A PROBABILISTIC POLYNOMIAL ALGORITHM FOR SOLVING A DIRECTED HAMILTONIAN PATH PROBLEM

by

G.L. Thompson & S. Singhal

December, 1983

DRC-70-16-83

A PROBABILISTIC POLYNOMIAL ALGORITHM FOR SOLVING A DIRECTED HAMILTONIAN PATH PROBLEM*

11 May 1983
(Revised July 1983)

by
Gerald L. Thompson
and
Shared Singhal

* **The original title of this paper was *A Polynomial Algorithm which has Positive Probability of solving a Hamiltonian Path Problem.***

This report was prepared as part of the activities of the Management Science Research Group, Carnegie-Mellon University, under Contract No. N00014-82-K-0329 NR 047-048 with the U.S. Office of Naval Research. Reproduction in whole or part is permitted for any purpose of the U.S. Government

Management Science Research Group
Graduate School of Industrial Adminstration
Carnegie-Mellon University
Pittsburgh, Pa 15213

Abstract

In this paper we present a graph-theoretic polynomial algorithm which has positive probability of finding a Hamiltonian Path in a given graph, if there is one; if the algorithm fails, it can be rerun with a randomly chosen starting solution, and there is again a positive probability it will find an answer. If there is no Hamiltonian Path, the algorithm will always terminate with failure.

Some basic theoretical results concerning spanning arborescences of a graph are given. The concept of a ramification index is defined and it is shown that ramification index of a Hamiltonian Path is zero. The algorithm starts with finding any spanning arborescence and by suitable pivots it endeavors to reduce the ramification index to zero. Probabilistic properties of the algorithm are discussed. Computational experience with graphs up to 30,000 nodes is included.

1. Introduction

Hamiltonian Cycle problem is the problem of finding a path in a graph which passes through each node exactly once. This problem is well known and has been discussed in most graph theory books such as [1, 3, 5].

A polynomial algorithm will be presented in this paper which has positive probability of finding a Hamiltonian Path in a given graph if there is one. If a graph has a Hamiltonian Path, this algorithm will either find it or end with a message that it cannot proceed further. However, if there is no Hamiltonian Path in the graph, the algorithm will always end with the "cannot proceed further" message.

Despite the fact that the algorithm can fail, we have found it to be of great practical value due to the following reasons :

- With positive probability the algorithm finds the Hamiltonian path if there is one in the graph. See Section 5 for a probabilistic analysis of the algorithm.

- In case the algorithm does not find a Hamiltonian path, it can be restarted with a different randomly chosen starting solution, where again there is a positive probability of finding a Hamiltonian Path.

- It takes only a few milliseconds to solve a problem of 100 nodes. Thus it is more efficient to repeatedly use this algorithm in the hope that eventually an answer may be obtained rather than use other large scale algorithms [2, 4, 7, 8].

- The algorithm has solved problems having 30.000 nodes. Such a large problem takes less than one minute of DEC-20 CPU time. If the algorithm does not find a Hamiltonian path, it is very economical to run it again with a different starting solution. Our method is constrained by memory limitations and not computation time. It would be possible to overcome the memory limitation by packing data or by making use of secondary storage devices.

2. Exact statement of the problem

Let G = (V,A) be a directed graph with V as the set of nodes and A as the set of arcs. G has *n* nodes and they are numbered sequentially from 1 to n. The indegree of node / is zero. The outdegree of node *n* is also zero. The problem is to find the HP from node / to node *n* which goes through all the intermediate nodes exactly once.

2.1. The Hamiltonian Cycle Problem

The problem of finding a Hamiltonian cycle in a directed graph can be easily converted to a problem of finding a HP as shown in Figure 11. We split any node of the graph into two nodes so that one node has no incoming arc and the other has no outgoing arc. The Hamiltonian Path between these two nodes would be a Hamiltonian cycle in the original graph. In Figure 2-1 HP between nodes 7 and *1a* is equivalent to a HC Thus we are solving the general Hamiltonian Path problem rather than for a specific type of graph.

