TOMLAB - An Environment for Solving Optimization Problems in MATLAB

Kenneth Holmström [1]

Applied Optimization and Modeling Group (**TOM**)
Center of Mathematical Modeling
Department of Mathematics and Physics
Mälardalen University, P.O. Box 883, S-721 23 Västerås, Sweden

Abstract

TOMLAB is a general purpose, open and integrated **MATLAB** environment for solving optimization problems on **UNIX** and **PC** systems. **TOMLAB** has meny systems and driver routines for the most common optimization problems and more than 50 algorithms implemented in the toolbox **NLPLIB** and the toolbox **OPERA**.

NLPLIB TB 1.0 is a **MATLAB** toolbox for nonlinear programming and parameter estimation and **OPERA TB 1.0** is a **MATLAB** toolbox for operational research, with emphasis on linear and discrete optimization. Of special interest in **NLPLIB TB 1.0** are the algorithms for general and separable nonlinear least squares parameter estimation.

TOMLAB is using MEX-file interfaces to call solvers written in C/C++ and FORTRAN. Currently MEX-file interfaces have been developed for the commercial solvers *MINOS*, *NPSOL*, *NPOPT*, *NLSSOL*, *LPOPT*, *QPOPT* and *LSSOL*. From **TOMLAB** it is also possible to call routines in the MathWorks Optimization Toolbox.

Interfaces are available for the model language **AMPL** and the **CUTE** (Constrained and Unconstrained Testing Environment). Optimization problems may be either be defined in **MATLAB** code or in the **CUTE** SIF language or **AMPL** model language. Included in **TOMLAB** are many example and demonstration files.

The motivation for **TOMLAB** is to simplify the research and solution of practical optimization problems, giving easy access to all types of solvers; at the same time having full access to the power of **MATLAB**. Using **MATLAB** as an environment for solving optimization problems offers much more possibilities for analysis, then currently available in modeling languages like AIMMS, AMPL, ASCEND, GAMS and LINGO.

TOMLAB is free for academic purposes. More information about **TOMLAB**, **NLPLIB TB 1.0** and **OPERA TB 1.0** is found at **http://www.ima.mdh.se/tom**.

[1] Contact by E-mail: hkh@mdh.se

1 Introduction

This paper presents **TOMLAB**, an environment in MATLAB for the solution of optimization problems. **TOMLAB** features menu systems and driver routines for the most common optimization problems. **TOMLAB** has many internal algorithms implemented in the toolbox **NLPLIB** [20] and **OPERA** [21] and also calls routines from the MathWorks Optimization Toolbox [18]. It also provides interfaces to optimization software in FORTRAN and C/C++. This is possible using MEX-file interfaces.

To solve optimization problems, traditionally the user has been forced to write a FORTRAN code that calls some standard solver written as a FORTRAN subroutine. For nonlinear problems the user must also write subroutines which computes the function value and constraint values. The needed derivatives has either been explicitly coded, computed by using numerical differences or nowadays using automatic differentiation techniques.

In recent years several modeling languages has been developed, like AIMMS [6], AMPL [14], ASCEND [27], GAMS [7, 9] and LINGO [1]. The modeling system acts as a preprocessor. The user describes his problem in detail in a very verbal language, an opposite to a concise mathematical description of the problem. This problem description file was normally modified in a text editor, with help from example files solving the same type of problem. Now much effort is directed to the development of more user friendly interfaces. The model system processes the input description file and calls any of the predefined solvers which interfaces are built for.

The modeling language approach is suitable for many management and decision problems, but may not al-

ways be the best way for engineering problems, which often are nonlinear and have complicated problem descriptions. For people with some mathematical background, modeling languages often seems a very tedious way to define an optimization problem. Using MATLAB as an environment for solving optimization problems offers much more possibilities for analysis than just the pure solution of the problem.

The idea of this paper, and the concept of **TOMLAB**, is to try to integrate all different systems, getting access to the best of all worlds. **TOMLAB** should be seen as a complement to existing model languages, for the user needing more power and flexibility than given by a model system.

This paper is organized as follows. In Section 2, we describe the main features of **TOMLAB**. We then describe the two most important parts of **TOMLAB**, the toolbox **OPERA** in Section 3 and the toolbox **NLPLIB** in Section 4. Finally we end with some conclusions and further work in Section 5.

