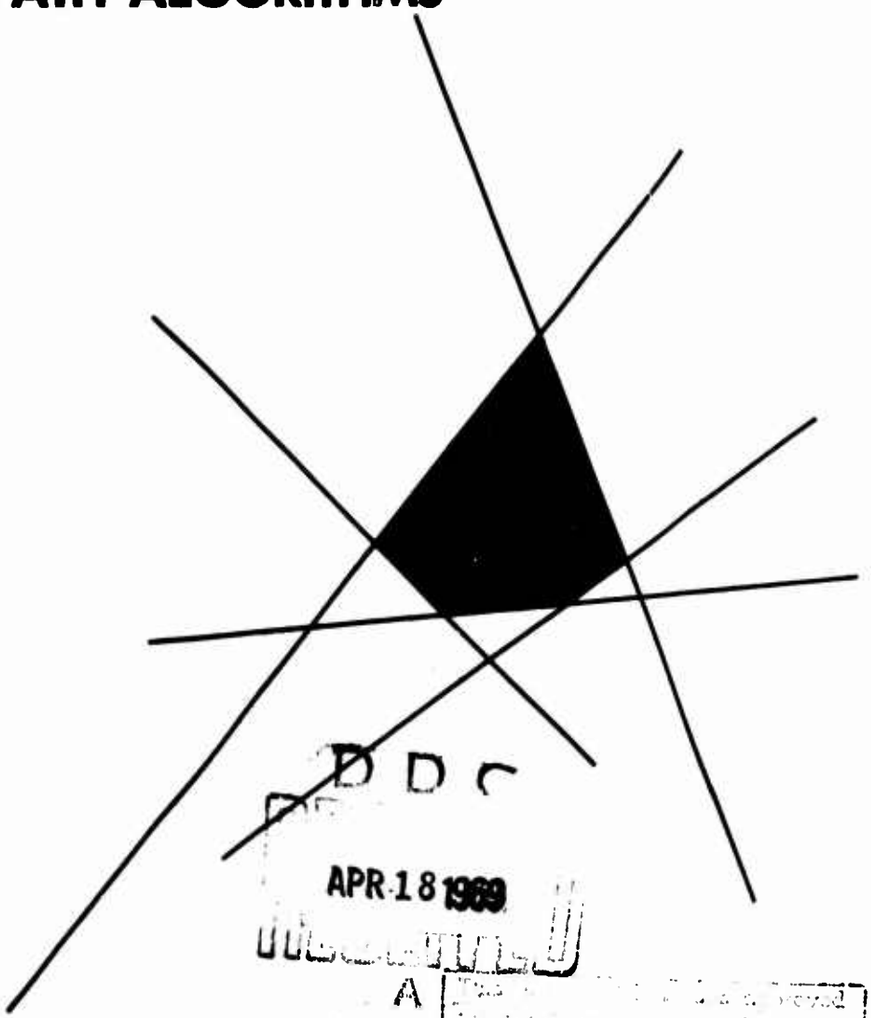
ORC 68-25
NOVEMBER 1968

A COMPARATIVE INVESTIGATION OF THE COMPUTATIONAL EFFICIENCY OF SHORTEST PATH ALGORITHMS

by
LEWIS E. HITCHNER



OPERATIONS RESEARCH CENTER

UNIVERSITY OF CALIFORNIA • BERKELEY

27

A COMPARATIVE INVESTIGATION OF THE COMPUTATIONAL EFFICIENCY OF SHORTEST PATH ALGORITHMS

by

Lewis E. Hitchner
Operations Research Center
University of California, Berkeley

November 1968 ORC 68-25

This research has been partially supported by the Office of Naval Research under Contract Nonr-222(83) and the National Science Foundation under Grant GP-8695 with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.

ABSTRACT

Efficiency of shortest path algorithms is a function of various network parameters. This paper reports the results of an investigation of five algorithms for finding the shortest path from a root node to all other nodes for different network structures. Parameters considered are number of nodes, number of links, range of data (i.e., arc lengths), and shape (if applicable). It is hoped that the results will suggest to potential users of these algorithms which one is best suited for their problem.

A COMPARATIVE INVESTIGATION OF THE COMPUTATIONAL EFFICIENCY OF SHORTEST PATH ALGORITHMS

by

Lewis E. Hitchner

INTRODUCTION

Research and planning in transportation, communication and other fields often involve examination of routes through large networks. In this paper, a network is considered to be a collection of nodes joined by directed links or arcs in which each arc has a nonnegative arc length associated with it. (This definition can be extended to include undirected arcs by substituting two parallel, oppositely directed arcs of equal length for each undirected arc.) The nodes could represent street intersections, telephone switchboards, or points in some state space. The arcs could stand for highways, telephone lines, or any abstract connections. Arc lengths might be time, distance, cost, etc.

The problem considered is that of finding the shortest path from a root or home node to all other nodes in the network. (This is equivalent to the problem of finding the shortest path to the root node from all other nodes.) A path is a connected route through the network which obeys the direction on each arc. Thus, what is called a path here is what network theory texts call a chain.[1] The shortest path is that path (not necessarily unique) which minimizes the sum of its arc lengths.

Since any reasonably large problem requires thousands of operations, these problems are naturally suited for solution by a digital computer. Therefore, the five algorithms examined were each programmed in FORTRAN in the most efficient method I could determine and were run on the CDC 6400 Computer at the University of California Computation Center. Conclusions are based on the results of these programs.

---

[1] Ford and Fulkerson, FLOWS IN NETWORKS, Princeton, p. 3, (1962).

METHOD OF ANALYSIS

The measure of efficiency used for comparing algorithms is the total number of basic operations required to find the shortest path. In the computer programs written, the operations counted are all add, subtract, and move (data transfer) operations not including those necessary for subscripting arrays or indexing loops. Thus, the operations counted are only those actually specified by each algorithm. This, hopefully, removes any bias due to the use of FORTRAN rather than an assembly or machine language. Since each operation requires very nearly the same amount of time in most computers, they are counted equally. (One algorithm also requires a relatively small number of divisions, but it requires so many more adds and subtracts than the other algorithms that counting a division equal to an add does not affect comparative results.)

Four network parameters which affect algorithm efficiency are considered. They are:

(1) Size = N = number of nodes.

(2) Density = ρ = number of links, L , as a proportion of the maximum possible number of links, i.e., $L/N(N - 1)$.

(3) Shape = dimensions of a network (if it can be conceived as a rectangle in a plane).

(4) Maximum arc length. This does not actually affect the network structure, but it is a factor which affects one of the algorithms.

For computational purposes, network structures are divided into three general classes. The first is called low density networks. This class only includes networks in which there are four arcs leading out of each node. (Thus, $L = 4N$, and density = $4/(N - 1)$, which is very low for large N .) It is divided into two sub-classes. The first, which is called low density-random,

consists of networks in which the four arcs out of each node are connected to randomly chosen nodes (with the restriction that at least one path from the root node to each of the other nodes exists). The second sub-class, low density-grid, consists of networks in which the nodes correspond to integer coordinates in the Euclidean plane within a rectangular boundary. Each node, except for those on the boundary, is linked to its four neighboring nodes. This class was chosen because of its similarity to a city street system. These are the only networks for which shape is meaningful. The shape parameters are length and width, measured in number of nodes. (Density is slightly less than as stated above since $L < 4N$.)

The other two classes are medium and high density networks. These are similar to the low density-random networks except that the number of arcs out of each node is also a random variable. The mean of this random variable is determined by N and the specified density, ρ . Medium density networks are those in the .25 to .70 range. High density networks are those from .70 to 1.0.

