ARAB REPUBLIC OF EGYPT

THE INSTITUTE OF NATIONAL PLANNING

Memo. No. (1575)

Solving Salesperson Problem with Artificial Intelligence Techniques

by

Dr. Abdalla A. El-Daoushy

March 1994

CAIRO
SALAH SALEM St-NASR CITY

SOLVING SALESPERSON PROBLEM WITH ARTIFICIAL INTELLIGENCE TECHNIQUES

BY

Dr. Abdalla A. El-Daoushy

CONTENTS

ABSTRACT AND PREFACE

Operations research (OR) problems have been solved by the well-known mathematical algorithms and the conventional programming languages such as FORTRAN, PASCAL, ...etc.

This memo. is concerned with solving some OR problems (specially the network problems) by using the artificial intelligence (AI) techniques such as expert systems (ESs) and the decalarative programming languages such as PROLOG* language.

The salesperson problem was taken as a classic network optimization problem to be solved with such AI techniques. A small ES was built for solving this problem. PROLOG language has been used as an ES tool since it is considered as an effective AI language. With minor modification to the suggested ES, the shortest route problem, the maximal flow problem, finding the spanning tree of a graph,...etc can be easily solved.

---

* PROLOG is a non-conventional programming language centered around a small set of basic mechanisms, including pattern matching, tree-based data strucuring, and automatic backtracking. This small set constitutes a surprisingly powerful and flexible programming framework. PROLOG is especially well suited for problems that involve objects ,in particular, structured objects and relations between them. For more details, refer to reference no. [1].

Chapter 1 introduces the traveling salesperson problem from operational research point of view. It also includes the FORTRAN program (stated in appendix) as a convensional approach to solve this problem.

Chapter 2 (the main chapter of this memo.) manipulates the same problem from artificial intelligence point of view. It introduces the the ES for solveng the problem. It also introduces a small expert system for solving the salesperson problem using TURBO PROLOG language. This chapter also refers to the solution of the salesperson problem by artificaial neural systems since researchers (in laboratories) showed that using artificial neural systems (ANS) enabled them to solve one salesperson problem on an ordinary micro-computer in 0.1 second compared to the optimal solution that required one hour of CPU time on a maineframe-computer using the conventional programming language [8].

Conclusion has been presented at the end of this memo.

CHAPTER 1

TRAVELING SALESPERSON PREOBLEM (OPERATIONS RESEARCH MANIPULATION)

PROBLEM DEFINITION :

Traveling salesperson's problem is considered as a formal model of many practical network optimiztion problems. The problem is defined by a network with N nodes (towns) and links (edges, arcs, or roads) between them. The task is to find a shortest route from some starting town, visiting all the other towns and ending in the starting town. No town, with the exception of the starting one, may appear on the route twice (an acyclic route).

PROBLEM FORMULATION :

Let c_{ij} = the cost (distnce) of travel from town i to town j.

$$X_{ij} = \begin{cases} 1 \text{ if the route uses a link from town i to town j.} \\ 0 \text{ otherwise.} \end{cases}$$

The total distance of a route become

$$Z = \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} \, X_{ij} \qquad (1)$$

In order for each town on a route to be visited only once, the salesperson must arrive at each town exactly once, and leave each town exactly once. This means that :

$$\sum_{j=1}^{N} X_{ij} = 1 \quad \forall \; i = 1,N \qquad \{\text{leaves each town exactly once}\} \qquad (2)$$

$$\sum_{i=1}^{N} X_{ij} = 1 \quad \forall \; j = 1,N \qquad \{\text{arrives each town exactly once}\} \qquad (3)$$

l), (2), and (3) constitute just an assignment problem, but due to
ıe circumstances of the salesperson's problem, this assignment
·oblem does not guarantee that the route will be connected.
ıerefore, extra constraints should be added to govern the
ısconnection as follows [9] :

$$u_i - u_j + N X_{ij} \leq N-1 \quad \forall\ i = 2,N,\ j = 2,N, \text{ and } i \neq j.$$

ıere

u_i represent N-1 auxiliary variables.

PROBLEM ALGORITHM WITH CONVENTIONAL COMPUTER PROGRAM (FORTRAN) [3] :

Although a great deal of research effort has been put forth in the past years to solve the general traveling salesperson problem, algorithms that find the exact optimal solution are still limitted to fewer than 100 towns. Algorithms that will find the exact solution for large problems are still very much in demand.

A number of branch-and-bound algorithms that find the exact optimal solution for small-to-moderate-size traveling salesperson problems (fewer than 50 towns) have appeared in literature during the last years; however, most ,if not all, are based on the Eastman algorithm. Eastman and Little et Al algorithms form the basis for all traveling salesperson problems.

The FORTRAN program as listed in the appendix solved the 10-town problem ,listed in example 2 at the end of this chapter, in 28 seconds on compatible IBM PC computer. However, computation times grow exponentially as the number of towns increases as we shall see in chapter 2. Generally, a 30-town problem can be solved in a reasonable amount of time on a mainframe computer, but larger problems get out of hand quickly [3].

The following illustrates the solution of two examples using this mentioned program.

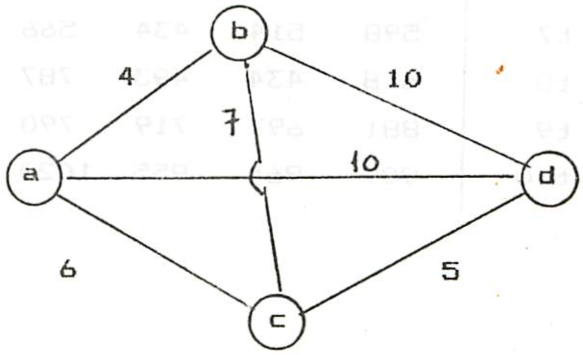
EXAMPLE 1 : A Four-towns problem . The input data and output results.

```
4
99999       4       6      10
  4 99999       7      10
    6     7 99999       5
  10    10       5 99999
```

To / From	1 a	2 b	3 c	4 d
1 a	-	4	6	10
2 b	4	-	7	10
3 c	6	7	-	5
4 d	10	10	5	-

```
THE DISTANCE MATRIX
-------------------

99999       4       6      10
   4   99999       7      10
     6       7   99999       5
   10       10       5   99999
```



```
THE OPTIMAL DISTANCE =       25.
-------------------------------------


THE ROUTE IS AS FOLLOWS :
-----------------------------


FROM TOWN   1 TO TOWN   2
FROM TOWN   2 TO TOWN   4
FROM TOWN   3 TO TOWN   1
FROM TOWN   4 TO TOWN   3
```

That is the shortest route is as follows :

a --> b --> d --> c --> a

EXAMPLE 2 : A Ten-towns problem. The input data and output results.

