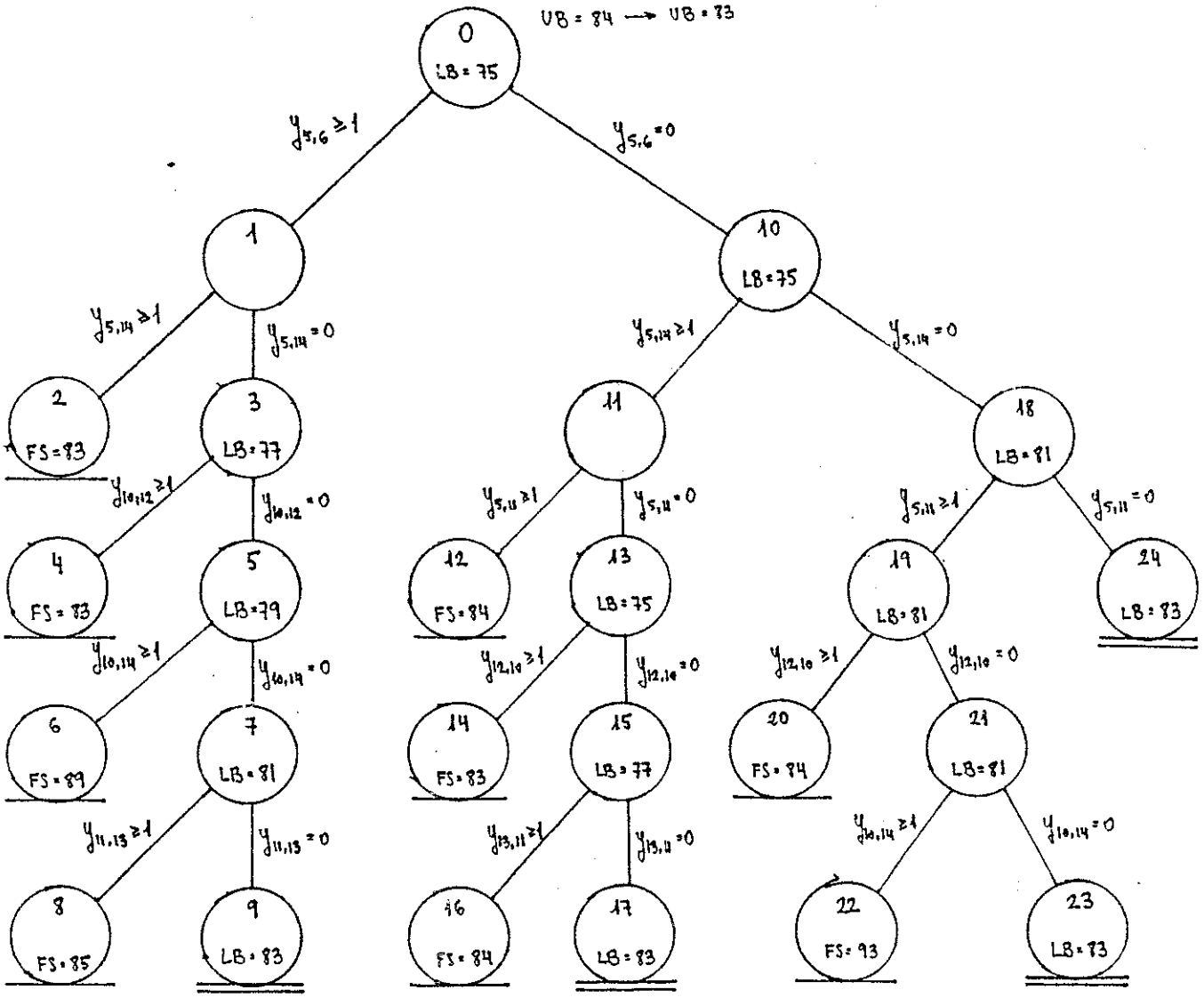


Figure 3. An optimal solution

Figure 4. THE BRANCH AND BOUND TREE OF THE EXAMPLE



LB : Lower Bound.

UB : Upper Bound.

FS : Feasible Solution to the RPP.

$y_{ij} \geq 1$: Shortest path from i to j is in the solution.

$= 0$: " " " " is not in the solution.

——— : node fathomed by the matching.

═══ : " " " " lower bound.