ORNL/TM-12869

ornl

OAK RIDGE NATIONAL LABORATORY

MARTIN MARIETTA

MAXBAND Version 3.1: Heuristic and Optimal Approach for Setting the Left Turn Phase Sequences in Signalized Networks

Rekha S. Pillai*
Ajay K. Rathi

Date Published: February 1995

Prepared for the Federal Highway Administration
U. S. Department of Transportation
Office of Safety and Traffic Operations
Research and Development
Washington, DC 20590
under Interagency Agreement Number 1883-E020-A1

*Postdoctoral Research Program, Oak Ridge Institute of Science and Education (ORISE),
Oak Ridge, TN 37831

Center for Transportation Analysis
OAK RIDGE NATIONAL LABORATORY
P.O. Box 2008
Oak Ridge, TN 37831, U.S.A.
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
DEPARTMENT OF ENERGY
under contract no. DE-AC05-840R21

ORNL/TM-12869

Energy Division

MAXBAND VERSION 3.1: HEURISTIC AND OPTIMAL APPROACH FOR SETTING THE LEFT TURN PHASE SEQUENCES IN SIGNALIZED NETWORKS

Rekha S. Pillai*
Ajay K. Rathi

Date Published: February 1995

Prepared for the Federal Highway Administration
U. S. Department of Transportation
Office of Safety and Traffic Operations
Research and Development
Washington, DC 20590
under Interagency Agreement Number 1883-E020-A1

*Postdoctoral Research Program, Oak Ridge Institute of Science and Education (ORISE), Oak Ridge, TN 37831

Center for Transportation Analysis
OAK RIDGE NATIONAL LABORATORY
P.O. Box 2008
Oak Ridge, TN 37831, U.S.A.
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
DEPARTMENT OF ENERGY
under contract no. DE-AC05-840R21400

DLC

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This report is based on the work supported by the Federal Highway Administration. This research was also supported in part by an appointment to the ORNL Postdoctoral Research Associates Program administered jointly by ORNL and by the Oak Ridge Institute for Science and Education. The authors are grateful to Dr. Rajendra Solanki of ORNL and Dr. Steven Cohen of the FHWA for their contribution to the development and implementation of the solution algorithms. In addition, we were benefitted from the comments and suggestions of our colleagues, Dr. Shaw-Pin Miaou and Dr. Oscar Franzese.

EXECUTIVE SUMMARY

The main objective of synchronized signal timing is to keep traffic moving along arterial in platoons throughout the signal system by proper setting of left turn phase sequence at signals along the arterials/networks. The synchronization of traffic signals located along the urban/suburban arterials in metropolitan areas is perhaps one of the most cost-effective methods for improving traffic flow along these streets. MAXBAND Version 2.1 (formerly known as MAXBAND-86), a progression-based optimization model, is used for generating signal timing plan for urban networks. This model formulates the problem as a mixed integer linear program and uses Land and Powell branch and bound search to arrive at the optimal solution. The computation time of MAXBAND Version 2.1 tends to be excessive for realistic multiarterial network problems due to the exhaustive nature of the branch and bound search technique. Furthermore, the Land and Powell branch and bound code is known to be numerically unstable, which results in suboptimal solutions for network problems with a range on the cycle time variable. This report presents the development of a new version of MAXBAND called MAXBAND Version 3.1. This new version has a fast heuristic algorithm and a fast optimal algorithm for generating signal timing plan for arterials and networks. MAXBAND 3.1 can generate optimal/near-optimal solutions in fraction of the time needed to compute the optimal solution by Version 2.1. The heuristic algorithm in the new model is based on restricted search using branch and bound technique. The algorithm for generating the optimal solution is faster and more efficient than version 2.1 algorithm. Furthermore, the new version is numerically stable. The efficiency of the of the new model is demonstrated by numerical results for a set of test problems.

1. INTRODUCTION

Efficient transportation is very important to a nation's economic health. Nearly all economic activity uses transportation directly or indirectly. The economic productivity of a nation is boosted by improving the efficiency of transportation systems. The synchronization of traffic signals, located along the urban/suburban arterials in metropolitan areas, is perhaps one of the most cost effective method for improving traffic flow along these areas. The main objective of synchronized signal timing is to keep traffic moving along an arterial in platoons throughout the signal system by proper synchronization of green signals along the arterials/networks.

Over time, traffic engineering research has resulted in a number of techniques for setting traffic signals along arterials and networks. These models can be classified into two major categories: on-line models and off-line models. The on-line (also referred to as traffic adaptive) models compute signal settings in real-time and are used for controlling traffic dynamically. Optimization Policies for Adaptive Control (OPAC) is an example of this type of model (Gartner, 1983). This model generates signal setting for single intersection.

Off-line signal optimization models were developed in the late 1960s and early 1970s, and are used for computing signal settings for recurrent traffic flow conditions. The existing models for off-line determination of signal settings on single/multiarterial networks fall into one of two major categories. One set of models is based on the criteria of minimizing system delays and stops, while the other maximizes the progression bandwidth along the arterials. Delay minimization models lead to signal settings that minimize the number of stops and delays experienced by vehicles at intersections. Bandwidth maximization models lead to signal settings that maximize the proportion of traffic flowing unimpeded through the signals. TRANSYT (Robertson, 1968) and SIGOP (Lieberman *et al.*, 1983) are models that determine signal settings that minimize delay. These models combine macroscopic simulation and nonlinear optimization based gradient searches to determine the optimal signal settings. MAXBAND (Messer *et al.* 1987, Chang *et al.* 1988), is a model that maximizes bandwidth for multiarterials. The underlying optimization model in MAXBAND is a Mixed Integer Linear Programming (MILP) model (Little, 1966). Cohen *et al.* (1983,1986) and Liu (1988) have experimented with combining MAXBAND and TRANSYT models and have obtained signal settings that minimize delay and maximize progression bandwidth.

TRANSYT is perhaps the most widely used model for setting signal timings in the practice of traffic engineering. TRANSYT minimizes delay-based disutility functions from which green bands cannot always be found. Furthermore, the TRANSYT model does not optimize left turn phase sequences. MAXBAND model maximizes green bands, optimizes left turn phase sequences, and computes the best cycle time from a range of cycle time for given green split. Studies have shown [Rogness (1981), Cohen *et al.* (1983)] that left turn phase sequence optimization can substantially improve performance of signal timing plans. But, experience with MAXBAND has shown that hours of computer time may be required to optimize a medium-sized network problem even on a mainframe computer. The computational inefficiencies make the current version of MAXBAND impractical for use by traffic engineering community.

This technical report documents recent enhancements to the MAXBAND model made for the purposes of improving its numerical stability and execution time when running on IBM compatible microcomputers. These enhancements include: i) development of a fast heuristic algorithm that is capable of generating optimal/near optimal solution for the MILP in fraction of time required by the old MAXBAND, ii) a new optimal algorithm, and iii) replacement of the old linear programming problem solver with a numerically stable linear programming problem solver.

The MAXBAND model with these enhancements is referred to as MAXBAND Version 3.1 by the Federal Highway Administration (FHWA) who sponsored this research. The speed up in execution time should make MAXBAND usage attractive for real-time applications, off-line usage in a microcomputer system, and for iterative use of MAXBAND with delay-minimization problems or simulation procedures.

This report is organized as follows. In the next section, the history and evolution of bandwidth maximization model and the limitations of the existing solution approaches are discussed briefly. Section 3 discusses the fast heuristic algorithm and the optimal algorithm developed for the MILP. This is followed by a discussion on the LP solver and computer implementation of MAXBAND Version 3.1 in Section 4. An overview of the new version is provide in Section 5. Section 6, reports the results for a number of network and arterial test problems. Finally, conclusions and directions for future work are discussed in Section 7.

2. MAXBAND

The original mixed integer linear programming formulation for bandwidth maximization for signal setting along single arterial was by Little (1966). This formulation was extended to triangular networks by Little, Kelson, and Gartner (1981). MAXBAND Version 2.1 (formerly known as MAXBAND-86), developed by Messer *et al.* (1987), extended the formulation to account for general grid networks and left turn phase sequences. Gartner *et al.* (1991) report the extension of the arterial MILP formulation to include multi-band capability. Chaudhary *et al.* (1993) report the development of bandwidth optimization formulation that include circular phasing of signals, the new model is called PASSER IV.