From \ To	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10
t1	99999	184	292	449	670	516	598	618	881	909
t2	184	99999	195	310	540	375	514	434	679	964
t3	292	195	99999	215	380	232	434	493	719	955
t4	499	310	215	99999	288	200	566	787	790	1020
t5	670	540	380	288	99999	211	436	814	632	974
t6	516	357	232	200	211	99999	381	642	697	952
t7	598	514	434	566	436	381	99999	295	224	541
t8	618	434	493	787	814	642	295	99999	320	341
t9	881	697	719	790	632	697	224	320	99999	318
t10	909	964	955	1020	974	952	541	341	318	99999

```
10
99999   184   292   449   670   516   598   618   881   909
  184 99999   195   310   540   375   514   434   679   964
  292   195 99999   215   380   232   434   493   719   955
  499   310   215 99999   288   200   566   787   790  1020
  670   540   380   288 99999   211   436   814   632   974
  516   357   232   200   211 99999   381   642   697   952
  598   514   434   566   436   381 99999   295   224   541
  618   434   493   787   814   642   295 99999   320   341
  881   697   719   790   632   697   224   320 99999   318
  909   964   955  1020   974   952   541   341   318 99999
```

```
THE DISTANCE MATRIX :
---------------------

99999    184    292    449    670    516    598    618    881    909
  184  99999    195    310    540    375    514    434    679    964
  292    195  99999    215    380    232    434    493    719    955
  499    310    215  99999    288    200    566    787    790   1020
  670    540    380    288  99999    211    436    814    632    974
  516    357    232    200    211  99999    381    642    697    952
  598    514    434    566    436    381  99999    295    224    541
  618    434    493    787    814    642    295  99999    320    341
  881    697    719    790    632    697    224    320  99999    318
  909    964    955   1020    974    952    541    341    318  99999

THE OPTIMAL DISTANCE =   2855.
-------------------------------


THE ROUTE IS AS FOLLOWS :
-------------------------


FROM TOWN  1 TO TOWN  2
FROM TOWN  2 TO TOWN  8
FROM TOWN  3 TO TOWN  1
FROM TOWN  4 TO TOWN  3
FROM TOWN  5 TO TOWN  6
FROM TOWN  6 TO TOWN  4
FROM TOWN  7 TO TOWN  5
FROM TOWN  8 TO TOWN 10
FROM TOWN  9 TO TOWN  7
FROM TOWN 10 TO TOWN  9
```

That is the route is as follows :

t1 -> t2 -> t8 -> t10 -> t9 -> t7 -> t5 -> t6 -> t4 -> t3 -> t1

CHAPTER 2

TRAVELING SALESPERSON PROBLEM (ARTIFICIAL INTELLIGENCE MANIPULATION)

PROBLEM DEFINITION AND PROBLEM ALGORITHM :

The analysis here will be quite different from that one mentioned in chapter 1. The problem is defined by a map with N towns and road distances between these towns. The task is to find a shortest route from starting town, visiting all towns on the map and ending in the starting town. No town ,except the starting one, may appear on the route more than once.

The salesperson here requires an expert system which will compute all thepossible routes between any two towns on the map, along with the shortest of these routes. Every route should comprises all of the towns in the map and every town should appear only once. That is every route is a hamiltonian route; that is an acyclic route comprising all of the towns on the map. This will enable him to :

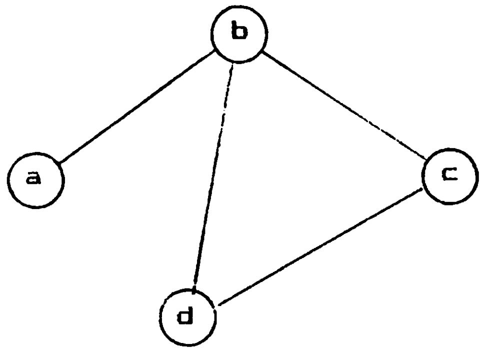
- plan routes which cover all towns on the map,
- choose routes which use particular roads with which he is familiar, and
- choose the shortest (i.e., cheapest) route.

GRAPH REPRESENTATION OF THE PROBLEM :

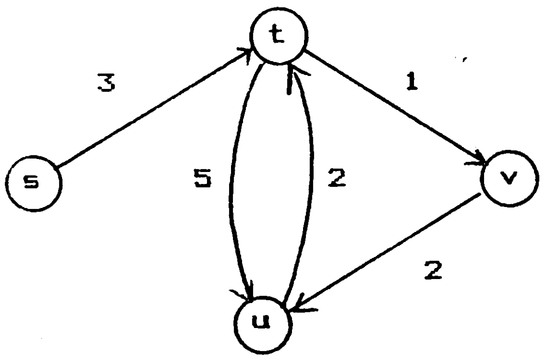
A graph is defined by a set of nodes and a set of edges, where each edge is a pair of nodes. When the edges are directed,they are also called arcs. Arcs are represented by ordered pairs. Such a graph is a directed graph. The edges can be attached costs, names, distances, or any kind of labels depending on the applications.

