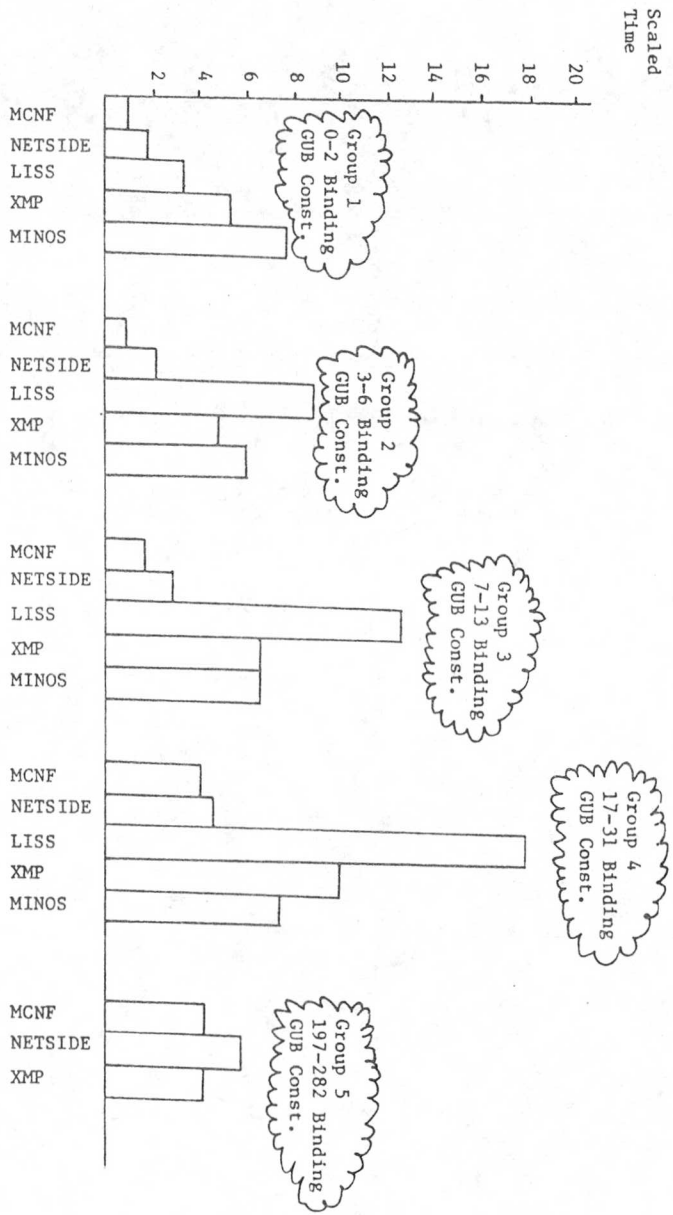

Figure 1. Summary of Computational Experience on 5 Sets of Problems With 5 Problems In Each Set (All problems have from 400 to 600 rows and from 700 to 1000 columns.)

COMPUTATIONAL EXPERIENCE WITH LARGE-SCALE CONSTRAINED GLOBAL OPTIMIZATION*

J.B. Rosen
University of Minnesota

1. Introduction

Significant progress has been made during the past two years in the solution of certain large scale constrained global minimization problems. The class of problems of primary interest here is that of finding the global minimum of a concave, quadratic function on a convex polytope. A variety of important applications lead to such a formulation, including plant location with economies of scale, quadratic assignment problems, and fixed charge problems (see for example [Hei81]). In fact, any zero-one integer or mixed integer linear programming problem can be converted to an equivalent problem of the type considered here by the use of a quadratic penalty function [Rag69, Kal82].

The problem of interest here can be stated as follows:

$$\text{(GM)} \qquad \underset{(x,y)\in P}{\text{global min}}\ \psi(x, y)$$

where

$$P = \left\{(x, y) \;\middle|\; \begin{matrix} A_1x + A_2y = b \\ x, y \geq 0 \end{matrix}\right\}$$

is a polytope in R^{n+k},

$$\psi(x, y) = \phi(x) + d^Ty,$$

and $\phi(x)$ is a concave quadratic function, given by

$$\phi(x) = c^Tx - \frac{1}{2}x^TQx,$$

where Q is a positive semi-definite symmetric matrix. We assume that $x \in R^n$, $y \in R^k$, $A_1 \in R^{m\times n}$ and $A_2 \in R^{m\times k}$. Since P is, by assumption, nonempty and bounded, the global minimum function value ψ^* will be attained at a vertex (x^*, y^*) of P, with $\psi(x^*, y^*) = \psi^*$.

It is well known that, as with a completely linear function, the global minimum of a concave function will occur at a vertex of P. In fact, every local minimum function value is attained at a vertex, and problems can easily

* This research was support in part by the National Science Foundation under Research Grant MCS8101214.

be constructed such that every vertex of P is a local minimum. Because of this, it may be necessary, in the worst case, to evaluate the objective function at a significant fraction of all the vertices in order to find the global minimum. This combinatorial aspect is confirmed by the fact that even the special case of finding the global minimum of a concave, quadratic function on the unit cube is known to be NP-hard (unless Q is diagonal or has other special properties).

In earlier published work only the case corresponding to $k = 0$ has been considered (see for example [Fal76]), that is, all variables are treated in the same manner. We will refer to this as the small scale case. Algorithms developed for this small scale case may, at worst, be exponential in the number of variables n. Therefore the maximum size problem which can be solved computationally in a reasonable time is definitely limited. Computational results quoted in the literature [Hei81] for this type of problem seem to be limited to $n \leq 25$. Somewhat larger problems of this type have been solved by us recently using the algorithm described in [Ros83A], but even this method is limited (probably to $n \leq 100$), when all variables are treated similarly.

For the practical solution of large scale global minimization problems one should consider the related situation in convex minimization and zero-one integer linear programming. Large scale, linearly constrained minimization problems with a convex objective function can be solved, provided most of the variables appear only linearly in the objective function. In that case a method such as MINOS [Mur78], which treats the linear variables in a different manner than those appearing nonlinearly, can be used efficiently. Similarly, large scale zero-one mixed integer linear programming problems can be solved in a reasonable time provided most of the variables are continuous.

