AMO - Advanced Modeling and Optimization Volume 1, Number 2, 1999

The TOMLAB OPERA Toolbox for Linear and Discrete Optimization [1]

Kenneth Holmström[2], Mattias Björkman[3] and Erik Dotzauer[4]
Center for Mathematical Modeling
Department of Mathematics and Physics
Mälardalen University, P.O. Box 883, SE-721 23 Västerås, Sweden

Abstract

The Matlab toolbox OPERA TB is a set of Matlab m-files, which solves basic linear and discrete optimization problems in operations research and mathematical programming. Included are routines for linear programming (LP), network programming (NP), integer programming (IP) and dynamic programming (DP). OPERA TB, like the nonlinear programming toolbox NLPLIB TB, is a part of TOMLAB; an environment in Matlab for research and teaching in optimization. Linear programs are solved either by direct call to a solver routine or to a multi-solver driver routine, or interactively, using the Graphical User Interface (GUI) or a menu system. From OPERA TB it is possible to call solvers in the Math Works Optimization Toolbox and, using a MEX-file interface, general-purpose solvers implemented in Fortran or C. The focus is on dense problems, but sparse linear programs may be solved using the commercial solver MINOS. Presently, OPERA TB implements about thirty algorithms and includes a set of test examples and demonstration files. This paper gives an overview of OPERA TB and presents test results for medium size LP problems. The tests show that the OPERA TB solver converges as fast as commercial Fortran solvers and is at least five times faster than the simplex LP solver in the Optimization Toolbox 2.0 and twice as fast as the primal-dual interior-point LP solver in the same toolbox. Running the commercial Fortran solvers using MEX-file interfaces gives a speed-up factor of five to thirty-five.

Keywords: Operations Research, Linear Programming, Network Programming, Integer Programming, MATLAB, CUTE, AMPL, Software Engineering, Mathematical Software, Optimization, Algorithms.

AMS Subject Classification: 90B99, 90C05, 90C10, 90B10

1 Introduction

The Matlab toolbox OPERA TB is a set of Matlab m-files, which solves basic linear and discrete optimization problems in operations research and mathematical programming. Included are routines for linear programming (LP), network programming (NP), integer programming (IP) and dynamic programming (DP).

OPERA TB like the toolbox NLPLIB TB [14], is a part of TOMLAB [12, 13]; an environment in Matlab for research and teaching in optimization. TOMLAB is well documented in about 170 pages in the User's Guide [15], including instructions on how to define and solve different types of optimization problems.

[1] Financed by the Mälardalen University Research Board, project *Applied Optimization and Modeling (TOM).*
[2] E-mail: **hkh@mdh.se**; URL: **http://www.ima.mdh.se/tom**.
[3] E-mail: **mbk@mdh.se**.
[4] E-mail: **Erik.Dotzauer@mdh.se**.

From OPERA TB it is possible to call solvers in the Math Works Optimization Toolbox [4] and, using a MEX-file interface, general-purpose solvers implemented in Fortran or C. The TOMLAB MEX-file interfaces, as well as the interfaces to problems defined in CUTE [3] and AMPL [7], are further discussed in [13].

Presently, OPERA TB implements about 30 algorithms and it includes a large set of test examples and demonstration files. The focus is on dense problems, but sparse linear programs may be solved using the commercial solver MINOS 5.5 [17]. The toolbox is running in Matlab 5.x and works on both PC (NT4.0, Win95/Win98, Windows 3.11) and Unix (SUN, HP) systems.

An overview of OPERA TB is given in Section 2, and in Section 3 the implemented algorithms are briefly described. For more detailed descriptions, see [15] and [11]. Section 4 gives a short discussion about the example and demonstration files, and in Section 5 some computational results are presented. Finally, conclusions and a discussion of possible extensions are given in Section 6.

2 Overview of the Toolbox OPERA TB

Currently OPERA TB consists of about 15 000 lines of Matlab m-file code partitioned into 57 files with algorithms and utilities, and 50 files with test problems.

The menu program *lpOpt* for linear programs is similar to the menu programs in NLPLIB TB [14]. It calls the multi-solver driver routine *lpRun*, which may call any of the predefined solvers written in Matlab, C or Fortran code. Also solvers in the Optimization Toolbox are callable. The user may either run *lpOpt*, or call the multi-solver driver routine *lpRun* or the OPERA TB solvers directly. Linear programming problems may also be solved from the TOMLAB Graphical User Interface [6].