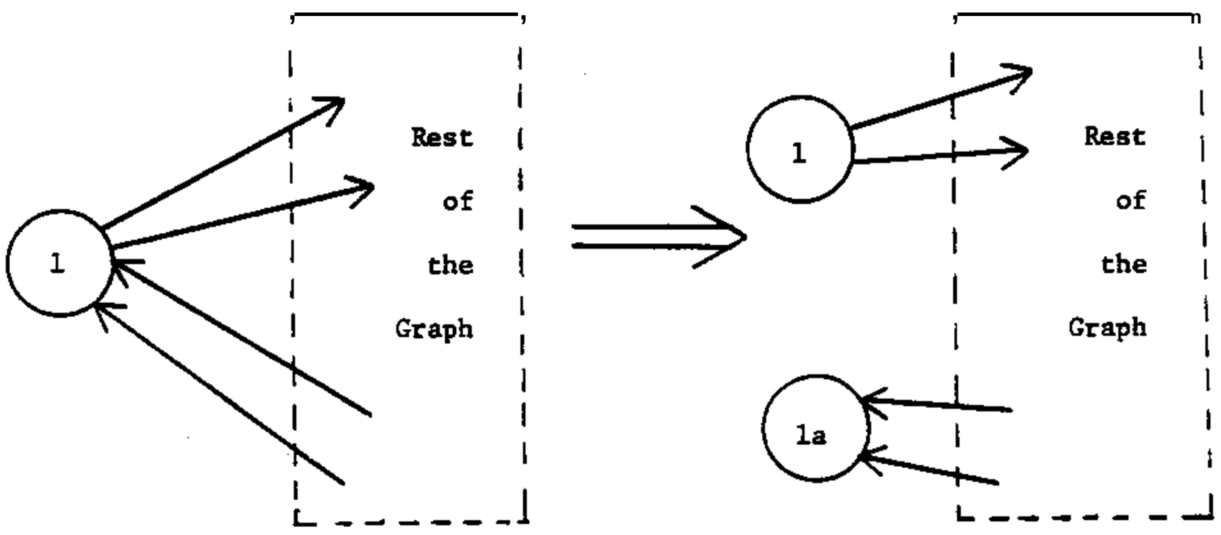

Figure 2.1: Converting a Hamiltonian Cycle Problem to a Hamiltonian Path Problem

2.2. The Postman Pickup Problem

This is a problem which can be translated into a Hamiltonian Path problem. A postman has to pick up one letter from each of *n* locations and deliver them to the main post office. Starting from his home, how can he do this while covering the minimum possible distance ?

Let node 1 represent his home and node n represent the post office. Nodes 2 through n-1 are the locations from which letters have to be picked up. The Hamiltonian Path from node 1 to node n is a possible path on which the postman should travel in order to minimize his total distance traveled.

3. Preliminary concepts and definitions

We give some preliminary definitions with interpretations related to the postman pickup problem.

Definition 1: An *arboresence* is a directed tree having one node called the *root node* which can be reached by a unique directed path from any node.

For the postman pickup problem, the root node is the post office.

Definition 2: Let G = (V,A) be a directed graph. Then arborescence T = (V,A_T) is a *spanning arborescence* of G if A_T is a subset of A.

Definition 3: Let i and j be two nodes of a spanning arborescence of G and (lj) € A - A_T.

1. If there exists a directed path in T from i to j then (ij) is said to be an *up arc.*

2. If there exists a directed path in T from j to i then (ij) is said to be a *down arc.*

3. Otherwise (ij) is called a *cross arc.*

Definition 4: If i, j € V and (ij) € A_T, then *predecessor node of i* is said to be j or symbolically p(i) = j.

One can interpret predecessor of i to be like the "father" of i in the sense of a family tree.

Definition 5: A node i € V is said to be a *junction node* of arborescence T = (V,A_T) if it is the end node of at least two arcs belonging to A_T.

Definition 6: A node i € V is said to be a *beginning node* if it is not the end node of any arc 6 A_T.

Definition 7: For each i € V we define the *successor set* V_i = {h : p(h) = ih Thus ¥ is the set of nodes which are immediately below i, i.e. the set of nodes for which i^1 is the predecessor node.

Lemma 8: If $\| Y_i \|$ = 0 then i is a beginning node.

Lemma 9: If $| I^*_i | \wedge 2$ then i is a junction node.

Definition 10: The *successor function* of i represented by s(i), for i G V is defined inductively as follows :

$$s(i) = 1 + \sum_{h \in V} S(h>$$

In other words, s(i) = 1 *+ number of nodes in T below i. For the postman pickup problem s(i) is the number of letters the postman will carry after leaving node i.