Three computer programs have been written for each of the three density ranges. The only major difference among the three is the method of data storage. The low density program uses an N by 8 matrix to store the arc length data. The first four columns of row i contain the nodes to which node i is connected. The second four columns contain the corresponding arc lengths. The medium and high density programs use an N by N matrix in which entry (i,j) is the length of the arc from node i to node j . If no such arc exists, the entry is infinity (10^6 was actually used for computational purposes). In each program the algorithms are programmed identically except that in the medium and high density programs, two algorithms are able to take slight advantage of the data storage method. Also, the medium density program examines each entry before it does any computations with it in order to skip those with infinite arc length.

Input data for each program specifies the number of nodes, which algorithms are to be run, the root node, the maximum and minimum bounds on arc lengths, and the

number of repetitions. For the low density program, the dimensions are also inputted. (N by zero specifies a low density-random network.) For the medium and high density programs, the density is inputted. The programs generate the specified network according to the input data. Arc lengths are random numbers uniformly distributed between the upper and lower arc length bounds. Each specified algorithm solves the shortest path problem for this network. New arc lengths are generated, and the algorithms solve the problem for the new network (with the same structure). When the desired number of repetitions is completed, the average number of operations for each algorithm is printed.

THE SHORTEST PATH ALGORITHMS

In each of the five algorithms examined, the method of solution involves assigning a tentative label to each node, which is the length of an attainable path from the root node to that node. Each label is initially set equal to infinity, and attempts are made iteratively to try to improve each label (i.e., decrease its value). When no more improvements can be made, the algorithm is done, and each label is the length of the shortest path to that node. A record can be kept of the preceding nodes in each shortest path and is used to obtain the actual route through the network.

The Ford Algorithm

Algorithm number 1 is a modification of a method first described by L. R. Ford in 1956[2] and elsewhere in later publications.[3] Basically, the label on node j is improved if the arc length from some other node, i, to node j plus the current label on node i is less than the current label on node j. Each label is examined in this way until no more improvements can be made. An improvement in this method is the use of a sequence list as described in a paper by C. Witzgall and attributed to the Bureau of Public Roads.[4] It is obvious that no improvements on a j label can be made by setting label j equal to label i plus the length of arc (i,j) unless label i has been improved. Thus, whenever a node label is changed, that node number is put on the sequence list (unless it is already on the list). Then only nodes directly linked to those nodes on the list are examined to see if their labels can be improved. After all the neighbors of a node on the list have been examined, that node is removed from the sequence list. When the list is finally empty, all labels are optimal.

---

[2] Ford, L. R., Jr., "Network Flow Theory," The RAND Corporation, P-923, (August 1956).

[3] Ford and Fulkerson, FLOWS IN NETWORKS, Princeton, pp. 130-133, (1962).

[4] Witzgall, C., "On Labeling Algorithm for Determining Shortest Paths in Networks," Boeing Scientific Research Laboratories, (unpublished as of June, 1968).

The other four algorithms are all modifications of a standard shortest path procedure first reported by Dijkstra.[5] This method is basically a dynamic programming algorithm. At each iteration one label becomes "permanent", and the next iteration makes use of the shortest paths already found. (Thus, only N iterations are required.) The four variations differ in their handling of "temporary" (i.e., not permanent) labels, but all adhere to the basic Dijkstra method.

Loubal Transportation Algorithm

The first of these four is a method described to us by P. Loubal and used at the Institute of Transportation and Traffic Engineering at the University of Calfironia, Berkeley. The actual description of the program can be found in the TRAFFIC ASSIGNMENT MANUAL[6] published by the Bureau of Public Roads. Essentially this follows the Dijkstra algorithm (referred to in the Manual as the Moore algorithm). It derives its efficiency from the manner in which it stores temporary labels. Rather than storing temporary label values in the storage locations for each node number, it stores node numbers in a table in which the row number of the table is the value of the label. This is, if the temporary label on node 207 is 16, say, the number 207 is stored in row 16. Thus, to find the lowest valued temporary label (which becomes the next permanent label), the program merely scans the table to find the next entry. The row number of that entry is the value of the label. Note that using this table requires commensurate arc lengths unless rounding is acceptable.

Also, it is possible to save space by using a "wrap-around" table which has as many rows as the number of units in the longest arc length plus 1. Thus, if arc lengths are very large (i.e., large relative to the units used to measure them),

---

[5] Dijkstra, E. W., "A Note on Two Problems in Connexion with Graphs," Numerische Mathematik, 1, pp. 269-271, (1959). See also Dreyfus, S. E., "An Appraisal of Some Shortest Path Algorithms," The RAND Corp., RM-5433-PR, (October 1967).

[6] U. S. Department of Commerce, Bureau of Public Roads, TRAFFIC ASSIGNMENT MANUAL, Chapter V, (June 1964).

the table will be very large and more searching of the table is involved. Another trick is the use of a counter for each row which tells how many node numbers are currently in that row. This facilitates the location of the next entry in the table.

Straight Dijkstra Algorithm

This is merely the regular Dijkstra algorithm as referenced above. No modifications have been made in any of its steps.

Dijkstra Ordered List Algorithm

Here the Dijkstra method is modified by storing the temporary labels in order of increasing magnitude. Another list is kept of the node numbers of each label in the list. Thus, the smallest temporary label is just the first node in the ordered label list. However, whenever a temporary label is updated (decreased) its position in the list must be changed. Its new location is found by the bisection method, and then it is placed in its new position and all higher valued temporary labels are moved back one slot in the list.

Dijkstra Threaded List Algorithm

This is similar to the ordered list method except that rather than actually physically ordering the labels in regard to magnitude, pointers are used to indicate the order of the labels. That is, each label has associated with it two pointers which give the node number of the label which precedes it and the node number of the label which follows it in the ordering of labels. Thus, the smallest label is the label on the follower of node zero. Also, when a label is updated, its new position in relation to the other labels is found by comparing its value to that of its predecessors. When the proper location in the list is found, the necessary pointers on the associated nodes are changed (which involves only six pointer changes). This procedure avoids having to relocate a large part of the list.

Note that the two modified Dijkstra methods involve keeping the labels in order during the whole process of the algorithm. This requires many changes to update the list of labels. The straight Dijkstra algorithm does not do this and, thus, does not require the extra work. However, at each iteration it must examine all the labels to find the smallest temporary label.

EMPIRICAL RESULTS

Low Density-Random

Test runs were made for all algorithms for N = 100, 200, 300 and 400. Since it became obvious after these four cases that the three Dijkstra methods were much worse than Ford or Loubal, only the latter two were tested for N = 500, 600, 700, 800, 900, 1000, 1250 and 1500. Arc lengths for all of these runs were between 1 and 100 with all lengths being integers.

Chart 1 shows all algorithms for N up to 400 and Ford and Loubal for N up to 1500. The Dijkstra methods seem to be growing exponentially and obviously are not as good as Ford or Loubal even for networks as small as 100 nodes. Ford and Loubal seem to grow very nearly linearly with an increase in network size. Ford is better up to about 400 or 500 nodes. From there on Loubal takes fewer operations. However, even at 1500 nodes Loubal is only about 15% better than Ford. (The irregularity of the Ford curve is due to a large variance in the number of operations for each run. All results in this paper are based on the average of 5 runs for each algorithm. For the Ford algorithm more runs would probably give smoother curves.)