A closely related approach is proposed for solving large scale constrained global minimization problems, by taking full advantage of the linearity of the y variables. Using such an approach it should be possible in a reasonable time to solve problems with 500 or more linear variables ($k \geq 500$), provided that the number of nonlinear variables is limited to 100 or less ($n \leq 100$).

The basic idea used is to initially determine a rectangular domain $R_x \subset R^n$, which contains the projection P_x of P on the x-space. In fact, we can find the smallest such rectangular domain by solving a single multiple-cost-row linear program with 2n cost vectors. The cost vectors used are $\pm u_i$, $i = 1, \ldots, n$, where the u_i are the orthogonal eigenvectors of Q. The rectangular domain R_x

8. Helgason, R., and J. Kennington, "A Product Form Representation of the Inverse of a Multicommodity Cycle Matrix", Networks, 7, 297-322, (1977).

9. Kennington, J., "A Primal Partitioning Code for Solving Multicommodity Network Flow Problems", Technical Report No. 79008, Department of Operations Research, Southern Methodist University, Dallas, Texas, (1979).

10. Kennington, J., and R. Helgason, Algorithms for Network Programming, John Wiley and Sons, New York, New York, (1980).

11. Marsten, R., "XMP: A Structured Library of Subroutines for Experimental Mathematical Programming", Technical Report No. 351, Department of Management Information Systems, The University of Arizona, Tucson, Arizona, (1979).

12. Marsten, Roy E., "The Design of the XMP Linear Programming Library", ACM Transactions on Mathematical Software, 7, 4, 481-497, (1981).

13. Murtagh, B. A., and M. A. Saunders, "MINOS User's Guide", Technical Report 77-9, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California, (1977).

14. Murtagh, B. A., and M. A. Saunders, "Large-Scale Linearly Constrained Optimization", Mathematical Programming, 14, 41-72, (1978).

We do not claim that these are the best times that can be achieved from any of these five codes. Almost certainly some fine tuning could produce lower times for all of these systems. Since we have used the default parameters for all systems, we believe that this is a fair comparison and that there is no particular bias for any of the codes tested. This computational experience presents our best judgement as to the expected savings for using specialized as opposed to general software for multicommodity problems.

Note:

This paper is a condensed version of [1] which has been submitted for publication elsewhere. Professor Robert R. Meyer invited us to prepare this paper for the COAL Newsletter. The complete version of this manuscript may be obtained by writing to the authors.

References

1. Ali, I., D. Barnett, K. Farhangian, J. Kennington, B. McCarl, B. Patty, B. Shetty, and P. Wong, "Multicommodity Network Problems: Applications and Computations", Technical Report 82-OR-4, Department of Operations Research, Southern Methodist University, Dallas, Texas 75275.

2. Ali, A., and J. Kennington, "MNETGN Program Documentation", Technical Report No. 77003, Department of Operations Research, Southern Methodist University, Dallas, Texas (1977).

3. Ali, A., and J. Kennington, "LISS: An-Incore Primal Simplex Code for Solving Linear Programs", Technical Report No. 80015, Department of Operations Research, Southern Methodist University, Dallas, Texas (1980).

4. Ali, A., R. Helgason, J. Kennington, and H. Lall, "Computational Comparison Among Three Multicommodity Network Flow Algorithms", Operations Research, 28, (4), 995-1000, (1980).

5. Barr, R. S., K. Farhangian, and J. L. Kennington, "Networks With Side Constraints: An LU Factorization Update", Technical Report 83-OR-4, Department of Operations Research, Southern Methodist University, Dallas, Texas, (June 1983).

6. Hartman, J., and L. Lasdon, "A Generalized Upper Bounding Algorithm for Multicommodity Network Flow Problems", Networks, 1, 333-354, (1972).

7. Helgason, R., and J. Kennington, "Spike Swapping in Basis Reinversion", Naval Research Logistics Quarterly, 27, 4, 697-702, (1980).

is then constructed with each pair of opposite faces parallel to one of the hyperplanes $u_i^T x = 0$. A linear underestimating function $\Gamma(x)$ to $\phi(x)$ is then easily computed, which has the property that

$$\Gamma(w_j) = \phi(w_j), \qquad j = 1, \ldots, 2^n$$

and

$$\Gamma(x) \leq \phi(x), \qquad \text{all } x \in R_x$$

where the w_j are the vertices of R_x. The following linear program is then solved:

$$\text{(LM)} \qquad \min_{(x,y)\in P} \Gamma(x) + d^T y.$$

The solution to (LM) will give a vertex, $\bar{v} = (\bar{x}, \bar{y})$, of P, which is a candidate for the global minimum, and has the function value $\psi(\bar{x}, \bar{y}) = \phi(\bar{x}) + d^T\bar{y}$, so that $\psi(\bar{x}, \bar{y})$ is an upper bound for ψ^*. Since $\Gamma(x)$ underestimates $\phi(x)$ on $P_x \subset R_x$, we also have that $\Gamma(\bar{x}) + d^T\bar{y} \leq \Gamma(x) + d^T y \leq \psi(x, y)$ for all $(x, y) \in P$. Thus $\Gamma(\bar{x}) + d^T\bar{y}$ is a lower bound for ψ^*. We now have $\Gamma(\bar{x}) + d^T\bar{y} \leq \psi^* \leq \psi(\bar{x}, \bar{y})$. If these lower and upper bounds are close enough, we may take $\psi(\bar{x}, \bar{y})$ as an acceptable approximation to the global minimum value, and $(\bar{x}, \bar{y})$ as an acceptable global minimum vertex. If not acceptable, we partition R_x into two or more subdomains, get an improved linear underestimating function over each subdomain, and solve (LM) again for each subdomain. This leads to a branch and bound algorithm with a strict decrease in the difference between the upper and lower bounds at each iteration.

2. Summary of Computational Results

The initial research objectives were the development of one or more algorithms suitable for solving large scale global minimization problems, implementation on a suitable computer system, and computational testing to determine the robustness and average behavior of each algorithm. A new algorithm for the small scale case ($k = 0$) was proposed [Ros83A] and has served as a starting point for subsequent work on the large scale problem. In that paper, the construction of an appropriate rectangular domain containing the polytope P was proposed, and its use to obtain a lower and upper bound on ψ^* described. This algorithm was implemented and tested computationally on small scale test problems with known global minimum solutions. Computational results for problems with $n \leq 100$ were obtained using this implementation on the Cyber 74/730 at the University of Minnesota Computing Center [Zil83]. All such problems were solved in CPU times less than 90 seconds. The small scale test problems used were constructed by a method described in [Sun82], which generates nontrivial problems with known global minima.