2 The features of TOMLAB

The main features of **TOMLAB** may be summarized as follows:

- MATLAB based environment
- Implements >50 optimization algorithms in toolbox **OPERA** and **NLPLIB**.
- Solves linear and discrete optimization problems:
  - linear programming
  - transportation programming problems
  - network programming problems
  - integer programming problems
  - dynamic programming problems

  using the toolbox **OPERA**
- Solves
  - unconstrained nonlinear optimization problems
  - quadratic programming problems
  - nonlinear constrained optimization problems
  - nonlinear least squares optimization problems
  - fitting of positive sums of exponential functions to data

  using the toolbox **NLPLIB**
- Portable, runs in MATLAB 5.1 and MATLAB 4.2c on
  - UNIX (SUN, HP) and
  - PC (NT4.0, Win95, Windows 3.11).
- Menu programs and driver routines makes **TOMLAB** very easy to use.
- MEX-file interfaces to standard optimization software. Works for both PC and UNIX. Currently MINOS, NPSOL, NPOPT, NLSSOL, QPOPT, LSSOL and LPOPT.
- **TOMLAB** interface to the **CUTE** SIF language and the standard set of test problems [8]. The CUTE distribution includes MATLAB interface routines.
- **TOMLAB** interface to the **AMPL** model language[14]. The MATLAB Interface to AMPL was built by Gay [15].
- Integrated possibility to call to almost all routines in the MathWorks Optimization Toolbox.
- You only need to define your problem once and use all available solvers!

TOMLAB is described in more detail in Holmström [22].

3 The OPERA Toolbox

The MATLAB toolbox **OPERA** is a collection of MATLAB m-files which solves many of the basic optimization problems in operations research and mathematical programming. Currently **OPERA** consists of:

- 10 500 lines of MATLAB code
- 53 files with algorithms and utilities
- 42 example files
- Menu program and driver routine for linear programming.
- Interactive routine for the definition and direct solution of linear programming problems

We now describe the algorithms implemented in **OPERA**. The actual MATLAB m-file names are given in parenthesis.

There are several algorithms implemented for **linear programming**. The standard revised simplex algorithm as formulated in Goldfarb and Todd [17, page 91] is used to solve the Phase II simplex problem (*lpsimp2*). A Phase I simplex strategy which formulates a LP problem with artificial variables is implemented (*lpsimp1*). This routine is using *lpsimp2* to solve the phase I problem. The dual simplex method [17, pages 105-106], usable when a dual basic feasible solution is available, is implemented in routine *lpdual*.

Two polynomial algorithms for linear programming are implemented. Karmakar's projective algorithm is implemented (*karmark*) using the description in Bazaraa et al. [5, page 386]. There is a choice of update, either according to Bazaraa or a rule by Goldfarb and Todd [17, chap. 9]. An affine scaling variant of Karmakar's method is implemented (*akarmark*) following the description in Bazaraa [17, pages 411-413]. As the purification algorithm a modification of the algorithm on page 385 in Bazaraa is used.

To solve **mixed linear inequality integer programs** two algorithms are implemented. The first implementation (*branch*) is a branch and bound algorithm from Nemhauser and Wolsey [26, chap. 8]. The second implementation (*cutplane*) is a cutting plane algorithm with Gomory cuts. Both routines are using other linear programming routines in the toolbox ((*lpsimp1*, *lpsimp2*, *lpdual*) to solve relaxed subproblems. Balas method for 0/1 integer programs restricted to integer coefficients is implemented in the routine *balas*.

Transportation problems are solved using an implementation of the transportation simplex method as described in Luenberger [25, chap 5.4]. Three algorithms to find a starting basic feasible solution for the transportation problem are included, the northwest corner method, the minimum cost method and Vogel's approximation method. The implementation of these algorithms follows the algorithm descriptions in Winston [32, chap. 7.2].