The complete MILP formulation for multiarterial networks contains mix of integer and continuous variables. The optimal solution approach to solve the MILP is to use the branch and bound algorithm. MAXBAND Version 2.1 uses the branch and bound code by Land and Powell (1973) to solve the underlying MILP model. This code is numerically unstable for bandwidth maximization problems where the optimal cycle time is to be selected from a range of cycle times. Numerical instability results in runs ending prematurely with either suboptimal or no solutions at all. Also, the execution time for network problems were excessive due to the exhaustive nature of the branch and bound search in the optimization code. Some modifications were made to stabilize the numerical computations, see Solanki, Rathi, and Cohen (1993). Multi-step heuristic algorithms (Two-step heuristic and Three-step heuristic) were developed by Chaudhary, Pinnoi and Messer (1991) to generate optimal/near optimal solutions. The execution times for network problems were not consistently better than simultaneous optimization for all network problem instances and continued to be excessive. The issue of numerical instability remained unresolved, since they were using the Land and Powell code to solve the MILP sub-problems within the heuristic algorithms.

3. HEURISTIC AND OPTIMAL APPROACH

The mixed integer linear programming formulations for multiarterial networks consists of blocks of constraints dealing with individual arterials and some additional constraints that impose restrictions on loops of multiple arterials. The derivation of the constraints and detailed MILP formulation are provided in Messer *et al.* (1987) and hence will not be discussed in this report. Only the integer variables of the MILP formulation are discussed. The difficulty in solving realistic network problems arises due to the large number of integer variables in the MILP formulation.

The branch and bound procedure is an implicit enumerative search method for finding the optimal integer solution from a set of feasible integer solutions. This procedure does not deal directly with the integer problem. Rather, it considers a continuous problem, (Linear Program, LP, which is simpler to solve), defined by relaxing the integer restrictions on the variables. Thus the solution space of the integer problem is only a subset of the continuous space. If the optimal continuous solution is all integer, then it is also optimum for the integer problem. Otherwise, the branch and bound algorithm partitions the continuous solution space into subspaces, which are also continuous (this is called the branching operation). Each of the created subproblems can now be solved as a continuous problem. When the solution of a subproblem is integer, the subproblem is not branched, otherwise further branching is necessary. The optimal objective value for each linearized subproblem created by branching sets an upper bound (assuming the objective is to be maximized) on the objective value associated with any of its integer feasible values (this is called the bounding operation). The optimum integer solution is the integer solution of the subproblem having the largest upper bound (maximization problem). Nemhauser and Wolsey (1988) provide a more detailed description of the branch and bound procedure. The complexity of the branch and bound technique depends on the large number of branches that may be created and on the computer storage required for the storing subproblems to be scanned later. The worst case complexity of the branch and bound algorithm is the same as complete enumeration of every integer solution in the feasible space.

The set of integer variables in the bandwidth maximization MILP formulation can be divided into three sets:

Intra-loop variables (m_{ij}): are a set of general integer variables. This variable denotes the number of cycles required to go from signal i to signal $i+1$ and back, on arterial j. The m_{ij}'s should assume integer values due to the fact that the progression bandwidth in a specified direction for arterial j should pass through the green interval of signal cycles at signal i and $i+1$. Little (1966) provides the analytical justification for the integral nature of this set of variables.

Inter-loop variables (n_l): are a set of general integer variables. This variable denotes the number of cycles required for traversing arterials in the loop. The inter-loop variables are the reflection of the network closure constraints which are required in a closed network consisting of intersecting arterials and running on a common cycle length. These variables state that the sum of the offsets around any closed loop in the network must be an integral multiple of the common cycle length. Messer *et al.* (1987) provide the analytical justification for the integer nature of this set of variables.

Left-turn-phase sequence variables (δ_{ij}): are a set of binary variables. These variables are used to define the left turn phase sequence pattern on intersection i of arterial j.

3.1 RESTRICTED BRANCH AND BOUND ALGORITHM

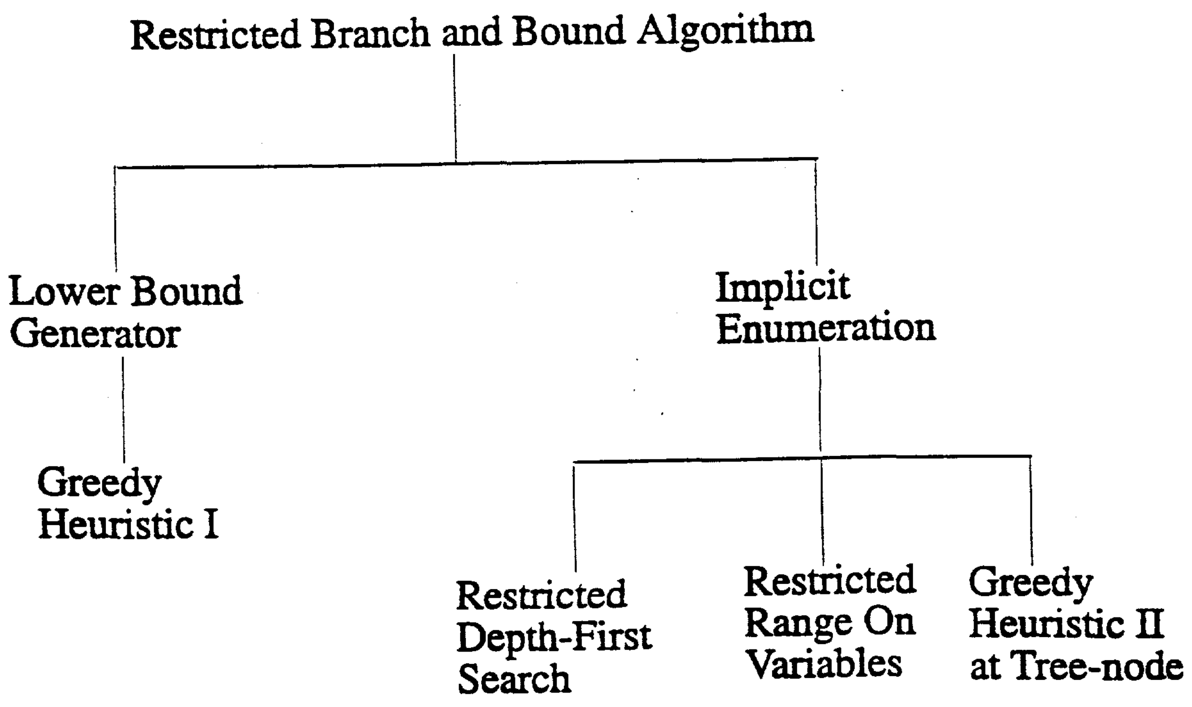
The heuristic procedure discussed in this report is a restricted search procedure for suitable values of the integer variables. The only known heuristics for the MILP are the *two-step* and *three-step* heuristic by Chaudhary *et al.* (1991,1993). The first step of the two-step heuristic relaxes the δ_{ij}'s to be continuous variables and searches for optimal m_{ij}'s and n_l's. Six of the best solutions, obtained during the search, are saved. (Note: These six best solutions found are not necessarily the best six solutions to the partial problem. They are the best six solutions to the partial problem which were identified as a part of the branch and bound search.) For each of these six solutions, the integer values of the m_{ij}'s and n_l's are fixed in the second step, which searches for optimal integer values of the δ_{ij}'s. Similarly, the three-step heuristic solves the integer values of the n_l's, m_{ij}'s and δ_{ij}'s in three steps, where the integer values obtained in one step are fixed in the next step. As expected, the two-step heuristic produces better solutions but consumes significantly more time compared with the three-step heuristic. In both heuristic methods, at each step an exhaustive branch and bound search is required to obtain optimal integer values. It was observed that, for some problem instances, the time required by the multi-step method could be more than the time required for the simultaneous optimization of all integer variables. This was because the six best solutions found during the execution of the first step are all kept and used in the subsequent steps, regardless of their relative merit and the exhaustive nature of branch and bound technique.

The key observation of a good heuristic design is to identify suitable problems that can be solved quickly and repetitively to generate improving solutions over iterations. The heuristic developed for MAXBAND Version 3.1 is a *restricted branch and bound algorithm*. The branch and bound search is restricted to portions of solution space which is likely to contain good solutions. Figure 1 gives an overview of the new heuristic. There are two key elements that characterize the algorithm described here:

1. a greedy heuristic to generate a good lower bound to be used at the root node of the branch and bound tree (*Greedy Heuristic I*), and
2. a *tree search* approach that combines branching and bounding techniques.

Efficient implementation of these key elements allow us to solve large problem instances of the MILP in reasonable time and memory allocations. Let P be the original MILP problem to be maximized. Let $V(P)$ be the optimal objective function value of P. Let P' be the LP relaxation of P, obtained by relaxing the integer variables m_{ij}'s, n_l's, and δ_{ij}'s. Then, $V(P')$ is the optimal objective value of P'. The fact that $V(P)$ _ $V(P')$ is a consequence of the linearity of the problem. If the optimal value of the solutions vector corresponding to the variables m_{ij}'s, n_l's, and δ_{ij}'s are integer in P', then the solution is optimal to the original problem P. The greedy heuristic, that generates a lower bound to be used in the tree search procedure, shall be discussed first. This report then continues to discuss the restricted branch and bound algorithm.