The results obtained with this algorithm for the small scale case motivated subsequent research on the extension to large scale problems with many linear variables. The key to this extension is the construction of the smallest rectangular domain R_x, which contains the projection P_x, of P, on the x-space. The construction of P_x itself is not practical, but as discussed earlier, R_x is readily obtained by solving a multiple-cost-row linear program. A linear function $\Gamma(x)$ is then easily computed which underestimates $\phi(x)$ on R_x, and therefore on P_x. This approach has the practical advantage that all calculations which might grow exponentially are limited to the relatively low-dimensional x-space.

The idea of constructing a linear underestimating function on R_x was first proposed in [Ros83B], in a somewhat different form. Its present form, as described above, is due to Kalantari and has been implemented in his PhD thesis, now being completed [Kal84]. Both the original form and that developed by Kalantari have been implemented by Zilverberg in his PhD thesis [Zil83]. As summarized at the end of the previous section the solution of the linear

form [8]. This working basis inverse has dimension equal to the number of binding GUB constraints. The initial basis is crashed using a multicommodity variation of the routine used in NETFLO [10]. A partial pricing scheme is used. The pricing tolerance is 1.E-6, while the pivot tolerance is 1.E-8.

2.2 Computational Experimentation

Twenty-five test problems were randomly generated using MNETGN [2]. These problems were the largest that could be solved by all five codes under a core storage limitation of 201K octal words. None of the codes was fine tuned for the problem set and the default parameters were used for all codes. The problems varied from pure network problems to problems with over 90% of the rows corresponding to GUB constraints.

A summary of our computational experience is shown in Figure 1. Each block represents the total solution time for five test problems. Note that MCNF dominates NETSIDE which dominates the LP codes except on those problems which have a large number of binding GUB constraints. Our experience on real world models indicates that those models are like those in groups I through IV rather than like the group V problems (see [1]). All runs were made on the CDC 6600 at Southern Methodist University using the FTN compiler with the optimization feature enabled. The times exclude input and output. Letting T(ALG) denote the CPU time required to solve the first 20 test problems using code ALG, the dominance relationships among these codes is given as follows:

T(LISS) = 5.24 T(MCNF)
T(LISS) = 3.55 T(NETSIDE)
T(MINOS) = 3.36 T(MCNF)
T(XMP) = 3.29 T(MCNF)
T(NETSIDE) = 1.48 T(MCNF)

Note that NETSIDE performs better when compared to MCNF as the number of binding linking constraints increases. This is because NETSIDE is tuned for the more difficult problems. MCNF and NETSIDE use the same algorithms and only slightly different data structures for the network part of the problem.

There were three problems which were attempted but not reported in Figure 1. On two of these problems, the optimal objective values obtained by XMP and MINOS differed by more than 10%. On the third problem, XMP experienced numerical difficulties and terminated with the message that the problem became infeasible during phase II. For the 25 problems of Figure 1, the objective values were all within 2% on all codes.

2.1 Description of Codes

MINOS (Murtagh and Saunders [13,14]) -- For linear programs, MINOS uses the revised simplex algorithm with all data and instructions residing in core storage. The basis inverse is maintained as an LU factorization using a Bartels-Golub update. The reinversion routine uses the Hellerman-Rarick pivot agenda algorithm. The initial basis was obtained by a CRASH routine. The inversion frequency was 50 iterations and the partial pricing scheme uses a single block. The pricing tolerance was 1.E-6 and the pivot tolerance was 8.43E-8. For our experiments, no problem required more than 201K octal words of core storage.

XMP (Marsten [11,12]) -- XMP is a library of FORTRAN subroutines which can be used to solve linear programs. The basis inverse is maintained in LU factored form. The Harwell Library LA05 routines are used in the execution of all operations involving the basis inverse. The initial basis consists of a combination of artificial and slack variables. The pricing routine uses a candidate list of size 6 with 200 columns being scanned each time the list is refreshed. The basis is reinverted every 50 iterations. Both the pricing and pivot tolerance are 1.E-10. For our experiments, no problem required more than 200K octal words of core storage.

LISS (Ali and Kennington [3]) -- LISS is an in-core LP solver with the basis inverse maintained in product form. The reinversion routine is a modification of the work of Hellerman and Rarick, incorporating a technique known as "splitting the bump" and uses the "spike swapping theory" of [7]. The basis inverse is refactored every 50 iterations. The initial basis consists of all artificials. A partial pricing scheme is used with 20 blocks. Both the pricing and pivot tolerance is 1.E-8.

NETSIDE (Barr, Farhangian, and Kennington [5]) -- NETSIDE is designed to solve the network with general side constraints model. It uses a specialization of the revised simplex method known as the primal partitioning algorithm. The basis inverse is maintained as a rooted spanning tree and a working basis inverse in product form. The reinversion routine is a modification of the work of Hellerman and Rarick, using the "spike swapping theory" of [7]. The initial working basis consists of a combination of artificial and slack variables. The working basis is reinverted every 60 iterations. The pricing routine uses a candidate list of size 6 with block size of 200. Both the pricing and pivot tolerances are 1.E-6.

MCNF (Kennington [4,9]) -- MCNF is designed to solve multicommodity network flow problems. It uses a specialization of the revised simplex algorithm. The basis inverse is maintained as a set of rooted spanning trees (one for each commodity) and a working basis inverse in product

program (LM) gives both a lower and an upper bound on ψ^*, and a feasible vertex $\bar{v}$, which is a candidate for the global minimum point.

In view of the difficulty of the problem (GM), it is not reasonable to insist on the exact solution, that is, a vertex at which the exact minimum function value is attained. Instead, we accept an ε-approximate solution defined in terms of a bound on the difference $\psi(\hat{x}, \hat{y}) - \psi^*$, where $(\hat{x}, \hat{y})$ is a current feasible vertex with minimum function value. Let $\Delta\phi(x) = \phi(x) - \Gamma(x)$, where $\Gamma(x)$ is the linear underestimating function previously described. Since we have

$$\Gamma(\hat{x}) + d^T y \le \psi^* \le \phi(\hat{x}) + d^T\hat{y} = \psi(\hat{x}, \hat{y})$$

it follows that

$$\psi(\hat{x}, \hat{y}) - \psi^* \le \Delta\phi(\hat{x}).$$

Thus if $\Delta\phi(\hat{x})$ is sufficiently small, we consider $\psi(\hat{x}, \hat{y})$ to be an acceptable approximation to the global minimum function value, with $(\hat{x}, \hat{y})$ as the corresponding approximate global minimum vertex.