Low Density-Grid

Charts 2, 3 and 4 display total number of operations versus number of nodes for low density-grid networks. Three ratios of width to length were tested, 1 to 4, 1 to 2 and 1 to 1. Here, again, it is obvious that the three Dijkstra methods are relatively very inefficient. Loubal is definitely the best method except, perhaps, for small networks of up to 400 nodes where Ford is about the same or a little better than Loubal.

Note the effect of making the networks more compact (i.e., more nearly square). Loubal and Dijkstra are very nearly constant for variation in dimensions. Dijkstra threaded and Dijkstra ordered improve about 30% to 40% if the width to length ratio

CHART 1

LOW DENSITY - RANDOM TOTAL NUMBER OF OPERATIONS

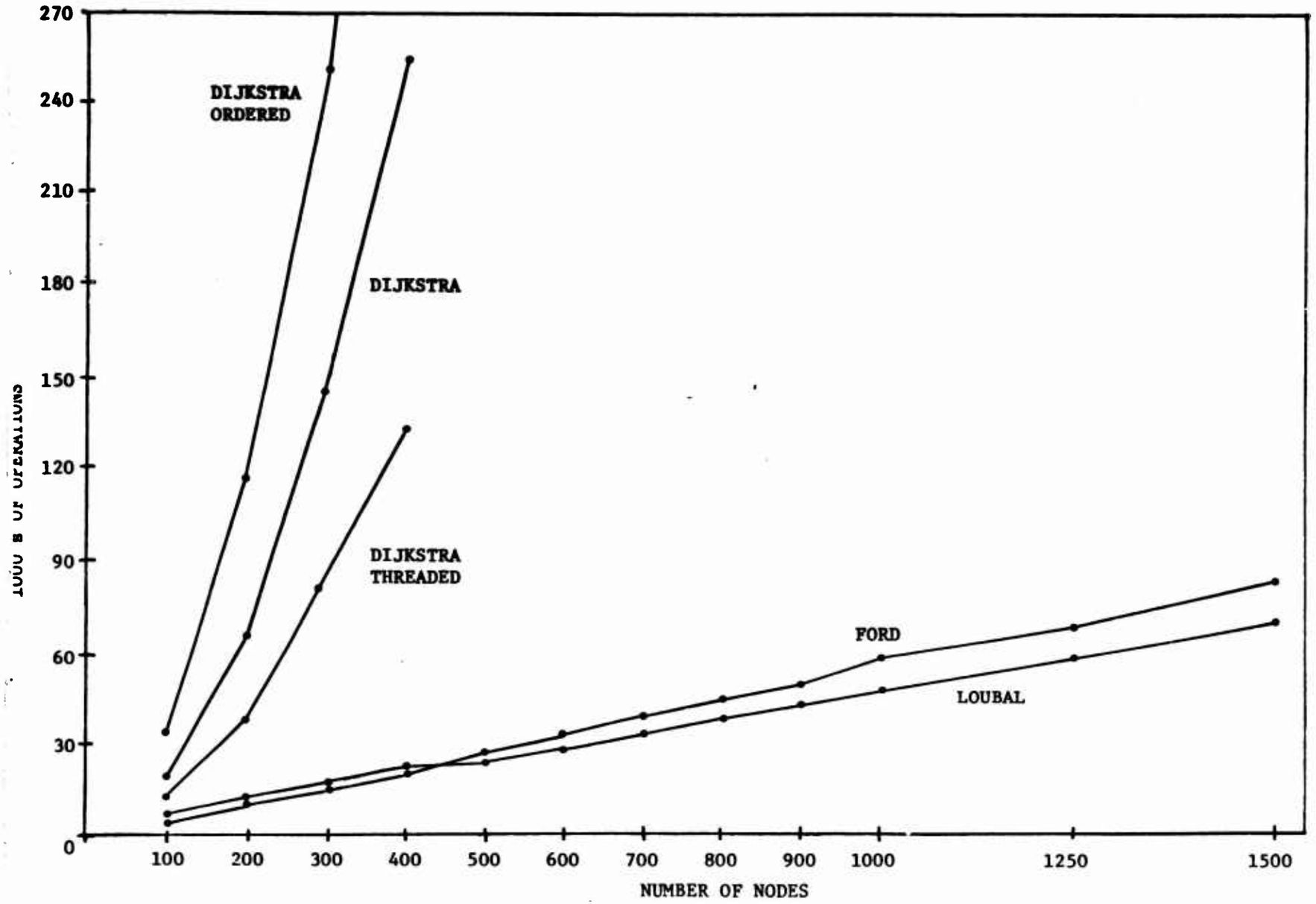


CHART 2

LOW DENSITY - GRID TOTAL NUMBER OF OPERATIONS WIDTH:LENGTH = 1:4

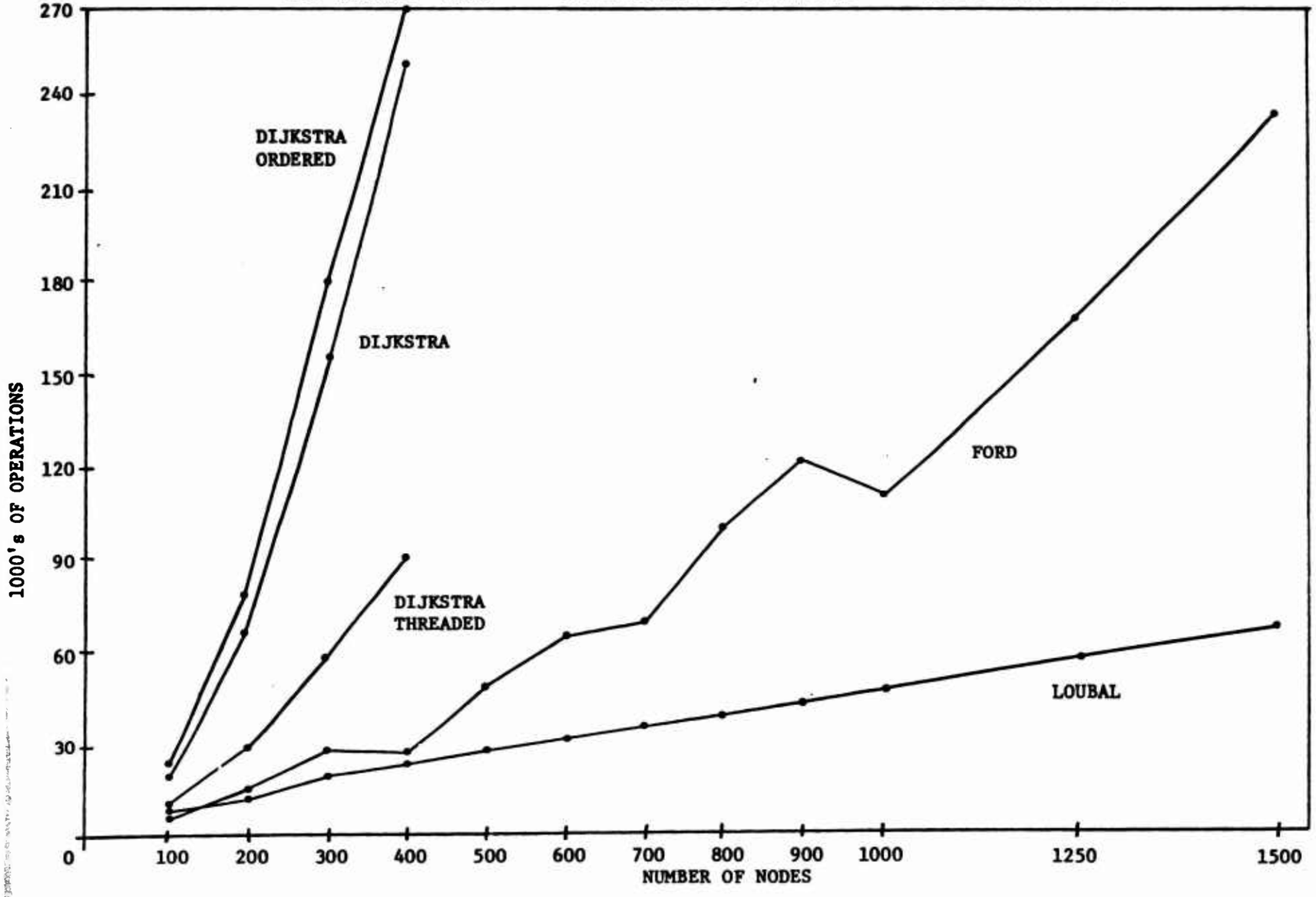


CHART 3

LOW DENSITY - GRID TOTAL NUMBER OF OPERATIONS WIDTH:LENGTH = 1:2

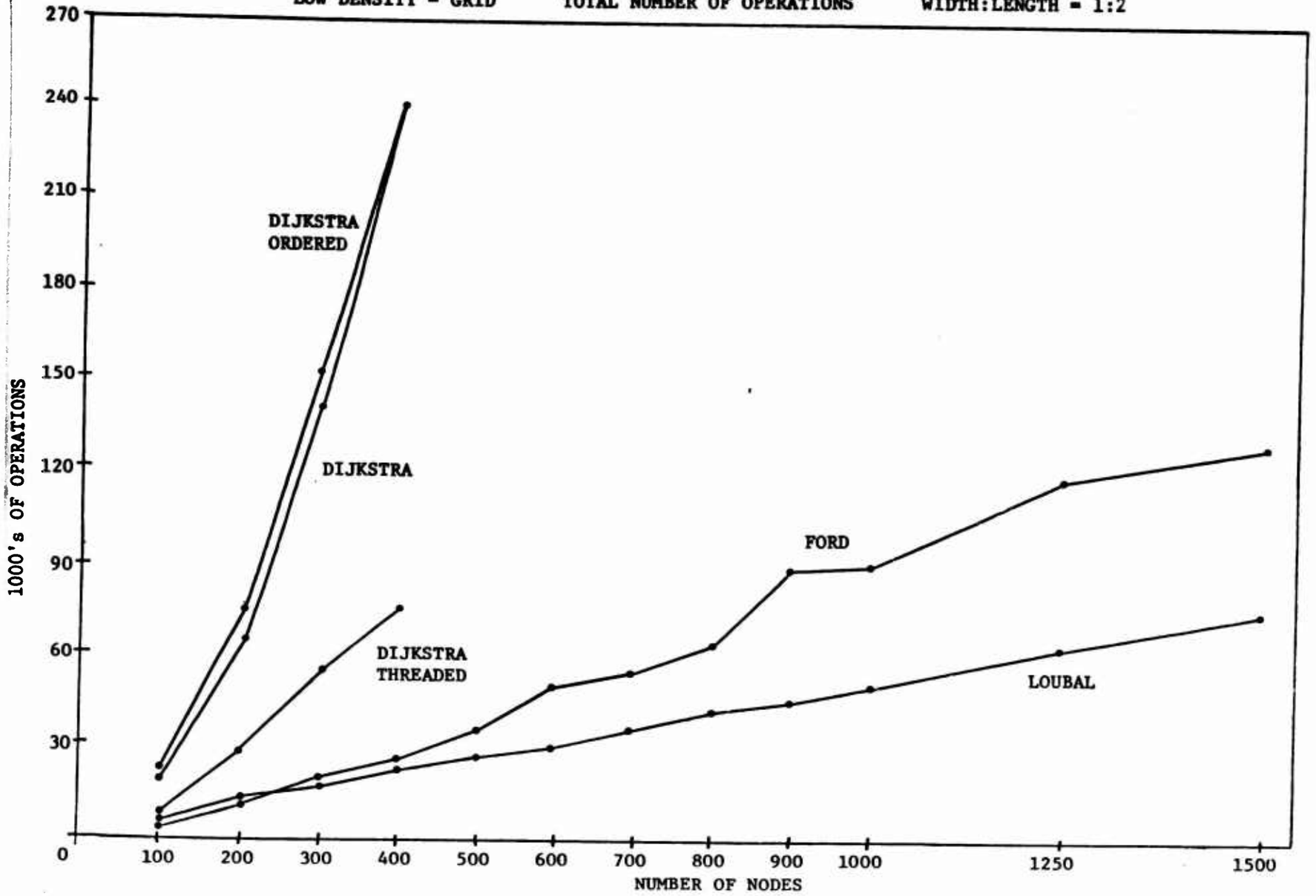
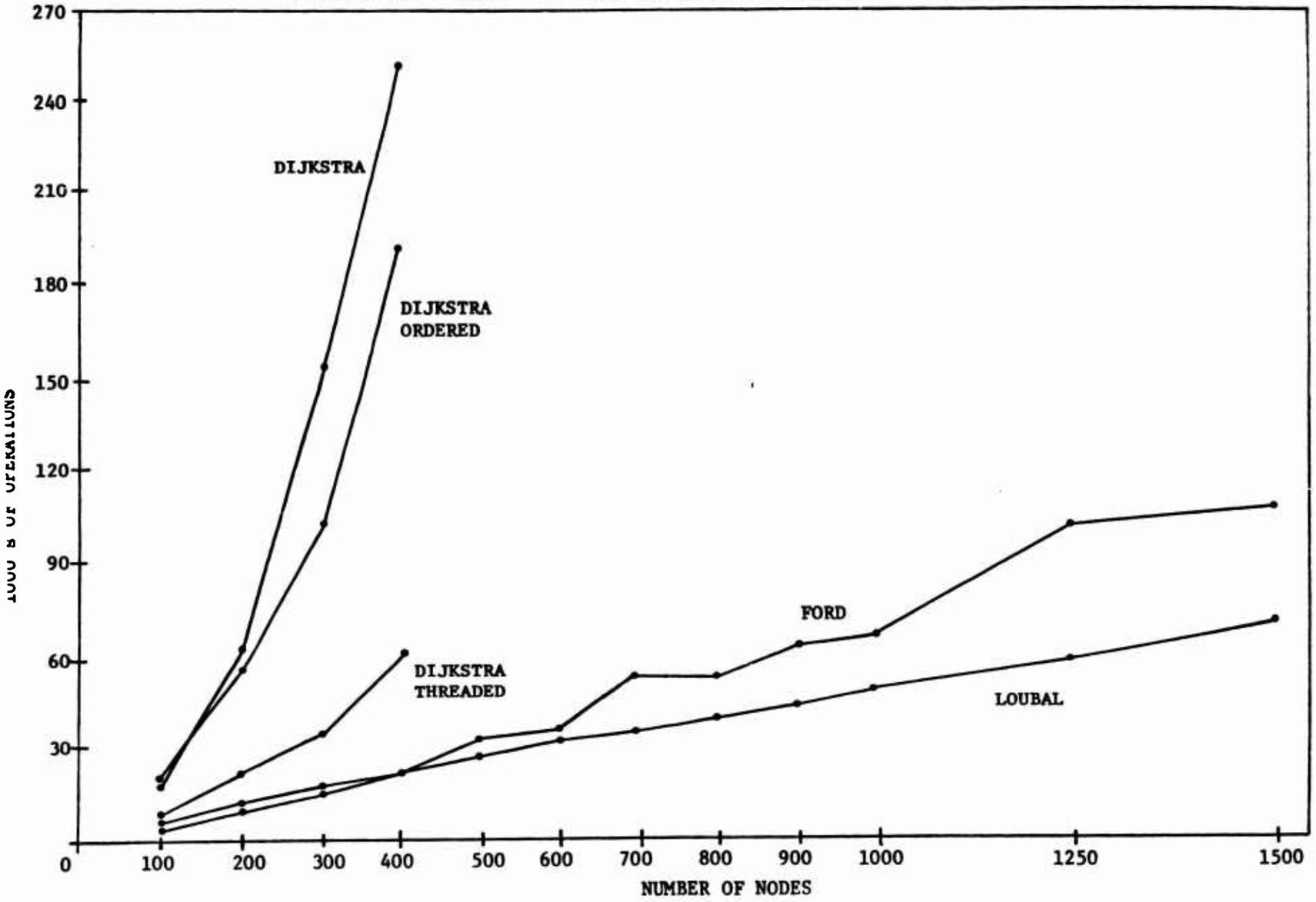