The implementation of the **Network Programming** algorithms are based on the forward and reverse star representation technique described for example in Ahuja et al. [3, pages 35-36]. The following algorithms are currently implemented:

- Search for all reachable nodes in a network using a stack approach (*gsearch*). The implementation is a variation of the Algorithm SEARCH in [2, pages 231-233].
- Search for all reachable nodes in a network using a queue approach (*gsearchq*). The implementation a variation of the Algorithm SEARCH in [2, pages 231-232].
- Find the minimal spanning tree of an undirected graph (*mintree*) with *Kruskal's algorithm* described in Ahuja et al. [3, page 520-521].
- Solve the shortest path problem using Dijkstras algorithm (*dijkstra*). A direct implementation of the Algorithm DIJKSTRA in [2, pages 250-251].
- Solve the shortest path problem using a label correcting method (*labelcor*). The implementation is based on Algorithm LABEL CORRECTING in [2, page 260].
- Solve the shortest path problem using a modified label correcting method (*modlabel*). The implementation is based on Algorithm MODIFIED LABEL CORRECTING in [2, page 262], with the addition of the heuristic rule discussed to improve running time in practice.
- Solve the maximum flow problem using the Ford-Fulkerson augmenting path method (*maxflow*). The implementation is based on the algorithm description in Luenberger [25, pages 144-145].
- Solve the minimum cost network flow problem (MCNFP) using a network simplex algorithm (*NWsimplx*). The implementation is based on Algorithm network simplex in Ahuja et al. [3, page 415].
- Solve the symmetric travelling salesman problem using Lagrangian relaxation and the sub gradient method with Polyak rule II (*salesman*), an algorithm by Held and Karp [19].

Two algorithmic examples of **dynamic programming** are included. Both algorithms are described in Winston [32, chap. 20]. Forward recursion is used to solve an inventory problem (*dpinvent*) and a knapsack problem (*dpknap*).

Lagrangian Relaxation is exemplified by a routine (*ksrelax*), which solves integer linear programming problems with linear inequality constraints and upper and lower bounds on the variables x. The problem is solved by relaxing all but one constraint and hence solving simple knapsack problems as subproblems in each iteration. The algorithm is based on the presentation in Fischer [11], using subgradient iterations and a simple line search rule.

4 The NLPLIB Toolbox

The current status of the **NLPLIB** toolbox is:

- 15 400 lines of MATLAB code.
- 101 files with algorithms, utilities and predefined problems
- Menu programs and driver routines for
  - unconstrained optimization,
  - quadratic optimization,
  - constrained optimization and
  - nonlinear parameter estimation - nonlinear least squares and fitting of sums of exponential functions to data

The main routine for unconstrained optimization, *ucsolve* and the main routine for nonlinear least squares, *gn*, are both written as a prototype algorithm which includes several of the popular methods for these problems. The prototype algorithm for nonlinear least squares is discussed in more detail in the next section 4.1.

The prototype routine for **unconstrained optimization** called *ucsolve* handles problems with bound constraints as described in Gill et al. [16]. This routine implements the Newtons method, the Quasi Newton BFGS method, the inverse BFGS, the Fletcher-Reeves conjugate gradient method and the Polak-Ribiere conjugate gradient method. Included is also a solver routine *strustr* that implements a structural trust region algorithm [10] combined with an initial trust region radius algorithm [29].

One of the menu options is to draw a contour plot of $f(x)$ together with the search steps. On each search step there are marks for each trial value the line search algorithm has computed a function value for. It is possible to follow the full iterative sequence on two-dimensional problems. In Figure 1 the result of optimizing the classical Rosenbrock banana function using *ucsolve* with Newtons method are displayed. There are a few steps where the line search has shortened the step. In contrast to this, see the behavior when running *ucsolve* with the Fletcher-Reeves conjugate gradient method in Figure 2. This method, when not using second derivative information, has a much more chaotic path to the solution.

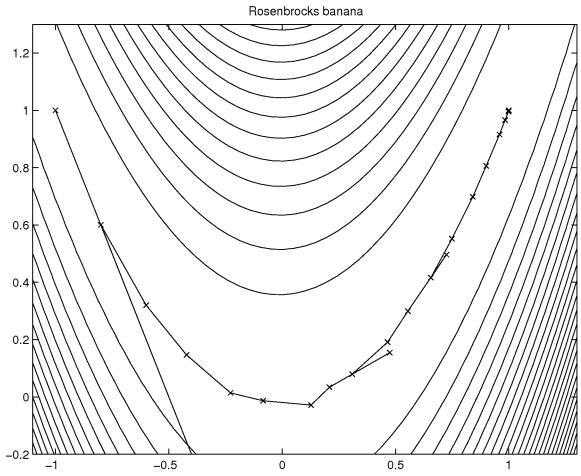

Figure 1: Rosenbrocks banana with search steps and line search for Newtons method

For **general nonlinear problems with nonlinear constraints** a sequential quadratic programming (SQP) method by Schittkowski [30] is implemented in routine *consolve*.

