(2)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

AD-A250 499

2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Final 1 JAN 88 - 31 DEC 91

4. TITLE AND SUBTITLE
"NEW DIRECTIONS IN NETWORK FLOWS"

5. FUNDING NUMBERS
61102F
2304/B1

6. AUTHOR(S)
Dr. James B. Orlin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
MIT
Sloan School of Management
50 Memorial Drive, E53-357
Cambridge MA 02139

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
Bldg 410
Bolling AFB DC 20332-6448

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFOSR-88-0088

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release;
Distribution unlimited

12b. DISTRIBUTION CODE
UL

13. ABSTRACT *(Maximum 200 words)*

A new, fast algorithm has been developed for the solution of problems using Lagrangian relaxation. This algorithm appears to improve running times by a factor of n-squared, where n is the number of variables.

14. SUBJECT TERMS

15. NUMBER OF PAGES
6

16. PRICE CODE

17 SECURITY CLASSIFICATION OF REPORT	18 SECURITY CLASSIFICATION OF THIS PAGE	19 SECURITY CLASSIFICATION OF ABSTRACT	20 LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

A Final Report for contract AFOSR-88-0088
20 March 1992

by James B. Orlin

MIT Sloan School of Management

This past year I have carried out research on several projects under the contract AFOSR-88-0088. These may be summarized as follows:

1. Developing faster algorithm for solving lagrangian relaxation problems.
2. Co-authoring a graduate text in the area of network flows.
3. Solving the shortest path problem from each node of a set of origin nodes to each node in a set of destination nodes.
4. Diagnosing infeasibilities in network flow problems.
5. Data compression for shortest path problems in Geographical Informations Systems.
6. Improved algorithms for the assignment problem.

These topics encompass a range of topics in which I am interested within the area of network optimization. I believe that several of these topics will be considered to be important contributions to the field of network optimization, and some of these may have important ramifications in practice. I will summarize each of the contributions below.

Faster Algorithms for Solving the Lagrangian Dual. (Joint with Dimitris Bertsimas.) This has led to the following paper:

> "A Technique for Speeding up the Solution of the Lagrangian Dual" with Dimitris Bertsimas. Operations Research Center working paper OR 248-91. April 1991. Accepted for publication by *Mathematical Programming*.

Moreover, the paper will presented at the upcoming IPCO conference.

Literally hundreds of papers have been written in the past 20 years in which problems have been solved using lagrangian relaxation. Mathematically, the problem can be formulated as follows. The problem is to find the optimal solution to P, where S is some polyhedral set. (For example, S can be the feasible solutions for some network flow

problem. The lagrangian subproblem is $P(\lambda)$, and the lagrangian dual is to maximize $z(\lambda)$ for $\lambda \in R^d$.

$$
\begin{array}{llll}
z^* = & \min & cx & \\
 & \text{s.t.} & Ax = b & P \\
 & & x \in S & \\
 & & & \\
z(\lambda) = & \min & cx - \lambda(Ax - b) & P(\lambda) \\
 & \text{s.t.} & x \in S &
\end{array}
$$

Remark. $z(\lambda) \leq z^*$ for all λ.

Lagrangian Dual: maximize $(z(\lambda) : \lambda \in R^d)$.

Lagrangian relaxations are a major application area for networks and other combinatorial problem since the problem $P(\lambda)$ is so often a network flow problem. Because the solution to lagrangian duals relies on solving the lagrangian subproblem a large number of times, the solution technique relies on the fact that network flow problems can be solved 100's of times faster than other linear programs.

The contribution of my joint paper with Dimitris Bertsimas to solving the lagrangian dual is the following: we have the best algorithms in terms of (worst case) performance guarantees. Moreover, our improvement over the best other algorithms is substantive. In some cases we have improved the running time by a factor n^2 or better. (And even though our paper talks about fast matrix multiplication, one does not need fast matrix multiplication to achieve these enormous speedups.) Our approach relies on solving the lagrangian dual problem using a linear programming method such as the ellipsoid algorithm.