In PROLOG, graphs can be represented in several ways:

(1) Each edge or arc is represented separately as one clause. For example :



graph {a}



graph {b}

For graph a, the clauses are :

```
edge(a,b).
edge(b,c).
edge(c,d).
  .
  .
```

For graph b, the clauses are :

```
arc(s,t,3).
arc(t,v,1).
arc(u,t,2).
 .
 .
```

(2) A whole graph can be represented as one data object. In this case, a graph can be represented as a pair of 2 sets : nodes and edges (or only a set of edges). Each set can be represented as a list; each edge is a pair of nodes. For the graphs above, we have for example :

```
G1 = graph([edge(a,b),edge(b,c),edge(c,d),edge(b,d)]).
G2 = graph([arc(s,t,3),art(t,v,1),arc(t,u,5),arc(u,t,2),
                                                  arc(v,u,2)]).
```

There are other methods for representing the graph. What will be the most suitable representation will depend on the application and on operations to be performed on graphs.

Two typical operations on graphs are :

(1) Finding a route(s) between two given nodes (e.g., finding the maximal or minimal flow in a network, finding the shortest route for the traveling salesperson problem, etc.).
(2) Finding a subgraph ,with some specified properties, of a graph (e.g., finding a spanning tree of a graph).

FINDING A ROUTE IN A GRAPH :

Let G be a graph; A and Z are two nodes (towns) in G. Let us define the relation :

route(A,Z,G,P)

This relation means that: P is an acyclic route between A and Z in G. P is represented as a list of nodes on the route. For example, for graph {a} above, we have two routes between nodes a and d as follows:

```
route(a,d,G,[a,b,d]).
route(a,d,G,[a,b,c,d]).
```

Since a route must not contain any cycle, a node can appear in the route at most once.

To find a route(s), one method is as follows:

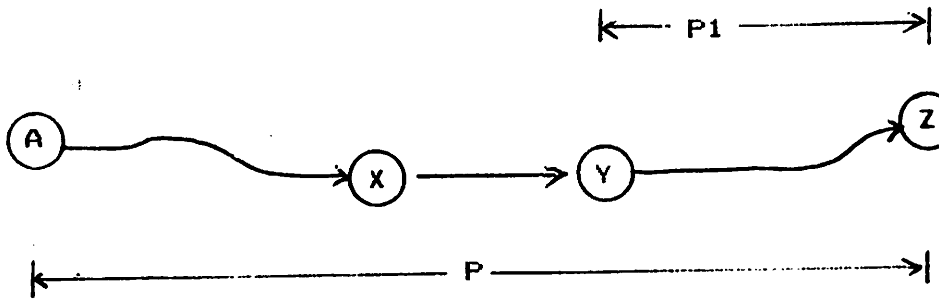
To find an acyclic route ,P, between A and Z in G, we have:

- If A = Z, then P = [A] and the relation becomes :

route(A,A,G,[A]).

otherwise,
- Find an acyclic route ,P1, from some node Y to Z and find a route from A to Y avoiding the nodes in P1. This formulation implies another relation ,route1, as follows

route1(A,P1,G,P)



Where,

P1 : is a route in G.
P : is an acyclic route in G that goes from A to the beginning of P1 and continues along P1 up to its end.

The relation between "route" and "route1" is :

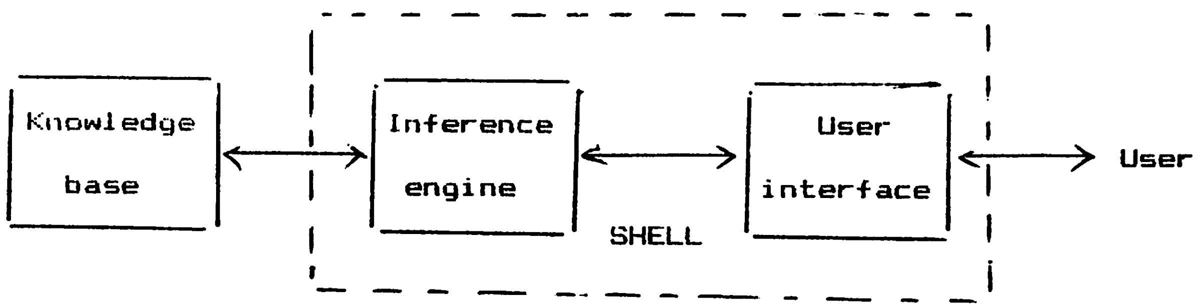
route(A,Z,G,P) :- route1(A,[Z],G,P).

A classical problem of finding a route in a graph is to find a hamiltonian route; that is, an acyclic route which comprises all nodes in the graph (the salesperson problem is an example of such a hamiltonian route(s)).

EXPERT SYSTEMS [*]

An expert system (ES) is an artificial intelligence program that behaves like an expert in some ,usually narrow, domain of application. ESs have to be capable of solving problems that require expert knowledge in a particular domain. They should possess that knowledge in some form. Therefore, they are also called knowledge-based systems. An ES also has to be capable ,in some way, of explaining its behavior and its decision to the user, as human experts usually do. Such an explanation feature is especially necessary in uncertain domains (such as medical diagnosis) in order to enhance the user's confidence in the system's advice, or to enable the user to detect a possible flow in the systems's reasoning. Therefore, ESs have to have a friendly user interaction capability that will make the system's reasoning transparent to the user. The main structure of an expert system is usually divided into 3 main modules :

1. a knowledge base,
2. an inference engine, and
3. a user interface.



A **knowledge base** comprises the knowledge that is specific to the domain of application, including such things as simple facts about the domain, rules that describe relations or phenomena in the domain, and possibly also methods, heuristics, and ideas for solving problems in this domain.

An **inference engine** knows how to actively use the knowledge in the knowledge base.

A **user interface** provides a smooth communication between the user and the system. It also provides the user with an insight into the problem-solving process carried out by the inference engine.

It is convenient to view the inference engine and the user interface as one module which is usually called an **ES SHELL**.