Quadratic programming (QP) problems are solved with a standard active set method [25], routine *qpi*. In **NLPLIB** there are also two routines for solving QP with equality constraints, either with a null space method (*qpe*) or with Lagranges method (*qplm*).

The line search algorithm *linesrch* used by the solvers in **NLPLIB** (*consolve*, *gn*, *ucsolve*) is a modified version the algorithm in Fletcher [12]. Quadratic (*intpol2*) or cubic interpolation (*intpol3*) is possible in the line search algorithm.

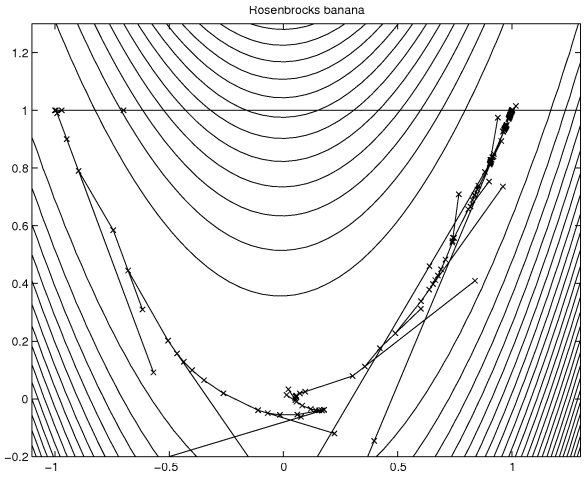

Figure 2: Search path of Fletcher-Reeves conjugate gradient method

4.1 Nonlinear Least Squares Algorithms

NLPLIB implements a prototype **nonlinear least squares** algorithm *gn* that also treat problems with bound constraints in a similar way as the routine *ucsolve* described in Section 4.

If rank problems occur the prototype algorithm is using subspace minimization, see Lindström [24]. The line search algorithm used is a modified version of the algorithm in Fletcher [12, chap. 2].

The prototype algorithms includes the following search step methods:

- Gauss-Newton
- Al-Baali-Fletcher hybrid method [4]
- Fletcher-Xu hybrid method [13].
- Huschens method [23].

As shown in Holmström [22], the *gn* routine performs very well on ill conditioned non linear least squares problems compared to other routines, like *leastsq* in the Optimization Toolbox.

In Figure 3 we see the result of running the prototype solver *gn* with the Al-Baali-Fletcher hybrid method on an approximation problem. The problem is to fit a circle to points in the plane. The data points are artificially generated, adding random noise. The dashed circle is the theoretical circle.

In **NLPLIB** a separable nonlinear least squares algorithm [28] is used to approximate positive sums of exponential functions to empirical data. To illustrate this facility we approximate two weighted exponential terms to the empirical data series by Steyn and Wyk [31]. Figure 4 shows the first part of the series, the approximating

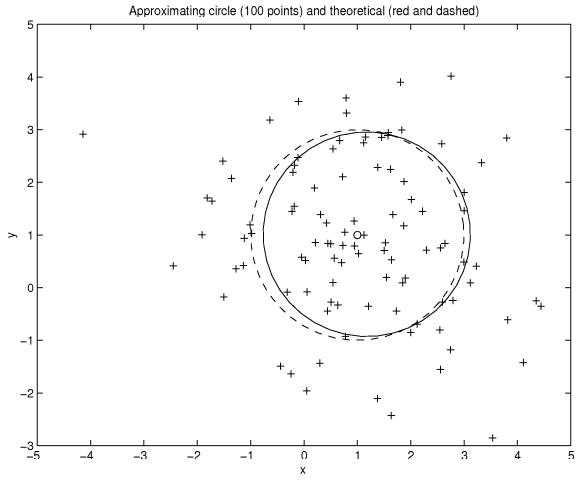

Figure 3: Using solver *gn* to find approximating circle to artifically generated and disturbed points in the plane

exponential model for the starting values used (dashdot line) and the optimized model (solid line). The same results are shown for the second part of the series in Figure 5.