We have not yet tested out our approach. We are optimistic about the performance in practice of our approach if we use as a subroutine an algorithm developed by Leonid Levin about 8 years ago. Levin's algorithm has the same performance guarantees as the ellipsoid algorithm and Vaidya's algorithm, but it seems to be much better suited for solving problems in practice. To our knowledge, no one has ever tested Levin's algorithms to see how well it does in practice. If our ideas work well, then we might solve many network optimization problems not previously solved in the literature.

Network Flow Book. (Joint with Ravi Ahuja and Tom Magnanti)

This book (approximately 800 pages) is intended both for classes in network optimization as well as by researchers. The book will be completed this fall, and it will be published by Prentice Hall next fall. It has a number of new research results within it on each of the following topics:

1. The shortest path problem
2. The maximum flow problem
3. The minimum cost flow problem
4. The multicommodity flow problem
5. The generalized flow problem
6. The convex cost flow problem

In addition, it brings together topics covered in our bibliography of approximately 500 papers, and presents a large number of disparate research results in a unified framework.

One of the key features of this book is a compilation of approximately 150 applications that have been culled from the Operations Research literature. These applications (each of which is described in sufficient detail) suggest the broad range of applicability of network flows. Moreover, the compilation of this large collection of applications may in and of itself lead O.R. practitioners to use network optimization models even more frequently in the future.

Another key feature of this book is the use of exercises to present additional technical material. We have included approximately 600 exercises, and so the book material goes significantly beyond that covered in the text of the chapters.

We also feel that we have made the last 10 years of research in networks accessible to a very broad audience, in addition to our covering the classical results.

The O-D shortest path problem (joint with Murali Kodialam).
This topic has appeared in the thesis of Murali Kodialam, and will be written as a technical paper within the next two months.

One of the most significant problems in network optimization is the problem of identifying the shortest (or least cost) path from one node to another node in a network. For example, in a communication network, one wants to find the cheapest way for one person to communicate with another. In a road network, one wants to find a shortest route from point A to point B. In addition, the shortest path problem has a wide range of

other applications in transportation systems, communication systems, project management, and other assorted contexts. Here we consider a generalization of the shortest path problem in which there is a set O of origin nodes, a set D of destination nodes, and the objective is to find the shortest path from each node in O to each node in D. We refer to this problem as the O-D shortest path problem.

Special cases of this problem include the following well studied problems: (1) the problem of finding the shortest path from a single origin s to a single destination t, (2) the problem of finding the shortest path from a single origin s to all other nodes, and (3) the problem of finding the shortest paths between all pairs of nodes in the network. In addition to generalizing these three famous shortest path problems, the O-D shortest path problem has other applications of importance.

The O-D shortest path problem arises whenever one wants to solve a restricted problem on a road network. For example, consider a traveling salesman problem in which a salesman needs to visit 100 customers. Determining the travel distances between each pair of customers is an O-D shortest path problem. As another example, in the Hitchcock transportation problem, one wants to deliver goods from a set S of warehouses to a set T of retailers, so as to minimize the total transportation cost. The cost of shipping from warehouse i to retailer j can be estimated using the minimum distance from i to j in a road network. To calculate the shortest distances one needs to solve the O-D shortest path problem between all pairs of points in S and all pairs of points to T.

This past year, we have considered a modification of the following algorithm for solving the O-D shortest path problem. For each origin node grow a shortest path tree starting from s using Dijkstra's algorithm (with appropriate data structures to enhance performance). Also, for each destination node t grow a shortest path tree terminating at t using (reverse) Dijkstra's algorithm. When all of the trees have intersected, then "patch" the solutions obtaining all of the O-D shortest paths.