---

* For more details, see reference no. [2] for the same author

SMALL EXPERT SYSTEM (TURBO PROLOG) FOR SOLVING THE PROBLEM :

```
%
% .....Travelling SalesPerson Problem.....
%
DOMAINS
      l = SYMBOL*
      s = SYMBOL
    int = INTEGER
%
PREDICATES
   route(s,s,l,l,int)
   member(s,l)
   go(s,s,int)
   act_on(s)
   length(l,int)
   RUN
%
CLAUSES
%
RUN :-
   MAKEWINDOW(1,5,4,"SalesPerson Problem",0,0,25,80),
%
   WRITE("Enter the no. of towns on the map: "),
   READINT(N),
   WRITE("Enter the guessing shortest route: "),
   READINT(MinDist),
%
   WRITE("  Enter the name of starting town: "),
   READLN(Town1),
   WRITE("    Enter the name of ending town: "),
   READLN(Town2),nl,nl,
   WRITE("All possible shortest routes are as follows:"),nl,
   WRITE("--------------------------------------------"),nl,nl,
%
```

```
   route(Town1,Town2,[Town1],Way,Dist),
   Dist <= MinDist,
   length(Way,N),
%
% the predicate length():
% this predicate limits searching for only one route
% in a way that it includes all the possible towns
% but visited only once.
%
   WRITE(Way),
   WRITE("-->"),
   WRITE("The distance is :"), WRITE(Dist),nl,
   act_on(y).
%
%
act_on(y) :- nl,FAIL.
act_on(n).
%
%
route(Town,Town,_,[Town],0) :- !.
route(Town1,Town2,Covered,[Town1|Way],Dist) :-
      go(Town1,Town3,Dist1),
      NOT(member(Town3,Covered)),
      route(Town3,Town2,[Town3|Covered],Way,Dist2),
      Dist = Dist1 + Dist2.
%
member(X,[X|_]).
member(X,[_|Y]) :- member(X,Y).
%
length([],0).
length([_|T],N) :- length(T,N1),N=N1+1.
%
% end of program. .
%
```

NOTICE :

It should be noted that this is a simple prototype system and represents a very small map. The system could easily be modified and extended to cover large maps by adding the necessary go facts.

Moreover, it should be noted that this is a very inefficient way for finding the minimal or maximal routes. This method is unselectively investigates possible routes and is completely unsuitable for large maps because of its high time complexity.

<u>More sophisticated methods for finding optimal routes will be dealt with in another memo. which will include the following items :</u>

- Heuristic search principle (the best 1st search and its applications)

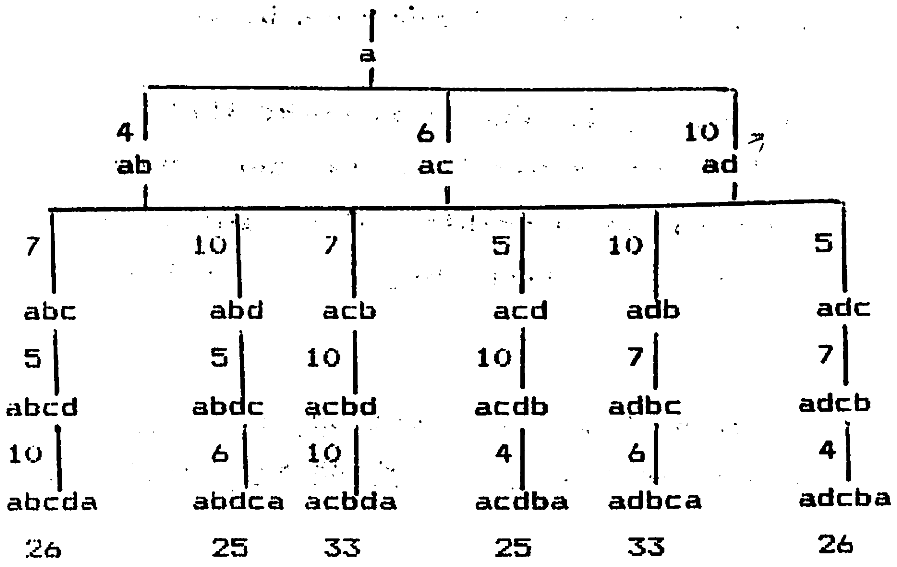
- Problem reduction and AND/OR graphs
  . AND/OR graph representation of problems.
  . Basic AND/OR search procedure.
  . Best 1st AND/OR search.

Anyhow, the following represents the solution of the same two examples solved in chapter 1 by using this small ES :

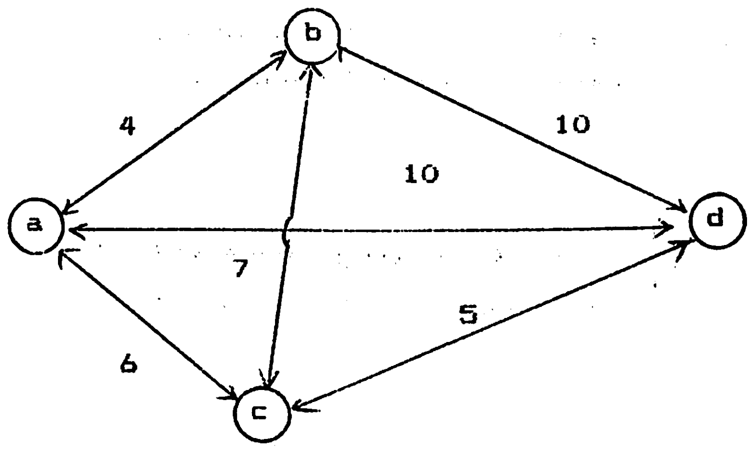
EXAMPLE 1 : Four towns problem.

The following shows the table of distances and the graphical representation of the problem :

To / From	a	b	c	d
a	–	4	6	10
b	4	–	7	10
c	6	7	–	5
d	10	10	5	–



```
%
go(a,b,4).
go(b,a,4).
go(a,c,6).
go(c,a,6).
go(a,d,10).
go(d,a,10).
go(b,c,7).
go(c,b,7).
go(b,d,10).
go(d,b,10).
go(c,d,5).
go(d,c,5).
```



The oputput results will look like the following :

```
                     SalesPerson Problem
Enter the no. of towns on the map: 4
Enter the guessing shortest route: 26
  Enter the name of starting town: a
    Enter the name of ending town: c

All possible shortest routes are as follows:
----------------------------------------------

["a";"b";"d";"c"]
---->The distance is :19
```

Notice that :

If we add the distance from c to a ,6, we get the shortest route with the same distance as in chapter 1 of the conventional algorithm.