In order to determine a satisfactory small value for $\Delta\phi$, a scaling factor is needed since ψ^* may have any value, including $\psi^* = 0$. A suitable scaling factor is the (approximate) range $\Delta\phi_{max}$ of ϕ on R_x, given by $\Delta\phi_{max} = \phi_{max} - \phi(\hat{x})$, where ϕ_{max} is an easily computed approximation to the maximum of ϕ on R_x. The vertex $(\hat{x}, \hat{y})$ is then accepted when $\Delta\phi(\hat{x}) \le \varepsilon\Delta\phi_{max}$, where ε is a specified tolerance. The value $\psi(\hat{x}, \hat{y})$ is called an ε-approximate solution to the problem (GM).

One of the surprising computational results obtained by both Zilverberg and Kalantari is that in a majority of the test cases run the vertex $\bar{v}$ obtained by (LM) gave a function value only slightly larger than ψ^* and in some cases was an ε-approximate solution. The confirmation of this fact however often required many additional iterations, in order to determine an improved lower bound that was sufficiently close to ψ^*.

The algorithms developed and implemented by Zilverberg and Kalantari are both similar in their initial phase, which finds a vertex $\bar{v}$, as previously described. Following this initial phase, Zilverberg's algorithm continues by partitioning R_x into 2n pyramids. We summarize this partitioning for the usual case where $\bar{v}$ is an interior point of R_x. Each pyramid P_i, $i = 1, \ldots, 2n$, has as its base one of the 2n faces of R_x, and has $\bar{v}$ as its common apex. The

union of these pyramids is the rectangular domain R_x. As shown by Zilverberg, a linear function $L_i(x)$ can easily be computed which underestimates $\phi(x)$ on P_i, and such that $L_i(\bar{v}) = \phi(\bar{v})$, and $L_i(w_j) = \phi(w_j)$, $j = 1, \ldots, 2^{n-1}$, where the w_j are the vertices of R_x contained in the face which forms the base of P_i. It follows from the concavity of ϕ that $L_i(x) \leq L_j(x)$, for any $x \in P_i$, and $j = 1, \ldots, 2n$. Furthermore, $\Gamma(x) \leq L_i(x) \leq \phi(x)$, for any $x \in P_i$, so that $L_i(x)$ is an improved underestimator for ϕ on the pyramid P_i. Therefore solving another multiple-cost-row linear program will give an improved lower bound, and usually an improved global minimum candidate (if $\bar{v}$ is not in fact the global minimum). More specifically, this is obtained by solving

$$\min_{(x,y)\in P} \; L_i(x) + d^T y, \qquad i = 1, \ldots, 2n.$$

Corresponding to each value of i, a vertex of P is obtained. Denote by $\hat{v}$ that vertex with the minimum function value. If $\hat{v} = \bar{v}$, then $\bar{v}$ is the global minimum point. If not, construct the pyramids with $\hat{v}$ as their common apex, and repeat.

The algorithm developed and implemented by Kalantari is somewhat simpler to describe. After the initial phase, the rectangular domain R_x is bisected with a cut parallel to one of its faces. A linear underestimating function is then obtained for each of the two resulting rectangular subdomains, and the problem (LM) solved on each. This bisection and solution of (LM) on each subdomain is then repeated at each interation, giving a branch and bound algorithm with an improved bound at each iteration [Kal84]. Kalantari shows that the difference in function values satisfies the following bound

$$\Delta\psi \equiv \psi(\bar{x}, \bar{y}) - \psi^* \leq \frac{1}{8} \sum_{i=1}^{n} \lambda_i \Delta\beta_i^2$$

where the λ_i are the (real, nonnegative) eigenvalues of Q, and the $\Delta\beta_i$, $i = 1, \ldots, n$ are the lengths of the edges of R_x. The cut is chosen so as to bisect that axis of R_x corresponding to the largest value of $\lambda_i \Delta\beta_i^2$, since this gives the greatest decrease in the bound at each iteration.

The algorithm for the large scale problem (GM) developed by Zilverberg has been implemented by him on the Cyber 74/730 at the University of Minnesota Computing Center (UCC). He tested it on 130 problems generated for this purpose. These computational results are presented in his thesis [Zil83], and are summarized below. The algorithm developed by Kalantari for the problem (GM) has been implemented by him on the Sperry 1100/80 series computer system. Machine time on this system was made available by courtesy of Sperry, as part of their contribution to the Industry/University Cooperative Research Program under which this research was supported.

$\underline{u}^k$ is a n^k column vector of individual upper bounds,

$\underline{0}$ is a vector of zeroes of proper dimension,

R_j is an index set giving the commodities for which arc j appears in a mutual capacity constraint, and

$\underline{x}^k$ is a n^k column vector of decision variables.

The type (4) constraints link the commodities and are called linking constraints or generalized upper bounding (GUB) constraints. We assume that each arc appears in, at most, one GUB constraint.

The linear program (1) - (3) is called the minimal cost network flow problem. It is well known that bases for (2) and (3) are lower triangular and correspond to spanning trees. Consequently, the two primal simplex computations which involve the basis inverse can be performed very efficiently using labeling operations directly on the basis trees. Specialized primal simplex software which exploits these and other observations has been developed. Computational experimentation has shown this software to be from one to two orders of magnitude faster than general purpose linear programming systems.

Bases for multicommodity problems are more complicated than bases for pure network problems. However, these bases can be partitioned in order to preserve part of the structure found in the pure problem. These bases may be partitioned as follows:

$$\begin{bmatrix} B & C & \\ \hline D & E & \\ F & G & I \end{bmatrix} \quad \begin{matrix} \} \text{ Network Constraints} \\ \} \text{ GUB Constraints} \end{matrix}$$

where B is lower triangular and corresponds to a spanning tree. The row size of E is the number of binding GUB constraints corresponding to that basis and I is an identity matrix. If the number of GUB constraints is small relative to the number of network constraints, then the algorithms used to develop efficient software for the pure network problem can be partially used to solve the multicommodity problem efficiently. If the number of binding GUB constraints is small relative to the number of GUB constraints, then additional efficiencies result.

