HEURISTIC ALGORITHM FOR GRAPH COLORING BASED ON MAXIMUM INDEPENDENT SET

Hilal Al Mara'beh, Amjad Suleiman

Department of Basic Science King Saud bin Abdulaziz University for Health Sciences (KSAU-HS), Saudi Arabia

ABSTRACT: A number of heuristic algorithms have been developed for the graph coloring problem, but unfortunately, on any given instance, they may produce colorings that are very far from optimal. In this paper we investigated and introduce a three heuristics algorithm to color a graph based on a maximum independent set. The select a node with minimum and maximum degree consecutively (Min_Max) algorithm is implemented, tested and compared with select a node with minimum degree first (SNMD), select a node with maximum degree first (SNXD) in terms of CPU time and size of graph with different densities (0.1,0.2,…,0.9). The result indicated that the Min_Max algorithm and SNXD is better than SNMD based on the time of first maximum independent set, running time of CPU and the number of coloring nodes.

KEYWORDS: Maximum independent set, Heuristic algorithm, adjacent nodes, Independent set.

I. INTRODUCTION

A coloring of graph G = (V,E) with vertex set V and edges set E is a mapping c: V → F, where F is a finite set. The elements of F will be referred to as the colors; so that colloquially, the definition basically implies that adjacent vertices cannot be assigned that same color; [AS99]. When a coloring exists, and |c(V)| =k, c is said to be a k-coloring of G. notice that a k-coloring partitions V into k sets V1,...,V_k, such that no two vertices in that same set are adjacent. The chromatic number ,X(G) , is the smallest number of colors needed to color G, that is, the smallest k such that three exists a coloring c for G and |c(V)|=k.

The simplest heuristic methods are sequential coloring strategies, and example can be found in [GJ97]. This algorithm can easily be implemented so that its worst-case complexity is $O(n^2)$. One of the most successful deterministic heuristic is the recursive largest first (RLF) producer due to Leighton [Lag02], which is based on the idea of coloring as much as possible with on color before introducing another. Johnson, et al. [Joh74] developed a randomize version of this method called XRLF that improves the performance of the simple heuristic. Met heuristic such as simulated annealing [Lei79], tabu search [J+91], genetic algorithm [KGV83] and GRASP [GL97, Hol75], have also been applied to graph coloring, Hertz and de Werra [FR89] present result of Appling simulated annealing to graph coloring. There are many application of graph coloring such as, software engineering, assigning channels to radio station and register allocation in a computer.

Solving graph coloring problem based on the maximum independent set problem (MIS), which that of finding (MIS) is important for many applications such as computer science, engineering.[FR95].

Graph coloring is a well-known NP-hard problem [CHW87]; this is why a variety of heuristic approaches have been Min_Max to produce good colorings in a reasonable amount of time.

The purpose of present work is to introduce more efficient way to find the minimum number of color needed to color any type of graph by finding number of maximum independent sets of a graph with polynomial complexity.

The next section will talk about SNMD algorithm, and solve an example by SNMD in order to show how it works. Then, section 3 discusses the SNXD algorithm and solves an example by SNXD in order to show how it works. The section 4, discusses the Min_Max algorithm in details. In section 5, we present the experimental performance of the three algorithm and some comparisons between some factors like number of nodes on a graph with independence number, and number of color required at different densities (0.1,0.2,…0.9) finally, concluding remarks are made in section 6.

II. SELECT A NODE WITH MINIMUM DEGREE FIRST (SNMD)

The process of this approach is to select the node that have the minimum degree among all nodes of a graph, add it to approximated maximum independent set, and delete the selected node and its neighbors from the graph. The process is repeated until the degree of the remaining graph become zero;[Joh74].

Procedure SNMD:
Input: G (V, E)
Output: MIS
/* Global : int max_ind_set[],char visite_node[];
Get_ind_set() /* function to get the independent sets
{While degree_node(G) <>0 /* return zero if there are no nodes
{v:=select_node(G) /* find the node that has smallest degree , if there are more than one choose the first one.
G:=G-v-neighbors(v);
max_ind_set=max_ind_set U {v}
} }

The running time bound for SNMD is $O(n^2)$, in Fig.2 we apply this algorithm on a graph in Fig.1.
As a result in Fig.2 the procedure is explained in the following steps:
Step 1: choose the node that has minimum degree first and put it in max_ind_set[], as you see there is tow node has the same degree , so we choose node that has ascending order [node 2].
Step 2: delete the selected node and its neighbors from the graph.
Step 3 : after deleting the graph containing tow nodes [3,5], apply algorithm on this sub graph to select another node, after applying the algorithm the Maximum Independent Set is [2,3].
Step 4: delete the Maximum Independent Set nodes from the main graph, and apply the algorithm once more to find all coloring until the degree of the remaining graph becomes zero.
When the algorithm finish, we will obtain on four colors for the entire main graph:
1: [2,3] MIS.
2: [1].
3: [4].
4: [5].

III. Select a Node With Maximum Degree First (SNXD)

The process of this approach is to select the node that have the maximum degree among all nodes of a graph, add it to approximated maximum independent set, and delete the selected node and its neighbors from the graph. The process is repeated until the degree of the remaining graph become zero;[GJ97].

Procedure SNXD:
Input: G(V,E)
Output: MIS
/* Global: int max_ind_set[],char visite_node[];
Get_ind_set() /* function to get the independent sets
{
While degree_node(G) <>0 /* return zero if there are no nodes
{
v:=select_node(G) /* find the node that has largest degree , if there are more than one choose the first one.
G:=G-v-neighbors(v);
max_ind_set=max_ind_set U {v} }}

The running time bound for SNXD is $O(n^2)$, in Fig.3 we apply this algorithm on a graph in Fig.1.
As a result in Fig.3 the procedure is explained in the following steps:
Step 1: choose the node that has maximum degree first and put it in max_ind_set[], as you see there is one node [1].
Step 2: delete the selected node and its neighbors from the graph.
Step 3: after deleting the graph containing three nodes zero nodes, the Maximum Independent Set is [1].
Step 4: delete the Maximum Independent Set nodes from the main graph, and apply the algorithm once more to find all coloring until the degree of the remaining graph becomes zero.
When the algorithm finish, we will obtain on three colors for all the main graph:
1: [1] MIS.
2: [4,3].
3: [2,5].