EXAMPLE 2 : Ten towns problem.

Consider the same data of example 2 in chapter 1.

```
%
 go(t1,t2,184).
 go(t1,t3,292).
 go(t1,t4,449).
 go(t1,t5,670).
 go(t1,t6,516).
 go(t1,t7,598).
 go(t1,t8,618).
 go(t1,t9,881).
 go(t1,t10,909).
%
 go(t2,t1,184).
 go(t2,t3,195).
 go(t2,t4,310).
 go(t2,t5,540).
 go(t2,t6,357).
 go(t2,t7,514).
 go(t2,t8,434).
 go(t2,t9,679).
 go(t2,t10,964).
%
 go(t3,t1,292).
 go(t3,t2,195).
 go(t3,t4,215).
 go(t3,t5,380).
 go(t3,t6,232).
 go(t3,t7,434).
 go(t3,t8,493).
 go(t3,t9,719).
 go(t3,t10,955).
%
 go(t4,t1,449).
 go(t4,t2,310).
go(t4,t3,215).
go(t4,t5,288).
go(t4,t6,200).
go(t4,t7,566).
go(t4,t8,787).
go(t4,t9,790).
go(t4,t10,1020).
%
go(t5,t1,670).
go(t5,t2,540).
go(t5,t3,380).
go(t5,t4,288).
go(t5,t6,211).
go(t5,t7,436).
go(t5,t8,814).
go(t5,t9,632).
go(t5,t10,974).
%
go(t6,t1,516).
go(t6,t2,357).
go(t6,t3,232).
go(t6,t4,200).
go(t6,t5,211).
go(t6,t7,381).
go(t6,t8,642).
go(t6,t9,697).
go(t6,t10,952).
%
go(t7,t1,598).
go(t7,t2,514).
go(t7,t3,434).
go(t7,t4,566).
go(t7,t5,436).
go(t7,t6,381).
go(t7,t8,295).
go(t7,t9,224).
go(t7,t10,541).
```

```
%
 go(t8,t1,618).
 go(t8,t2,434).
 go(t8,t3,493).
 go(t8,t4,787).
 go(t8,t5,814).
 go(t8,t6,642).
 go(t8,t7,295).
 go(t8,t9,320).
 go(t8,t10,341).
%
 go(t9,t1,881).
 go(t9,t2,679).
 go(t9,t3,719).
 go(t9,t4,790).
 go(t9,t5,632).
 go(t9,t6,697).
 go(t9,t7,224).
 go(t9,t8,320).
 go(t9,t10,318).
%
 go(t10,t1,909).
 go(t10,t2,946).
 go(t10,t3,955).
 go(t10,t4,1020).
 go(t10,t5,974).
 go(t10,t6,952).
 go(t10,t7,541).
 go(t10,t8,341).
 go(t10,t9,318).
%
```

The output results will look like the following :

```
                              SalesPerson Problem
 Enter the no. of towns on the map: 10
 Enter the guessing shortest route: 2600
   Enter the name of starting town: t1
     Enter the name of ending town: t3


 All possible shortest routes are as follows:
 --------------------------------------------
 ["t1";"t2";"t8";"t10";"t9";"t7";"t5";"t6";"t4";"t3"]
 ---->The distance is :2563

 ["t1";"t2";"t8";"t10";"t9";"t7";"t6";"t5";"t4";"t3"]
 ----> The distance is :2596
```

Notice that :

If we add the distance from t3 to t1 ,292, we get the shortest route with the same distance as in chapter 1 of the conventional algorithm.

ARTIFICIAL NEUKAL SYSTEM FOR TRAVELING SALESPERSON PROBLEM [8] :

The solution of the salesperson problem is important because it is the classical problem faced in optimizing the routing of signals in a telecommunicatios system. Optimizing routing is important in minimizing the travel time, and thus effeciency and speed.

The following table shows the possible routes for 1 to 4 towns on a map :

No. of towns	Possible routes
1	1
2	1-2-1
3	1-2-3-1 1-3-2-1
4	1-2-3-4-1 1-2-4-3-1 1-3-2-4-1 1-3-4-2-1 1-4-2-3-1 1-4-3-2-1

Notice that the number of possible routes for a map of N towns is proportional to (N-1)!.

While there are 9!=362880 routes for 10 towns, there are 29!=8.8E30 possible routes for 30 towns. The traveling salesperson problem is a classic example of combinatorial explosion because the number of possible routes increases so rapidly that there are no practical solutions for realistic number of cities. If it takes 1 hour of a main frame CPU time to solve 30 towns, it will take 30 hours for 31 towns and 330 hours for 32 towns. These are actually very small numbers when compared to the thousands of telecommunication switches and towns that are used in routing of data packets and real items.

A new development in computer science arose in 1980's called artificial neural system (ANS), based on how the brain processes information. This science is sometimes called connectionism because it models solutions by training simulated neurons connected in a network.

ANS has had remarkable success in providing real-time response to complex pattern recognition problems. In one case, a neural net running on an ordinary micro-computer obtained a very good solution to the traveling salesperson problem in 0.1 seconds comared to the optimum solution that requires 1 hour of CPU time on a main frame computer.

A neural net can solve the 10-town case just as fast as the 30-town case while a conventional computer takes much larger. Although neural nets may not always give the optimum answer, they can provide a best guess in real time. In many cases, as 99.9999999% correct answer in 0.1 second is much better than a 100% correct answer in 30 hours [8].

CONCLUSION

The salesperson problem is solved in this memo. using two different approaches.

The results obtained from the two approaches show that the solution by the non-conventional approach is better than the one obtained from applying the conventional approach for many reasons :

1, The database in conventional programming is certain and static, while it is dynamic and may be uncertain for PROLOG programming.

2, The searching space for PROLOG programming is much small compared to the one in FORTRAN programming.

