P-Agents Seasonal Job Completion Model

Soma Sekhar Srinivas V.K, Suresh Babu.C, Madhu Mohan Reddy. P

Dept. of Mathematics, S. V. University, Tirupati, Andhra Pradesh, India.

Abstract- In this paper we study a **P-AGENTS SEASONAL JOB COMPLETION MODEL** problem with seasonal job completion model. There are N-cities (N=1, 2, 3….n) and J jobs (1, 2,…q). The distance between each pair of cities is mentioned and is denoted by $d_{ij.}$ For each city seasonal jobs have been mentioned for the agents whom they have to perform them in two seasons/periods. In the distance matrix the distance between each pair of the cities is **symmetric.** They have to do **m** jobs (truncated,i.e m<q) the agents starts from Head Quarter City and returns to it in the same path while completing all the **m** in both season jobs. The agents perform the first seasonal jobs at the cities when they move from Head Quarter City to the cities and they perform second seasonal jobs while return in the same path to the Head Quarter city. The aim of the problem is to find the paths of the agents such that the total distance travelled by them is minimum while completing all the **m** jobs. We consider a suitable numerical example and solved the problem by lexicographic search approach using pattern recognition technique.

Index Terms- Integer programming, Lexi-Search approach using Pattern recognition technique , Symmetric distance.

I. Introduction

The Travelling Salesman Problem is one of the most intensively studied problems in computational mathematics and is a kind of mathematical puzzle with a long enough history Crores, G.A[1].Many solution procedures have been developed such as Flood, M.M[3], Hardgrave, W.W & Nembanser, L.G[4] and Little, D.C.I.et al[7] for the travelling salesman problem.

Suppose a salesman wants to visit a certain number of cities allotted to him. He knows the distance/cost/time of journey between every pair of city 'i' and city 'j' denoted $C_{ij.}$ The problem is to select a rout that starts from a given home city to passes through each and every city once and only once and returns to his starting city in the shortest path. Here in the present study we have considered a variation of the above and called the salesman as agent.

II. Variations of Travelling Salesman Problem

There are many algorithms for usual one man TSP developed by researchers from time to time. But the problem has not received much attention in its restricted context. However, literature which is available with regard to the TSP with variations are discussed Das Shila [2,13], Jaillet P [5], Kubo & Kasugai [6], Pandit[8], Ramesh [9], Raviganesh G.S. Murthy & Das [11] and Srivastava Kumar, R.C. Garg and P. Sen [16].

Travelling purchaser problem (Ramesh T [9]) is another kind of variation to the TSP. Here there is of 'm' markets and a set of 'n' commodities. The cost of travel between each pair of markets and cost of each commodity at each market are known. A purchaser starts from one market and returns to it after purchasing all the commodities he needs. He may not visit all the markets, also may not purchase any commodity even if passes through a market. The purchaser is to find an optimal tour such that the total of the cost of travel and the cost of purchasing all the commodities is a minimum.

III. Travelling Salesman Problem with Multiple Jobs

Bansal and kumar (1971) considered the above problem and proposed a dynamic programming approach for its solution. Nizmudhin [13]has considered the above problem with 10 stations and 20 jobs and solved the problem by lexicographic search approach. Suresh Babu [15]did a problem related to the TSP with multiple job facilities and solved the model by using pattern recognition based lexicographic search approach. He stated, there is a set of stations containing N elements and a set of jobs containing M elements those are to be performed by the salesman. The distance between each pair of stations and facilities for jobs at each station are known. A salesman in his tour may or may not visit all the stations and should not visit a station more than once. The problem is to find a tour of the salesman such that the total distance travelled by him is minimum, while completing all the M jobs.

In this problem there N {1,2,3,….n} cities. '**P-Agents'** have to do **m(<n**) jobs. The distance between each pair of cities is mentioned and is denoted by $d_{ij.}$ The agents starts from a head quarter city he visits few cities and finish season 1 jobs while returns to head quarter city in the same path he completes season 2 jobs (total **m** jobs) in **P**- paths. The objective is to find minimum distance while completing all the **m** jobs in P- paths. In this sequel we developed a Lexi-algorithm using "Pattern Recognition Technique" to solve this problem which takes care of simple combinatorial structure of this problem.

IV. Lexicographic Search using Pattern Recognition Technique

Lexicographic Search Approach is a systematized Branch and Bound approach, developed by Pandit [8] in the context of solving of loading problem in 1962. In principle, it is essentially similar to the Branch and Bound method as adopted by Little et.al.[7]. This approach has been found to be productive in many of the Combinatorial Programming Problems. It is significance mentioning that Branch and Bound can be viewed as a particular case of Lexicographic Search approach [Pandit - 1965]. The

name Lexicographic Search itself suggests that, the search for an optimal solution is done in a systematic manner, just as one searches for the meaning of a word in a dictionary and it is derived from Lexicography the science of effective storage and retrieval of information. This approach is based on the following grounds [Pandit - 1963].

(i) It is possible to list all the solutions or related configurations in a structural hierarchy which also reflects a hierarchical ordering of the corresponding values of these configurations.