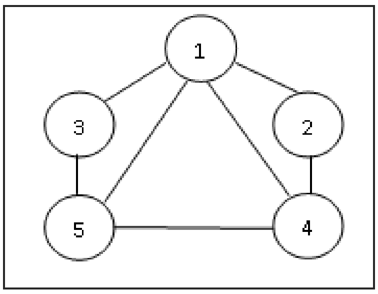


Fig.1. G=(V,E)

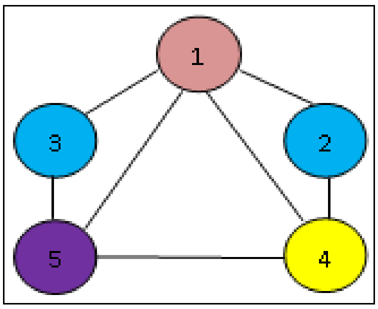


Fig.2. SNMD Coloring Graph

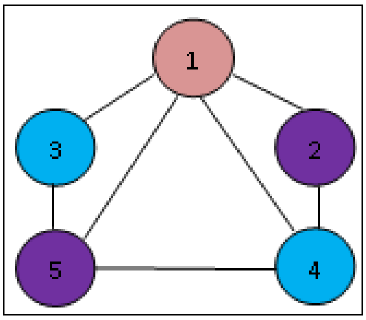


Fig.3. SNXD Coloring Graph

IV.SELECT A NODE WITH MINIMUM AND MAXIMUM DEGREE CONSECUTIVELY (MIN_MAX)

The main function of this algorithm is to find the maximum independent set of any graph, which might be used, among many other services in the graph coloring. If we want to find all Independent Set's, after finding the first MIS, we eliminate the nodes that are in MIS and their edges from the graph, and call the procedure again.

In Fig.4 we apply this algorithm on a graph in Fig.1 and the idea of the algorithm is as the following:

Step 1: Select the node that has minimum degree first and put it in max_ind_set[], as you see there is tow node has the same degree , so we select node that has ascending order [node 2].

Step 2: delete the selected node and its neighbors from the graph.

Step 3 : after deleting the graph containing tow nodes [3,5], apply algorithm on this sub graph to select another node, after applying the algorithm the Maximum Independent Set is [2,3].

Step 4: delete the Maximum Independent Set nodes from the main graph step 5: select the node that has maximum degree and put it in max_ind_set[], as you see there is three node has the same degree, so we select node that has ascending order.[node1].

Step 6: delete the selected node and its neighbors from the graph.

Step 7: after deleting the degree of the graph is zero so the second independent set (IS) is node [1].

Step 8: delete this node from the graph, and apply the algorithm by select the node that has minimum degree.[node 4]. And delete it with its neighbors. Third (IS) is [4].

Step 9: delete this node from the graph, and apply the algorithm by select the node that has maximum degree. [node 5]. And delete it with its neighbors. Fourth (IS) is [5].

When the algorithm finish, we will obtain on four colors for the entire main graph:

1: [2,3] MIS.
2: [1].
3: [4].
4: [5].

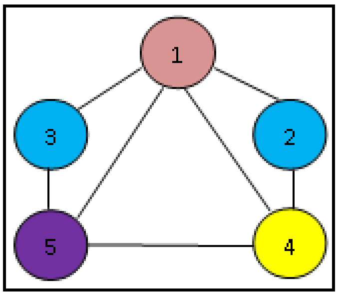


Fig.4 Min_Max Algorithm Graph Coloring

The following algorithm which summarized the above discussion consists of three procedures:

1. Degree ( ): return zero if there are no nodes in G.
2. select_min_node ( ): return vertex which has smallest degree.
3. Select_max_node ( ): return vertex which has largest degree.

Procedure Min_Max:
Input: G(V,E)
Output: MIS
/*Global: intmax_ind_set[], char visite_node[], select_min_node, select_max_node;
Get_ind_set() /* function to get the independent sets
{
While degree_node(G) <>0 /* return zero if there are no nodes
{
if g is odd /* variable to determine which procedure select (min or max)
{
v:=select_min_node(G) /* find the node that has largest degree , if there are more than one choose the first one
G:=G-v-neighbors(v);
Visite_node[V]='T'
max_ind_set=max_ind_set U {v}
}
Else {
v:=select_max_node (G) /* find the node that has largest degree , if there are more than one choose the first one
G:=G-v-neighbors(v);
Visite_node[V]='T'
max_ind_set=max_ind_set U {v}
}} }
/*Global: intmax_ind_set[], char visite_node[], select_min_node, select_max_node;
Get_ind_set() /* function to get the independent sets
{
While degree_node(G) <>0 /* return zero if there are no nodes
{

```
if g is odd /* variable to determine which procedure select (min or max)
        {
    v:=select_min_node( G ) /* find the node that has largest degree , if there are more than one choose the first one
  G:=G-v-neighbors(v);
  Visite_node[V]='T'
   max_ind_set=max_ind_set U {v}
 }
Else {
     v:=select_max_node ( G )  /* find the node that has largest degree , if there are more than one choose the first one
  G:=G-v-neighbors(v);
  Visite_node[V]='T'
   max_ind_set=max_ind_set U {v}   }} }
```

V. COMPUTATIONAL RESULTS

The three algorithms are programmed in Visual C++ 6.0, compiled and executed on an Intel Pentium 4 CPU 1400 MHZ, 128 RAM. under MS Win XP.

The computational results show the difference in performance between the algorithms in terms the CPU run time and the size of graph. The test graph is generated randomly by choosing the size (N) of the graph and the density (D). The size of the graph is varied between 20 up to 5000 at different densities (0.1, 0.2,…, 0.9). 10 graphs are generated and considered their average result.

In table (1 to 3) we have the computational results for testing the random graphs of three algorithms (SNMD, SNXD, Min_Max), each with different size 20,…,5000 and different densities 0.1,05,0.9.

The table's shows size of graphs size of the MIS's, CPU run time performance for finding the MIS.

The variation of CPU time between the three algorithm is shown in (figure 1.1 to 1.9) at different densities (0.1,…, 0.9) respectively. In fact the CPU time form density 0.1,..,0.4 is smaller in Min_Max algorithm, SNXD at node 3000 than SNMD, but when density is 0.5,..,0.9 the SNMD is smaller time.

In figure (2.1, 2,2 and 2.3) shows the variation between the MIS and size of graph and different densities 0.1, …, 0.9, as you see the number of nodes in MIS in Min_Max algorithm and SNMX is equal, but in SNXD the number of nodes in MIS is smaller.

In figure (3.1,…, 3.3) shows the variation time for MIS and the graph size with different densities 0.1,..,0.9 ,the SNXD and Min_Max algorithm is more efficient form SNMD to find the MIS with less time.