3, The number of towns can be very large compared to the limitted number of towns in conventional programming.

4, In future, it would be possible to use the ANS in real world applications in which case, the current ES inference mechanism will be replaced by the new ANS inference one --- which it seems as a science fiction film; however, stanger things have happend !.

APPENDIX

THE CONVENTIONAL COMPUTER PROGRAM (FORTRAN) [3] :

```
C     This program has been quated with minor modification from
C     Reference no. [3].
C
C     EASTMAN's Algorithm for Traveling Salesperson Problem
C
C      This program will solve the traveling salesperson problem when
C        15 or fewer cities are involved, by using the HUNGARIAN
C     ALGORITHM to solve the associated ASSIGNMENT problem.
C     This program is designed to read from a FILE called SP.DAT the :
C     No. of cities involved on the map.
C     The distance matrix of order N called IDIST(I,J) ; I,J =1,N.
C     The output results of this program will be :
C     CLUB : The distance of the optimal feasible solution.
C      IAS : The assignment matrix.
C
      DIMENSION IAS(15),IDIST(15,15),ISDIST(15,15),JCOMP(15),
     2BFSD(15),IIAS(500,15),BRANCH(500),R(500),C(500),P(500),
     3TD(500),NARCS(500),SUBT(500)
C
      OPEN(UNIT=1,FILE='SP.DAT')
      OPEN(UNIT=2,FILE='SP.RST')
C
   50 READ(1,*,END=2000)N
      DO 2 I=1,N
    2 READ(1,*)(IDIST(I,J),J=1,N)
C
      KODE=0
      WRITE(2,*)' THE DISTANCE MATRIX'
      WRITE(2,*)
      DO 707 I=1,N
  707 WRITE(2,702)(IDIST(I,J),J=1,N)
  702 FORMAT(1 [illegible]/)
C
C      STEP 1 :
C      Set minimum total distance (CLUB) to large +ve number.
C
       CLUB=0.1E11
C
C      STEP 2 :
C      Call subroutine HUNGRY to solve the associated ASSIGNMENT
C      problem by the HUNGARIAN method.
C        If at least one subtour exists, go to step 3. Otherwise, an
C         optimal solution of the ASSIGNMENT problem is the OPTIMAL
C      solution of the TRAVELING SALESPERSON problem, so STOP.
C
       CALL HUNGRY(N,IDIST,IAS,MD,KODE)
C
       NN=N-2
       DO 71 I=1,N
   71  JCOMP(I)=IAS(I)
       DO 70 II=2,NN
       DO 70 I=1,N
       J=JCOMP(I)
       JCOMP(I)=IAS(J)
       IF(JCOMP(I) .EQ. I) GO TO 80
   70  CONTINUE
       GO TO 200
C
C      STEP 3 :
C      A subtour exists of which we have the SHORTEST. Determine the
C      parameters of the SHORTEST subtour.
C
   80  IARS=II
       INT=I
       DO 701 KK=1,N
  701  IIAS(1,KK)=IAS(KK)
       K=1
C
```

```
C      STEPS 4,5
C      Branch into K subproblems
C      Solve the K subproblems as a new ASSIOGNMENT problems, and let
C      each solution distance be a LOWER BOUND for the corresponding
C      subproblem
C
       P(1)=1
       IPRED=1
       GO TO 110
  700  LL1=IPRED
C
  111  IF(P(LL1) .EQ. 1) GO TO 112
       LL2=R(LL1)
       LL3=C(LL1)
       ISDIST(LL2,LL3)=IDIST(LL2,LL3)
       IDIST(LL2,LL3)=99999
       LL1=P(LL1)
       GO TO 111
  112  LL2=R(LL1)
       LL3=C(LL1)
       ISDIST(LL2,LL3)=IDIST(LL2,LL3)
       IDIST(LL2,LL3)=99999
       IARS=NARCS(IPRED)
       INT=SUBT(IPRED)
  110  ITERM=INT
       DO 90 IJ=1,IARS
       IBEGIN=ITERM
       ITERM=IIAS(IPRED,IBEGIN)
       K=K+1
       R(K)=IBEGIN
       C(K)=ITERM
       ITEMP=IDIST(IBEGIN,ITERM)
       IDIST(IBEGIN,ITERM)=99999
C
       CALL HUNGRY(N,IDIST,IAS,MD,KODE)
C
       ID(K)=MD
```

```
      P(K)=IPRED
      IDIST(IBEGIN,ITERM)=ITEMP
      DO 125 MM=1,N
  125 IIAS(K,MM)=IAS(MM)
C
      DO 73 I=1,N
   73 JCOMP(I)=IAS(I)
      DO 75 II=2,NN
      DO 75 I=1,N
      J=JCOMP(I)
      JCOMP(I)=IAS(J)
      IF(JCOMP(I) .EQ. I)GO TO 131
   75 CONTINUE
C
C     STEP 6
C     If there exists one or more feasible solutions from STEP 5, and
C      if the smallest TOTAL distance for that feasible solution is
C      smaller than CLUB, st CLUB = STD and save the corresponding
C     feasible solution.
C
      IF(CLUB .LT. TD(K))GO TO 130
      CLUB=TD(K)
      DO 129 JJ=1,N
  129 BFSD(JJ)=IAS(JJ)
  130 BRANCH(K)=0
      GO TO 90
  131 BRANCH(K)=1
      SUBT(K)=I
      NARCS(K)=II
C
   90 CONTINUE
C
      IF(IPRED .EQ. 1)GO TO 160
      LL1=IPRED
  150 IF(P(LL1) .EQ. 1)GO TO 152
      LL2=R(LL1)
      LL3=C(LL1)
      LL1=P(LL1)
      IDIST(LL2,LL3)=ISDIST(LL2,LL3)
      GOTO 150
  152 LL2=R(LL1)
      LL3=C(LL1)
      IDIST(LL2,LL3)=ISDIST(LL2,LL3)
C
C     STEP 7
C     If CLUB is less than the LOWER BOUNDS on all of the unexplored
C     subproblems, the solution corresponding to CLUB is an OPTIMAL
C     soltion, so STOP. Otherwise, go to STEP 8.
C
  160 DO 132 I=2,K
      IF(BRANCH(I) .EQ. 0)GOTO 132
      IF(CLUB .GT. TD(I))GOTO 134
  132 CONTINUE
      GOTO 135
C
C     STEP 8
C     Select the unexplored subproblem with the smallest value.
C     Find the minimum distance for the unfeasible NODES.
C     Set the initial minimal NODEs to the value of the counter in the
C     previous loop, and return for the further branching.
C
  134 MINND=I
C
C
      DO 142 J=1,K
      IF(BRANCH(J) .EQ. 0)GOTO 142
      IF(TD(J) .LT. TD(MINND)) MINND=J
  142 CONTINUE
      BRANCH(MINND)=0
      IPRED=MINND
      GOTO 700
C
C     Output the final MINIMAL distance.
C     Output the ASSIGNMENT for the OPTIMAL solution.
```

```
C
  135 WRITE(2,222)CLUB