Lemma 11: If i is a beginning node then s(i) = 1.

Definition 12: The *ramification index* R of an arborescence T = (V,A_T) is defined as follows:

$$R \equiv R(T) = n*(n-l)/2 - \sum_{i \in V \{n\}} s(i)$$

Lemma 13: The maximum possible ramification index of an arborescence is $(n-1)(n-2)/2$

Proof: The arborescence for which the ramification index is maximum is the one for which the only junction node is the root node and each of the *n-1* arcs in A_T has the root node as its end node. For this case the values of the successor function at each node i for i = 1.....n-1 is 1 and we get from Definition 12 :

$$R = n*(n-1)/2 - (n-1) = (n-1)*(n-2)/2$$

Definition 14: Let J be the set of junction nodes of arborescence T. The *ramification index of a junction node* j € J is defined as :

$$R_j = \sum_{i_1, i_2 \in \Psi_j,\ i_1 < i_2} s(i_1) * s(i_2)$$

The significance of Definition 14 is due to the following theorem.

Theorem 15: The ramification index of an arborescence T can be computed from the following equation:

$$R \equiv R(T) = \sum_{j \in J} R_j$$

Proof: Assume T = (V,A_T) has *n* nodes. We will first prove the proposition for the case in which T has only one junction node j. The nodes above j do not contribute to the ramification index because as will be seen in Theorem 16 (4 => 5), a tree with no junction nodes has zero ramification index. Let s_1 to s_m be the successor functions of the m arcs coming into j. Note that j=n implies $s_1+s_2+...+s_m = n-1$. According to Definition 12

$$R \equiv R(T) = n*(n-1)/2 - \sum_{i \in V-\{n\}} s(i)$$

or

$$R_j = R = \sum_{j=1}^{m} \sum_{i=1}^{s_i} i$$

$$= \left[\sum_{i=1}^{s_1} i - \sum_{i=1}^{s_1} i \right] + \left[\sum_{i=s_1+1}^{s_1+s_2} i - \sum_{i=1}^{s_2} i \right] + \ldots$$

$$\ldots + \left[\sum_{i=s_1+\ldots+s_{m-1}+1}^{s_1+\ldots+s_m} i - \sum_{i=1}^{s_m} i \right]$$

$$= s_1*s_2 + (s_1+s_2)*s_3 + \ldots + (s_1+s_2+..+s_{m-1})*s_m$$

The last expression is equivalent to the expression in Definition 14.

We have proved the proposition when there is just one junction node in T. By induction, assume that the proposition is true for up to k-1 junction nodes in T. We will now show that the proposition is also true for k junction nodes in T.

Let J be the set of k junction nodes of T with j designating the top most junction node. We again assume that j=n. The proof is stated for the case in which (i_1,j) and (i_2,j) are the only two arcs coming into j. Let successor functions of i_1 and i_2 be s_1 and s_2 respectively. Let V_1 denote all the nodes below and including i_1 and V_2 denote all the nodes below and including i_2. Furthermore, let J_1 and J_2 denote the sets of junction nodes below i_1 and i_2 respectively. Then by Definition 12

R(T) = bxx+s+Mt+s3)/2 – X, ,, v s(i) - X. i ε v2 s(i)
v

Since | J1 || £ k-1 and || J2 1 £ k-1. we can write

R(T) = $(s_i+s,)(s+s.+l)/2$ - [$s_i(s_i+l)/2$ - 21 .j f J1 Rj J -

[$s_2(v\text{-}l)/2$ - X, ^ R,]

= s_1*s: • X, f , V X, ; i, RJ

= Σ R

This completes the proof. If there were more then two arcs entering j, the proof would simply require more summation terms.

Theorem 16: Let T = (V,A_T) be a spanning arborescence of a directed graph G = (V,A). The following statements are equivalent :

1. T is a Hamiltonian Path starting from node 1 and ending at node n.
2. There is a directed path in T from node 1 to any other node x 6 V.
3. T has node 1 as its only beginning node.
4. T has no junction nodes.
5. T has zero ramification index.

Proof: Proof f o r l = £ 2 = £ 3 ^ 4 i s obvious.

4 => 5 The assertion follows directly from Theorem 15. Because J = *f* there are no terms in the first summation and thus ramification index must be zero.