CHART 4

LOW DENSITY - GRID TOTAL NUMBER OF OPERATIONS WIDTH:LENGTH = 1:1



is changed from 1:4 to 1:1. Ford is affected even more so. The square network requires about 60% fewer operations than the 1:4 ratio network. Perhaps a nearly circular grid would make Ford even more efficient. It seems very doubtful, however, that it would be better than the Loubal algorithm.

It is worthwhile noting that for the low density-random networks Ford behaves as if the random network were square while Dijkstra ordered and Dijkstra threaded seem to behave as if the random network were very much elongated.

Since the Loubal algorithm is affected by the number of units in the data range, it is possible that the Ford algorithm may be more effecient if the data range is very large. Chart 5 shows the effect of maximum arc length ranges between 10 and 200 for four different network sizes. The effect is obviously linear. However, it seems that unless that data range is extremely large, Loubal will still be most efficient except for very small networks. Also, it is often possible to round off arc lengths so that larger units of measurement can be used.

Medium Density

Charts 6 and 7 show the results of all algorithms for densities of 25%, 40%, 55% and 70%. In this case, Dijkstra is obviously the best method. For the lower two densities Loubal is a fairly close second. However, the gap between Dijkstra and Loubal increases as the density increases. The other three methods are not even worth considering for these networks as they all require many more operations.

High Density

Charts 8 and 9 display results for 70%, 85% and 100% dense networks. Here, Dijkstra is still the best algorithm and the others are obviously out of contention. Note that Loubal becomes very much more inefficient as the network density approaches 100%.

As stated above, the difference between the high and medium density programs is that for medium density a check is made to see whether an arc length is infinite.