  222 FORMAT(//,' THE OPTIMAL DISTANCE = ',F7.0//)
      WRITE(2,*)' THE ROUTE IS AS FOLLOWS :'
      WRITE(2,*)
      DO 136 I=1,N
      JJJ=BFSD(I)
  136 WRITE(2,220)I,JJJ
  220 FORMAT(/,10H FROM TOWN,I3,8H TO TOWN,I3)
      GOTO 50
C
C     The solution to the ASSIGNMENT problem is the OPTIMAL solution
C     to the TRAVELING SALESPERSON problem.
C
  200 WRITE(2,*)
      WRITE(2,*)
      WRITE(2,*)' THE SOLUTION HAS NO SUBROUTES, HENCE THE OPTIMAL'
      WRITE(2,*)' SOLUTION OF THE ASSIGNMENT PROBLEM IS ALSO AN OPTIMAL'
      WRITE(2,*)' SOLUTIONOF THE SALESPERSON PROBLEM '

      WRITE(2,*)
      WRITE(2,*)
C
      WRITE(2,202)MD
  202 FORMAT(26H THE TOTAL MINIMUM ROUTE IS ,I7//)
      WRITE(2,*) 'THE ROUTE IS AS FOLLOWS'
      WRITE(2,*)
      DO 203 I=1,N
  203 WRITE(2,204)I,IAS(I)
  204 FORMAT(10H FROM TOWN,I3,7HTO TOWN,I3)
      GOTO 50
C
 2000 STOP
C
      END
C
C
```

```
C
      SUBROUTINE HUNGRY(N,IIF,IROW,TOTAL,KODE)
C
      DIMENSION IIF(15,15),IEFF(15,15),IROW(15),ICOL(15),ICHECK(15),
     2JCHECK(15)
C
      INTEGER TOTAL
C
C     STEP 1
C
C     CONSTRUCT THE EFFECTIVENESS MATRIX IF TOTAL EFFECTIVENESS IS TO
C     BE MAXIMIZED. LET THE EFFECTIVENESS MATRIX BE THE -VE OF THE
C     ORIGINAL MATRIX; OTHEWISE, THE EFFECTIVENESS MATRIX = THE ORIGINAL
C     MATRIX.
C
      IF(KODE .EQ. 1)GOTO 100
      DO 110 I=1,N
      DO 110 J=1,N
  110 IEFF(I,J)=IIF(I,J)
      GOTO 200
  100 DO 120 I=1,N
      DO 120 J=1,N
  120 IEFF(I,J) = - IIF(I,J)
C
C     STEP 2
C
C     FIND THE SMALLEST ELEMENT OF EACH ROW AND SUBTRACT IT FROM
C     OTHER ELEMENTS OF THE SAME ROW.
C
  200 DO 210 I=1,N
      MINROW=IEFF(I,1)
      DO 220 J=1,N
      IF(IEFF(I,J) .LE. MINROW) MINROW=IEFF(I,J)
  220 CONTINUE
      DO 210 J1=1,N
  210 IEFF(I,J1)=IEFF(I,J1)-MINROW
```

```
C
C     STEP 3
C
C     FIND THE SMALLEST ELEMENT OF EACH COUMN AND SUBTRACT IT FROM
C     OTHER ELEMENTS OF THE SAME COLUMN.
C
      DO 300 J=1,N
      MINCOL=IEFF(1,J)
      DO 310 I=1,N
      IF(IEFF(I,J) .LE. MINCOL) MINCOL=IEFF(I,J)
  310 CONTINUE
      DO 300 I1=1,N
  300 IEFF(I1,J)=IEFF(I1,J)-MINCOL
C
C     STEP 4
C
C     ROW ASSIGNMENTS . . .
C
C
  400 DO 410 I=1,N
      IROW(1)=0
  410 ICOL(I)=0
      NOMADE=0
  420 LOOP=0
      NZEROS=0
      DO 430 I=1,N
      NOZR=0
      IF(IROW(I) .NE. 0)GOTO 430
      DO 440 J=1,N
      IF(ICOL(J) .NE. 0)GOTO 440
      IF(IEFF(1,J) .NE. 0)GOTO 440
      NOZR=NOZR+1
      NZEROS=NZEROS+1
      NRPT=J
      NREFF=I
  440 CONTINUE
C
```

```
      IF(NOZR .NE. 1)GOTO 430
      IROW(1)=NRPT
      ICOL(NRPT)=1
      NOMADE=NOMADE+1
      LOOP=LOOP+1
  430 CONTINUE
C
C
C
C     STEP 5
C
C     COLUMNS ASSIGNMENTS . . .
C
      DO 500 I=1,N
      NOZC=0
      IF(ICOL(I) .NE. 0)GOTO 500
      DO 510 J=1,N
      IF(IROW(J) .NE. 0)GOTO 510
      IF(IEFF(J,I) .NE. 0)GOTO 510
      NOZC=NOZC+1
      NZEROS=NZEROS+1
      IRPT=J
  510 CONTINUE
      IF(NOZC .NE. 1)GOTO 500
      ICOL(I)=1
      IROW(IRPT)=I
      NOMADE=NOMADE+1
      LOOP=LOOP+1
  500 CONTINUE
C
C     STEP 6
C
C     CHECK RESULTS SO FAR. IF COMPLETE ASSIGNMENT HAS BEEN MADE,
C     GO TO STEP 15. IF AT LEAST ONE ASSIGNMENT HAS BEEN MADE, RETURN TO
C     STEP 4 TO MAKE ADDITIONAL ASSIGNMENTS.
C
      IF(NOMADE .EQ. N)GOTO 1500
      IF(LOOP .NE. 0)GOTO 420
      IF(NZEROS .EQ. 0)GOTO 800
C
C     STEP 7
C
C     PICK UP AN ARBITRARY ZERO AND RETURN TO STEP 4 . . .
C
      IROW(NREFF)=NRPT
      NOMADE=NOMADE+1
      ICOL(NRPT)=1
      GOTO 420
C
C     STEP 8
C
C     CHECK FOR UNASSIGNED ROW . . .
C
  800 DO 810 I=1,N
      ICHECK(I)=0
      JCHECK(I)=0
      IF(IROW(I) .NE. 0)GOTO 810
      ICHECK(I)=I
  810 CONTINUE
C
C     STEP 9
C
C     CHECK COLUMNS THAT HAVE ZEROS IN CHECKED ROWS . . .
C
  900 NCHECK=0
      DO 910 I=1,N
      IF(ICHECK(I) .EQ. 0)GOTO 910
      DO 920 J=1,N
      IF(JCHECK(J) .NE. 0)GOTO 920
      IF(IEFF(I,J) .NE. 0)GOTO 920
      JCHECK(J)=J
      NCHECK=NCHECK+1
  920 CONTINUE
  910 CONTINUE
```

```
C
C      STEPS 10,11,12
C
C      CHECK ROWS THAT HAVE ASSIGNMENT IN CHECKED COLUMNS.
C      REPEAT UNTIL CHAIN OF CHECKING IS COMPLETED.
C      ELIMINATE ALL UNCHECKED ROWS AND ALL CHECKED COLUMNS.
C
       IF(NCHECK .EQ. 0)GOTO 1300
       DO 1010 I=1,N
       IF(JCHECK(I) .LE. 0)GOTO 1010
       DO 1020 J=1,N
       IF(JCHECK(I) .NE. IROW(J))GOTO 1020
       ICHECK(J)=J
       NCHECK=NCHECK+1
 1020  CONTINUE
C
 1010  CONTINUE
C
       IF(NCHECK .NE. 0)GOTO 900
C
C      STEP 13
C
C      FIND THE MINIMUM UNMARKED ELEMENT . . .
C
 1300  MINELM=99999
       DO 1310 I=1,N
       IF(ICHECK(I) .EQ. 0)GOTO 1310
       DO 1320 J=1,N
       IF(JCHECK(J) .NE. 0)GOTO 1320
       IF(IEFF(I,J) .GE. MINELM)GOTO 1320
       MINELM=IEFF(I,J)
 1320  CONTINUE
 1310  CONTINUE
C
C      STEP 14
C
C      REDUCE MATRIX AND GO TO STEP 4 . . .
```

```
C
      DO 1400 I=1,N
      DO 1400 J=1,N
      IF(ICHECK(I) .LT. 0)GOTO 1400
      IF(ICHECK(I) .EQ. 0)GOTO 1410
      IF(JCHECK(J) .NE. 0)GOTO 1400
      IEFF(I,J)=IEFF(I,J)-MINELM
      GOTO 1400
 1410 IF(JCHECK(J) .LE. 0)GOTO 1400
      IEFF(I,J)=IEFF(I,J)+MINELM
 1400 CONTINUE
      GOTO 400
C
C     STEP 15
C
C     CALCULATE TOTAL AND RETURN TO MAIN PROGRAM . . .
C
 1500 TOTAL=0
      DO 1510 I=1,N
      K=IROW(I)
 1510 TOTAL=TOTAL+IIF(I,K)
      RETURN
      END
```