Greedy heuristic I is based on the concept of local search in the space of integer variables. Heuristic algorithms based on local searches have been found to be very effective in a large



Restricted Branch and Bound Algorithm
Lower Bound Generator
Implicit Enumeration
Greedy Heuristic I
Restricted Depth-First Search
Restricted Range On Variables
Greedy Heuristic II at Tree-node

Figure 1: Overview of Heuristic Algorithm

variety of integer programming problems. The key to this algorithm is to restrict the integer variable ranges to those values which are likely to yield good solutions. The local search is performed by fixing the values of some of the integer variables. The objective of the restricted problem is evaluated by solving the resulting restricted linear program.

3.1.1 Algorithm I: Greedy Heuristic I

Input: *P', the set of integer variables*

Step 1: Initialize the current incumbent, $Z^* = -\infty$.

Step 2: Perform steps 3 through 9 two times. Go to step 10.

Step 3: Order the set of integer variables as follows $I = \{n_1, \ldots., n_L, m_{11}, \ldots., m_{KN}\}$

Step 4: Solve LP problem P'.

Step 5: If the set I is empty then go to step 8; otherwise, pick the next variable from the ordered set I (say variable x_{ij}) and delete it from set I.

Step 6: Set the upper and lower bounds of the variable x_{ij} as follows:
$b^{ij}_{low} = b^{ij}_{up} = Int(x_{ij} + 0.5)$, i.e. set the upper and lower bounds of the integer variable to the integer value nearest the LP solution.

Step 7: Solve the restricted LP. If the current LP is infeasible then reset the variable last set to the other end of the LP optimal solution (obtained in step 6) and re-solve. Go to step 5.

Step 8: The algorithm reaches this step once all the integer variables have been set to the LP solution upper or lower bound. If the objective is greater than the current incumbent, save the current solution as the incumbent. Reset the bounds of all the integer variables.

Step 9: Reverse the order of the integer variables and put it in set I, i.e. this time the variable m_{KN} is the first variable and variable n_1 is the last variable. Go through steps 4 through 8.

Step 10: Fix the m_{ij}'s and n_l's at their best values and use branch and bound code to integerize the δ_{ij}'s.

The solution obtained at the end of step 10 of greedy heuristic I serves as a lower bound (best incumbent) in the branch and bound procedure. Such a bound restricts the growth of the tree and hence helps in faster resolution of the optimal solution. The restricted tree search algorithm, (also called restricted branch and bound), developed for bandwidth optimization can then be described as follows: In the tree-search procedure, the range over which the integer variables, m_{ij}'s and n_l's, can vary, are restricted. Integer variable m_{ij}'s are allowed only two values and integer variable n_l's are allowed three values. The three values of n_l's are selected such that the incumbent value is the middle value. The two values of m_{ij}'s are selected such that the incumbent value is the upper bound of this variable. For ease of exposition let an integer variable be denoted x_{ij}. Let the set F_l be the set of the integer variables fixed at the lower bound during the branch and bound procedure i.e. $F_l = \{x_{ij} / b_{low} \leq x_{ij} \leq b_{low}\}$. Let F_m be the set of integer variables fixed at the middle value i.e. $F_m = \{x_{ij} / b_{low}+1 \leq x_{ij} \leq b_{low}+1\}$, and F_r be the set of integer variables fixed at the upper bound i.e. $F_r = \{x_{ij} / b_{up} \leq x_{ij} \leq b_{up}\}$. Then, let S be a family of ordered triple of node sets $<F_l, F_m, F_r>$, and let Z^*, referred to as an *incumbent*, be the incidence vector of some integer feasible solution.

To describe the restricted branch and bound algorithm the following terminologies are used. Let a *tree-node*, associated with the ordered set $<F_l, F_m, F_r>$, be the problem $P(F_l,F_m,F_r)$. This is a problem of finding a signal timing plan whose solution vector satisfies the inequalities (3.1a), (3.1b), and (3.1c) given below:

$$b_{low} \leq x_{ij} \leq b_{low} \text{ for all integer variables in the set } F_l \qquad (3.1a)$$
$$b_{low}+1 \leq x_{ij} \leq b_{low}+1 \text{ for all integer variables in the set } F_m \qquad (3.1b)$$
$$b_{up} \leq x_{ij} \leq b_{up} \text{ for all integer variables in the set } F_r \qquad (3.1c)$$

Then, $P'(F_l,F_m,F_r)$ is the *linear relaxation* of $P(F_l,F_m,F_r)$ obtained by relaxing the integer variable not in the set F_l, F_m, and F_r. The tree-nodes are recorded by the ordered triple corresponding to it. A tree-node is considered fathomed if one or more of the following conditions are satisfied:

i. the optimal LP objective i.e. $V(P'(F_l,F_m,F_r))$, at this node is less than the current incumbent,
ii. the depth of this tree-node is equal to the maximum depth (*mdepth*) specified,
iii. the LP, $P'(F_l,F_m,F_r)$, is infeasible, or
iv. the optimal LP results in an integer feasible solution.

If the optimal solution of the current LP relaxation is fractional and the current depth (number of integer variables fixed) is less than maximum depth, the algorithm selects a *branching variable* x_{ij} and *branches*, thus providing up to three new tree-nodes ($<F_l \cup \{x_{ij}\}, F_m, F_r>$, $<F_l, F_m \cup \{x_{ij}\}, F_r>$, $<F_l, F_m, F_r \cup \{x_{ij}\}>$). The *root-node* of the search-tree is the tree-node $<\varnothing, \varnothing, \varnothing>$. During the algorithm the tree-nodes of the search-tree that are in S are called *active tree-nodes*. The restricted branch and bound algorithm can then be described as follows:

3.1.2 Algorithm II: Restricted Branch and Bound

Input: *Z^*, the LP problem P', the set of integer variables I.*

Step 1: (Initialization) Set $S = \{ <\varnothing, \varnothing, \varnothing> \}$. Limit the ranges of the m_{ij}'s and n_L's. Select the maximum depth *(mdepth)* of the tree to be half of the number of integer variables in the problem. Number the integer variables such that the first consecutive number, (starting with number 1), are given to the m_{ij}'s, the next consecutive numbers are given to n_L's and finally number the δ_{ij}'s.

Step 2: (Select a tree-node for evaluation). If $S = \varnothing$ then stop - the current incumbent is a local optima. Otherwise choose an ordered set $<F_l, F_m, F_r>$ from S and set $S = S \backslash <F_l, F_m, F_r>$.

Step 3: (Greedy heuristic II). Fix the all integer variables that are not fixed yet, (i.e. the set of integer variables $\{x_{ij} / x_{ij} \in I \backslash (F_l \cup F_m \cup F_r)\}$), to an integer value nearest to the LP solution, i.e. set $b^{ij}_{low} = b^{ij}_{up} = Int(x_{ij} + 0.5)$. Solve the new LP. If the optimal objective is greater than Z^*, save the solution and reset the variables fixed in this step.

Step 4: (Evaluation of tree-node). Solve the linear program, $P'(F_l, F_m, F_r)$, with the additional restriction. Let Z' be its optimal solution. If Z' _ Z^*, go to step 2.

Step 5: (Check for new incumbent) If Z' is integer feasible, and the optimal objective value is greater than Z^*, then set Z^* to Z'. Go to step 2.

Step 6: (Create new set of tree-nodes) If the depth of the tree is greater than the maximum depth specified for the problem instance then go to step 2. Otherwise, select a fractional integer variable x_{ij} to branch on. Such a variable will be in $I \backslash (F_l \cup F_m \cup F_r)$. Set $S = S \cup <F_l \cup \{x_{ij}\}, F_m, F_r> \cup <F_l, F_m \cup \{x_{ij}\}, F_r> \cup <F_l, F_m, F_r \cup \{x_{ij}\}>$ and go to step 2.

Once an ordered triple is removed from S it is never again generated in Step 6, so the algorithm terminates in a finite number of steps. When the algorithm stops, Z^* is a local optima. The performance of Algorithm II depends significantly on certain implementation details. In particular, the following issues are key to the algorithm's performance:

(a) Whether or not early tree-nodes can be fathomed depends on the starting Z^*. If this value is close to the optimum, the search-tree will consist of few tree-nodes. Therefore it is necessary to generate good feasible solutions early in the procedure. This objective is achieved by the Greedy Heuristic I and Greedy Heuristic II.