II. Computational Comparison among General Purpose Codes and Specialized Code

We have made a computational comparison among three well-known linear programming codes, our specialized multicommodity code, and our specialized network with side constraint code. All codes are written in FORTRAN and have not been tailored to either our equipment or our FORTRAN compiler. None of the codes were tuned for this problem set.

A COMPUTATIONAL COMPARISON OF SPECIALIZED VERSUS GENERAL CODES FOR MULTICOMMODITY NETWORK PROBLEMS

by

Jeff L. Kennington and Bruce W. Patty

Department of Operations Research
Southern Methodist University
Dallas, Texas 75275
(214) 692-3072

It is well documented that pure network problems can be solved from ten to one hundred times faster using specialized primal simplex software as compared to general linear programming systems. For multicommodity network flow problems, the computational savings are a function of the number of tight side constraints. In this study, we compare a specialized multicommodity network code with a network with side constraint code and three linear programming codes. We conclude that our specialized multicommodity network code is three times as fast as a general code, while a specialized network with general side constraints code has twice the speed of a general LP code.

I. Introduction

The linear multicommodity network flow problem is a special structured linear program which may be stated as follows:

$$\text{minimize} \quad \sum_k \underline{c}^k \underline{x}^k \tag{1}$$

$$\text{subject to} \quad A^k \underline{x}^k = \underline{r}^k \ , \ (\text{all } k) \tag{2}$$

$$\underline{0} \le \underline{x}^k \le \underline{u}^k, \ (\text{all } k) \tag{3}$$

$$\sum_{k \in R_j} x_j^k \le b_j \ , \ (\text{all } j) \tag{4}$$

where k is the commodity index,

A^k is a m^k x n^k node-arc incidence matrix,

$\underline{c}^k$ is a n^k row vector of unit costs,

$\underline{r}^k$ is a m^k column vector of node requirements,

b_j is a scalar (called the mutual arc capacity for arc j),

The results of the computational testing to date show that those problems for which the maximum of ϕ occurs at an interior point of P are more difficult to solve. This is expected since ϕ may then have constrained local minima in any direction. It has also been confirmed that problems for which $\phi(x)$ dominates the linear term are more difficult. Thus the most difficult problems of type (GM) are those for which ϕ has an interior maximum point and $d = 0$ (or is relatively small).

A theoretical analysis of the computational performance of approximate algorithms of this type has recently been completed [Ros83D]. It is shown there that with reasonable assumptions the total number of flops F_T, required to obtain an ε-approximate solution, satisfies the bound

$$F_T \le m^2(n+k)\delta\mu(n, \varepsilon)$$

where δ is the fractional density of the matrix $[A_1 \ A_2]$, and $\mu(n, \varepsilon)$ depends on n, ε and the specific algorithm used, but is independent of m and k. Under suitable conditions it is shown that $\mu(n, \varepsilon) = O(\frac{1}{\varepsilon})$, for fixed n. The computational results obtained so far confirm the predicted behavior with respect to k, that is, the computing time increases linearly with k, for fixed m, n and ε. This behavior is important if large scale problems with $k >> n$ are to be solved in a reasonable time. Extensive additional computational testing will be required to determine if the theoretical analysis [Ros83D] is valid for these algorithms with respect to n and ε.

Comparing the two algorithms, it is found that Zilverberg's algorithm requires more computation per iteration, but generally obtains a good upper bound (and corresponding feasible vertex) more quickly than Kalantari's algorithm. However, Kalantari's algorithm is more efficient in finding a good lower bound, so that $\Delta\phi$ is reduced more rapidly in his algorithm. Further computational testing on larger problems is needed to properly evaluate the relative merits of these two algorithms.

References

(Fal76) Falk, J.E. and K.R. Hoffman, "A successive underestimation method for concave minimization problems," Math. Oper. Res. 1, 251-259.

(Hei81) Heising-Goodman, C.D., "A survey of methodology for the global minimization of concave functions subject to convex constraints", OMEGA, Intl. Jl. of Mgmt. Sct. 9, 313-319.

(Kal82) Kalantari, B. and J.B. Rosen, "Penalty for zero-one integer equivalent problem," Math. Programming 24, 229-232.

(Kal84) Kalantari, B., "Large scale concave quadratic minimization and extensions," PhD thesis, Computer Science Dept., Univ. of Minnesota, March, 1984.

(Mar81) Marsten, R.E. "The disign of the XMP linear programming library," ACM Trans. Math. Softw. 7, 481-497.

(Mur78) Murtagh, B.A. and M.A. Saunders, "Large scale linearly constrained optimization,"Math. Programming 14, 41-72 .

(Rag69) Raghavachari, M., "On connection between zero-one integer programming and concave minimization under linear constraints," Operations Research 17, 680-684

(Ros83A) Rosen, J.B., "Global minimization of a linearly constrained concave function by partition of feasible domain," Math. Oper. Res. 8, 215-230.

(Ros83B) Rosen, J.B. "Parametric global minimization for large scale problems," Tech. Rept. 83-11 (revised), Computer Science Dept., Univ. of Minnesota.

(Ros83C) Rosen, J.B. and B. Kalantari, "Construction of large scale global minimum concave quadratic test problems." To be published in J. Optim. Theory & Applic.

(Ros83D) Rosen, J.B., "Performance of approximate algorithms for global minimization." To be published in Proceedings of Oberwolfach Conference on Mathematical Programming.

(Sun82) Sung, Y.Y. and J.B. Rosen, "Global minimum test problem construction," Math. Programming 24, 353-355

(Zil83) Zilverberg, N., "Global minimization for large scale linearly constrained systems," PhD thesis, Computer Science Dept., Univ. of Minnesota, September 1983.