REFERENCES (alphbetically)

1, Abdalla El-Daoushy, (1991)
An Introduction to Artificial Intelligence through TURBO PROLOG, Memo. No. 898, Institute of National Planning, Cairo, EGYPT.

2, Abdalla El-Daoushy, (1993)
Expert Systems : An Overview, Memo. No. 1571 ,Institute of National Planning, Cairo, EGYPT.

3, Billy E. Gillett, (1979)
Introduction to Operations Research,
A computer-oriented algorithmic approach, TATA McGraw-Hill Publishing Company, New delhi.

4, Christopher John Hoggor, (1984)
Introduction to logic programming, Academic Press, Inc.

5, Claudia Marcus, (1986)
PROLOG Programming--Applications for dataBase Systems, Expert Systems, and Natural Language Systems, Addison-Wesley Publishing Company.

6, Ivan Bratko, (1991)
PROLOG--Programming for Artificial Intelligence, 2nd Edition, Addison-Wesley Publishing Company.

7, Jean B. Rogers, (1987)
A TURBO PROLOG Primer, Addison-Wesley Publishing Company, Inc.

8, Joseph Giarratano and Gary Riley, (1989)
Expert Systems, Principles and Programming, PWS-KENT Publishing Company, Boston.

9, Joseph Ecker and Michael Kupferschmid, (1988)
Introduction to Operations Research, John Wiley & Sons, New York.

10, Kelly Rich, Phillip R. Robison, (1988)
Using TURBO PROLOG, Borland Osborne-McGraw Hill.

11, Neil G. Rowe, (1988)
Artificial Intelligence through PROLOG, Prentice-Hall International, Inc.

12, Peter Smith, (1988)
Expert System Development in PROLOG and TURBO PROLOG, Sigma Press, UK, and Halsted Press, a division of John-Wiley & Sons.

13, W. F. Clocksin, C. S. Mellish, (1984)
Programming in PROLOG, 2nd Edition, Springer-Verlag.