In figure (4.1,4.5 and 4.9) shows the relation between the graph size and number of colors with different densities 0.1,…,0.9. The number of colors in Min_Max algorithm and SNXD is less small than SNMD.

In figure (5.1 and 5.1.1) shows the relation between the CPU time and different density in Min_Max algorithm. Figure (5.2 and 5.2.1) shows the relation between the CPU time and different density in SNMD and figure (5.3 and 5.3.1) shows the relation between the CPU time and different density in SNXD. In fact the CPU time form density 0.1,..,0.4 is smaller in Min_Max algorithm, SNXD at node 3000 than SNMD, but when density is 0.5,..,0.9 the SNMD is smaller time.

In figure (6.1,6.2,6.3,6.4) shows there relation between density and the time for MIS, number of color, CPU time and MIS in Min_Max algorithm at graph size 5000 with different densities 0.1,…,0.9 .

In figure (7.1, 7.2, 7.3, 7.4) shows there relation between density and the time for MIS, number of color, CPU time and MIS SNMD algorithm at graph size 5000 with different densities 0.1,…,0.9 . As mentioned the time is decrease when the density is increase, the MIS is decrease and the number of coloring is increase, but the CPU time is increase and decrease depends on the density. This result is continuously rising by increasing the size of the graph.

As mentioned the time is decrease when the density is increase, the MIS is decrease and the number of coloring is increase, but the CPU time is increase and decrease depends on the density, as also in figure (8.1, 8.2, 8.3, and 8.4) for SNXD.

MIS and the minimum number of colors required to color a graph is at small density in general.

The time needed to find the MIS is very small relatively at 0.8 or 0.9 densities.

For further researches plans to compare the Min_Max algorithm with other exact algorithms are suggested. Plans are also recommended to verify that the selection node is always better when it has the highest degree.

VI. CONCLUSION

In this paper we introduce an approximate algorithm to color graph based on MIS finding. We have compared the Min_Max algorithm, SNMD, SNXD. Experimentation proved that that the Min_Max algorithm and SNXD is more efficient than the SNMD algorithm in terms of the number of nodes in a graph and the time complexity. The worst case for the MIS and the minimum number of colors required to color a graph is at small density in general.

The time needed to find the MIS is very small relatively at 0.8 or 0.9 densities.

For further researches plans to compare the Min_Max algorithm with other exact algorithms are suggested.

Plans are also recommended to verify that the selection node is always better when it has the highest degree.

REFERENCES

[AS99] **A. Al-Jaber, A. Sharieh -** Algorithms Based on Weight Factors for Maximum Independent Set. Jordan: University of Jordan, 1999.

[CHW87] **M. Chams, A. Hertz, D. de Werra -** Some Experiments with Simulated Annealing for Coloring Graphs, European Journal Of Operational Research, 32,(1987) 260-266.

[FR89] **T. A. R. E Feo, M. G. C Resende -** A Probabilistic Heuristic for Computationally Diffecult Set Covering Problem, Operations Research Letters,8,(1989) 67-71.

[FR95] **T. A. R. E Feo, M. G. C Resende -** Greedy Randomize Adaptive Search Procedures, Journal Of Global Optimization 2,(1995) 1-27

[GJ97] **M. R. Garey, D. S. Johnson -** Computers and Intractability: A Guide to the theory of NP-Completeness, W.H. Freeman and Co., Francisco, 1979.

[GL97] **F. Golver, M. Laguna -** Tabu Search, Kluwer Academic Publishers, 1997.

[Hol75] **J. H. Holland -** Adaptation in natural language and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

[Joh74] **D. S. Johnson –** "Worst-case behavior of graph coloring algorithm",Proceedings of the Fifth Southeastern Conference on Combinatorics, Graph Theory and Computing,Utilitas Mathematica Publishing Winnipeg, Canada (1974) 513-528.

[J+S91] **D. S. Johnson, C. A. Aragon, L. A. Mcgeoch, C. Schevon** - Optimization by Simulated Annealing: An Experimental Evaluation – Part II (Graph Coloring and Number Partitioning), Operations Research, 31,(1991) 378-406.

[KGV83] **S. Kirkpatrick, C. D Gelatt Jr., M. P. Vecchi** - Optimization by Simulated Annealing, Science, 220,(1983) 61-680.

[Lag02] **Manuel Laguna -** Tabu Search with Simple Ejection Chain for Coloring Graphs. Leeds School of Business, University of Colorado, Boulder, CO 80309-0419, USA, February 14,2002.

[Lei79] **Leighton, F.T.A.** - Graph Coloring Algorithm for large Scheduling Problems,J.Res. Nat. Bur. Standard, 84,(1979) 789-506.

Appendix

TABLE I Average Time in second Min_Max Algorithm, SNMD and SNXD Graph size = (20, 40,…..5000), Density = (0.1)

		Min_Max Algorithm				SNMD				SNXD			
Num of Node	Density	MIS	Time(sec) MIS	Time(sec) All Color	Num of color	MIS	Time(sec) MIS	Time(sec) All Color	Num of color	MIS	Time(sec) MIS	Time(sec) All Color	Num of color
20		11	0	0	3	11	0	0	3	10	0	0	3
40		19	0	0	5	19	0	0	5	12	0	0	5
60		24	0	0	6	24	0	0	6	15	0	0	6
80		27	0	0	7	27	0	0	8	17	0	0	7
100		32	0	0	8	32	0	0	8	18	0	0	8
120		35	0	0.01	10	35	0	0.01	11	20	0	0.01	9
200	0.1	45	0.01	0.05	14	45	0.01	0.02	13	33	0	0.01	13
400		56	0.07	0.25	22	56	0.08	0.31	23	36	0.02	0.1	22
800		68	0.29	1.692	38	68	0.29	1.932	40	39	0.05	0.65	36
1000		72	0.45	2.824	44	72	0.461	3.525	48	39	0.08	1.202	43
2000		87	1.763	18.546	81	87	1.772	22.731	87	48	0.351	8.482	74
3000		96	4.096	56.728	110	96	4.176	68.685	124	50	0.812	26.146	98
4000		96	6.69	122.369	139	96	6.75	150.908	153	53	1.462	59.632	126
5000		100	10.935	228.065	165	100	29.731	296.348	184	56	2.333	114.748	147

TABLE II Average Time in second Min_Max Algorithm, SNMD and SNXD Graph size = (20, 40,…..5000), Density = (0.5)