(b) Steps 3 and 4 must be executed many times before a good solution is obtained. A large portion of the final execution time of the algorithm is devoted to solving the LPs. Therefore, it is important to use the LP-optimizer as efficiently as possible. The LP-optimizer of MINOS is very fast and numerically very stable since it uses the state-of-art techniques of numerical analysis and linear programming for updating basis and performing basis inversions.

(c) The efficiency of the algorithm with respect to run time and memory usage depends on two things: the way the ordered triple is chosen in Step 2, and the way the tree-nodes are created. The tree-nodes were processed in a depth-first fashion (LIFO). The integer variables are ordered as follows: (m_{ij}'s, n_l's, and δ_{ij}'s). Through experimentation, it was found that this particular order led to incumbents that are close to optimal early on in the search tree. The depth of the search-tree was also restricted to half the number of integer variables, m_{ij}'s and n_l's. This restricted the number of tree-nodes generated and, hence, restricted the growth of the search tree. Further restriction on the range of integer variables also limited the number of tree-nodes generated. As will be seen in the numerical results both types of restriction helped in faster resolution of the optimal solution. The experimentation with δ variables revealed that these naturally turn out to be integer or can be rendered integer by a minimal amount of branching in the branch and bound search. Thus the δ values are searched using the exact optimization technique.

3.2 OPTIMAL ALGORITHM

The algorithmic steps for the optimal approach, in MAXBAND Version 3.1, is the same as that of the heuristic approach (discussed earlier). In the optimal algorithm, during the tree-search procedure the integer variables and the depth of the search tree are not restricted. Figure 2 gives an overview of the optimal method in MAXBAND Version 3.1.

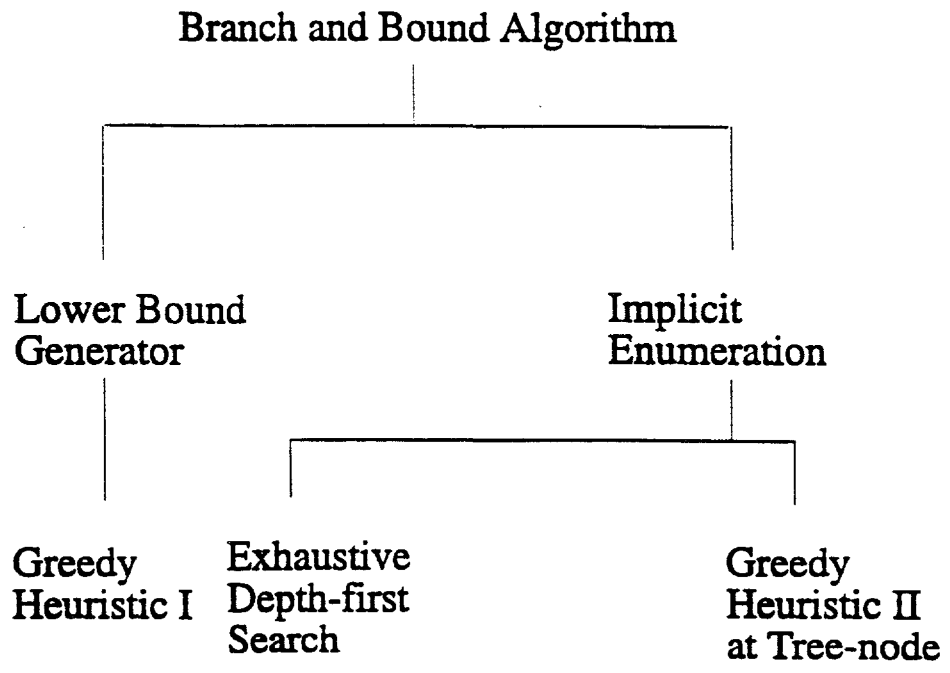

Branch and Bound Algorithm
Lower Bound Generator
Implicit Enumeration
Greedy Heuristic I
Exhaustive Depth-first Search
Greedy Heuristic II at Tree-node

Figure 2: Overview of Optimal Algorithm

MAXBAND Version 3.1 allows the user to select either the optimal approach or the heuristic approach for solving a network problem. This option can be exercised by setting the proper flag in the input data (discussed in the section on 'Getting Started'). The arterial problems are always solved optimally. This is because arterial problems could be solved very fast (in few seconds).

4. OVERVIEW OF MAXBAND VERSION 3.1

Figure 3 provides an overview of the structure of the revised MAXBAND Version 3.1. The new structure is not significantly different from the structure of MAXBAND Version 2.1. The routines in the box shown by dashed lines are the new modules. In MAXBAND Version 2.1 this box contained the MATGEN and the MPCODE modules. The branch and bound procedure (by Land and Powell) used in MAXBAND Version 2.1, is called MPCODE. The matrix generator routine used to generate the MILP model for MPCODE is called MATGEN. In this research, MATGEN and MPCODE routines were replaced by a model generator routine call MPSGEN and a MILP solver routine called MODMINOS respectively. The MODMINOS module is comprised of subroutines from MINOS 5.4 (1993) modified for use in MAXBAND Version 3.1. Details of the MINOS code will be provided in section 5.0. To the MINOS code, heuristic algorithms and new branching and bounding strategies were added for faster resolution of the optimal bandwidth. The MODMINOS module is capable of solving both arterial and network problems.

In MAXBAND Version 3.1, the arterial problems are always solved optimally. The network problems can be solved either heuristically or optimally. The MPSGEN routine generates the MILP model in the MPS format, (format discussed in Appendix A), required by MINOS. The OUTPUT module was also modified to accept the signal timing plans in the form provided by MINOS. Subroutines that would print the MINOS run statistics (e.g. number of solutions obtained, number of iterations, etc.) were added to the OUTPUT module.