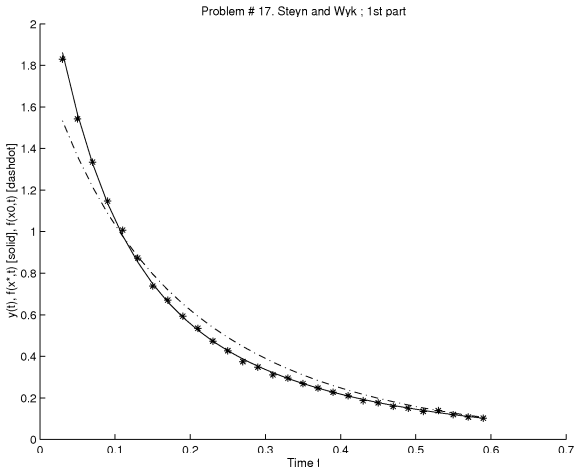

Figure 4: Using solver *gn* to find an exponential sum model that fits empirical data. Part 1.

5 Conclusions

TOMLAB together with the toolbox **NLPLIB** and the toolbox **OPERA** offers a powerful and unique environment for research, teaching and the practical solution of optimization problems. **TOMLAB** is a flexible tool, with both menu programs and driver routines.

TOMLAB is in continuous development. The number of external solvers possible to call with MEX-file interfaces will increase as well as the number of internal solvers. The graphics and menus should be improved and implemented using the latest facilities in MATLAB.

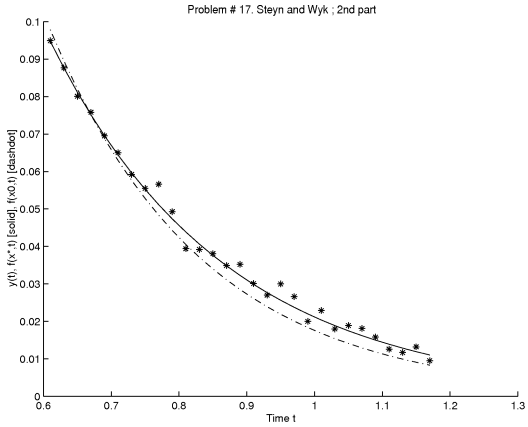

Figure 5: Using solver *gn* to find an exponential sum model that fits empirical data. Part 2.

References

[1] *LINGO - The Modeling Language and Optimizer.* LINDO Systems Inc., Chicago, IL, 1995.

[2] R. K. Ahuja, T.L. Magnanti, and J. B. Orlin. Network flows. In G. L. Nemhauser, A.H.G. Rinnooy Kan, and M.J.Todd, editors, *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*. Elsevier/North Holland, Amsterdam, The Netherlands, 1989.

[3] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice-Hall Inc., Kanpur and Cambridge, 1993.

[4] M. Al-Baali and R. Fletcher. Variational methods for non-linear least squares. *J. Oper. Res. Soc.*, 36:405–421, 1985.

[5] Mokhtar S. Bazaraa, John J. Jarvis, and Hanif D. Sherali. *Linear Programming and Network Flows.* John Wiley and Sons, New York, 2nd edition, 1990.

[6] J. Bisschop and R. Entriken. *AIMMS - The Modeling System.* Paragon Decision Technology, Haarlem, The Netherlands, 1993.

[7] J. Bisschop and A. Meeraus. On the development of a general algebraic modeling system in a strategic planning environment. *Mathematical Programming Study*, 20:1–29, 1982.

[8] I. Bongartz, A. R. Conn, N.I.M. Gould, and PH. L. Toint. CUTE: Constrained and Unconstrained Testing Environment. *ACM Transactions on Mathematical Software*, 21(1):123–160, 1995.

[9] A. Brooke, D. Kendrick, and A. Meeraus. *GAMS - A User's Guide.* The Scientific Press, Redwood City, CA, 1988.