NAME OF THE CODE:
AUTHOR(S):
MATHEMATICAL PROBLEM:
DOMAIN OF APPLICATIONS:
MATHEMATICAL METHOD:
PROGRAMMING LANGUAGE:
COMPUTER SYSTEMS WHERE THE CODE HAS BEEN IMPLEMENTED:
SPECIAL PROGRAM FEATURES:
PRACTICAL APPLICATION PROBLEMS SOLVED BY THE CODE:
CONDITION OF AVAILABILITY:
CHARGE:
ADDRESS FROM WHERE THE CODE CAN BE ORDERED:
REFERENCES:

Please support our efforts to establish a qualified and extensive software fair. Feel free to distribute this request to other colleagues who might be interested in a participation. Address your response to:

Klaus Schittkowski
Institut für Informatik
Universität Stuttgart
Azenbergstr. 12
D-7000 Stuttgart 1, Germany F.R.

SOFTWARE FAIR

Klaus Schittkowski

The Committee on Algorithms organizes a NATO Advanced Study Institute (ASI) on Computational Mathematical Programming to be held in Bad Windsheim, W. Germany, from July 23 to August 2, 1984. Please confer the preceding Newsletter for details. The registrations show that a large percentage of participants are practioneers who need optimization algorithms as a tool in their application models, e.g. in mechanical engineering or operations research. Their main motivation is to learn new solution methods for solving optimization problems, in particular they are very much interested in obtaining some information about corresponding computer programs. Therefore COAL plans to establish a software fair in the sense that information material about available optimization codes is to be layed out during the ASI and questionnaires are to be published in the conference proceedings. We all know that it is quite difficult to spread out software information to possible users, and COAL offers the chance for making some advertising for mathematical programming codes.

If you want to participate in the software fair, then two actions are required:

a) Submit three copies of information material (e.g. code descriptions, user's guides, reports), which will be layed out during the ASI. Please understand that the available conference budget is limited and cannot be used for refunding your expenses.

b) Type one page of condensed information for each code you want to describe, which is to be retyped and published in the proceedings of the ASI. The form should contain the following items:

ON THE LENGTHS OF SIMPLEX PATHS

Snjólfur Ólafsson and P.O. Lindberg
Department of Mathematics, Royal Institute of Technology,
S-100 44 Stockholm

1. Introduction

This is a short presentation of an experimental study of the number of iterations for the Stochastic Simplex Method on Assignment and Transportation Problems (reported in Lindberg & Ólafsson (1980, 1983) and Ólafsson & Lindberg (1983, 1984)).
It is wellknown that the Simplex Method is very efficient in practice, but also that there are worst case problem classes giving iteration counts exponential in the data.

It was and is our belief that a large part of the practical efficiency of the simplex method may be ascribed to the existence of many short paths from any starting point to the optimum. To test this hypothesis we have employed the Stochastic Simplex Method, which chooses the variable to enter the basis completely at random among all variables with negative reduced cost. With a Simplex Path we will understand a basis sequence obtainable by the Stochastic Simplex Method. Low iteration counts for this method imply that simplex paths are short on the average, which in turn explains why the Simplex Method is so efficient in practice.

In choosing problem classes to study we have started with Assignment and Transportation Problems. The reason for this is threefold.
First one needs comparatively few parameters to define problems of these classes, which means less parameters to control in the experiments. Second, they have a well defined structure; taking a general LP problem it would be hard to choose a structure that is typical. Finally, these problem classes are important in practical applications.

The results we have obtained strongly confirm our beliefs on simplex path lengths.
For $d \times d$ Assignment Problems we found, by experiments on 17 problems with d between 20 and 230, that the mean and the standard deviation

of the iteration count for the stochastic simplex method behave as

$$\mu_d = 1.19\ d^{1.83} \quad \text{and}$$

$$\sigma_d = 0.32\ d^{1.51} \quad \text{respectively.}$$

For the standard simplex method (in the form of PNET) on the other hand the iteration count grows as

$$\mu_d^s = 1.10\ d^{1.57}.$$

Similarly, for the iteration count on m x n Transportation Problems we get

$$\mu_{m,n} = 1.69\ (mn)^{.59}(m+n-1)^{.44} \quad \text{and}$$

$$\sigma_{m,n} = 0.79\ (mn)^{.57}$$

for the mean and standard deviation for the stochastic simplex method and

$$\mu_{m,n}^s = 1.40\ (mn)^{.68}(m+n-1)^{.09}$$

for PNET. This is based on 27 problems with up to 22500 variables.

Hence we see, both for Assignment and Transportation Problems, that a large part of the efficiency of the Simplex Method may indeed be attributed to the fact that simplex paths are short on the average.

These results are nice in themselves, but they have a strong statistical underpinning. In the sequel, we will describe how we have gone about to reduce the number of parameters to control on a statistically sound basis. We believe that our methodology might be useful in other cases, and that is why we present it in this newsletter.

2. The carrying through of the study

A special feature of our study is that we make several (independent) runs on one and the same problem. The iteration count for one run of the Stochastic Simplex Method on a given problem is a random variable. By making several runs on one problem we of course get a better estimate of the mean iteration count. But we also get estimates of its variance. This may be used advantageaously in the supsequent regression.

this prize, please contact John Tomlin at Ketron, Inc, Old Mill Office Center Suite 208A, 201 San Antonio Circle, Mountain View, California, 94040. He welcomes your assistance in this endeavor.

In January, current and past officers of COAL met with officers of The Special Interest Group on Mathematical Programming (SIGMAP) of ACM and officers of the Computer Science Technical Section of ORSA to investigate the possibility of merging some of the activities of these groups. It appears that each of these groups has a large membership interested in computational mathematical programming. At that meeting it was agreed that coordination and cooperation among these groups would benefit all three groups. Suggestions for joint activities include: 1) an annual special-interest meeting which includes a "SHARE"-type interaction between vendors of mathematical programming software and the software users, 2) cooperation between the newsletters of these groups, and 3) the promotion of technical publications in this research area.

Finally I wish to take this opportunity to compliment Jan Telgen and Robert Meyer on the excellent job they are doing. This newsletter gets better and better with each issue and I know it is largely due to their tireless efforts. Thank you so much Jan and Robert.

Karla Hoffman

CHAIRMAN'S COLUMN

The Council of the Mathematical Programming Society has approved the awarding of a society prize to be called "The Mathematical Programming Society Orchard-Hays Prize for Excellence in Computational Mathematical Programming". The first such prize will be awarded at the Twelfth Symposium of the Society held in Boston in 1985. It will be a cash award of $500.