		Min_Max Algorithm				SNMD				SNXD			
Num of Node	Density	MIS	Time(sec) MIS	Time(sec) All Color	Num of color	MIS	Time(sec) MIS	Time(sec) All Color	Num of color	MIS	Time(sec) MIS	Time(sec) All Color	Num of color
20		5	0	0	8	5	0	0	8	3	0	0	8
40		9	0	0	13	9	0	0	13	3	0	0	10
60		9	0	0	16	9	0	0	18	4	0	0	13
80		10	0	0.01	22	10	0	0.01	24	4	0	0.01	24
100		10	0	0.01	26	10	0	0.01	29	5	0	0.01	24
120		10	0	0.01	30	10	0	0.01	31	5	0	0.01	28
200		11	0	0.04	43	11	0	0.04	50	5	0	0.04	42
400	0.5	13	0.01	0.24	79	13	0.01	0.24	82	7	0.01	0.221	79
800		15	0.05	1.591	145	15	0.05	1.592	154	8	0.03	1.482	140
1000		16	0.08	3.004	179	16	0.08	3.035	196	8	0.04	2.814	165
2000		16	0.31	20.77	322	16	0.31	20.489	347	8	0.15	19.78	302
3000		18	0.691	65.264	476	18	0.701	64.423	509	10	0.351	61.53	435
4000		18	1.182	147.08	593	18	1.202	142.856	628	10	0.641	143.5	555
5000		19	1.912	282.11	713	19	2.844	273.663	742	11	1.061	273.7	657

TABLE III Average Time in second Min_Max Algorithm, SNMD and SNXD Graph size = (20, 40,…..5000) Density = (0.9)

		Min_Max Algorithm				SNMD				SNXD			
Num of Node	Density	MIS	Time(sec)	Time(sec) All Color	Num of color	MIS	Time(sec)	Time(sec) All Color	Num of color	MIS	Time(sec)	Time(sec) All Color	Num of color
20		3	0	0	15	3	0	0	15	1	0	0	4
40		3	0	0	25	3	0	0	25	2	0	0	22
60		3	0	0	36	3	0	0	36	2	0	0	31
80		4	0	0	48	4	0	0.01	47	2	0	0.01	40
100		4	0	0.01	55	4	0	0.011	57	2	0	0.017	49
120		4	0	0.01	67	4	0	0.02	69	2	0	0.02	59
200	0.9	4	0	0.03	101	4	0	0.04	102	2	0	0.048	100
400		5	0	0.22	185	5	0	0.161	185	2	0.01	0.351	186
800		5	0.01	1.453	339	5	0.01	1.092	362	3	0.01	2.284	341
1000		5	0.02	2.734	417	5	0.023	2.033	447	3	0.01	4.266	420
2000		6	0.06	19.327	782	6	0.065	13.9	806	3	0.05	30.06	780
3000		6	0.14	61.909	1148	6	0.19	44.544	1187	3	0.12	97.65	1138
4000		6	0.241	142.17	1464	6	0.254	100.354	1494	3	0.231	226.8	1462
5000		7	0.3	266.88	1760	7	0.313	189.333	1782	4	0.291	425.3	1777

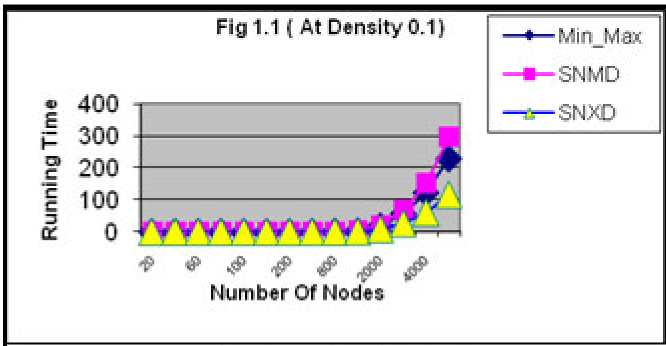


Fig 1.1 (At Density 0.1)
Min_Max
SNMD
SNXD
Running Time
400
300
200
100
0
20 60 100 200 800 2000 4000
Number Of Nodes

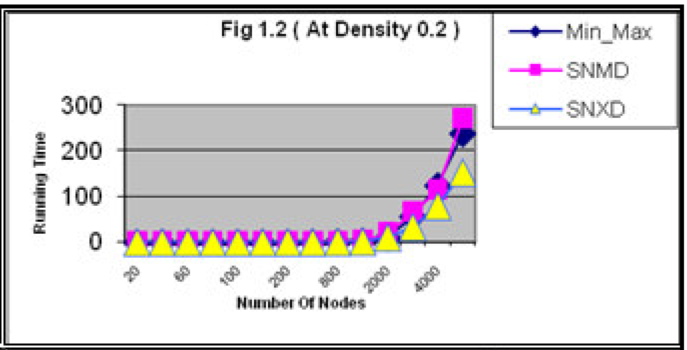


Fig 1.2 (At Density 0.2)
Min_Max
SNMD
SNXD
Running Time
300
200
100
0
20 60 100 200 800 2000 4000
Number Of Nodes

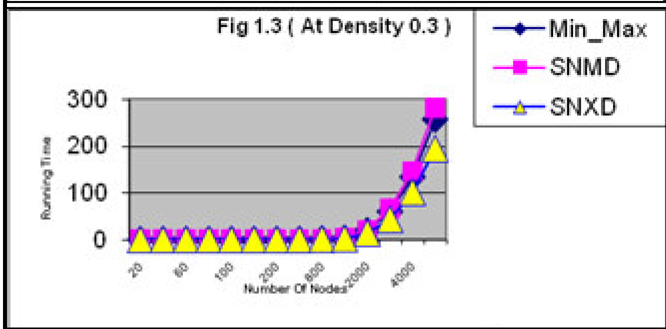


Fig 1.3 (At Density 0.3)
Min_Max
SNMD
SNXD
Running Time
300
200
100
0
20 60 100 200 800 2000 4000
Number Of Nodes