We conjecture that this algorithm will be effective under a wide range of circumstances. For the case of random graphs, where each arc has an equal chance of being in the graph, and where arc lengths are uniformly or exponentially distributed, we can prove that the algorithm is remarkably good. For the case that the number of origin nodes and the number of destination nodes is less than $\sqrt{n/\log n}$ where n is the number of nodes, then the expected running time of the algorithm is linear in the data. In other words, solving the problem takes no more time than reading the data. And we have solved nearly n shortest path problems in this time!

In practice, graphs are not random (in the sense usually considered in the literature), and the algorithm will not be as effective; however, it is still likely that the algorithm will be quite good. This awaits our computational testing of the algorithm.

Diagnosing Infeasibility in Network Flow Problems. (Joint with Jianxiu Hao). This paper employs some of the techniques of the following paper to solve a problem of concern in practical implementations of network flow problems:

"A faster algorithm for Finding a Minimum Cut in a Graph" with Jianxiu Hao. Sloan School Working Paper 3372-92. December, 1992.

In the case that there is no feasible flow for a minimum cost network flow model, the modeler may want to diagnose the source of the infeasibility and correct it if possible. A "proof of infeasibility" (or violating set) is a set S of nodes whose net supply exceeds the net capacity of arcs leaving S. In general, there may be a large number of different violating sets. Harvey Greenberg posed the problem of developing tools for a modeler so that the modeler can find desirable violating sets, i.e., violating sets that really help the user to diagnose the problem and correct. We have determined procedures for finding violating sets with certain desirable properties including the following: (1) the set with the most infeasibility, (2) the set with the most infeasibility per node, and (3) violating sets S that are minimal, i.e., no proper subset of S is violating.

The first result is actually a well-known application of the max flow algorithm. The second result is an application of some deep results of Gallo, Grigoriadis and Tarjan. The third result is our primary contribution. We can show that one can find minimal violating sets by solving n different max flow problems (where n is the number of nodes), and one can take advantage of the special structure of these problems to solve all of the n max flow problems in the time the time that it normally takes to solve just one max flow problem.

We will write these results as a technical paper during this upcoming summer.

Data Compression for Shortest Path problems. (Joint with Diane Misra.)

We propose to consider shortest path problems in a road network. One of the key questions that we will deal with is the following: Suppose that one has solved the shortest path problem from each node in a subset S. To store this information in the standard way requires the storage of a spanning tree for each node in S. Thus, for a

network with n nodes, the amount of storage is O(n |S|). This amount of storage is prohibitively excessive for many applications. One alternative is to store only differences between spanning trees. Thus if we want to store spanning trees T_1, T_2, T_3, etc., then we need only store the arcs that are in T_j but not in T_{j-1} for each j = 2, 3, 4 etc. The primary difficulty with this storage technique is that the construction of each tree may take too long.

We have developed an approach that is comparably efficient to the above scheme in its data compression. We have found that the average number of arcs stored per tree is typically around $\sqrt{n}$ rather than n, and this leads to about a factor 100 savings in space for all pairs shortest path problems on networks with 10,000 nodes. In addition, our approach has the advantage of recovering any decompressing any spanning tree in O(n) time and decompressing any shortest path of k arcs between an origin and destination node in O(n log n) steps.

The Assignment Problem

We have developed a very fast approach for solving the assignment problem. Our basic approach is to use the successive shortest path algorithm with a variety of heuristics to improve the algorithm, including the following, (1) stopping each augmenting path phase earlier than is normally done, (2) using a technique to improve the performance of binary heaps, and (3) using bidirectional shortest path techniques to improve the performance of the augmenting path phases.

We anticipate that our approach will prove to be the fastest algorithm for the assignment problem. In addition, we have done extensive analysis which suggests that the running time for the algorithm on random assignment problems with n nodes and m arcs is $O(n \log^2 n + m)$. Surprisingly, this running time is not determined by curve fitting, but by an interesting mix of theory and computational testing.