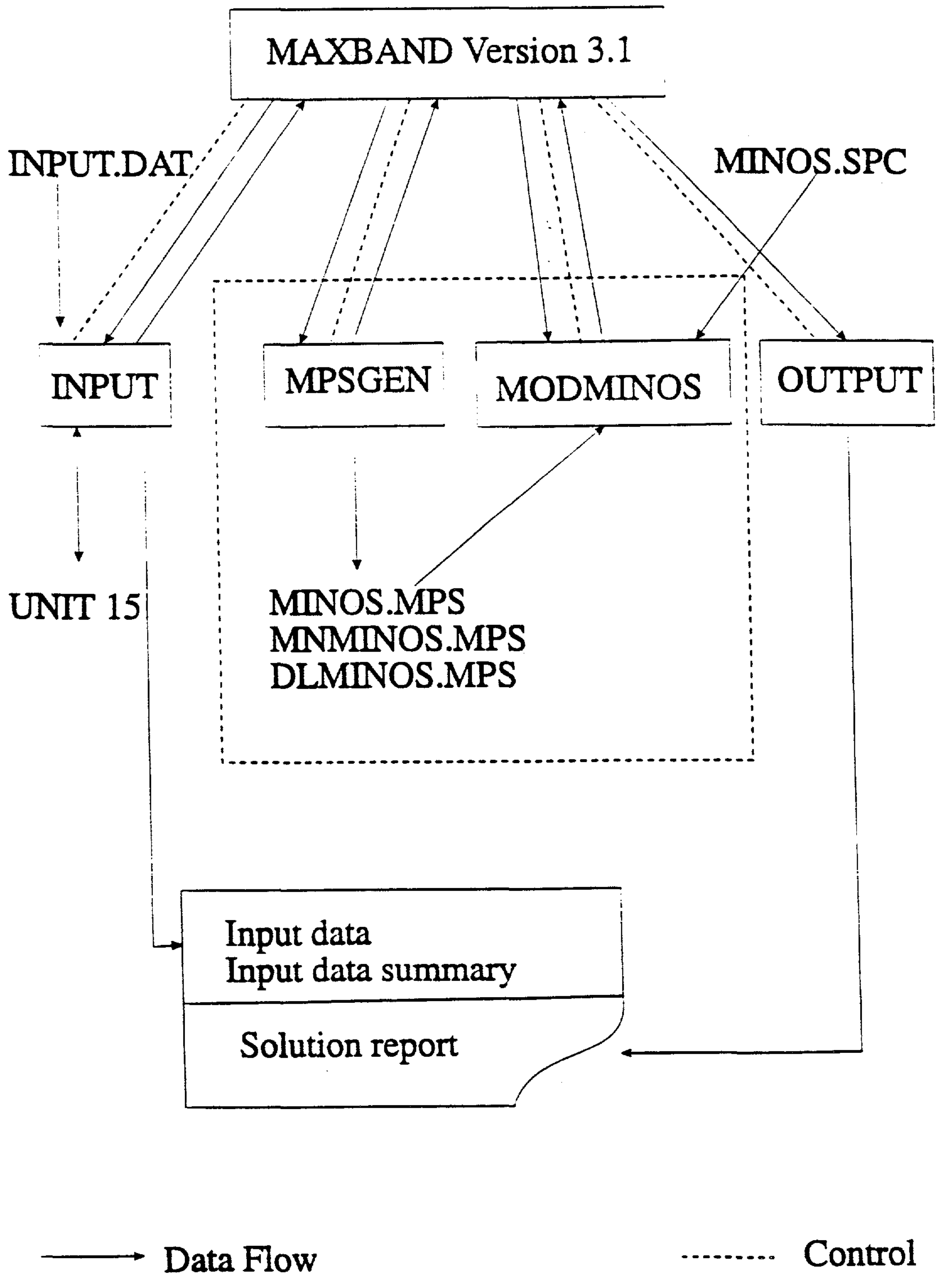


Figure 3: Overview of MAXBAND Version 3.1

5. THE LP SOLVER

Linear program solver routines from MINOS 5.4 (1993) were used to solve the LPs in Algorithms I and II. MINOS is a FORTRAN-based computer system designed to solve large-scale linear and nonlinear optimization problems. It has a collection of high-performance mathematical subroutines which can be called from application programs. From this package the subroutines required for solving linear programming problems were used. The main reasons for choosing MINOS over other LP solvers were the cost and the availability of source code. The availability of source code allowed customization and therefore helped to speed up the executable.

MPCODE, the MILP solver in MAXBAND Version 2.1, is a straightforward implementation of the revised simplex algorithm presented in any elementary Linear Programming textbook. Advances in numerical analysis techniques and operations research techniques have led to improved revised simplex routines for updating basis, inverting basis, choosing entering and leaving columns, etc. These advances have led to faster and numerically stable algorithms for solving linear programming problems. MINOS implementation takes advantage of the recent advances in numerical analysis and linear programming techniques, e.g. using scaling as a simple cure for ill conditioned matrices. MINOS performs scaling of rows, right hand side vectors, and columns by choosing appropriate scale factors to make its rows and columns roughly the same length, in some appropriate norm during the solution process; whereas, in MPCODE the scaling of a problem instance had to be performed by the user externally. In MINOS, the constraints and variables are scaled by an iterative procedure that attempts to make the matrix coefficients as close as possible to 1. This improves the solution performance. Some of the other techniques adopted by MINOS to improve stability and efficiency are discussed in the following paragraphs.

Data (both input and output) is stored within a work array that is partitioned by a set of pointers to starting locations of individual arrays needed by the procedure, each with an appropriate number of bytes that depends on whether the array is integer, single, or double precision floating-point. This makes implementation largely independent of data structures and it is then relatively easy to unplug one set of data structure and substitute another.

An elementary way to solve a nonsingular square system of linear equations that arise in our case, within the cycle of primal simplex algorithm, is to use Guassian elimination. LU factorization is a reformulation, in matrix terms, of Gaussian Elimination. During LU factorization the near zero pivot elements lead to uncontrollable growth in the elements and fill-in of L and U. This in turn results in large numerical errors and large computational times. The solution is to choose pivot elements suitably so as to prevent such element growth and fill-in growth. MINOS implementation is based on the *Markowitz pivoting strategy* that balances considerations of stability and sparsity. The basis updating strategy used by MINOS is the *Bartels-Golub basis updating* strategy in which updating is carried out with a pivot strategy that balances considerations of stability and sparsity. The basis inverse is maintained implicitly in product form. For complete details of the Markowitz pivoting strategy, Bartels-Golub basis updating strategy, and implementation details see Reid (1976, 1982). MINOS has also implemented various selection strategies for actually making the choice of entering and exiting variables. These strategies lead to faster resolution of the optimal solution, degeneracy resolution and also doesn't lead to numerical instability.

Certain disk files are accessed by MINOS subroutines as follows:

Input files
SPEC file (input file)
MPS file (created during run time)
INTEGER VARIABLE SPECIFICATION files (created during run time)

Output files
OUTPUT file

The SPEC file sets various run time parameters that describe the nature of the problem being solved. This file is an input file and is required along with the network data file and the executable for MAXBAND. The file consists of a sequence of card images, each of which contains a *keyword* and certain associated values. The first keyword is BEGIN and the last keyword is END. The SPEC file format, the parameters, default values and a sample SPEC file are given in APPENDIX A.

The MPS file is required for all problems to specify names for the variables and constants, and to define the constraints themselves. This file is generated by MAXBAND Version 3.1. A very fixed format must be used for the MPS file; this means that each item of data must appear in a specific column. The MPS file, called 'MINOS.MPS', is created by MPSGEN routines of MAXBAND Version 3.1. The MPS file format and a sample MPS file are given in APPENDIX B.

The files 'MNMINOS.MPS' and 'DLMINOS.MPS', are used to specify which variables in the linear program described by the specifications file and the MPS data file are either integer values or are integer multiples of given increments. These two files use a format similar to that found in an MPS file. These files are also created by the MPSGEN routines. The INTEGER VARIABLE SPECIFICATION file format and a sample file are given in APPENDIX C.

Warnings and errors encountered during the solution of problems are printed in the OUTPUT file. The solution to the problem is also printed in the OUTPUT file in the format specified by MAXBAND Version 2.1. The following information could be written by the MINOS routines to the OUTPUT file during the solution of each problem:

1. A listing of warnings/errors encountered in the MPS file (if any).
2. Extra storage requirements (if any).
3. Abnormal exit conditions (if any).
4. Various run statistics: number of feasible integer solutions found, number of tree-nodes created, etc.

6. NUMERICAL RESULTS

A number of network and single arterial problems were solved using the new model. The test data sets were supplied by FHWA. Tables 1 through 4 report the solution quality and the computation times for these problems. Tables 1 and 2 show the results of the arterial test problems. Tables 3 and 4 correspond to the network problem runs. The columns of Tables 3 and 4 can be described as follows. Column 1 specifies the names of data set, as provided by FHWA. Column 2 contains the problem size showing the number of arterials and total number of intersections. Column 3 contains the optimal objective value. Column 4 contains the objective function value at the end of the LP based heuristic (greedy heuristic I); the numbers in parentheses show how close this value is to the optimal value. Column 5 contains the time in seconds for greedy heuristic I. Column 6 contains the objective at the end of the restricted branch and bound procedure; the number in parentheses shows how close this value is to the optimal objective value. The numbers in column 7 show the time taken in seconds for the entire algorithm. The computation times are reported for an 80486/66 MHz personal computer. As is observed from the Tables 3 and 4, the heuristic performs very well in generating optimal/near-optimal solutions in a short amount of time. The utility of the heuristic increases as the size of the problem grows and an exact search requires excessive computation time. All of the arterial problems were solved using the optimal approach.

Table 1: Arterial Problem Without Left-Turn Phase Sequence Variables

PROBLEM	SIZE	Opt. Obj.	Time (Sec.)	MAXBAND Version 2.1 Time (sec)**
Hawth	(1,13)	0.2305	12.14	106
N33rd	(1,9)	0.1292	7.47	19
Nicholas	(1,12)	0.5454	11.53	56
Univ	(1,10)	0.2166	4.89	19
Fredrica	(1,12)	0.6726	15.32	79
Mstreet	(1,8)	0.6993	2.52	4

** CPU times on a 486/33 MHz personal computer.

Table 2: Arterial Problem With Left-Turn Phase Sequence Variables

PROBLEM	SIZE	Opt. Obj.	Time (Sec.)	MAXBAND Version 2.1 Time (sec)**
Hawth.ph	(1,13)	0.5796	15.38	889
N33rd.ph	(1,9)	0.4735	16.53	44
Nicholas.ph	(1,12)	0.6254	16.92	257
Univ.ph	(1,10)	0.5000	7.36	23
Fredrica.ph	(1,12)	0.7106	33.83	209
Felipe.ph	(1,12)	0.6000	3.74	21

** CPU times on a 486/33 MHz personal computer.

Table 3: Network Problem Without Left-Turn Phase Sequence Variables

PROBLEM	SIZE	Opt. Obj.	Heu.Obj. (%Opt)	Time (Sec.)	Heu. B&B Obj. (%Opt)	Time (Sec.)	MAXBAND Version 2.1 Time (sec)**
Daytona	(7,12)	1.6368	1.5381(94.0%)	46.58	1.5403(94.1%)	235.52	244592
Annarbor	(8,14)	3.3235	2.8010(84.3%)	57.40	3.3235(100%)	198.18	8912
Houston	(8,13)	2.5079	1.5816(63.1%)	58.50	2.3723(94.6%)	113.26	11765
Memphis	(8,17)	3.0479	2.5768(84.5%)	50.10	3.0044(98.6%)	582.59	28306
Ogden	(8,13)	2.6192	2.2981(87.7%)	62.40	2.5902(98.9%)	548.49	33365
Baycity	(8,16)	2.9658	2.5040(84.4%)	75.63	2.7094(91.4%)	348.18	26854
Owosso	(8,16)	4.0057	3.2217(80.4%)	45.21	3.9119(97.7%)	111.99	3804
Lax	(8,15)	3.4278	3.2157(93.8%)	65.63	3.4278(100%)	394.86	15745
Wlntcrk	(6,13)	2.0925	1.5478(74.0%)	28.62	2.0925(100%)	148.30	3525
Sanramon	(6,17)	1.8848	1.5416(81.8%)	70.03	1.8848(100%)	348.45	32812
Annarbo1	(9,20)	2.5599	1.6168(63.2%)	125.18	2.3363(91.3%)	1782.11	na
Wlntcrk1	(9,22)	2.6737	2.4303(90.9%)	75.80	2.4303(90.9%)	658.12	na
Annarbo2	(9,26)	3.0861	2.5016(81.1%)	161.81	2.7579(89.4%)	1443.50	na
Wlntcrk2	(10,27)	2.5763*	1.8660(72.4%)	133.36	2.4834(96.4%)	985.20	na

* Best known objective, not necessarily optimal.

** CPU times on a 486-33 MHz microprocessor.

Table 4: Network Problem With Left-Turn Phase Sequence Variables†

PROBLEM	SIZE	Opt. Obj.	Heu.Obj (%Opt)	Time (Sec.)	Heu. B&B Obj (%Opt)	Time (Sec.)**
Daytona.ph	(7,12)	2.9445	2.2949(77.9%)	43.12	2.9445(100%)	474.72
Annarbor.ph	(8,14)	3.7652	3.4915(92.7%)	34.71	3.7095(98.5%)	314.56
Houston.ph	(8,13)	2.8171	2.0992(74.5%)	48.83	2.8171(100%)	106.78
Memphis.ph	(8,17)	3.4682	3.2214(92.9%)	31.14	3.4474(99.4%)	319.89
Ogden.ph	(8,13)	3.1220	2.9627(94.9%)	48.94	3.1029(99.4%)	654.11
Baycity.ph	(8,16)	3.8354	2.6903(70.1%)	54.98	3.7395(97.5%)	221.95
Owosso.ph	(8,16)	4.2978	3.5324(82.2%)	27.14	4.2321(98.5%)	79.48
Lax.ph	(8,15)	4.0004	2.9501(73.8%)	117.76	3.8862(97.2%)	939.01
Wintcrk.ph	(6,13)	2.8304	2.3120(81.7%)	26.09	2.7231(96.2%)	103.37
Sanramon.ph	(6,17)	2.8012	2.3150(82.6%)	31.41	2.8012(100%)	181.26
Annarbo1.ph	(9,20)	2.9380	1.8481(62.9%)	118.09	2.7929(94.7%)	7389.95
Wintcrk1.ph	(9,22)	3.3667	2.5443(75.6%)	62.29	3.2071(95.3%)	1494.02
Annarbo2.ph	(9,26)	3.6215*	1.9657(54.3%)	223.38	3.4078(94.1%)	6545.64
Wintcrk2.ph	(10,27)	3.6644*	3.3740(92.1%)	125.62	3.6798(100.4%)	6869.75

† MAXBAND Version 2.1 was unable to solve these problems in 5 days on a 486-33MHz personal computer.

* Best known objective, not necessarily optimal.

** CPU times on a 486-33 MHz microprocessor.

7. CONCLUSIONS AND FUTURE WORK

This report describes the development of a new version of MAXBAND called MAXBAND Version 3.1. The new version has a fast heuristic algorithm and an new optimal algorithm. The two algorithms are faster than the existing approach for bandwidth maximization problem. Furthermore, the new version has removed the numerical instabilities that existed in the previous version. The reduction in computation times for difficult network problems are substantial, and should make MAXBAND usage attractive for real-time applications, off-line usage in a microcomputer system, and for repetitive solutions of the bandwidth maximization problem in conjunction with the delay minimization problem or simulation procedure. All of the arterial problems were solved optimally since they only required a few seconds to solve.

Further work can be done to enhance the MILP formulation to include circular phasing and multiband capability. Work can also be done in building a combined model based on bandwidth maximization, delay minimization, and simulation.

REFERENCES

E.C.P. Chang, S.L. Cohen, C.C. Liu, N.A. Chaudhary, and C.J. Messer. **MAXBAND-86: Program for Optimizing Left Turn Phase Sequence in Multiarterial Closed Networks.** *Transportation Research Record* 1181:61–67. 1988.

N.A. Chaudhary, A. Pinnoi, and C.J. Messer. **Proposed Enhancements to MAXBAND-86 Program**. *Transportation Research Record*, 98–104. 1991.

N.A. Chaudhary and C.J. Messer. **PASSER-IV: A Program for Optimizing Signal Timing in Grid Networks**. *Paper presented at the 72nd Annual Meeting of the Transportation Research Board, Washington D.C.* 1993.

H. Chen. **An Adaptive Control Strategy for Signalized Intersections**. *Proceedings of the North American Conference on Microcomputers in Transportation, Boston MA*. 1987a.

H. Chen. **Simulation Study of OPAC: A Demand Responsive Strategy for Traffic Signal Control**. *Proceedings for the Tenth International Symposium on Transportation and Traffic Theory, Cambridge MA*. 1987b.

S.L. Cohen. **Concurrent use of MAXBAND and TRANSYT Signal Timing Programs for Arterial Signal Optimization**. *Transportation Research Record*, 906:81–84. 1983.

S.L. Cohen and C.C. Liu. **The Bandwidth-Constrained TRANSYT Signal Optimization Program**. *Transportation Research Record*, 1057:1–7. 1986.

R. Fourer. **Solving Staircase Linear Programs by the Simplex Method**. *Math. Prog. 23, 274–313.* 1982.

N.H. Gartner, M.H. Kaltenbach, and M.M. Miyamato. **Demand Responsive Decentralized Urban Traffic Control, Part II: Network Extensions**. *Technical Report DOT/RSPA/DPB, Department of Transportation, Washington D.C.* 1983.

N.H. Gartner, S.F. Assmann, F.L. Lasaga, and D. L. Hout. **A Multi-Band Approach To Arterial Traffic Signal Optimization**. *Transportation Research Board*, Vol. 25B, No. 1, pp. 55–74. 1991.

A.H. Land and S. Powell. **Fortran Codes for Mathematical Programming: Linear, Quadratic and Discrete.** *John Wiley & Sons, London.* 1973.

E.B. Lieberman. **An Advanced Approach to Meeting Saturated Flow Requirements**. *Paper Presented at the 72nd Annual meeting of the Transportation Research Board, Washington D.C.* 1993.

E.B. Lieberman, J. Lai, and R.E. Ellington. **SIGOP III User's Manual: Technical Report,** *Federal Highway Administration, Washington D.C.* 1983.

J.D.C. Little. **The Synchronization of Traffic Signals by Mixed-Integer Linear Programming**. *Operations Research,* 14:568–594. 1966.

J.D.C. Little, M.D. Kelson, and N.H. Gartner. **MAXBAND: A Program for Setting Signals on Arterials and Triangular Networks**. *Transportation Research Record,* 795. 1981.

C.C. Liu. **Bandwidth Constrained Delay Optimization for Signal Systems**. *ITE Journal,* 58:21–26. 1988.

C.J. Messer, G.L. Hogg, N.A. Chaudhary, and E.C.P. Chang. **Optimization of Left Turn Phase Sequence in Signalized Networks using MAXBAND 86, Vol.1 Summary Report.** *Technical Report FHWA/RD-87/109,* FHWA. 1987.

G.L. Nemhauser, L.A. Wolsey, **Integer and Combinatorial Optimization**. *A Wiley-Interscience Publication,* John Wiley and Sons, New York, 1988.

J.K. Reid, **Fortran Subroutines for Handling Sparse Linear Programming Bases**. *Technical report R8269, Atomic Energy Research Establishment,* Harwell, England, 1976.

J.K. Reid, **A Sparsity-Exploiting Variant of the Bartels-Golub Decomposition for Linear Programming Bases**. *Mathematical Programming,* 24, 55–69, 1982.

D.I. Robertson, **TRANSYT: Traffic Network Study Tool,** *Fourth International Symposium on the Theory of Traffic Flow, Karlsruhe, Germany.* 1968.

R.O.Rogness. **Evaluation of a Heuristic Programming Approach to Arterial Street Signal Timing Operation**. *Ph.D. Dissertation, Texas A&M University.* 1981.

R.S. Solanki, A.K. Rathi, and S.L. Cohen. **Numerical Stabilization of MAXBAND 86 on PC**. *Paper presented at the 72nd Annual Conference of the Transportation Research Board, Washington D.C.* 1993.

MINOS 5.4, **Stanford Business Software,** *Stanford University, Stanford, CA.* 1993.

APPENDIX A

SPECS FILE

Each line in the SPECS file contains a sequence of items in free format (they may appear anywhere in columns 1 to 72). The items are separated by spaces or equal signs (' ' or '='). Those selected from each line are:

1. The first word (the *keyword*). Only the first 3 characters are significant.

2. The second word (if any). Sometimes this is the keyword's associated name value, an 8 character name. More often it qualifies the keyword, and its first 4 characters are significant.

3. The first number (if any). This may be an integer value or a real value, specified by up to 8 characters in Fortran's I, F, E or D format.

The following example SPECS file shows all valid keywords and their default values relevant for MAXBAND Version 3.1. A more extensive list is provided in the MINOS 5.4 [1993] manual. Keywords are grouped according to the function they perform. Blank lines and comment lines may be used for readability. A comment begins with an asterisk ('*') and includes subsequent characters on the same line. Some of the default values depend on ϵ, the relative precision of the machine being used. The values given here correspond to double-precision arithmetic on IBM 360 and 370 systems and their successors ($\epsilon \approx 2.22 \times 10^{-16}$). Similar values would apply to any machine having about 15 decimal digits of precision.

```
BEGIN * check list of SPECS file parameters and their default values
*
* Keywords for the MPS file
*
    MAXIMIZE                                       * opposite of MINIMIZE
    ROWS                         100               * (or less)
    COLUMNS                      300               * or 3*rows (or less)
    ELEMENTS                     1500              * or 5*columns (or less)
    MPS file                     ?                 * depends on installation




    AIJ TOLERANCE           1.0E-10      *
    LOWER BOUND             0.0          *
    UPPER BOUND                      1.0E+20       * Plus infinity*

* Keywords for the simplex method
*
    FACTORIZATION FREQUENCY          50            * refactorize the basis matrix
    SCALE                            YES           * linear constraints and
variables
*
* Convergence and Stability tolerances
*
    FEASIBILITY TOLERANCE            1.0E-6  * for satisfying bounds
*
* Keywords for Mixed Integer Program
*
    CYCLE LIMIT                      1             * limits number of B&B
problems
    WORKSPACE (USER)                 0             * allocated for B&B routine
    WORKSPACE (TOTAL)          ?             * depends on installation
*
END * of SPECS file checklist
```

SPECS FILE DEFINITIONS

The following is an list of recognized SPECS file keyword definitions. A use of each keyword is given, along with a definition of the quantities involved and comments on usage. In some cases the value associated with a keyword is denoted by a letter such as *k*, and allowable values for *k* are subsequently defined.

AIJ TOLERANCE (*t*): During the input of the MPS file, matrix coefficients a_{ij} will be ignored if $|a_{ij}| < t$, i.e. the coefficient is set to zero.

COLUMNS (*n*): This must specify an *over-estimate* of the number of columns in the constraint matrix (excluding slack variables). If *n* proves to be too small, MINOS will continue reading the MPS file to determine the true value of *n*, and an appropriate warning message will be issued in the OUTPUT file.

CYCLE LIMIT (*l*): The number of branch and bound subproblems solved is controlled by the CYCLE LIMIT parameter found in the SPECS file. For this reason, *l* should be set to be more than the maximum number of subproblems to be solved.

ELEMENTS (*e*): This must specify an *over-estimate* of the number of nonzero elements (coefficients a_{ij}) in the constraint matrix. COEFFICIENTS is a valid alternative keyword. If *e* proves to be small, MINOS continues in the manner described under COLUMNS.

FACTORIZATION FREQUENCY (*k*): At most *k* basis changes will occur between factorizations of the basis matrix. In LP, the basis factors are usually updated every iteration. The default *k* is reasonable for typical problems. Higher values of *k* (say, up to $k=100$) may be more efficient on problems that are extremely sparse and well scaled.

FEASIBILITY TOLERANCE (*t*): A feasible solution is one in which all variables satisfy their upper and lower bounds to within the absolute tolerance, *t* (this includes slack variables). The linear constraints are also satisfied to within *t*. MINOS attempts to find a feasible point before optimizing the objective function. If the sum of infeasibilities cannot be reduced to zero, the problem is declared INFEASIBLE. Let SINF be the corresponding sum of infeasibilities. If SINF is quite small, it may be appropriate to raise *t* by a factor of 10 or 100. Otherwise, some error in the data should be suspected. If SCALE is used, feasibility is defined in terms of the scaled problem (since it is then more likely to be meaningful).

LOWER BOUND (*l*): Before the BOUNDS section of the MPS file is read, all structural variables are given the default lower bound *l*. (Individual variables may subsequently have their lower bound altered by a BOUND set in the MPS file). LOWER BOUND = 1.0E-5 (say) is a useful method for bounding all variables to remove singularities at zero. If all or most variables are to be FREE, use LOWER BOUND = -1.0E+20 to specify 'minus infinity'.

MAXIMIZE: This specifies the required direction of optimization.

MPS FILE (*f*): This is the file number for the MPS file. The default value is the system file reader ($f=5$). INPUT FILE is a valid alternative keyword.

ROWS (*m*): This must specify an *over-estimate* of the number of rows in the constraint matrix. If *m* proves to be too small, MINOS continues in the manner described under COLUMNS.

SCALE: The constraints and variables are scaled by an iterative procedure that attempts to make the matrix coefficients as close to 1 as possible (see Fourer [1982]). This will sometimes improve the performance of the solution procedures.

UPPER BOUND (*u*): Before the BOUNDS section of the MPS file is read, all structural variables are given the default upper bound *u*. (Individual variables may subsequently have their upper bound altered by the BOUNDS section in the MPS file.)

WORKSPACE (USER) (*maxw*): You will need to reserve a portion of the MINOS workspace for use by the integer programming routines. The variable *maxw* is the number of spaces in the workspace array that will be allocated to the arrays needed for integer programming. A good rule of thumb for selecting *maxw* is to set it to at least $6n+2m+4l$, where n is the number of variables, m is the number of constraints, and l is the CYCLE LIMIT.

WORKSPACE (TOTAL) ***(maxz)***: Default *maxz=NWCORE*. These keywords define the limits of the region of storage that MINOS may use for solving the current problem. The main work array is declared in the main program (MODMINOS), along with its length, by statements of the form

```
DOUBLE PRECISION Z(200000)
DATA          NWCORE/200000/
```

where the actual length of Z must be specified at the compilation time. The values specified by the WORKSPACE keywords are stored in

```
COMMON /M2MAPZ/ MAXW, MAXZ
```

and workspace may be shared according to the following rules:

1. Z(1) through Z(MAXW) is available for branch and bound routines.

2. Z(MAXW+1) through Z(MAXZ) is available to the LP solver in MINOS, and is not to be altered by MAXBAND 93.

3. Z(MAXZ+1) through (ZWCORE) is unused.

The WORKSPACE parameters are most useful on machines with a virtual (paged) store. Some systems will allow NWCORE to be set to a very large number (say 500000) with no overhead in saving the resulting object code. In general it is far better to have too much storage than not enough.

AN EXAMPLE:

```
Begin Example  * sample spec file
   Maximize
   Rows              1000
   Columns           1000
```

```
      Elements                  20000
      Cycle Limit               100000
      MPS file                  10
      Feasibility Tolerance     1.0E-5
      Workspace (User)  90000
      Workspace (total)         200000
End Example
```

APPENDIX B

MPS FILE

MPS format is the industry standard. Files of this kind are recognized by all commercial mathematical programming systems. In contrast to the relatively free format allowed in the SPECS file, a very fixed format must be used for the MPS file. Various 'header lines' divide the MPS file into several sections as follows:

```
NAME
ROWS
  .
COLUMNS
  .
RHS
  .
RANGES        (optional)
  .
BOUNDS        (optional)
  .
ENDATA
```

Each header line must begin in column 1. The intervening lines (indicated by '.' above) all have the following data format:

Columns	*2-3*	*5-12*	*15-22*	*25-36*	*40-47*	*50-61*
Contents	*Key*	*Name0*	*Name1*	*Value1*	*Name2*	*Value2*

The NAME section: This section contains the word NAME in columns 1-4, and a name for the problem in columns 15-22. The name may be from 1 to 8 characters of any kind, or it may be blank. The name is used to label the solution output. The NAME line is normally the first line in the MPS file, but it may be preceded or followed by comment lines.

eg. NAME   MAXBAND 93

The ROWS Section: The constraints are referred to as *rows*. The ROWS section contains one line each for each constraint (i.e. for each row). *Key* defines what type the constraint is, and *Name0* gives the constraint an 8-character name. The various row-types are as follows:

Key	*Row-type*
E	=
G	$\geq$
L	$\leq$
N	Objective

Row-types E, G, and L are easily understood. Row-type N stands for 'non binding', also known as 'free'. It is used to define the objective row, and also to prevent a constraint from actually being a constraint.

eg. ROWS

```
     N   MAXBAND
     E   LC001
```

The COLUMNS Section: For each variable x_j the COLUMNS section defines a name for x_j and lists the nonzero entries a_{ij} in the corresponding column of the constraint matrix. The nonzeros for the first column must be grouped together before those for the second column, and so on. If a column has several nonzeros, it does not matter in which order they appear (as long as they appear before the next column). In general, *key* is blank (except for comments), *Name0* is the column name, and *Name1, Value1* give a row name and value for some coefficient in that column. If there is another row name and value for the same column, they may appear as *Name2, Value2* on the same line, or they may appear in the next line.

eg.

```
    COLUMNS
         X01            MAXBAND
         X01            LC001
         X02            MAXBAND
         X02            LC001
```

The RHS Section: This section specifies the elements of the right hand side of the constraints. Only the nonzero coefficients need to be specified. They may appear in any order. The format is exactly the same as the COLUMNS section, with *Name0* giving a name to the right-hand side.

eg. RHS

```
              RHS01   LC001   7.5
```

The RANGES Section: Ranges are used for constraints of the form

$$l \leq a^T \leq u,$$

where both l and u are finite. The range of the constraint is $r=u-l$. Either l or u is specified in the RHS sections (as b, say), and r is defined in the RANGES section. The resulting l and u depend on the row-type of the constraint and the sign of r as follows:

Row-type	*Sign of r*	*Lower limit, l*	*Upper limit, u*
E	+	b	$b+\|r\|$
E	-	$b - \|r\|$	b
G	+ or -	b	$b + \|r\|$
L	+ or -	$b - \|r\|$	b

The format is exactly the same as in the COLUMNS section, with *Name0* giving a name to the range set.

The BOUNDS Section (Optional): The default bounds on all variables x_j (excluding slacks) are $0 \leq x_j \leq \infty$. If necessary, the default values 0 and ∞ can be changed in the SPECS file to $l \leq x_j \leq u$ by the LOWER and UPPER keywords respectively. In this section *Key* gives the type of bound required, *Name0* is the name of the bound set, and *Name1* and *Value1* are the column name and bound value. (*Name2* and *Value2* are ignored). Let l and u be the default bounds just mentioned, and let x and b be the column and value specified. The various bound-types allowed are as follows:

Key	*Bound-type*	*Resulting bounds*
LO	Lower bound	$b \leq x \leq u$
UP	Upper bound	$l \leq x \leq b$
FX	Fixed variable	$b \leq x \leq b$
FR	Free variable	$-\infty \leq x \leq \infty$
MI	Minus infinity	$-\infty \leq x \leq u$
PL	Plus infinity	$l \leq x \leq \infty$

AN EXAMPLE:

```
NAME              EXAMPLE           ROWS
 N                COST
 E                ROW1
 L                ROW2
 L                ROW3
COLUMNS
     COLUMN1      COST              -7.0
     COLUMN1      ROW1              -1.0
     COLUMN1      ROW2               5.0
     COLUMN1      ROW3              -2.0
     COLUMN2      COST              -2.0
     COLUMN2      ROW1               2.0
     COLUMN2      ROW2               1.0
     COLUMN2      ROW3              -2.0
     COLUMN3      ROW1              16.0
RHS
     DEMANDS      ROW1               4.0
     DEMANDS      ROW2              20.0
     DEMANDS      ROW3              -7.0
ENDATA
```

APPENDIX C

INTEGER VARIABLE SPECIFICATION FILE

MNMINOS.MPS and DLMINOS.MPS use a format similar to that found in an MPS file; there is a header line, an end-of-data line, and the lines in between must have the following data format:

Columns	5-12	25-36
Contents	*Variable name*	*Increment value*

In other words, the variable name is placed left justified in columns 5-12 of the line, and the increment value is placed in columns 25-36. The increment value used in MAXBAND Version 3.1 is 1.

AN EXAMPLE:

```
INTEGERS
    COLUMN1                 1.0
    COLUMN2                 1.0
ENDATA
```

APPENDIX D

GETTING STARTED

Installing the MAXBAND Version 3.1 System

Use the DOS 'copy' command to transfer all the files from the diskette. Unzip all the files in 'MAXBND31.ZIP' using the command 'PKUNZIP MAXBND31'.

The following files will be created in the current directory:

- all the source modules of the system
- a 'makefile', that can be used for compiling and linking the FORTRAN source code using Lahey FORTRAN extended memory version 5.10.
- MAXBAND.EXE, the executable code
- 'MINOS.SPC' file which is needed to run the system
- all the data files mentioned in the tables of the result.

Running MAXBAND Version 3.1 System

Type 'MAXBAND' at the DOS prompt. This will start the MAXBAND Version 3.1 system running. The default option on all the data sets is the heuristic method (except arterial problems). In order to exercise the optimal solution method of MAXBAND Version 3.1 the input data file will have to be altered. A flag should be reset in the input file. In the data line 'MPCODE', columns 51-55 is used for this purpose. A '1' in one of these columns will indicate that the optimal solution technique is to be used to solve the problem, otherwise the heuristic approach is used to solve that problem.

Output Results

The results of the run, i.e. the best signal timing plan, is printed on the output file specified at the beginning of this run by the user.

Software Availability and Technical Support

The new release, MAXBAND Version 3.1, software is available from *MCTrans* and *Pc-trans*. For any technical questions, comments, and suggestions please contact the main author at the address given below either via letter, phone or e-mail.

IVHS Research Group
Center for Transportation Analysis, ORNL
P.O. Box 2006, MS 6206
Oak Ridge, TN 37831
PH: (615) 574 1402
FAX: (615) 574 0202
e-mail: puv@ornl.gov

INTERNAL DISTRIBUTION

1.	M. S. Bronzini	18.	A. C. Schaffhauser
2.	J. B. Cannon	19.	R. Shelton
3.	O. Franzese	20.	R. Solanki
4.	S. Gordon	21.	F. Southworth
5.	D. Greene	22.	S. Stevens
6.	R. Honea	23.	P. Wolf
7.	S. Joshi	24.	ORNL Patent Office
8.	H. Knee	25.	Central Research Library
9.	S. P. Miaou	26.	Document Reference Section
10.	D. P. Middendorf	27-29.	Laboratory Records
11.	R. S. Pillai	30.	Laboratory Records-RC
12-17.	A. K. Rathi		

EXTERNAL DISTRIBUTION

31. Christina Andrews, Manager, Transportation Research , Farradyne Systems, Inc., 3204 Tower Oaks Boulevard, Rockville, MD 20852

32. Dr. Douglas R. Bohi, Director, Energy and Natural Resources Division, Resources for the Future, 1616 P Street, N.W., Washington, DC 20036

33. James E. Clark, Federal Highway Administration, HSR-11, 6300 Georgetown Pike, McLean, Virginia, 22101-2296

34. Dr. Steve Cohen, Private Consultant, P.O. Box 7322, Falls Church, Virginia 22046

35. Dr. Thomas E. Drabek, Professor, Department of Sociology, University of Denver, Denver, Colorado 80208-0209

36. Mr. Calvin D. MacCracken, President, Calmac Manufacturing Corporation, 101 West Sheffield Avenue, P.O. Box 710, Englewood, New Jersey 07631

37. Alberto J. Santiago, Federal Highway Administration, Traffic Systems Division, HSR-10, 6300 Georgetown Pike, McLean, Virginia 22101-2296

38. Mr. George F. Sowers, P.E., Senior Vice President, Law Companies Group, Inc., 114 Townpark Drive, Suite 250, Kennesaw, Georgia 30144-5599

39. Dr. C. Michael Walton, Ernest H. Cockrell Centennial Chair in Engineering and Chairman, Department of Civil Engineering, University of Texas at Austin, Austin, Texas 78712-1076

40-41. OSTI, U.S. Department of Energy, P.O. Box 62, Oak Ridge, TN 37831

42. Office of Assistant Manager of Energy Research and Development, DOE/ORO, P.O. Box 2008, Oak Ridge, TN 37831-6269