CHART 5

LOW DENSITY - GRID TOTAL NUMBER OF OPERATIONS LOUBAL ALGORITHM

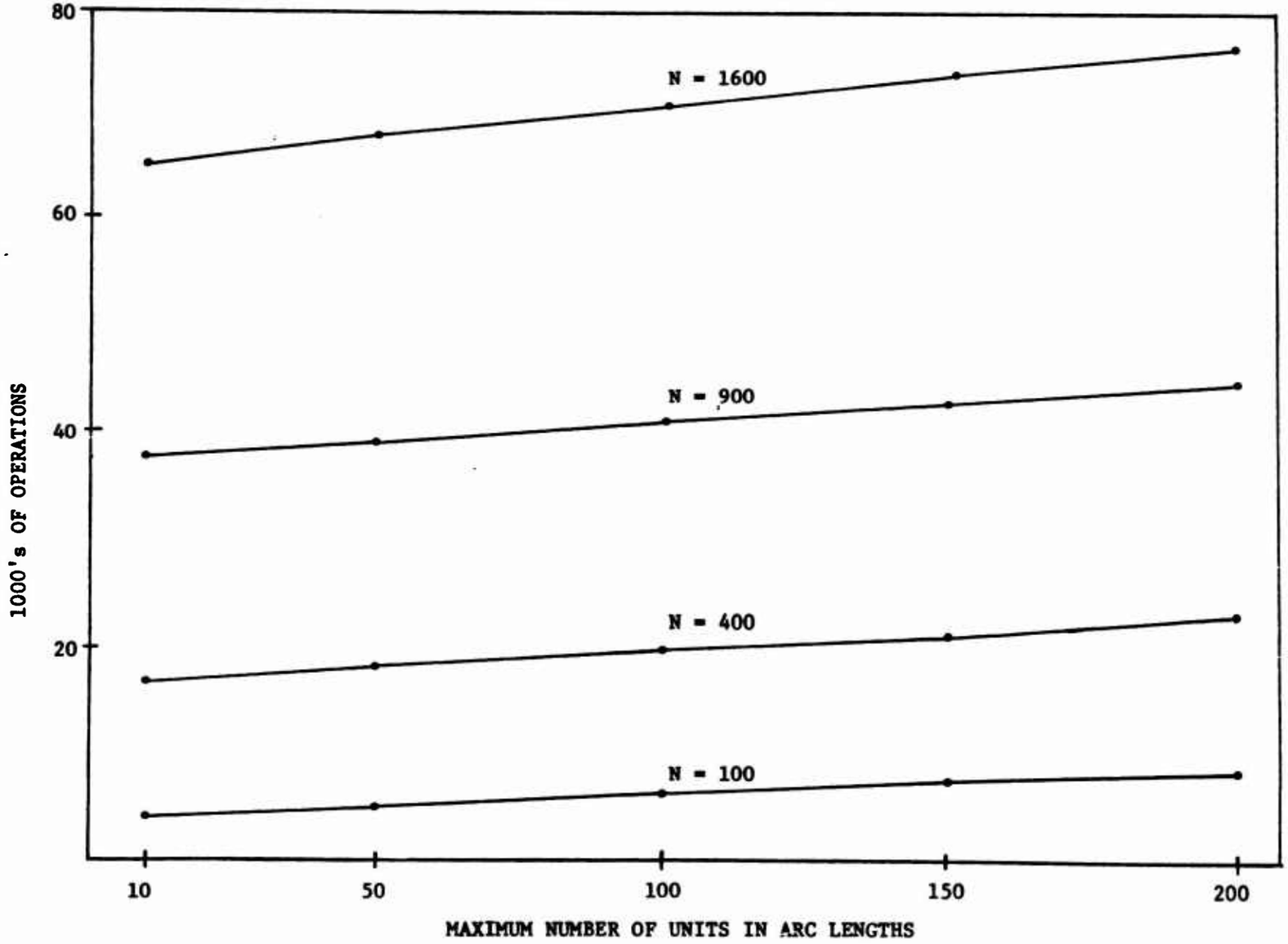


CHART 6

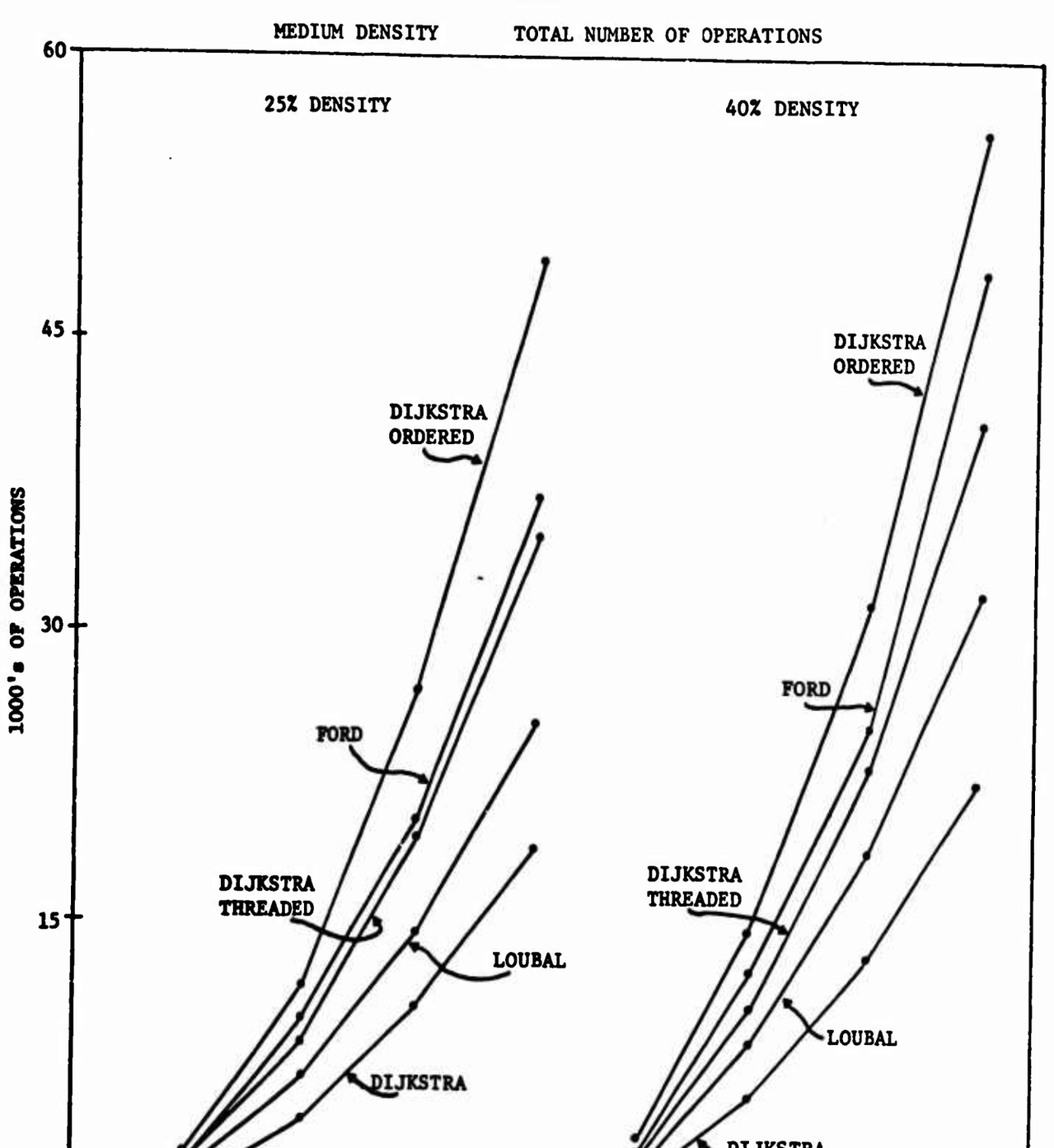

MEDIUM DENSITY
TOTAL NUMBER OF OPERATIONS
25% DENSITY
40% DENSITY
60
45
30
15
1000's OF OPERATIONS
DIJKSTRA ORDERED
FORD
DIJKSTRA THREADED
LOUBAL
DIJKSTRA
25
50
75
100
NUMBER OF NODES