"Computational Mathematical Programming" includes

1) experimental evaluation of mathematical programming algorithms,
2) development of high-quality mathematical programming software,
3) development of new computational methods together with experimental evidence of their effectiveness,
4) the development of new methods for the emperical testing and evaluation of mathematical programming software, and
5) methodolgy for enhancing mathematical programming modeling efforts.

I encourage each of you to reflect on who might be a worthy recipient of this prize and send nominations to Prof. George Dantzig, Chairman, Orchard-Hays Prize Committee, Stanford University, Stanford, California, 94305.

The prize is one which will be awarded triennually. A funding committee consisting of John Tomlin (chair), Harvey Greenberg, David Hirshfeld and A.C. Williams is actively seeking industrial sponsors for the prize. If you know of organizations which have benefited from mathematical programming software currently available and believe that those organizations would be willing to encourage further research in this area via sponsorship of

analysis. One may namely tell, to what extent the deviations from the regression "lines" depend on random fluctuations for the iteration count within the given problems and how much on variations between problems. Happily enough it turned out that a large (more than 50%) of the deviations could be ascribed to random fluctuation within the given problems.

Another feature of the study is that we tried and succeeded to demonstrate that the iteration count for a fixed problem is approximately normal. This permitted us to use different powerful statistical tests based on normality.

Even if Assignment and Transportation Problems have comparatively few parameters, there are still enough parameters to make a controlled experiment difficult. Hence there is a pressing need to identify the important parameters.

Let us first look at Assignment Problems. They are uniquely defined by their objective funtion coefficients. It is reasonable to believe that the distribution of the iteration count should not depend on the individual objective coefficients but rather on their distribution. This was verified in our tests. There was no significant difference between problems with objective coefficients drawn from the same distribution.
Another datareduction or standardization has to do with the dispersion of the objective coefficients. Most codes for Assignment or Network Flow Problems assume integer cost coefficients. So if these are drawn according to a continuous distribution, they have to be truncated to integers. But then the resulting problem will depend on the scaling of the costs. We indeed found the iteration count to be dependent of the scaling. With cost coefficients uniformly distributed in the intervall $[0,m_d]$ for a $d \times d$ Assignment Problem, the iteration count initially increased with m_d, but stabilized and remained constant when m_d reached about d^2. Hence, one may say that problems with $m_d < d^2$ are untypically simple. In our study we have thus investigated the case $m_d \geq d^2$ and put $m_d = 20\ d^2$ to be on the safe side.

Finally we have studied the dependence on the distribution for the cost coefficients. It turned out that with enough dispersion in the cost coefficients there is no significant difference in the iteration count for different cost coefficient distributions.

Thus we have reduced the number of parameters to control for a d × d Assignment Problem from d^2 to none. Hence the distribution of the iteration count depends only on the dimension, d.

Now let us take a quick look at Transportation Problems. For these we must in excess to cost coefficients specify right hand side, i.e. supplies and demands. Concerning the costs, their effect on the iteration count turned out to be similar to the Assignment case.

Concerning the right hand side, we have mainly generated problems with supplies and demands drawn from almost uniform distributions. However, using normal distribution did not give any statistically significant change in the iteration count.
Also, the outcome of the generation of the right hand side coefficients, did not effect the iteration count significantly.

Finally, the scaling of the supplies and demands before truncating to integers had a peculiar effect. It did not change the iteration count, but only the proportion of degenerate pivots, which went down to about zero, when the sum of supplies was about 1000 mn. (Here m is the number of supply points, n is the number of demand points.)

In order to get contrast to the Assignment case, where more than 90% of the iterations are degererate, we thus chose this sum large (5000 mn). This left us with only two parameters to control, i.e. m and n.

Finally some further comparisons showed that our standardized Transportation Problems give almost the same iteration counts as one real world problem and a problem with maximal number of extreme points, showing that our problems seem to be representative for "general" transportation problems. However, Zadeh's "pathological" Transportation Problems (Zadeh (1973)) give larger iteration counts than our "typical" problems.

Now let us turn to the financial situation of the COAL Newsletter. The Newsletter is produced on a relatively low budget (US $ 1500/year vs US $ 4000 for Optima) provided by MPS. Apart from 600 MPS members, approximately 90 non-MPS members (friends of COAL) have indicated that they want to receive the Newsletter too.

This implies that US $ 0.30 per MPS member per year (average costs, not marginal costs) is used for distributing the Newsletter to non-MPS members, spread all over the world. Admittedly, there is some injustice in this situation, but the alternative (US $ 2.00 contribution per year for non-members) is not without drawbacks either; for some people it is virtually impossible (e.g. no foreign currency export) to transfer this sum. We are currently discussing these matters with the MPS council.

In conclusion, The COAL Newsletter serves a purpose not currently served by Optima or Mathematical Programming. The editorial policy adopted (screening and direct communication with authors) reduces publication lags to a minimum and ensures rapid and timely dissemination of important material. The regular features (chairman's message, announcements) contribute to the performance of COAL as a subcommittee of MPS and constitute a valuable part of the Newsletter. We do not intend to alter the COAL Newsletter with regard to these points.
An exception might be the installment of a page limit both to emphasize the character of the Newsletter as a vehicle for short communications and to keep costs (production and mailing) down. Reduction of total costs and the finances of the COAL Newsletter are the subject of continued attention.

Robert R. Meyer
Jan Telgen

Apart from its function for COAL, the Newsletter conveys factual information on algorithms, software, tests, experiments and validations. Judging from the growing readership of the COAL Newsletter, this function is considered to be a valuable one too.

The relation of the COAL Newsletter to the other publications of the Mathematical Programming Society merits some attention. Whereas Mathematical Programming is a scientific journal of high quality and Optima is an MPS Newsletter, the COAL Newsletter provides a vehicle for the rapid publication of short computationally oriented papers as well as announcements. A formal indication of the fact that MPS does not consider Optima to be a duplication of the COAL Newsletter is provided by the inception of Optima years after the first COAL Newsletter was published. In order to avoid the possibility that the COAL Newsletter and Optima might eventually lead to some duplication, a working relation between the editors of the two has been created.

The only real duplication in the three publications is the inclusion of the MPS calendar. We do not regard this as an undesirable situation since this is basic information that requires only one page.

The COAL Newsletter contains papers that are closely related to COAL's objectives regarding testing, comparing and evaluating algorithms and software, Both other publications have a much broader scope and they do not try to match the COAL Newsletter in timeliness of the material published.