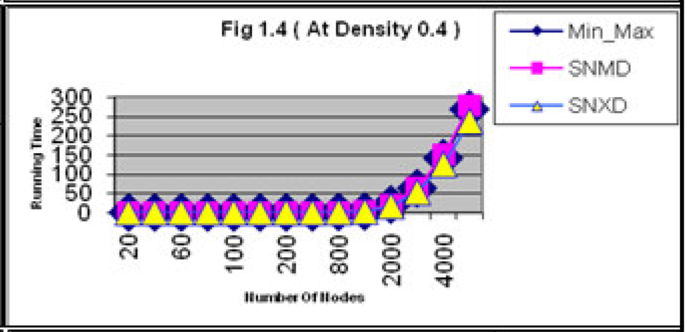


Fig 1.4 (At Density 0.4)
Min_Max
SNMD
SNXD
Running Time
300
250
200
150
100
50
0
20 60 100 200 800 2000 4000
Number Of Nodes

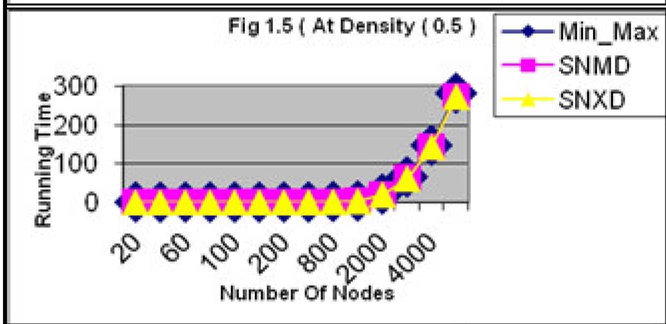


Fig 1.5 (At Density (0.5)
Min_Max
SNMD
SNXD
Running Time
300
200
100
0
20 60 100 200 800 2000 4000
Number Of Nodes

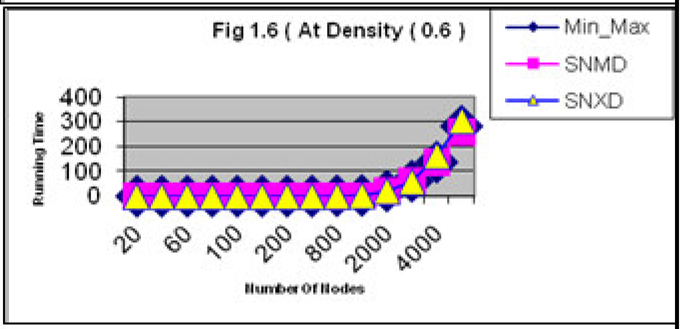


Fig 1.6 (At Density (0.6)
Min_Max
SNMD
SNXD
Running Time
400
300
200
100
0
20 60 100 200 800 2000 4000
Number Of Nodes

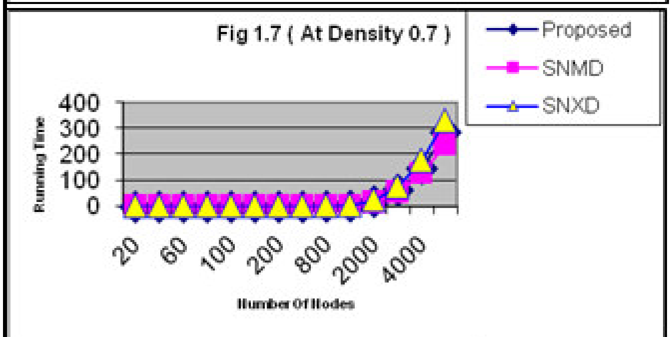


Fig 1.7 (At Density 0.7)
Proposed
SNMD
SNXD
Running Time
400
300
200
100
0
20 60 100 200 800 2000 4000
Number Of Nodes

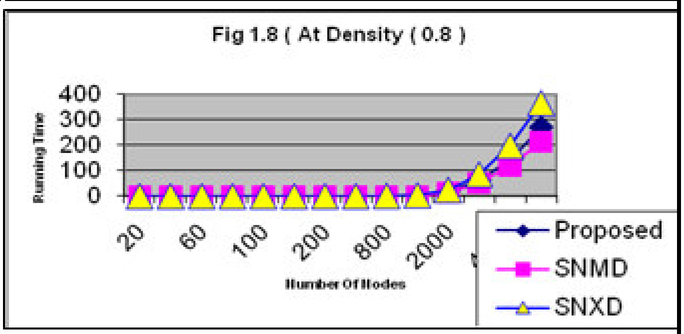


Fig 1.8 (At Density (0.8)
Proposed
SNMD
SNXD
Running Time
400
300
200
100
0
20 60 100 200 800 2000
Number Of Nodes

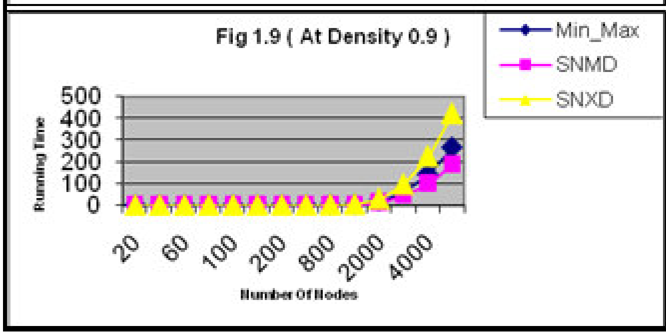


Fig 1.9 (At Density 0.9)
Min_Max
SNMD
SNXD
Running Time
500
400
300
200
100
0
20 60 100 200 800 2000 4000
Number Of Nodes

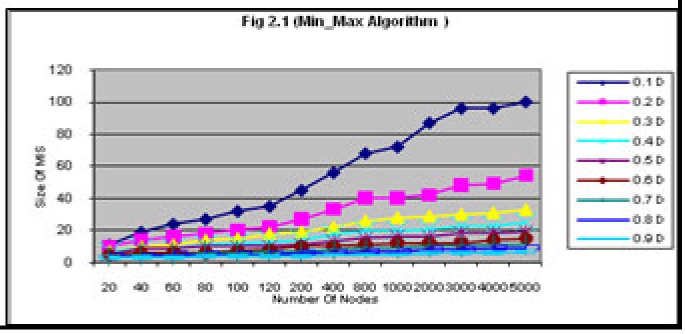


Fig 2.1 (Min_Max Algorithm)
0.1 D
0.2 D
0.3 D
0.4 D
0.5 D
0.6 D
0.7 D
0.8 D
0.9 D
Size Of MIS
120
100
80
60
40
20
0
20 40 60 80 100 120 200 400 800 1000 2000 3000 4000 5000
Number Of Nodes

Fig 2.2 (SNXD Algorithm)