[10] A. R. Conn, Nick Gould, A. Sartenaer, and Ph. L. Toint. Convergence properties of minimization algorithms for convex constraints using a structured

trust region. *SIAM Journal on Scientific and Statistical Computing*, 6(4):1059–1086, 1996.

[11] Marshall L. Fisher. An Application Oriented Guide to Lagrangian Relaxation. *Interfaces 15:2*, pages 10–21, March-April 1985.

[12] R. Fletcher. *Practical Methods of Optimization.* John Wiley and Sons, New York, 2nd edition, 1987.

[13] R. Fletcher and C. Xu. Hybrid methods for nonlinear least squares. *IMA Journal of Numerical Analysis*, 7:371–389, 1987.

[14] R. Fourer, D.M. Gay, and B.W.Kernighan. *AMPL - A Modeling Language for Mathematical Programming.* The Scientific Press, Redwood City, CA, 1993.

[15] David M. Gay. Hooking your solver to AMPL. Technical report, Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974, 1997.

[16] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press, London, 1982.

[17] D. Goldfarb and M.J. Todd. Linear programming. In G. L. Nemhauser, A.H.G. Rinnooy Kan, and M.J.Todd, editors, *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science.* Elsevier/North Holland, Amsterdam, The Netherlands, 1989.

[18] Andrew Grace. Optimization Toolbox User's Guide - For Use with MATLAB. Technical report, The Math Works, Inc., Natick, Mass. 01760, 1993.

[19] Michael Held and Richard M. Karp. The Traveling-Salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1:6–25, 1971.

[20] Kenneth Holmström. NLPLIB TB 1.0 - A Matlab Toolbox for Nonlinear Optimization and Parameter Estimation. Technical Report IMa-TOM-1997-2, Department of Mathematics and Physics, Mälardalen University, Sweden, 1997.

[21] Kenneth Holmström. OPERA TB 1.0 - A Matlab Toolbox for Optimization Algorithms in Operations Research. Technical Report IMa-TOM-1997-1, Department of Mathematics and Physics, Mälardalen University, Sweden, 1997.

[22] Kenneth Holmström. TOMLAB - A General Purpose, Open MATLAB Environment for Research and Teaching in Optimization. Technical Report IMa-TOM-1997-3, Department of Mathematics and Physics, Mälardalen University, Sweden, 1997. Presented at the 16th International Symposium on Mathematical Programming, Lausanne, Switzerland, August 24-29, 1997.

[23] J. Huschens. On the use of product structure in secant methods for nonlinear least squares problems. *SIAM Journal on Optimization*, 4(1):108–129, february 1994.

[24] P. Lindström. *Algorithms for Nonlinear Least Squares - Particularly Problems with Constraints.* PhD thesis, Inst. of Information Processing, University of Umeå, Sweden, 1983.

[25] David G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company, Reading, Massachusetts, 2nd edition, 1984.

[26] G. L. Nemhauser and L.A. Wolsey. Integer programming. In G. L. Nemhauser, A.H.G. Rinnooy Kan, and M.J.Todd, editors, *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science.* Elsevier/North Holland, Amsterdam, The Netherlands, 1989.

[27] P.C. Piela, T.G. Epperly, K.M. Westerberg, and A.W. Westerberg. ASCEND: An object-oriented computer environment for modeling and analysis: The modeling language. *Computers and Chemical Engineering*, 15:53–72, 1991.

[28] A. Ruhe and P-Å Wedin. Algorithms for Separable Nonlinear Least Squares Problems. *SIAM Review*, 22:318–337, 1980.

[29] A. Sartenaer. Automatic determination of an initial trust region in nonlinear programming. Technical Report 95/4, Department of Mathematics, Facultés Universitaires ND de la Paix, Bruxelles, Belgium, 1995.

[30] K. Schittkowski. On the convergence of a sequential quadratic programming method with an augmented lagrangian line search function. Technical report, Systems Optimization laboratory, Stanford University, Stanford, CA, 1982.

[31] H.S. Steyn and J.W.J. van Wyk. Some methods for fitting compartment models to data. Technical report, Wetenskaplike bydraes van die pu vir cho, Potchefstroomse Universiteit vir CHO, 1977.

[32] Wayne L. Winston. *Operations Research: Applications and Algorithms.* International Thomson Publishing, Duxbury Press, Belmont, California, 3rd edition, 1994.