The quality of the papers published in the COAL Newsletter is not monitored through a formal refereeing process, but papers are thoroughly screened by the editors who may call on others to assist them in this task. To give an example, for the issues published in 1983, 13 papers were considered of which 6 were published, most of them after changes suggested by the editors were incorporated. We believe that for the time being this procedure is sufficient to maintain the quality of the material published in the Newsletter, and we intend to continue working along these lines.

References

P.O. Lindberg & S. Ólafsson (1980): "On the lengths of simplex paths: The assignment case", TRITA-MAT-1980-29, Department of Mathematics, Royal Institute of Technology, Stockholm, Sweden.

P.O. Lindberg & S. Ólafsson (1983): "On the lengths of simplex paths: The assignment case", TRITA-MAT-1983-3, ibid.; to appear in Mathematical Programming.

S. Ólafsson & P.O. Lindberg (1983): "On the lengths of simplex paths: The transportation case", TRITA-MAT-1983-4, ibid.

S. Ólafsson & P.O. Lindberg (1984): "On the lengths of simplex paths: The transportation case", TRITA-MAT-1983-6, ibid., in preparation.

N. Zadeh (1973): "A bad network problem for the simplex method and other minimum cost flow algorithms", Math. Progr. 5, 255-266.

MATHEMATICAL PROGRAMMING

a publication of the mathematical programming society

AIMS & SCOPE

MATHEMATICAL PROGRAMMING publishes original work dealing with every theoretical, computational, and applicational aspect of mathematical programming; that is, everything of direct or indirect use for questions surrounding the problem of finding the extreme values of functions of many variables.

Included, along with the conventional topics of linear, nonlinear, integer and stochastic programming, are computer experimentation, techniques for formulating and applying mathematical programming models, computer programming devices of special interest to the subject; unconstrained optimization, convexity, polytopes, and control and game theory done in the spirit of mathematical programming.

Expositions and surveys, original research, reports on computer routines and computer experimentation, and new or ingenious practical applications are published.

MATHEMATICAL PROGRAMMING contains a 'Short Communication' section consisting of brief and timely announcements of new results, applications, codes or experiments. These submissions benefit from accelerated refereeing. Papers omitting proofs and details are encouraged if accompanied by supporting material establishing the validity of the assertions made.

THE MATHEMATICAL PROGRAMMING SOCIETY

ENROLLMENT

I hereby enroll as a member of the Society for the calendar year 1984

PLEASE PRINT Name _____

Mailing address _____

_____

_____

My subscription to *Mathematical Programming* is for my personal use and not for the benefit of any library or other institution.

Signed _____

The dues for 1983 are:
32 Dollars (U.S.A.)
20 Pounds (U.K.)
68 Francs (Switzerland)
250 Francs (France)
81 Marks (Fed. Rep. Germany)
94 Guilders (Netherlands)

Please send this application with your dues to:
The Mathematical Programming Society
%The International Statistical Institute
428 Prinses Beatrixlaan
2270 AZ Voorburg, Netherlands

Student Applications: Dues are one-half the above rates. Have a faculty member verify your student status below and send application with dues to: Professor John Mulvey, School of Engineering and Applied Science, Princeton University, Princeton, N.J. 08544.

Faculty verifying status _____ Institution _____

THE COAL NEWSLETTER : PAST AND FUTURE

After 6 years of operation on the same basis it is time to sit back and reconsider the development of the COAL Newsletter in the past and its directions for the future.

The Committee on Algorithms (COAL) itself has been very active since it was established in 1974. For outsiders this is most apparent from the special COAL-sponsored sessions at numerous meetings of MPS, ORSA, TIMS, ACM and EURO. In addition, two separate COAL conferences have taken place: in 1977 the Advanced Study Institute on "Design and Implementation of Optimization Software" in Urbino (Italy) and in 1981 the conference on "Evaluating Mathematical Programming Techniques" in Boulder Co (U.S.A.). In 1984 an Advanced Study Institute on "Computational Mathematical Programming" will be organized in Bad Windsheim (F.R.G.). The very existence of these COAL meetings and the generous financial support provided by NATO, NSF, NBS and MPS are clear indications of the importance and appreciation for the work done by COAL. The quality of this work is reflected in publications, either officially sponsored by COAL (such as the volumes arising from conferences) or written by individual COAL members. Many more things have been accomplished by COAL (surveys, comparisons etc.) of which perhaps the most important is the development of quidelines for reporting compitational experiments. These quidelines have been adopted as standards by the leading scientific journals in the field.

The COAL Newsletter has been the primary contact between the formal memebers of COAL and the many friends of COAL (interested but not necessarily active in the field). Trough the years it has proved to be a very effective means of communication not only with the Mathematical Programming community but also within COAL. By keeping people informed about ongoing activities - both announcements of meetings and descriptions of experiments, tests and results - recent developments in computational mathematical programming are known to a large community of users. COAL's objectives of excellence are best achieved via the Newsletter. Thus, the Newsletter itself has been instrumental in keeping COAL alive.

COMMITTEE ON ALGORITHMS of the
MATHEMATICAL PROGRAMMING SOCIETY

COAL OBJECTIVES

The Committee on Algorithms is involved in computational developments in mathematical programming. There are three major goals: (1) ensuring a suitable basis for comparing algorithms, (2) acting as a focal point for computer programs that are available for general calculations and for test problems, and (3) encouraging those who distribute programs to meet certain standards of portability, testing, ease of use and documentation.

NEWSLETTER OBJECTIVES

The newsletter's primary objective is to serve as a forum for the Friends of COAL. Through an informal exchange of opinions, members have an opportunity to share their experiences. To date, our profession has not developed a clear understanding on the issues of how computational tests should be carried out, how the results of these tests should be presented in the literature, or how mathematical programming algorithms should be properly evaluated and compared. These issues will be addressed in the newsletter.

Dr. JAN TELGEN
RABOBANK NEDERLAND
LAAN VAN EIKENSTEIN 9
3705 AR ZEIST
THE NETHERLANDS

Mr. T. Steihaug,
STATOIL,
Forus, P. O. Box 300,
STAVANGER, 4001,
Norway.

Mathematical Programming Society

Committee on Algorithms Newsletter

No. 10
March 1984

JAN TELGEN
BOB MEYER
EDITORS

Contents: