ALGORITHM CDT: A SUBROUTINE FOR THE EXACT SOLUTION OF LARGE-SCALE ASYMMETRIC TRAVELLING SALESMAN PROBLEMS

G. Carpaneto (+), M. Dell'Amico (†), P. Toth (‡)

(+) Dipartimento di Economia Politica, University of Modena, Italy.
(†) Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy.
(‡) D.E.I.S., University of Bologna, Italy.

Categories and Subject Descriptors: G.2.1 [**Discrete Mathematics**]: Combinatorics-*combinatorial algorithms*; G.2.2 [**Discrete Mathematics**]: Graph Theory-*graph – algorithms, path and circuit problems.*

General Terms : Algorithms

Additional Key Words and Phrases: Branch and Bound, Asymmetric Travelling Salesman Problem, Assignment Problem, Subtour Elimination, Reduction Procedure.

Abstract

The FORTRAN code CDT, implementing the algorithm of Carpaneto, Dell'Amico and Toth [3] for the *Asymmetric Travelling Salesman Problem*, is presented. The method is based on the *Assignment Problem relaxation* and on a *subtour elimination branching scheme.*
The effectiveness of the implementation derives from reduction procedures and parametric solution of the relaxed problems associated with the nodes of the branch-decision tree.

1.1. INTRODUCTION

Consider a complete *digraph* $G = (V, A)$ with vertex set $V = \{1, ..., n\}$, arc set $A = \{(i, j) : \ i \in V, \ j \in V\}$, and a cost $a_{i,j}$ associated with each arc $(i, j) \in A$ ($a_{i,i} = \infty \ \ \forall i \in V$). The *Asymmetric Travelling Salesman Problem* (ATSP) is to find a circuit visiting all vertices in V once (*circuit*) and such that the sum z^* of the costs of its arcs is minimum. Without loss of generatily, we will assume that costs are non-negative integers. The problem is known to be $NP - hard$. The code, implementing the lowest first branch and bound algorithm CDT presented in Carpaneto, Dell'Amico

and Toth [3], is based on the *Assignment Problem* (AP) relaxation and a *subtour elimination* branching scheme. At each node h of the decision tree algorithm CDT solves a *Modified Assignment Problem* (MAP_h), that is, an AP with additional constraints associated with arc subsets E_h and I_h, where:
$\{E_h = (i,j) \in A$: arc (i,j) is excluded from the optimal solution$\}$;
$\{I_h = (i,j) \in A$: arc (i,j) belongs to the optimal solution$\}$.
If the optimal solution to MAP_h does not define a Hamiltonian circuit and its value LB_h (giving the lower bound associated with node h) is less than the current optimal solution value, say UB, then descending nodes are generated from node h according to a subtour elimination branching scheme (see Carpaneto and Toth [2]) derived from that of Bellmore and Malone [1].

There are two kinds of nodes in the decision tree: *active* nodes (i.e. nodes not yet branched) and *passive* nodes (i.e. branched or fathomed). To store the information associated with the nodes, a vector V and two matrices MF and MV are used; vector V contains the *scalar information*, the matrices the *vectorial information*. The former is used to describe the decision tree structure, the vectorial information associated with node h is used for the parametric solution of the MAP's corresponding to nodes descending from h.

To increase the effectiveness of the implementation, a reduction procedure has been applied at the root node of the branch-decision tree so as to remove from G the arcs which cannot belong to an optimal solution. In this way the original digraph G can be transformed into a sparse one, say $\widetilde{G} = (V, \widetilde{A})$, allowing sparse cost matrix procedures to be used for the solution of the MAP's associated with the nodes of the branch-decision tree.

2. PROGRAM

The algorithm was coded in American National Standard FORTRAN as a main subroutine (CDT) calling 29 subroutines and 2 functions. The code has been tested on a CONVEX C120, on a SUN SPARC/2, on a VAX 6000/400, on a DECstation 5000/240, on a SGI Challenge and on a PC 486/33; moreover, it has been checked for portability using the PFORT Verifier [4].

The whole package is completely self-contained and communication with

it is achieved solely through the parameter list of CDT. The package can be invoked with the statement

```
CALL CDT (N, ORDX, X, MAXND, INF, ALPHA,
          ZSTAR, FSTAR, LB0, LBC, NEXP, NPROBQ NASS,
          ACTIVE, LOPT, SPARS, AVSON, ERR)
```

The input parameters are:

N = number of vertices (n);
X = working used array to store all the information needed for the branch and bound algorithm: in input it contains the original cost matrix, stored column by column, so as to store $a_{i,j}$ in the $k-th$ element of X, with $k = (j-1)\ n + i$;
ORDX = size of array X;
MAXND = maximum allowed number of $MAP's$ considered (set to -1 if no limitation is imposed);
INF = very large positive integer (with INF+ $max\{a_{i,j} \ : \ (i,j) \in A\}$ less than the maximum integer value representable in the computer);
ALPHA = parameter used to define an *artificial* upper bound UB:
if ALPHA > 0 then $UB = z(AP) \times$ALPHA
if ALPHA ≤ 0 an upper bound is computed;
ZEUR = value used to define a true upper bound, if ALPHA ≤ 0:
if ZEUR > 0 then CDT uses as upper bound the minimum between ZEUR and the value provided by the patching heuristic;
if ZEUR ≤ 0 then CDT uses the value given by the patching heuristic.

The value of ORDX must satisfy:

$$\text{ORDX} > n^2 + 21n + 2 + \mid \tilde{A} \mid .$$

The output values are:

ZEUR = upper bound value used by CDT;
ZSTAR = value of the optimal solution (z^* or value of the best solution so far);
FSTAR = solution vector corresponding to ZSTAR (FSTAR(i) $= j$ if arc (i,j) is in the optimal Hamiltonian circuit, $i = 1, ..., n$);

LB0 = value of the AP solved at the root node;
LBC = value of the highest lower-bound found so far, when algorithm stops; i.e. value of the lower bound associated with the last problem extracted from the queue. This is a valid lower bound for the original instance;
NEXP = number of explored nodes;
NPROBQ = number of problems stored in the queue;
NASS = number of completely solved MAP's;
ACTIVE = number of active nodes in the queue, when the program stops;
LOPT = level of the optimal solution in the branch-decision tree;
SPARS = percentage sparsity of the reduced matrix;
AVSON = average number of son nodes;
ERR = error condition; it can assume the following values:

-1: an error condition occurred and an explicative message was printed on logical unit 6
0 optimal solution found.

All the parameters are integer. After execution, all the input parameters are unchanged, except the first n^2 elements of X. The program needs no additional internal arrays, hence its global core memory requirements are ORDX+n elements.

When the program terminates, two situations may occurr: (i) the number of solved MAP's is less then the input parameter MAXND (or MAXND was set to -1); (ii) the number of solved MAP's is equal to MAXND. In case (i) two subcases must be distinguished. If the value ZSTAR is equal to the input value of ZEUR, then the upper bound ZEUR is optimal and the solution in vector FSTAR does not correspond to the optimal one. Otherwhise (ZSTAR less than the input value of ZEUR), the optimal solution is defined by vector FSTAR. In case (ii), if ZEUR is less than its input value and ALPHA ≤ 0, then vector FSTAR defines a heuristic solution for the instance, otherwise FSTAR has no meaning. In any case the value LBC is a valid lower bound for the instance.

Finally we note that the CPU time required to solve a single instance may be very large, so periodic printings have been introduced to monitor the correct running of the program. At the root node, when the first assignment problem has been solved and an upper bound has been determined, CDT prints the following message:

ROOT NODE: ZSTAR= xxx LB0= yyy

were xxx is the current best solution value (possibly artificial if ALPHA > 0), and yyy is the lower bound value at the root node. During the exploration of the branch decision tree, CDT prints the following message, every 1,000 nodes inserted in the queue:

ZSTAR= xxx LBC= yyy NPROBQ= zzz ACTIVE= www

were xxx is the current best solution value, yyy is the value of the lower bound associated with the last problem extracted from the queue, zzz is the number of nodes currently inserted in the queue and www is the number of active problems in the queue. The same printing is made when a new (better) solution is found, or value LBC is updated.

3. EXAMPLE

The subroutine CDT can be invoked by means of the following main program:

```
      PROGRAM MAIN
C
C     SAMPLE CALLING PROGRAM FOR CDT
C
      INTEGER ERR,ORDX,ZSTAR,ZEUR,ACTIVE
      INTEGER X(10000),FSTAR(100)
      ORDX = 10000
      INF = 99999999
      ALPHA= -1.
      MAXND= -1
      ZEUR = -1
C     READ N (WITH N .LE. 100) AND THE COST MATRIX
      OPEN(UNIT=1,FILE='INP.DAT',STATUS='OLD')
      READ(1,*) N
      CALL READA(N,X(1))
      CLOSE(1)
C
      CALL CDT(N,ORDX,X,MAXND,INF,ALPHA,ZEUR,ZSTAR,FSTAR,LBO,LBC,
     1         NEXP,NPROBQ,NASS,ACTIVE,LOPT,SPARS,AVSON,ERR)
C
      IF (ERR.EQ.0) GOTO 10
         WRITE(6,'('' SOLUTION NOT OPTIMAL '')')
 10   WRITE(6,'('' ZSTAR='',I8,'' LBO     ='',I8,'' LBC  ='',I8,
```

```
     1 '' SPARS ='',F8.4)')  ZSTAR,LB0,LBC,SPARS
       WRITE(6,'('' N.EXP='',I8,'' N.PROB.Q='',I8,'' N.ASS='',I8,
     1 '' ACTIVE='',I8,'' AV.SON='',F8.2)')NEXP,NPROBQ,NASS,ACTIVE,
     1 AVSON
       WRITE(6,'(20I4)') (FSTAR(I),I=1,N)
       STOP
       END
C
       SUBROUTINE READA(N,A)
       INTEGER A(N,N)
       DO 10 I=1,N
 10    READ(1,*) (A(I,J),J=1,N)
       RETURN
       END
```

The program reads all input data from logical unit 1 and write the output on logical unit 6. To read the problem costs easily in matricial form, instead of explicitly defining the entries of vector X, subroutine READIN was introduced.
A problem with 10 vertices was considered as an example; the following data define the instance.

```
  10
9999   964   786   990   345    63   386   999   361   126
 943  9999   961   706   800   488   482   198   743   190
 224   472  9999   326   695   362   420   193   203     0
 853   605   499  9999   963   781   179   370   531   289
  99   386   770   634  9999   420   295   487   355    36
 919   864   123   455   482  9999   156   585   350   812
 618   711   810    20   160   180  9999   129   897   245
 438   488   750    21   620   631   251  9999   233   156
  60   524   203   944   281   167    22   880  9999   734
 417   750   470   474    98   314   866   714   841  9999
```

The output of the program is:

```
ROOT NODE: ZSTAR=      1695 LB0=      1360
ZSTAR=    1695 LBC=    1452 NPROBQ=     3 ACTIVE=    1
ZSTAR=    1574 LBC=    1452 NPROBQ=     3 ACTIVE=    1
ZSTAR=    1574 LBC=    1461 NPROBQ=     3 ACTIVE=    0
ZSTAR=    1553 LBC=    1461 NPROBQ=     3 ACTIVE=    0
ZSTAR= 1553 LB0=    1360 LBC  = 1461 SPARS= 51.1111
N.EXP=    3 N.PROB.Q=  3 N.ASS=    5 ACTIVE=    0 AV.SON= 3.00
  6  10   9   2   1   3   8   4   7   5
```

The corresponding branch-decision tree is given in Fig. 1: the numbers inside the circles represent the lower bounds, those near the circles give the order in which subproblems were generated. The initial upper bound computed at the root node is 1695. The crossed nodes correspond to subproblems fathomed by the current upper bound. The solution of the assignment problem associated with node 8 is a Hamiltonian Circuit and corresponds to the optimal solution.

REFERENCES

1. Bellmore, M., Malone, J. C., "Pathology of Traveling Salesman Subtour Elimination Algorithms", **Op. Res.** 19, 1971, 278-307.

2. Carpaneto, G., Toth, P., "Some new Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem", **Management Science** 26, 1980, 736-743.

3. Carpaneto, G., Dell'Amico, M., Toth, P., "Exact Solution of Large-Scale Asymmetric Travelling Salesman Problems" Tech. Report, Dipartimento di Economia Politica, University of Modena, Italy, 1990.

4. Ryder, B.G., Hall, A.D., "The PFORT verifier", Bell Laboratories Computer Science report/2 Murray Hill, N.J., (May 1973-Jan. 1981).

.

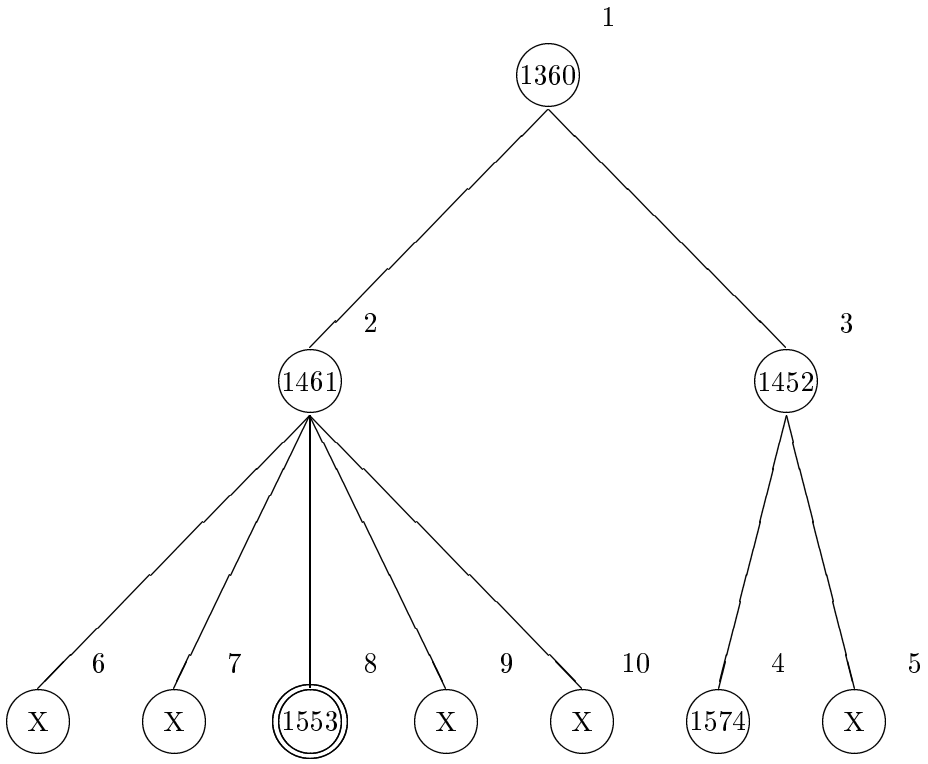

Figure 1. Branch-decision-tree of the example