5 =£ 1 Because ramification index of T is 0, from Definition 12, we have

^ , . v-{.> s(i) - n * to"" , 2

Since all s(i) are integer, the only way in which this is possible is if there is an ordering of the nodes such that $s(i_1)=1$, s(i^)=2 . . . . . $s(i_1)=n\text{-}l$. Because the number of successors mcreasel sequentially, it YoUows that T must be a Hamiltonian Path. Because indegree of node 1 is 0, i =1. Also because the outdegree of n is 0 we can choose i_n = n. *

4. A Polynomial Algorithm Which Has Positive Probability Of Finding A Hamiltonian Path (If there is one)

Let G=(V,A) be a directed graph with n nodes as described in Section 2. The algorithm to be described makes use of Theorem 16 which says that the ramification index of a Hamiltonian Path is zero. We first find a spanning arborescence of G denoted by T by constructing a greedy starting solution. By performing a sequence of pivots we try to reduce the ramification index of T and when (and if) the ramification index of T becomes 0, we have found a Hamiltonian path.

The algorithm can get stuck at a given arborescence T with R(T) > 0 if it is unable to find a pivot which will result in a decrease in the ramification index. This may mean : (a) that the graph has no Hamiltonian Path, or (b) that the algorithm made an unfortunate choice of pivots which led it to get stuck. The alternative to stopping in this case is to choose a different starting solution and go through the algorithm again. As evident from the discussion in Section 5, there exists a starting solution which will lead with positive probability to a Hamiltonian Path, if there is one, after a finite number of pivots.

For ease of exposition, we present the algorithm in three separate parts.

4.1. Algorithm 1 : Finding a Spanning Arborescence

We want to find a spanning arborescence T = (V,A_T) of a directed connected graph G = (V,A). For this purpose we define M which we call the set of marked nodes. This algorithm begins with A_T = *f* and ends when set A_T has been completely defined

- *Step 0 :* Set M = {nh A_T = ^, and k=0.
- *Step 7 :* Choose any arc (ij) G A such that i ^ M and j 6 M.
- *Step 2 :* Set M = M U «}. $A_?$ = A_y U «i,j)} and k=k+l.
- *Step 3 :* If k = n - 1 then STOP, else go to Step 1.

The above is the greedy form of the algorithm for finding-a spanning arborescence because it chooses as many as possible of those arcs to enter the solution which have their end points already marked to belong to T. A random greedy starting arborescence can be found by letting p be any number satisfying 0 < p < 1 and altering Step 2 as follows :

- *Step 2* * : Let x be a random number between 0 and 1; if x < p go to step 1, otherwise set M = M U {j} and $A_?$ = A_T U {(i,jH.

4.2. Algorithm 2 : Calculating the Change in ramification index

Given a spanning arborescence, it is possible to compute its ramification index either by using Definition 12 or else by using the equations of Theorem 15. However we now present a method of calculating the change in ramification index which is much faster than those methods.

Given an arborescence, a cross arc (i,j) is to be brought into the solution. Let there be m_1 arcs on the path from node i to node n and m_2 arcs on the path from node j to node n. Let the successor function of i be denoted by δ. Noting that an increase in successor function means an equal decrease in the ramification index, consider what happens when the cross arc (i,j) is brought into the solution :

1. The successor function of each of the nodes on the path from node i to node n (excluding node n) goes down by δ so that the corresponding change in ramification index is
$$\Delta R_1 = m_1 * \delta \tag{1}$$
2. The successor function of each of the nodes on the path from node j to node n (excluding node n) goes up by δ so that the corresponding change here is
$$\Delta R_2 = - m_2 * \delta \tag{2}$$
3. Finally, the extra arc (i,j) is added to the arborescence giving us
$$\Delta R_3 = - \delta \tag{3}$$

The sum of equations 1 to 3 is the total change in ramification index if arc (i,j) is brought into the solution. We have now proved the following Theorem.

Theorem 17: The change in ramification index of an arborescence due to the introduction of a cross arc (i,j) into the solution is given by
$$\Delta R = (m_2 - m_1 - 1) * \delta$$
where

m_1	= Number of arcs on the path from node i to node n.
m_2	= Number of arcs on the path from node j to node n.
δ	= Successor function of node i before arc (i,j) is brought into the solution.

Now Algorithm 2 can be very simply stated.

- Step 1 : Find a cross arc.
- Step 2 : Using Theorem 17 find the change in ramification index if that arc were to be brought into the solution.

4.3. Algorithm 3 : Finding A Hamiltonian Path

- **Step 0 :** *Find Initial Solution.* **Find a directed rooted spanning arborescence of G with *n* as the root node by using Algorithm 1.**

- **Step 1 :** *Calculate the successor function and ramification index of* **T. Using Definition 10 calculate s(i) for all i € V. Then from Definition 12 or from Theorem 15 calculate R(T). If R(T) = 0 STOP — T is a Hamiltonian Path. Else go to next step.**

- **Step 2 :** *Find incoming arc.* **For each cross arc (i,j) 6 A - A_T calculate the ramification index if that arc is brought into A_T using Algorithm 1. Let R_{new} be the minimum of these ramification indexes and lei $(i_1 J_1)$ be the corresponding arc.**

- **Step 3 :** *Check possible cases.* **If**

R_{new} =0	**STOP - A Hamiltonian Path has been found.**
R_{new} £ R	**STOP - The algorithm failed to find a Hamiltonian path on this trial If another trial is desired go back to Step 0 and generate a new, different spanning arborescence.**
R_{new} < R	**Go to next step**

- **Step 4 :** *Perform Pivoting.* **Bring the cross arc (i j) having ramification index R into the solution by setting A_y = $A_?$ -(j^k) G V_T} U Ui^H. Let R = RT_{new} Update the successor function. Go to Step[1] 2.**

4.4. Algorithm 4 : Finding a maximal R Spanning Arborescence

It is interesting to note that with minor modifications Algorithm 3 can be used to find a spanning arborescence with maximal ramification index. The modified algorithm is described now.

- **Step 0 :** *Find Initial Solution.* **Find a spanning arborescence of G using Algorithm 1.**

- **Step 1 :** *Calculate successor function and ramification index of T.* **Using Definition 10 calculate s(i) for all i 6 V. Then from Definition 12 or from Theorem 15 calculate R(T).**

- **Step 2 :** *Find incoming arc.* **For each cross arc (i,j) G A - A_T calculate the ramification index if that arc is brought into A using algorithm 2. Let R_{new} be the maximum of these ramification indexes and let $6_1 j_1$) be the corresponding arc.**

- **Step 3 :** *Check possible cases.* **If**

R_{new} ≤ R	**STOP - A maximal R spanning arborescence has been found.**
R_{new} < R	**Go to next step**

- **Step 4 :** *Perform Pivoting.* **Bring the cross arc $(i_1 J_1)$ having ramification index R_{new}**

into the solution by setting $A_? = A_? - \{(j\hat{}k) : (j,k) \in A_T\} \cup \{(i\hat{}\hat{})\}$- Let $R = R_{new}$. Update the successor function. Go to step i

There are applications in which spanning arborescences which have maximal ramification index can be useful. One such application is the design of computer terminal networks in which signal concentrators are located at junction nodes [61.

It is obvious that a random choice of incoming arcs can be made in Step 2 of each of the last two algorithms by letting R_{new} be any incoming arc which decreases the ramification index in Algorithm 3 (or increase the ramification index in algorithm 4). We will use the probabilistic versions of these algorithms in the next section.

		STATE OF NATURE FOR G	
		There exists **HP**	There is *no* **HP**
PROBABILITY OF ALGORITHM	**Succeeding**	**1**	0
	Failing	p^k	$1 - p^k$

Note that p < 1 and $p^k \Rightarrow 0$ as $k \Rightarrow$ infinity

Figure 4.1: Probabilities associated with Algorithm 3

5. Probabilistic Analysis of the algorithm

Let A be the set of all spanning arborescences of a graph G. Then for any starting arborescence $T \in A$, let p be the probability of the algorithm failing to find the Hamiltonian Path when there is one. We know that p is less than one for every graph because this algorithm starts with a greedy random starting arborescence which has a miniscule chance of being a Hamiltonian Path. Since $p^k \to 0$ as $k \to$ infinity we know that this algorithm converges in probability to the Hamiltonian Path. In other words, if G has a Hamilton Path, then Algorithm 3 will find it if it is run sufficient number of times. From empirical runs it appears that p is very small so that for a reasonably small k we can bring the probability of the algorithm failing when there *is* a Hamilton Path as close to zero as we please. This can be seen graphically from Figure 4.1 which is valid for any algorithm which has a positive probability of success.