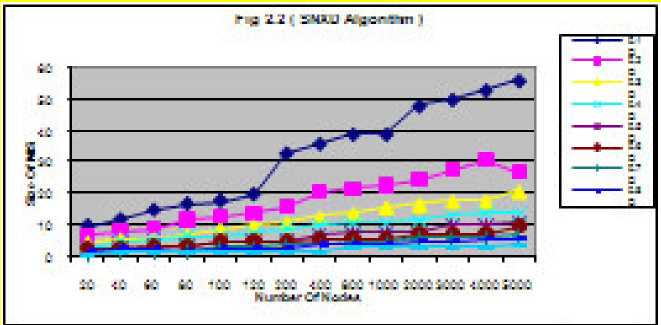


Fig 2.3 (SNMD Algorithm)

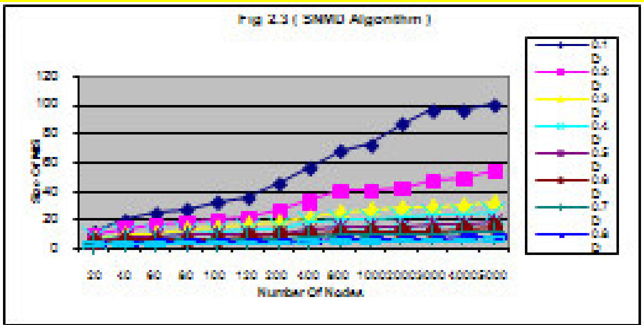


Fig 3.1 (Proposed Algorithm)

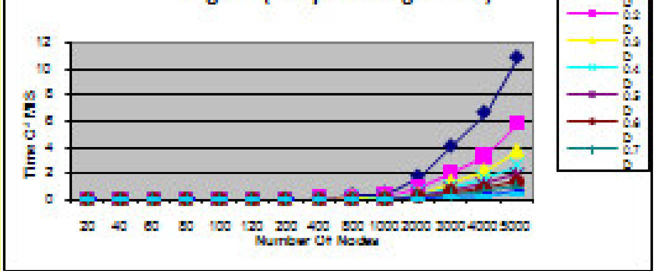


Fig 3.2 (SNMD Algorithm)

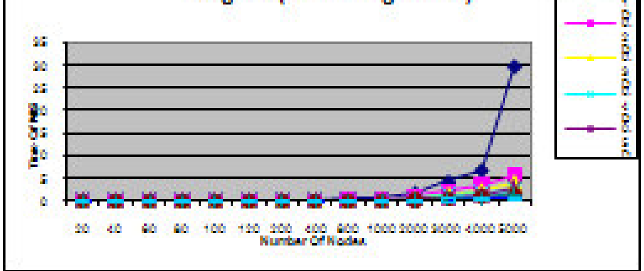


Fig 3.3 (SNXD Algorithm)

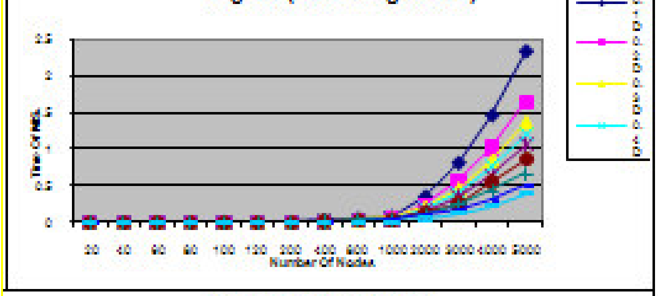


Fig 4.1 (At Density 0.1)

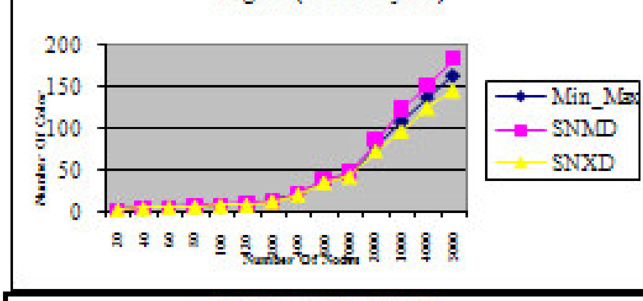


Fig 4.5 (At Density 0.5)

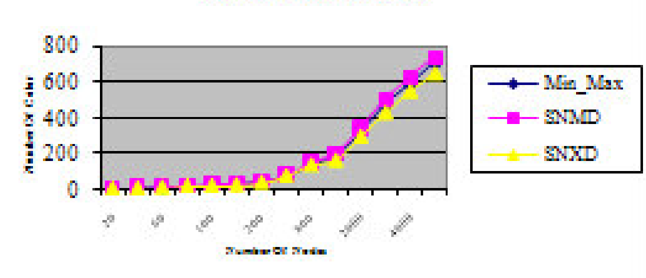


Fig 4.9 (At Density 0.9)

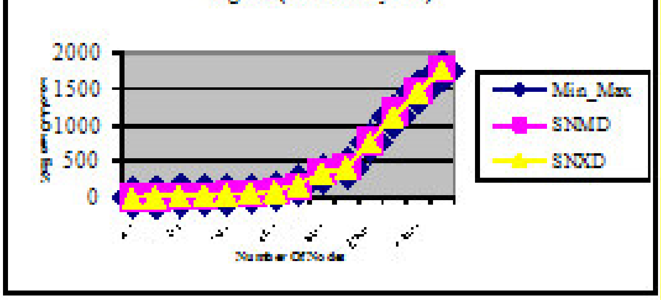


Fig 5.1 (Proposed Algorithm, Time And Density)

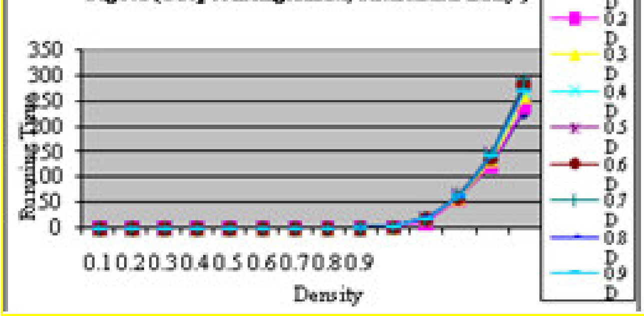
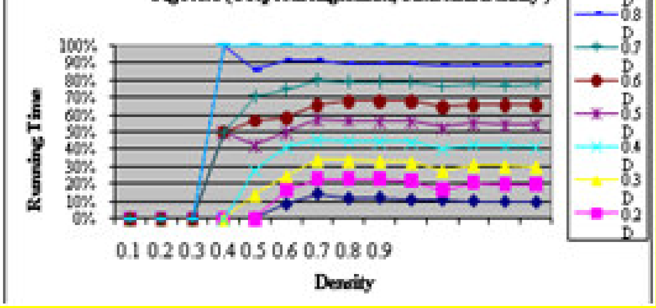


Fig 5.1.1 (Proposed Algorithm, Time And Density)



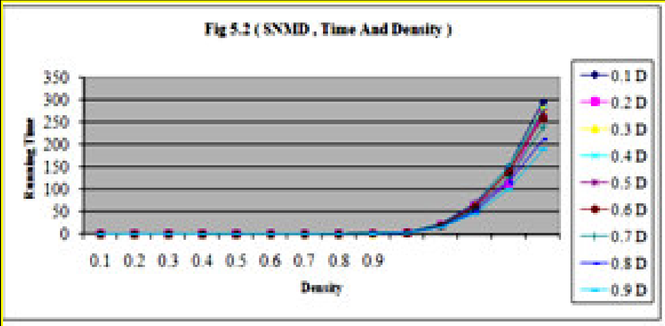


Fig 5.2 (SNMD , Time And Density)
Running Time
Density
0.1 D
0.2 D
0.3 D
0.4 D
0.5 D
0.6 D
0.7 D
0.8 D
0.9 D

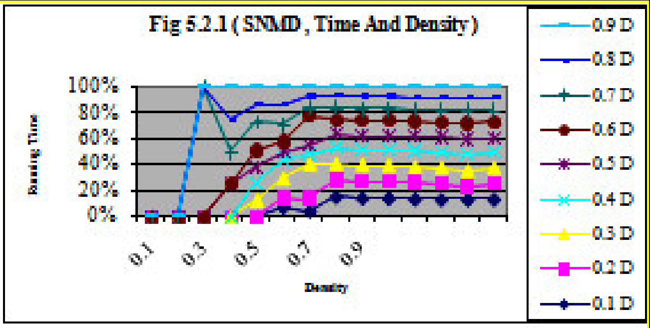


Fig 5.2.1 (SNMD , Time And Density)
Running Time
Density
0.9 D
0.8 D
0.7 D
0.6 D
0.5 D
0.4 D
0.3 D
0.2 D
0.1 D

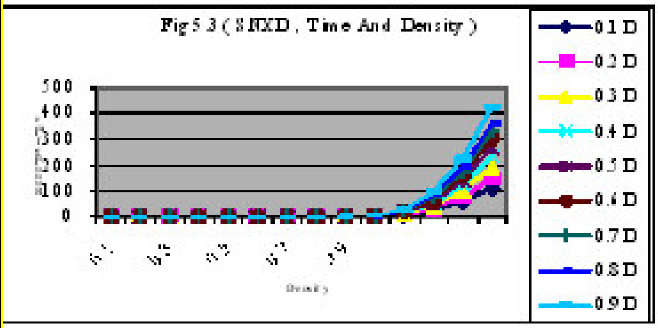


Fig 5.3 (SNXD , Time And Density)
0.1 D
0.2 D
0.3 D
0.4 D
0.5 D
0.6 D
0.7 D
0.8 D
0.9 D

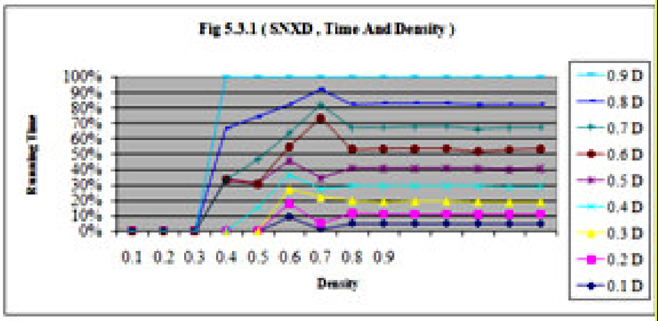


Fig 5.3.1 (SNXD , Time And Density)
Running Time
Density
0.9 D
0.8 D
0.7 D
0.6 D
0.5 D
0.4 D
0.3 D
0.2 D
0.1 D

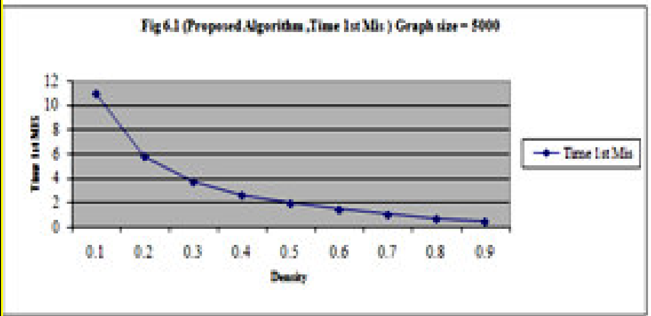


Fig 6.1 (Proposed Algorithm ,Time 1st Mis) Graph size = 5000
Time 1st MIS
Density
Time 1st Mis

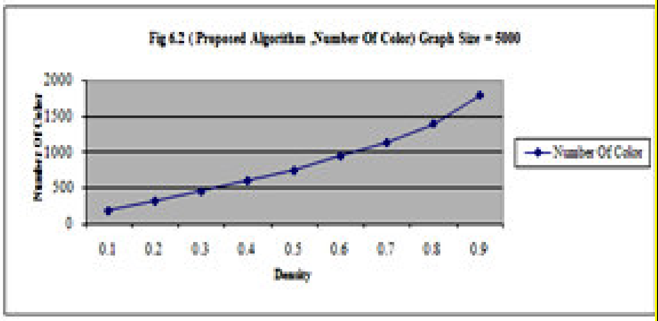


Fig 6.2 (Proposed Algorithm ,Number Of Color) Graph Size = 5000
Number Of Color
Density
Number Of Color

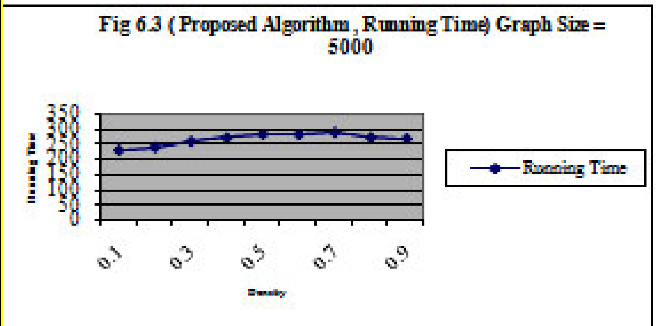


Fig 6.3 (Proposed Algorithm , Running Time) Graph Size = 5000
Density
Running Time

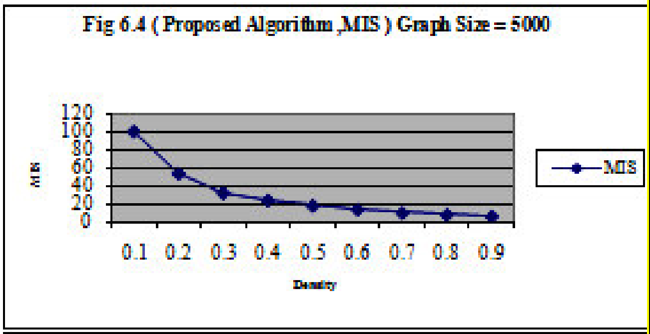


Fig 6.4 (Proposed Algorithm ,MIS) Graph Size = 5000
MIS
Density
MIS

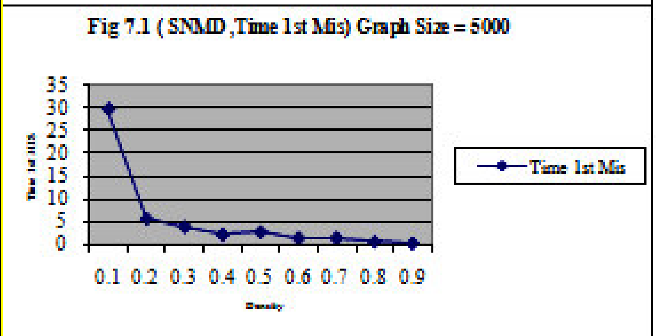


Fig 7.1 (SNMD ,Time 1st Mis) Graph Size = 5000
Density
Time 1st Mis

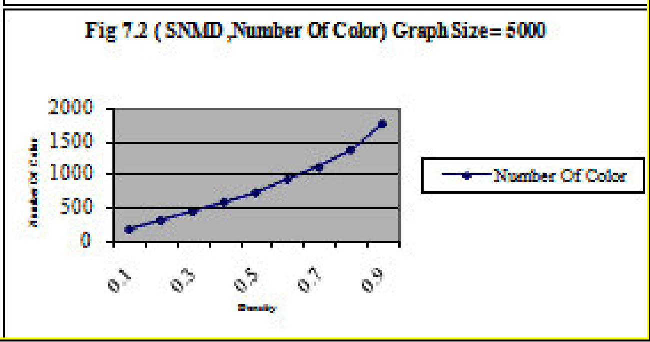


Fig 7.2 (SNMD ,Number Of Color) Graph Size= 5000
Density
Number Of Color

Fig 7.3 (SNMD ,Running Time) Graph Size = 5000

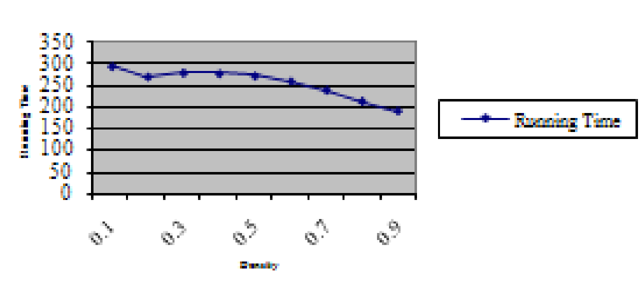


Fig 7.4 (SNMD ,MIS) Graph Size = 5000

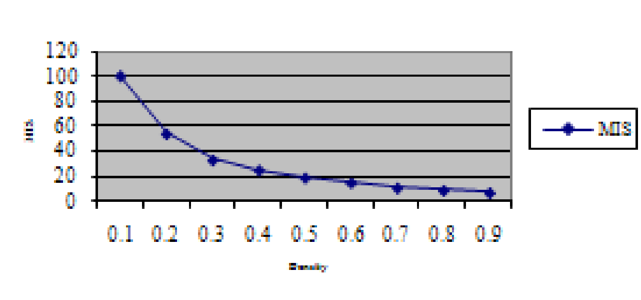


Fig 8.1 (SNXD ,Time 1st Mis) Graph Size = 5000

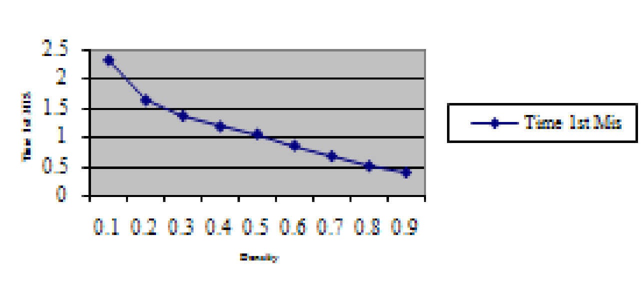


Fig 8.2 (SNXD , Number Of Color) Graph Size = 5000

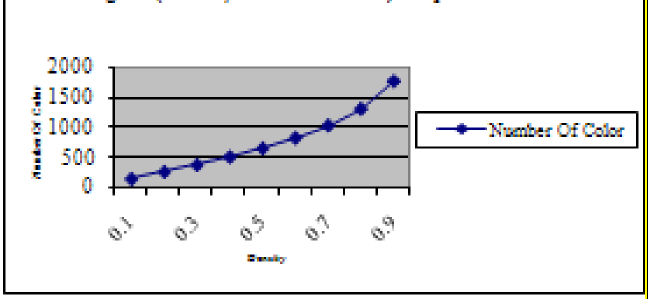


Fig 8.3 (SNXD , Running Time) Graph Size= 5000

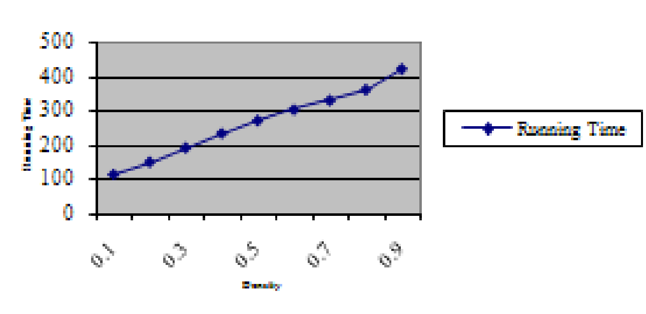


Fig 8.4 (SNXD ,MIS) Graph Size = 5000