CHART 7

MEDIUM DENSITY TOTAL NUMBER OF OPERATIONS

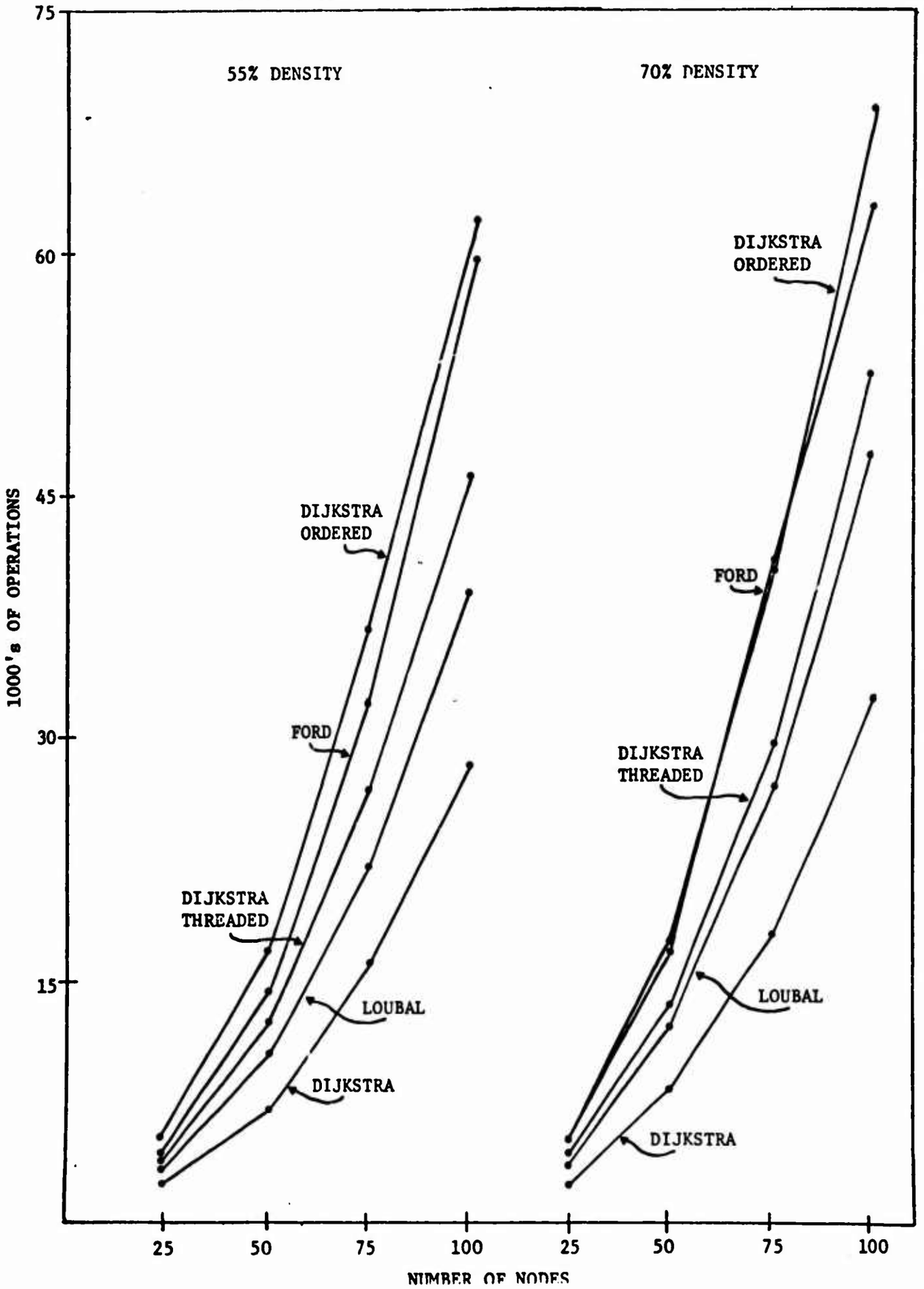


CHART 8

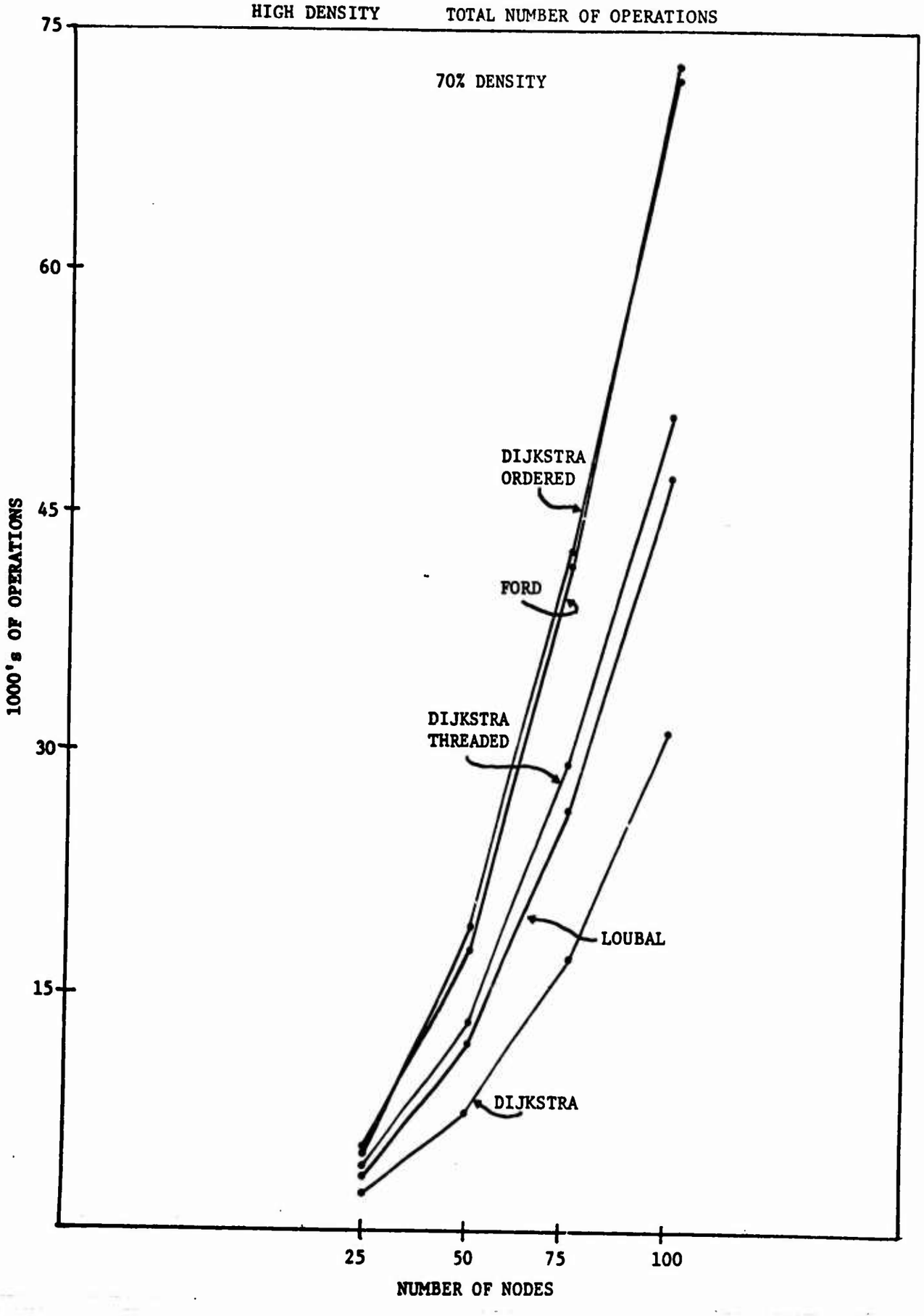
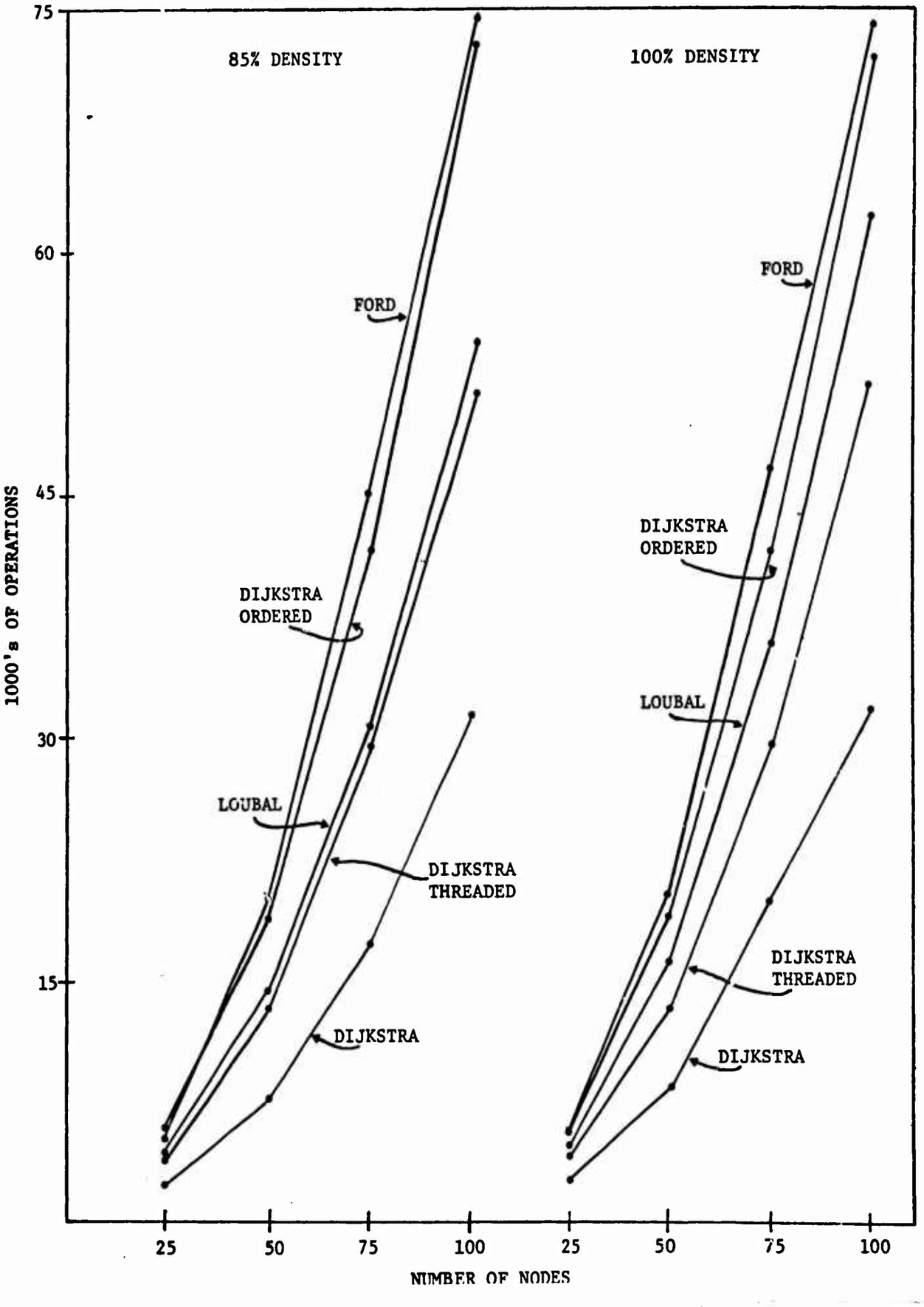