Theorem 18: Algorithm 3 is a polynomial algorithm.

Proof: The maximum possible loop value of an arborescence is of the order of $O(n^2)$ [Lemma 13]. We decrease the loop value by at least 1 in each pivot, and each pivot step is linear in number of arcs plus number of nodes, so that Algorithm

3 is also (Xn^2).

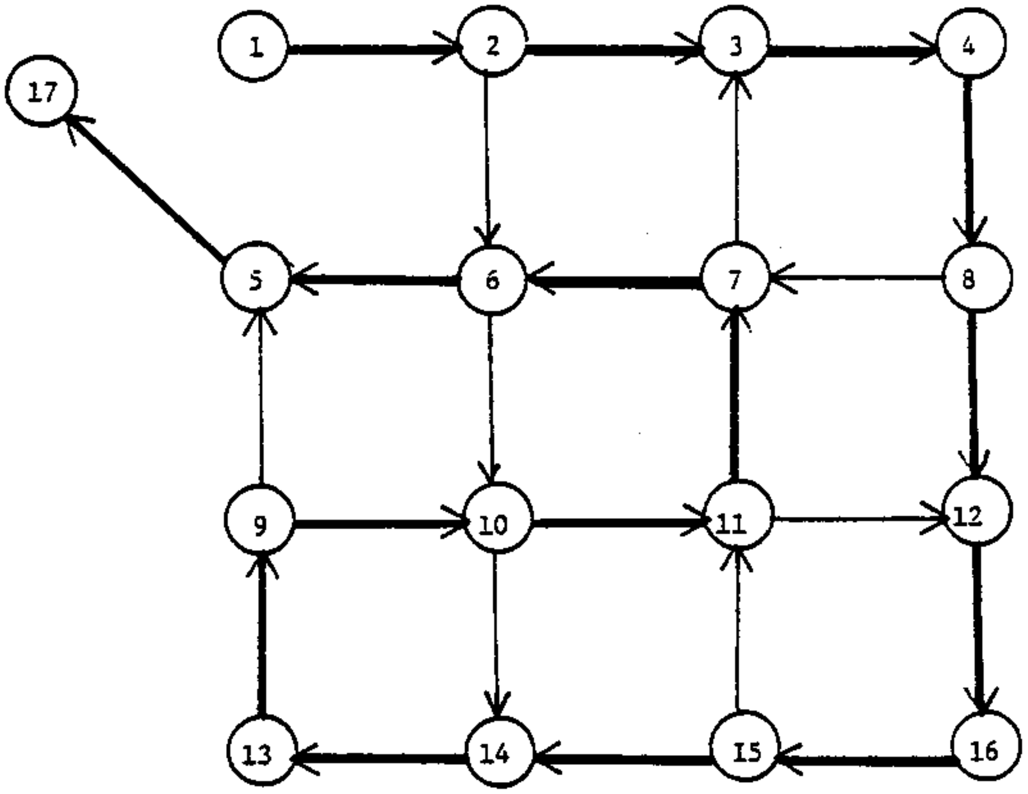

Figure 5.1: Example of a 4X4 Directed Rectangular Lattice Graph. A Hamilton Path is indicated by the darkened edges.

6. Computational Experience

We ran Algorithm 3 on Directed Rectangular Lattice Graphs[1] because they are easy to generate and it is possible to predict whether there is or is not a Hamilton Path in a given DRLG. The computational experience is shown in Figure 6.1.

We now make some observations about the computational results.

- The algorithm was coded in FORTRAN and all the runs were performed on the DECSYSTEM-20 at Carnegie-Mellon University.
- The starting value of ramification index is about 98% of the maximum possible value of ramification index [Lemma 13]. A large value of R indicates a bushy tree which is a good start since it has less chance of getting stuck.