(ii) Effective bounds can be set to the values of the objective function, when structural combinatorial restraints are placed on the Allowable configurations.

The basic principle is described as follows [Rajbhongshi [10]]. Consider a set of symbols A= (1, 2, 3,……, n) and the different possible sequences of length k of these symbols. Thus (α_1, α_2, …, α_k) is a k-word, formed from the alphabet of n symbols 1, 2, 3 ….. , n. The i^{th} letter in this word is $\alpha_i \varepsilon$ A. By defining an alphabetic order on the elements of A, we will be able to define a unique ordered list of words of length not exceeding m, where m is finite. Words of length $k \leq m$ are called incomplete words standing for the set or block of the (m-k)! Words of length k. Searching for an optimum word is a problem of finding the word of minimum value (in the case of a minimizing problem) in the Lexi Search defined by the solution of the problem. The search efficiency of a Lexi Search algorithm is based in this approach depends on the choice of an appropriate Alphabet-Table, where two conflicting characteristics of the search list have to be taken into account: one is the difficulty in setting bounds to the values of the partial words (that defines partial solutions representing subsets of solutions). The other difficulty is in checking the feasibility of a partial word. Thus we get two situations in the choice of the alphabet-table [Sundara Murthy [14]]. In this problem we get the process of checking the feasibility of a partial word is easy, while the calculation of a lower bound is bulky.

When the process of feasibility checking of a partial word becomes difficult and the lower bound computation is easy, a modified Lexi Search i.e. Lexi Search with recognizing the Pattern of the Solution known as **Pattern Recognition Technique** can be adopted. In this method, in order to improve the efficiency of the algorithm, first the bounds are calculated and then the partial word, for which the value is less than the initial (trial) value are checked for the feasibility. The pattern-recognition technique can be described as follows.

"A unique pattern is associated with each solution of a problem. Partial pattern defines a partial solution. An alphabet-table is defined with the help of which the words, representing the pattern are listed in a Lexicographic order. During the search for an optimal word, when a partial word is considered, first bounds are calculated and then the partial words for which the value is less than the trail value are checked for the feasibility"

Using Pattern Recognition technique reduces the dimensions requirement of the problem. For this problem to find an optimal solution **X** which is a two dimensional array the problem can be reduced to a linear form of finding an optimal word of length n. This reduction in the dimension for some problems reduces the computational work in getting an optimal solution [Sundara Murthy [14], Vidyullata [17], Ramana and Umashankar [12]. The present paper uses the Lexicographic Search in general and makes use of the Pattern Recognition approach.

V. Problem Description

In this problem we have a variation of Travelling Salesman Problem called **"P-AGENTS SEASONAL JOB COMPLETION MODEL".** Let there be N (1,2,3…n) be cities and J (1,2,3…q) jobs in which **P-Agents** has to do while visiting the cities. The distance matrix d (i, j) is given in which distance is symmetric and also jobs at different cities are also given. The agent can do a job in each city of first season and second season. The restriction of the problem is the agents have to perform first season jobs at cities while moving away to the headquarter city to the cities and perform the second season jobs at the cities while return to the head quarter city in the same path .The total number of jobs by P-Agents must be **m (truncated** ,i.e.m<q) in 'P' paths. The problem is to find minimum total distance to finish **m** jobs by P-agents in P-paths subject to the above consideration. i.e. Where i, j Є N; N = {1,2, . . ., n}. Let D (i, j) be the distance from i^{th} city to j^{th} city which is symmetric. For this we develop an algorithm called as Lexi-Search algorithm based on the pattern recognition Technique and it is illustrated with a suitable numerical example for two agents.

VI. Mathematical Formulation

Minimize (Z) $X=\sum_{i\in N}\sum_{J\in N} D(i,j)X(i,j)$ ---------- **(1)**

Where N= (1, 2, 3…n)

Subject to the constraints

Let (1, $\alpha_{s1,}$ $\alpha_{s2\ldots}\alpha_{sns}$) cities in the s^{th} path---------- **(2)**

(Where s=1,2,….p)

If X (i, j) =1

JX (J (i, 1)) =1

JX (J (i, 2)) = 1

JX (J (j, 1)) =1 ---------- (3)

JX (J (j, 2)) =1

$\sum_{i=1}^{n} JX(i) \geq m$

X (i, j) = 0 or 1---------- (4)

Z(X) = 2α ---------- (5)

Equation (1) represents that the objective function of the problem, i.e. to find total minimum distance to connect all the cities from head quarter city. Equation (2) represents that the

total number of cities travelled by an **agent** in s^{th} path. Equation (3) represent that the total number of jobs done by the salesmen in two paths. Equation (4) describes that if a city *'i'* is connected to city j then X (i, j) = 1.Otherwise it will be equal to 0. Equation (5) represents the total distance travelled by the P-Agents is 2α (distance) , since the agent travelled is α (distance) while completing first season jobs and α (distance) in return while completing second season jobs(symmetric distance).