HIGH DENSITY
TOTAL NUMBER OF OPERATIONS
70% DENSITY
75
60
45
30
15
1000's OF OPERATIONS
DIJKSTRA
ORDERED
FORD
DIJKSTRA
THREADED
LOUBAL
DIJKSTRA
25
50
75
100
NUMBER OF NODES

CHART 9

HIGH DENSITY TOTAL NUMBER OF OPERATIONS



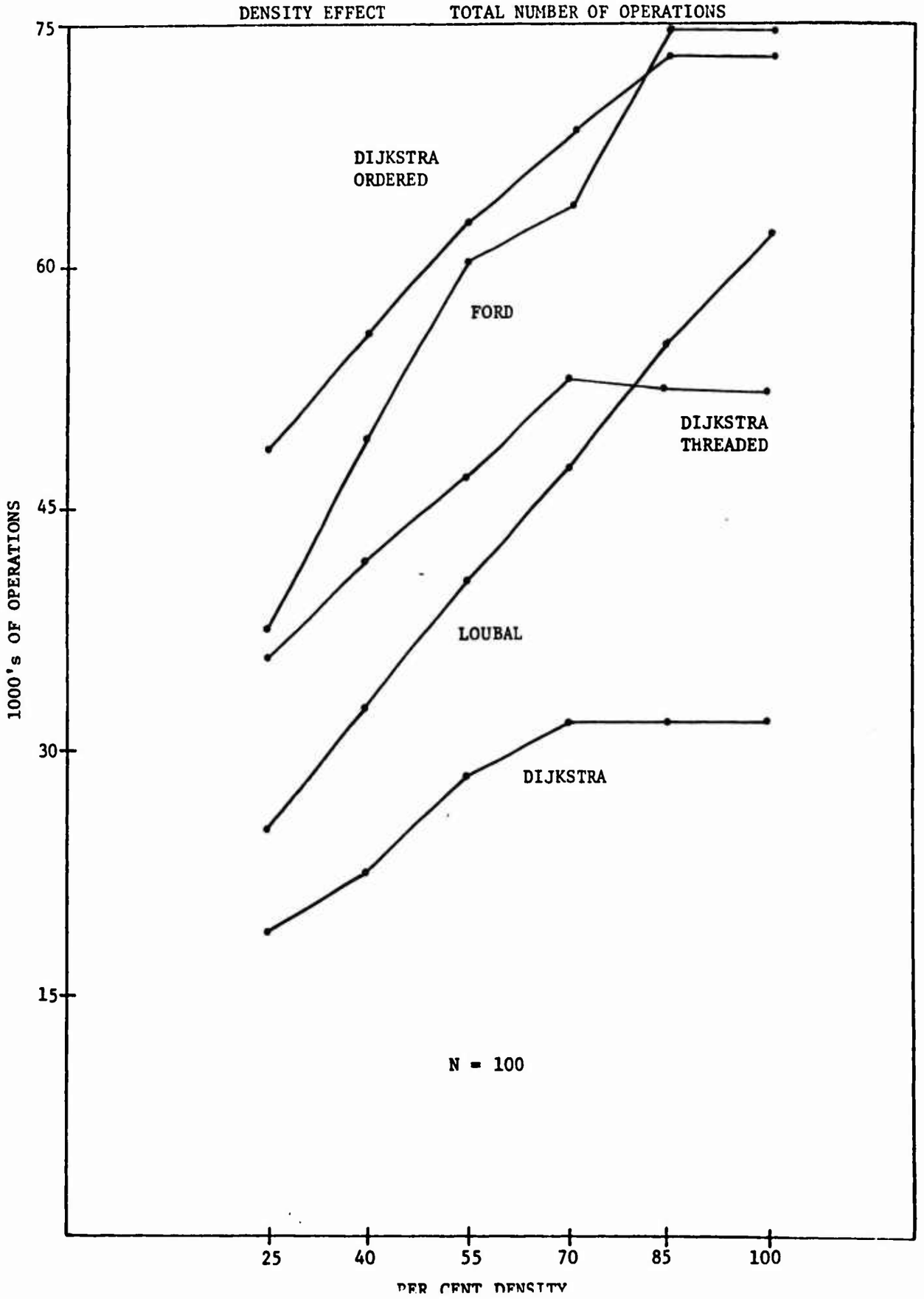
If it is, no computation is done which involves that arc length. For the high density program the amount of work to make such a check becomes more than the work saved by checking and, therefore, is not done. Results show that 70% density is, roughly, the break-even point for checking, though it varies somewhat with each algorithm.

Chart 10 shows the effect of density on each algorithm for a network of 100 nodes. All show a definite increase in number of operations for an increase in density. However, part of this effect is due to the arc length checking feature of the medium density program which reduces the number of operations in the lower density cases. Other runs of the high density program at medium densities show that Dijkstra is constant with respect to density, and all but Loubal only increase slightly as density increases. Loubal, though, is very much affected as Chart 10 demonstrates.

There were no test runs made for networks between 25% density and those of the four arcs per node type. This is because for very low densities when connections between nodes are randomly generated it requires an excessive amount of computation to assure that a spanning tree exists for each network. Thus, it cannot be determined at what point the Dijkstra algorithm becomes more efficient than Loubal. For networks of this type, all that can be suggested is to try both methods and see which is better. This may also be necessary for networks which have peculiar structures and do not fall into any of the categories included in this paper.

CHART 10

DENSITY EFFECT TOTAL NUMBER OF OPERATIONS



SUMMARY

This investigation of shortest path algorithms shows that for low density networks with four arcs per node the Loubal algorithm is the most efficient. For certain cases the Ford algorithm is very nearly as efficient as Loubal. For networks with 25% density or more, the Dijkstra algorithm is most efficient. For networks less than 70% dense the Loubal algorithm is second best while for higher density networks the Dijkstra algorithm using a threaded ordering of node labels is second best.

Unclassified

Security Classification

DOCUMENT CONTROL DATA · R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

University of California, Berkeley

2a REPORT SECURITY CLASSIFICATION

Unclassified

2b GROUP

3 REPORT TITLE

A COMPARATIVE INVESTIGATION OF THE COMPUTATIONAL EFFICIENCY OF SHORTEST PATH ALGORITHMS

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

Research Report

5 AUTHOR(S) (Last name, first name, initial)

HITCHNER, Lewis E.

6. REPORT DATE	7a TOTAL NO. OF PAGES	7b. NO. OF REFS
November 1968	22	6

8a. CONTRACT OR GRANT NO.	9a ORIGINATOR'S REPORT NUMBER(S)
Nonr-222(83) b. PROJECT NO. NR 047 033 c. Research Project No.: RR 003 07 01 d.	ORC 68-25 9b OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. AVAILABILITY/LIMITATION NOTICES

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
Also supported by the National Science Foundation under Grant GP-8695.	MATHEMATICAL SCIENCE DIVISION

13. ABSTRACT

Efficiency of shortest path algorithms is a function of various network parameters. This paper reports the results of an investigation of five algorithms for finding the shortest path from a root node to all other nodes for different network structures. Parameters considered are number of nodes, number of links, range of data (i.e., arc lengths), and shape (if applicable). It is hoped that the results will suggest to potential users of these algorithms which one is best suited for their problem.

DD FORM 1 JAN 64 1473

Unclassified

Security Classification

Unclassified

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Shortest Path Algorithms						
Computational Efficiency						
Dynamic Programming						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, roles, and weights is optional.

DD FORM 1 JAN 64 1473 (BACK)

Unclassified

Security Classification