[1] Directed Rectangular Lattice Graphs or DRLG's arc planar graphs which look like J grid as shown in Figure 5.1. They always have a Hamilton Path when the number of nodes is even belore runic 1 is split. Sec [9] for details

Grid	#Nodes	# Arcs	Starting R	#Pivots	CPU Time(Sec)
10	100	180	4,080	24	.062
20	400	760	72,360	99	.142
30	900	1,740	378,840	224	.292
40	1,600	3,120	1,217,520	399	.631
50	2,500	4,900	3,002,400	624	1.138
60	3,600	7,080	6,267,480	899	1.868
70	4,900	9,660	11,666,760	1,224	2.792
80	6,400	12,640	19,974,240	1,599	4.113
90	8,100	16,020	32,083,920	2,024	5.688
100	10,000	19,800	49,009,800	2,499	8.118
110	12,100	23,980	71,885,880	3,024	10.365
120	14,400	28,560	101,966,160	3,599	14.232
130	16,900	33,540	140,624,640	4,224	17.531
140	19,600	38,920	189,355,320	4,899	21.587
150	22,500	44,700	249,772,200	5,624	26.204
160	25,600	50,800	323,609,280	6,399	39.976
170	28,900	57,460	412,720,560	7,224	43.856
174	30,276	60,204	453,079,992	7,568	38.434

Figure 6.1: Computational experience for Algorithm 3 with a Greedy Start

# Nodes	# Arcs	Starting R	# Pivots	CPU Time(sec)
100	180	4,064	46	.118
400	760	72,008	177	.442
900	1,740	337,488	481	1.773
1600	3,120	1,213,976	852	7.474
2500	4,900	2,994,828	1,323	11.965
3600	7,080	6,251,360	2,151	22.966
4900	9,660	11,654,952	2,889	35.652
6400	12,640	19,949,896	3,994	86.459

Figure 6.2: Computational Results for Random Starting Solution

- According to Theorem 18, The upper bound on the number of pivots is the maximum possible ramification index. However, we can see from Figures 6.1 and 6.2 that number of pivots required to obtain the solution is much less than the starting value of the ramification index.

- When a random starting solution is chosen, Algorithm 3 takes little more than twice the computing time compared to when a greedy starting solution is used. The number of pivots required are also double. Why this is to be expected is explained next

- A greedy starting solution for a DRLG happens to be a good starting solution having a structure reasonably close to a Hamiltonian Path. As can be seen from Figure 6.1, ft is even possible to predict the number of pivots required by 1 subsequent problem using the formula

  Change in number of Pivots = One-fourth the change in number of nodes

  For a random start, there is no regular pattern.

- Number of CPU seconds required per pivot per node is a constant indicating empirically that each pivot requires a linear amount of work.

7. Conclusions

- We have devised a way of improving the performance of algorithm 2 which is expected to decrease the CPU times noted in Figures 6.1 and 6.2.

- We have proved that Algorithm 3 will never fail when applied to DRLG with even number of nodes (before any transformations are performed). A DRLG with odd number of nodes does not have any HP so that the algorithm will always fail.

- Algorithm 3 as stated is very inefficient for symmetric graphs[2] . We have developed a modification of Algorithm 3 which has proved to be very effective for finding a Hamiltonian Path on an undirected graph. The results will be reported in a later paper.

[2]in a Symmetric graph, if ihcrc is a direeled arc between nodes i and j then there i> also a directed arc between nodes j and i.

References

1. Bondy J. A. and Murty U. S. R.. *Graph Theory with Applications.* North Holland, 1978.

2. Christofides, Nicos and Eilon Samuel. "Algorithms for Large-scale Travelling Salesman Problems." *Op/ Res. Q. 23* (1972), 511-518.

3. Christofides, Nicos. *Graph Theory - An Algorithmic Approach.* Academic Press, London, 1975.

4. Crowder, Harlan and Padberg, Manford W. "Solving Large Scale Symmetric Travelling Salesman Problem to Optimality." *Management Science 26,* 5 (May 1980).

5. Garfinkel, Robert S. and Nemhauser, George L.. *Integer Programming.* John Wiley and Sons, 1972.

6. Gavish, BezaleL Topological Design of Centralized Computer Networks - Mathematical Formulations. Working Paper 1981.

7. Smith, T. H. C *A UFO Implicit Enumeration Alorithm for the Asymmetric Travelling Salesman Problem Using a One-arborescence Relaxation.* Ph.D. Th., Carnegie-Mellon University, 1975.

8. Smith, T. H. C and Thompson, Gerald L "A LIFO Implicit Search Algorithm for the Symmetric Travelling Salesman Problem Using Held and Karp's One Tree Relaxation." *Annals of Discrete Mathematics 1* (1977).

9. Thompson, Gerald L "Hamiltonian Tours and Paths in Rectangular Lattice Graphs." *Mathematics Magazine* 50, 3 (May 1977), 147-150.