VII. Numerical Illustration

The concepts and the algorithm developed will be illustrated by a numerical example number of cities/stations, **n** = 7, number of total jobs q=10 ,number of jobs to be perform **m** =8.Let JB={1,2,3,4,5,6,7,8,9,10}. N = {1, 2, 3, 4, 5, 6, 7} Then the cost array D is given below.

TABLE -1

D=

∞	02	08	05	06	03	07
02	∞	04	07	10	11	01
08	04	∞	09	08	10	03
05	07	09	∞	02	01	06
06	10	08	02	∞	05	09
03	11	10	01	05	∞	04
07	01	03	06	09	04	∞

In the above numerical example given in Table – 1, D (i, j) = '∞'= D (i, i), where (i=1,2,...n & j=1,2…n) these distance pairs are not relevant in the solution paths finding tours of the agent. Though all the D (i, j)'s taken as non negative integers it can be easily seen that this is not a necessary condition and the cost can be any positive quantity. The distance between cities are symmetric, i.e. D (3, 4) = (4, 3) =9, means cost of connecting the city 3 to 4 is same as city 4 to 3 is 9. Then the information about the cities is given below.

TABLE-2

NJ

CITIES	**SEASON 1 JOBS(s_1)**	**SEASON 2 JOBS(s_2)**
1	----	-----
2	7	4
3	5	2
4	9	8
5	1	10
6	2	3
7	3	6

The above table – 2 represents the array of jobs assigned to the corresponding Cities of Season **1** and **Season 2** denoted by NJ(i ,s_b **)** , **(**where b=1,2). **T**he city **1** (headquarter city) has no jobs. i.e. for example NJ (2, s_1) =7, means city '2' has 7^{th} job in season one.

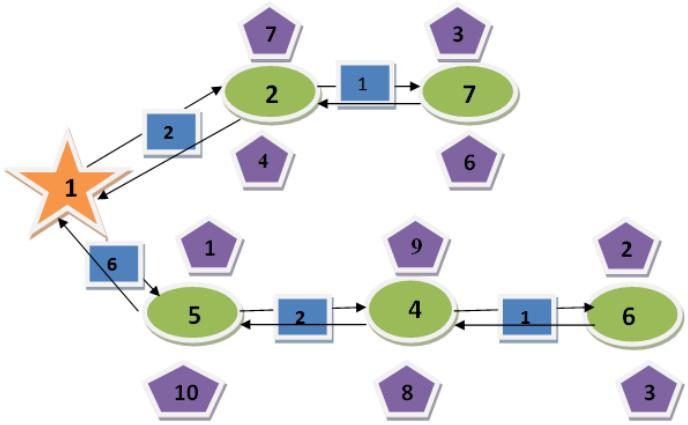
VIII. Concept and Definitions

8.1 Definition of a pattern: An indicator two-dimensional array which is associated the assignment is called a "pattern". A pattern is said to be feasible if X is solution $V(X) = \sum_{i \in N} \sum_{j \in N} D(i,j)\, X\,(i,j)$ .The value 2 V(X) gives the total distance travelled by them. In the algorithm, which is developed in the sequel a search is made for a feasible pattern with the least value, each pattern of the solution X is represented by the set of ordered pairs X(i, j)=1, which understanding that the other X(i, j) 's are zeros.

8.2 Feasible Solution:

In the following figure -1, the value in star represents the Head Quarter city, the values in the ellipses represents the cities allotted to above city, the value in the pentagon shape which are at top of the each city represents in first season particular job performed at that city while moving from Head Quarter city to the cities and value in pentagon shape at the bottom of the city represents in second season particular job performed by the agent at particular city while return to the Head Quarter city from that city, the value in the rectangle shape on the both (forward and backward) connected arrows represents the distance between the corresponding cities ,which is same in forward direction and in backward direction ,i.e. distance is symmetric.

Figure-1



In the above Figure-1, in first path the agent starts his trip at Head Quarter city{ 1} reaches to city 2 and performs the job'7'of season "1", from city 2 he visits the city 7 and performs the jobs '3'of season "1", while return in the same path he performs job '6' of season "2" in city 7, job '4'of season "2" in city 2 and reaches to the headquarter city{1}. In second path another agent visits the city 5 from city 1 and performs the job'1'of season "1", from city 5 he visits to city 4 and performs the job'9' of season "1", from city 4 he visits city 6 and performs the job '2' in season "1",while return in the same path he performs job'3' of season "2"in city 6, job '8' of season "2" in city4 ,job '10' of season "2" in city 5 and reaches to the Head Quarter city . Hence the solution is

Z = D (1, 2) + D (2, 7) + D (1, 5) + D (5, 4) + D (4, 6)
= 2+1+6+2+1 = 12.

Hence the above feasible solution of X pattern given in table-3, with the ordered pairs set {(1, 2), (2, 7), (1, 5), (5, 4) and (4, 6)} represents the feasible solution.

Table – 3

$$X(i,j) = \begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The above table -3 represents a feasible solution. The above solution X (1, 2) =1, represents that city 1 is connected to city 2. In similar way X (1, 5) represents city 1 is connected to city 5 and all the cities are connected to head quarter 1 either directly or indirectly.

IX. Alphabet Table

There are n x n ordered pairs in the two-dimensional array X. For convenience these are arranged in ascending order of their corresponding cost/distance and are indexed from 1,2…. (Sundara Murthy-1979). Let SN= [1, 2, 3…] be the set of indices. Let D be the corresponding array of cost. If a, b $\in$ SN and a < b then D (a) $\leq$ D (b). Also let the arrays R, C be the array of row and column indices of the ordered pair represented by SN and CD be the array of cumulative sum of the elements of D. The arrays SN, D, CD, R, C, for the numerical example are given in the table- 4. If p $\in$ SN then (R (p), C (p)) is the ordered pair and D (a) = D(R (a), C (a)) is the value of the ordered pair and $D(a) = \sum_{i=1}^{a} D(i)$

TABLE-4
Alphabet Table

SN	D	CD	R	C
01	1	1	4	6
			6	4
02	1	2	2	7
			7	1
03	2	4	1	2
			2	1
04	2	6	4	5
			5	4
05	3	9	1	6
			6	1
06	3	12	3	7
			7	3
07	4	16	2	3
			3	2
08	4	20	6	7
			7	6
09	5	25	1	4
			4	1
10	5	30	5	6
			6	5
11	6	36	1	5
			5	1
12	6	42	4	7
			7	4

13	7	49	2	4
			4	2
14	7	56	1	7
			7	1
15	8	64	1	3
			3	1
16	8	72	3	5
			5	3
17	9	81	3	4
			4	3
18	9	90	5	7
			7	5
19	10	100	2	5
			5	2
20	10	110	3	6
			6	3
21	11	121	2	6
			6	2

Let us consider 11 SN. It represents the ordered pair (R (11)), C (11)) = (1, 5) = (5,1). Then D (11) = 6 and CD (11) = 36.

X. Definition of an Alphabet - Table and a Word

Let SN = (1,2,…) be the set of indices, D be an array of corresponding costs of the ordered triples and CD be the array of cumulative sum of elements in D. Let arrays R, C be respectively, the row and column of the indices of ordered pairs. Let $L_k = \{a_1, a_2, - - - - - a_k\}$, $a_i \in$ SN be an ordered sequence of k indices of SN. The pattern represented by the ordered pair whose indices are given by L_k is independent of the order of a_i in the sequence. Hence for uniqueness the indices are arranged in the increasing order such that $a_i \leq a_{i+1}$, i = 1, 2, - - - -, k-1. The set SN is defined as the "Alphabet-Table" with alphabetic order as (1, 2, - - - -, n^2) and the ordered sequence L_k is defined as a "word" of length k. A word L_k is called a "sensible word". If $a_i < a_{i+1}$, for i =1, 2, - - - -, k-1 and if this condition is not met it is called an "insensible word". A word L_k is said to be feasible if the corresponding pattern X is feasible and same is with the case of infeasible and partial feasible pattern. A Partial word L_k is said to be feasible if the block of words represented by L_k has at least one feasible word or, equivalently the partial pattern represented by L_k should not have any inconsistency.

Any of the letters in SN can occupy the first place in the partial word L_k. Our interest is only in set of words of length almost equation, since the words of length greater than **n** are necessarily infeasible, as any feasible pattern can have only **n** unit entries in it. If k < n, L_k is called a partial word and if k = n, it is a full length word or simply a word. A partial word L_k represents, a block of words with L_k as a leader i.e. as its first k letters. A leader is said to be feasible, if the block of word, defined by it has at least one feasible word.

10.1 Value of the Word:

The value of the (partial) word L_k, V (L_k) is defined recursively as V (L_k) = V (L_{k-1}) + D (a_k) with V (L_k) = 0, where D (a_k) is the cost array arranged such that D (a_k) < D (a_{k+1}). V (L_k) and V(x) the values of the pattern X will be the same. Since X is the (partial) pattern represented by L_k (Sundara Murthy – 1979).

Consider the partial word L_4 = (01, 02, 03, 04 (1))
Then V (L_4) = 1+1+2+2 = 6.

10.2 Lower bound of a partial word LB (L_k):

A lower bound LB (L_k) for the values of the block of words represented by L_k= (a_i, a2,….a_k) can be defined as follows.

LB (I) = V (I)+CD[J + NA -I] – CD [J]
For Example: consider for the above partial word as follows
LB (4) = 6 + CD [4 + 5- 4] – CD [4]
LB (4) = 6+3 = 9

10.3.Feasibility criterion of a Partial Word:

An algorithm was developed, in order to check the feasibility of a partial word $L_{k+1} = (a_1, a_2, - - - -- a_k, a_{k+1})$ given that L_k is a feasible word. We will introduce some more notations which will be useful in the sequel.

- **IR** be an array where IR (i) = 1, $i \in N$ indicates that the sales man is visiting some city from city i Otherwise IR (i) = 0
- **IC** be an array where IC (i) = 1, $i \in N$ indicates that the sales man is coming to city i from another city, otherwise IC (i) = 0
- **SW** be an array where SW (i) = j indicates that the sales man is visiting city j from city i, Otherwise SW (i) = 0
- **L** be an array where L[i] = α_i, $i \in N$ is the letter in the i^{th} position of a word.

The values of the arrays L, IR, IC and SW are as follows

IR $(R(a_i)) = 1$, i = 1, 2, - - - - - , k and IR (j) = 0 for other elements of j

IC $(C(a_i)) = 1$, i = 1, 2, - - - - - , k and IC (j) = 0 for other elements of j

$SW(R(a_i)) = C(a_i)$, i = 1, 2, - - - , k and SW (j) = 0 for other elements of j

L (i) = a_i, i = 1, 2, - - - - -, k, and L (j) = 0, for other elements of j.

For example consider a sensible partial word L_5 = (1, 2, 3, 4(1) and 11) which is feasible. The array L, IR, IC and SW takes the values represented in table – 5 given below.

Table – 5

	1	**2**	**3**	**4**	**5**	**6**	**7**
L	1	2	3	4(1)			
IR	1	1		1	1		
IC		1		1		1	1
SW	2	7		6	4		

The recursive algorithm for checking the feasibility of a partial word L_p is given as follows. In the algorithm first we equate IX = 0. At the end if IX = 1 then the partial word is feasible, otherwise it is infeasible. For this algorithm we have IR = $R(a_{p+1})$ and IC = $C(a_{p+1})$

XI. ALGORITHMS

11.1.Algorithm-1

STEP 0 :	IX = 0	
STEP 1 :	IS IR = HC	IF YES {PA = PA+1} GO TO 2 IF NO GO TO 3
STEP 2 :	IS PA >2	IF YES GO TO 12 IF NO GOTO 4
STEP 3 :	IS IR (TR) = 1	IF YES GO TO 12 IF NO GO TO 4
STEP 4 :	IS IC (TC) = 1	IF YES GO TO 12 IF NO GOTO 5
STEP 5 :	W = TC	GO TO 6
STEP 6 :	IS SW (W) = 0	IF YES GO TO 9 IF NO GO TO 7
STEP 7 :	IS W = TR	IF YES GO TO 12 IF NO GO TO 8
STEP 8 :	S W=SW (TC)	GOTO 6 IF NO GO TO 11
STEP 9 :	CJ ≥ m – 2 (p-1)	IF YES GO TO 10 IF NO GO TO 12
STEP 10:	RJ = m-CJ Z = P – PA IS RJ ≤ 2Z	IF YES GO TO 11 IF NO GO TO 12
STEP 11:	IX = 1	
STEP 12 :	END.	

We start with the partial word $L_1 = (a_1) = (1)$. A partial word L_k is constructed as

$L_k = L_{k-1}$ * Where * indicates chain formulation. We will calculate the values of V (L_k) and LB (L_k) simultaneously. Then two situations arises one for branching and other for continuing the search.

1. LB (L_k) < VT. Then we check whether L_k is feasible or not. If it is feasible we proceed to consider a partial word of under (k+1). Which represents a sub-block of the block of words represented by L_k? If L_k is not feasible then consider the next partial word p by taking another letter which succeeds a_k in the position. If all the words of order p are exhausted then we consider the next partial word of order (k-1).

2. LB (L_k) $\geq$ VT. In this case we reject the partial word L_k. We reject the block of word with L_k as leader as not having optimum feasible solution and also reject all partial words of order p that succeeds L_k.

11.2. Lexi-Search Algorithm:

STEP 0 : (initialization): The arrays SN, D, CD, R, C, N, JX, CJ,NA, NN, K and Z the values are made available and IR, IC, L, V, LB, PA,SUM1,SUM2 and SW are initialized to zero. The values I=1, J=0, P=3, m = 8, VT =999, MAX=NZ-1.

STEP1: J_1= FW — IF YES GO TO 2; IF NO GO TO 2A

STEP 2: J=J +1 — IF YES GO TO 21

IS J> MAX — IF NO GO TO 3

STEP 2A: J_1=RW; TR = C (J); TC = R (J) — GO TO 4

STEP 3: L (I) =J; IR = R (J); IC = C (J) — GO TO 4

STEP 4: V (I) = V (I-1) + D (J) — GO TO 5

STEP 5: NA(TR) = NA (TR) +1; NA (TC) = NA (TC) +1 — GO TO 6

STEP 6: IS NA (i) =1, (i= 1,2,…n) — IF YES (SUM1 = SUM1 +1) GO TO 7; IF NO (SUM1 = SUM1) GO TO 7

STEP 7: NN = SUM1 — GO TO 8

STEP 8: JB (NJ (i,1)) = JB (NJ (i,1)) +1; JB (NJ (i,2)) = JB (NJ (i,2)) +1; JB (NJ (j,1)) = JB(NJ (j,1)) +1; JB (NJ (j,1)) = JB (NJ (j,1)) +1 — GO TO 9

STEP 9: IS JB (JX) =1, (JX = 1,2,3….q) — IF YES (SUM2 = SUM2 +1) GO TO 10; IF NO (SUM2 = SUM2) GO TO 10

STEP 10: IS CJ = SUM2 — GO TO 11

STEP 11: NA = NA_0 +Z-1 — GO TO 12

STEP 12: IS CJ+2 (Z-1) < m — IF YES GO TO 13; IF NO GO TO 2

STEP 13: K = m – CJ +2(Z-1); IS K = 0 — IF YES GO TO 15; IF NO GO TO 14

STEP 14: IS K = EVEN — IF YES NA = NA +K/2 GO TO 15; IF NO NA = NA +(K+1)/2 GO TO 15

STEP 15: LB (I) = V(I)+DC[J + NA -I] – DC (J); IS LB (i) $\geq$ VT — IF YES GO TO 21; IF NO GO TO 16

STEP 16: (CHECK FEASIBILITY BY USING ALGORITHM); IS IX=0 — IF YES GO TO 17; IF NO GO TO 24

STEP 17: NA(TR) = NA (TR) - 1; NA (TC) = NA (TC) - 1 — GO TO 18

STEP 18: IS NA (i) =1, (i= 1,2,…n) — IF YES (SUM1 = SUM1 - 1) GO TO 19; IF NO (SUM1 = SUM1) GO TO 19

STEP 19: NN = SUM1 — GO TO 20

STEP 20: JB (NJ (i,1)) = JB (NJ (i,1)) - 1
JB (NJ (i,2)) = JB (NJ (i,2)) - 1
JB (NJ (j,1)) = JB (NJ (j,1)) - 1
J B(NJ (j,1)) = JB (NJ (j,1)) - 1 GO TO 21
STEP 21: IS JB (JX) =1, (JX = 1,2,3….q) IF YES (SUM2 = SUM2 - 1) GO TO22
IF NO (SUM2 = SUM2) GO TO 22
STEP 22: IS CJ = SUM2 GO TO 2
STEP 23: L (I) = J
IC (TC) = 1
SW (TR) = TC
IS IR (TR) = HC IF YES (PA = PA+1) GOTO 24
IF NO (IR (TR) =1) GO TO 24
STEP 24: I=I+1 GO TO 25

STEP 25: IS I = NN – 1 IF YES GO TO 26
IF NO GO TO 2
STEP 26: IS PA = P IF YES GO TO 27
IF NO GO TO 2
STEP 27: IS SUM2 = m IF YES GO TO 28
IF NO GO TO 2

STEP 28: L (I) = J
L (I) IS FULL LENGTH WORD AND IS FEASIBLE
VT = V (I), RECORD L (I), VT GO TO 29
STEP 29: I=I-1 GO TO 30

STEP 30: IS I=1 IF YES GO TO 32
IF NO GO TO 31

STEP 31: J= L (I)
TC = C (J)
TR = R (J)
IC (TC) = 0
SW (TR) = 0 IF YES (PA = PA-1) GO TO 17
IS IR (TR) = HC IF NO (IR (TR) = 0) GO TO 17
STEP 32: STOP

XII. SEARCH-TABLE

The working details of getting an optimal word using the above algorithm for the illustrative numerical example is given in the Table – 6. The columns named (1), (2), (3),…, gives the letters in the first, second, third and so on places respectively. The columns R, C gives the row, column indices of the letter. The last column gives the remarks regarding the acceptability of the partial words. In the following table A indicates ACCEPT and R indicates REJECT.

TABLE-6
Search Table

SN	1	2	3	4	5	V	LB	R	C	CJ	REM
1	1					01	06	4	6	4	A
2		2				02	09	2	7	7	A
3			3			04	09	1	2	7	A
4				4		06	09	4	5		R
5				4(1)		06	09	5	4	8	A
6					05	09	09	1	6		R
7					05(1)	09	09	6	1		R
8					06	09	09	3	7		R
9					06(1)	09	09	7	3		R

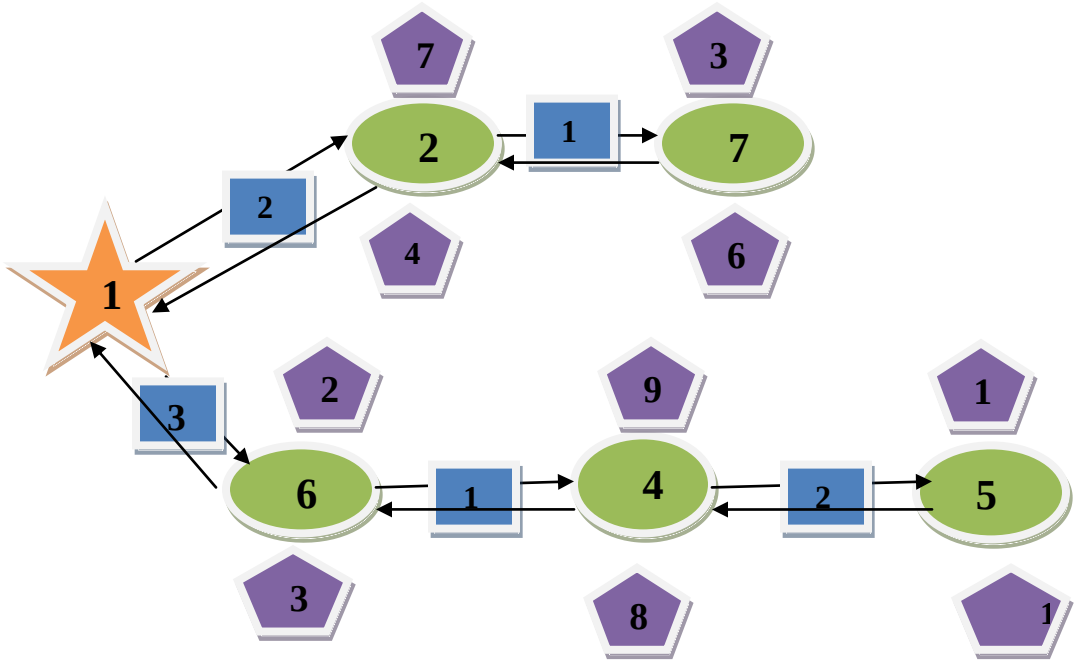
10					07	10	10	2	3		R
11					07(1)	10	10	3	2		R
12					08	10	10	6	7		R
13					08(1)	10	10	7	6		R
14					09	11	11	1	4		R
15					09(1)	11	11	4	1		R
16					10	11	11	5	6		R
17					10(1)	11	11	6	5		R
18					11	12	12VT	1	5		A
19				05		07	10	1	6		R
20				05(1)		07	10	6	1		R
21				06		07	11	3	7		R
22				06(1)		07	11	7	3	8	A
23					07	11	11	2	3		R
24					07(1)	11	11	3	2		R
26					08(1)	11	11	7	6		R
27					09	12	12	1	4		R,=VT
28				07		08	12	2	3		R,=VT
29			3(1)			04	09	2	1		R
30			4			04	10	4	5		R
31			4(1)			04	14	5	4		R,>VT
32		2(1)				02	09	7	2	7	A
33			3			04	09	1	2		R
34			3(1)			04	09	2	1		R
35			4			04	10	4	5		R
36			4(1)			04	10	5	4	8	A
37				5		07	10	1	6		R
38				5(1)		07	10	6	1		R
39				6		07	11	3	7		R
40				6(1)		07	11	7	3		R
41				7		08	12	2	3		R,>VT
42			5			05	12	1	6		R,>VT
43		3				03	08	1	2	6	A
44			4			05	08	4	5		R
45			4(1)			05	08	5	4	8	A
46				5		08	08	1	6		R
47				5(1)		08	08	6	1		R
48				6		08	08	3	7		R
49				6(1)		08	08	7	3		R
50				7		09	09	2	3		R
51				7(1)		09	09	3	2		R
52				8		09	09	6	7		R
53				8(1)		09	09	7	6		R
54				9		10	10	1	4		R
55				9(1)		10	10	4	1		R
56				10		10	10	5	6		R
57				10(1)		10	10	6	5		R
58				11		11	11VT	1	5		A
59			5			06	09	1	6		R
60			5(1)			06	09	6	1		R
61			6			06	10	3	7		R

62			6(1)			06	10	7	3		R
63			7			07	16	2	3	7	R,>VT
64		3(1)				03	08	2	1		R
65		4				03	09	4	5		R
66		4(1)				03	09	5	4	6	A
67			5			06	09	1	6		R
68			5(1)			06	09	6	1		R
69			6			06	10	3	7		R
70			6(1)			06	10	7	3		R
71			7			07	11	2	3		R,=VT
72		5				04	11	1	6		R,=VT
73	1(1)					01	06	6	4	4	A
74		2				02	09	2	7	7	A
75			3			04	09	1	2	7	A
76				4		06	09	4	5	8	A
77					5	09	09VT	1	6		A
78				4(1)		06	09	5	4		R,=VT
79			3(1)			04	09	2	1		R,=VT
80		2(1)				02	09	7	2		R,=VT
81	2					01	08	2	7	4	A
82		3				03	08	1	2	4	A
83			4			05	08	4	5	8	A
84				5		08	08	1	6		R
85				5(1)		08	08	6	1		R
86				6		08	08	3	7		R
87				6(1)		08	08	7	3		R
88				7		09	09	2	3		R,=VT
89			4(1)			05	08	5	4	8	A
90				5		08	08	1	6		R
91				5(1)		08	08	6	1		R
92				6		08	08	3	7		R
93				6(1)		08	08	7	3		R
94				7		09	09	2	3		R,=VT
95			5			06	13	1	6	5	R,>VT
96		3(1)				03	08	2	1		R
97		4				03	09	4	5	8	R,=VT
98	2(1)					01	08	7	2	4	A
99		3				03	08	1	2		R
100		3(1)				03	08	2	1		R
101		4				03	09	4	5		R,=VT
102	3					02	10	1	2		R,>VT

The above table – 6, gives optimal solution of the taken numerical example. The search table has an optimum solution VT is 09. It is in the 77th row of the search table.

At the end of the search table the optimum solution value of VT is 09 and the optimal feasible word L_5 = {1(1),2,3,4,5}.The following figure-2, represents the optimal solution to the network.

Figure-2



In the above Figure-2, in first path the agent starts his trip at Head Quarter city{ 1} reaches to city 2 and performs the job'7'of season "1", from city 2 he visits the city 7 and performs the jobs '3'of season "1", while return in the same path he performs job '6' of season "2" in city 7, job '4'of season "2" in city 2 and reaches to the headquarter city{1}. In second path another agent visits the city 6 from city 1 and performs the job'2'of season "1", from city 6 he visits to city 4 and performs the job'9' of season "1", from city 4 he visits city 5 and performs the job '1' in season "1",while return in the same path he performs job'10' of season "2"in city 5, job '8' of season "2" in city4 ,job '3' of season "2" in city 6 and reaches to the Head Quarter city . Hence the solution is

Z = D (1, 2) + D (2, 7) + D (1, 6) + D (6, 4) + D (4, 5)
= 2+1+3+1+2 = 9.

XIII. CONCLUSION

In this chapter we developed a Lexi-search algorithm to solve

P-AGENTS SEASONAL JOB COMPLETION MODEL by a suitable numerical example. Lexi-search algorithm using pattern recognition technique is used to get an optimum solution. A numerical example developed to understand the concepts and the steps involved in the algorithms. Lexi – search algorithm are proved to be more efficient in many combinatorial problems. I suggest that these algorithms can perform larger size problems also.

ACKNOWLEDGEMENTS

The first author express my deep sense of reverence and gratitude to the research supervisor Prof. M .Sundara Murthy,(Retd), Department of Mathematics, S.V.University, Tirupati, Andhra Pradesh, for suggesting this problem for investigation. It is solely due to his immense interest, critical analysis, exceptional guidance and concrete suggestions enlightened discussions, which cumulatively are responsible for the successful execution of this work.

REFERENCES

[1] Crores G.A. "A method for solving TSP" Operational Research,Vol.6, No.3, pg.791(1958)

[2] Das Shila: "Routing and Allied Combinatorial Problems" - Ph.D., Thesis, Dibrugrah University, Dibrugrah Assam, India.(1976)

[3] Flood, M.M: The TSP. Opns. Res., 41, pp. 61- 75 (1956)

[4] Hardgrave,W.W. & G.L. Nambhauser: "On the Relation between the Travelling Salesman and the Longest Path Problems" Opens. Res., Vol. 10, p. 647 (1962)

[5] Jaillet.P : "Probabilistic TSP " PhD thesis Technical Report No. 185, Opns. Res., Center, Massachusetts Institute of Technology, Cambridge, M.A. (1985)

[6] Kubo, M and Kasugai, H: "The precedence constrained TSP, J. of Opns. Res., of Japan, Vol. 34, p. 152 (1991)

[7] Little, J.D.C. Murthy K.G. Sweeny, D.W : "An algorithm for the TSP. Opns. Res., 11, pp.972-982 (1963)

[8] Pandit, S.N.N.:"Some observations on the Longest Path Problems". Opns. Res. Vol. 11, p. 361 (1964)

[9] Ramesh, T : "Traveling Purchaser Problem". Opens. Res., Vol. 18, No.2, p.781 (1981)

[10] Rajbongshi, M : "Sequencing and allied combinatorial programming problems" Ph.D. Thesis, Osmania University, Hyderabad, A.P., India.(1982)

[11] Raviganesh, G.S. Murty and S.Das: "Solution of TSP in a Protean Network. IJOMAS, Vol. 14, No.1, p. 99. (1998)

[12] Raman V.V.V. and Umashankar. C: "A class of combinatorial programming problems - using Lexi-search approach, Ph.D. Thesis, S.K.D. University, P.G. Centre, Kurnool, A.P., India.(1994)

[13] Shila Das, Nazimuddin Ahmed, A Travelling Salesman Problem (TSP) with Multiple Job Facilities published in OPSEARCH, Vol. 38(4), 2595-2600, (2001)

[14] Sundara Murthy. M: "Combinatorial Programming - A Pattern Recognition Approach" PhD, Thesis REC, Warangal, India.(1979)

[15] Suresh babu C "lexi-search exact algorithms for variant TSP, Bulk TP and network spanning models", Ph. D. thesis, S. V. University, Tirupati, A.P, (2012)

[16] Srivastava, S.S. Santhosh Kumar Garg, R.C&Sen, P: "Generalized TSP through n sets of nodes" Opns.Res.Vol.7, No.2pp. 97-101.(1969)

[17] Vidyullatha. A : "Combinatorial Programming Problems. M.Phil, dissertation, S.V.U.C.E., Tirupati, India. (1992).

AUTHORS

First Author – Soma Sekhar Srinivas V.K, Dept. of Mathematics, S. V. University, Tirupati, Andhra Pradesh, India., Email:somasekharsrinivas@gmail.com

Second Author – Suresh Babu.C, Dept. of Mathematics, S. V. University, Tirupati, Andhra Pradesh, India. , Email: suresh8044@gmail.com

Third Author – Madhu Mohan Reddy. P, Dept. of Mathematics, S. V. University, Tirupati, Andhra Pradesh, India. , Email: mmrphdsv@gmail.com