Presently, MEX-file interfaces have been developed for six general-purpose solvers available from the Systems Optimization Laboratory, Department of Operations Research, Stanford University, California; NPSOL 5.02 [8], NPOPT 1.0-10 (updated version of NPSOL), NLSSOL 5.0-2, QPOPT 1.0-10, LSSOL 1.05 and LPOPT 1.0-10. Furthermore, an interface to MINOS 5.5 [17] has been developed. MEX-file interfaces are available for both Unix and PC systems. MINOS 5.5 and LPOPT 1.0-10 are intended for LP problems, but it is also possible to use the other solvers to solve LP problems.

There is a routine *simplex* to interactively define and solve LP problems in canonical standard form. When the problem is defined, *simplex* calls the OPERA TB LP solvers *lpsimp1* and *lpsimp2*.

OPERA TB includes algorithms for network programming, integer programming and dynamic programming. The algorithms in OPERA TB are further described in Section 3.

3 Algorithms in OPERA TB

In this section brief descriptions of the algorithms in OPERA TB are given.

There are several algorithms implemented for **linear programming**, which are divided into two groups; numerically stable solvers and class-room solvers for education in optimization. A general **linear program** (**LP**) is defined as

$$\begin{aligned} \min_x \quad & f(x) = c^T x \\ s.t. \quad & \begin{array}{ccccc} x_L & \leq & x & \leq & x_U \\ b_L & \leq & Ax & \leq & b_U, \end{array} \end{aligned} \tag{1}$$

where $c, x, x_L, x_U \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b_L, b_U \in \mathbb{R}^m$. This problem is solved by *lpSolve*, which is using the routines *Phase1Simplex* and *Phase2Simplex* in the solution process, see Table 1.

The routines *Phase1Simplex*, *Phase2Simplex* and *DualSolve* are generalized and improved versions of *lpsimp1*, *lpsimp2* and *lpdual*, respectively. These three class-room solvers, together with some basic interior point solvers for linear programming, are listed in Table 2.

Table 1: Numerically stable solvers for linear programming.

Function	**Description**
lpSolve	General solver for linear programming problems. Calls *Phase1Simplex* and *Phase2Simplex*.
Phase1Simplex	The Phase I simplex algorithm. Finds a basic feasible solution (bfs) using artificial variables. Calls *Phase2Simplex*.
Phase2Simplex	The Phase II revised simplex algorithm with three selection rules.
DualSolve	The dual simplex algorithm.

Table 2: Class-room solvers for linear programming.

Function	**Description**
lpsimp1	The Phase I simplex algorithm. Finds a basic feasible solution (bfs) using artificial variables. Calls *lpsimp2*.
lpsimp2	The Phase II revised simplex algorithm with three selection rules.
karmark	Karmakar's algorithm. Kanonical form.
lpkarma	Solves LP on equality form, by converting and calling *karmark*.
lpdual	The dual simplex algorithm.
akarmark	Affine scaling variant of Karmarkar's algorithm.

The implementation of *lpsimp2* is based on the standard revised simplex algorithm by Goldfarb and Todd [9, page 91], used for the solution of Phase II LP problems. *lpsimp1* implements a Phase I simplex strategy by formulating a LP problem with artificial variables. This routine is using *lpsimp2* to solve the Phase I problem. The dual simplex method, *lpdual*, usable when a dual feasible solution is available instead of a primal feasible, implements the algorithm presented in [9, pages 105-106].

Two polynomial algorithms for linear programming are implemented. Karmakar's projective algorithm (*karmark*) is based on the algorithm in Bazaraa et al. [2, page 386]. There is a choice of update rule, either according to Bazaraa or to Goldfarb and Todd [9, chap. 9]. The affine scaling variant of Karmakar's method (*akarmark*) is an implementation of the algorithm from Bazaraa [9, pages 411-413]. As purification algorithm a modified version of the algorithm on page 385 in Bazaraa is used.

The OPERA TB internal linear programming solvers *lpsimp2* and *lpdual* both have three rules for variable selection implemented. Bland's cycling prevention rule is the choice if fear of cycling exists. There are two variants of minimum reduced cost variable selection, the original Dantzig's rule and one which sorts the variables in increasing order in each iteration. The same selection rules are used in *Phase2Simplex* and *DualSolve*.

Transportation problems are solved using an implementation (*TPsimplx*) of the transportation simplex method described in Luenberger [16, chap 5.4]. Three simple algorithms to find a starting basic feasible solution for the transportation problem are included; the northwest corner method (*TPnw*), the minimum cost method (*TPmc*) and Vogel's approximation method (*TPvogel*). The implementation of these algorithms follows the algorithm descriptions in Winston [19, chap. 7.2].

The implementation of the **network programming** algorithms are based on the forward and reverse star representation technique described in Ahuja et al. [1, pages 35-36]. The network programming routines are listed in Table 3.

To solve mixed linear inequality constrained **integer programs** two algorithms are implemented. The first implementation (*mipSolve*) is a branch-and-bound algorithm from Nemhauser and Wolsey [18, chap. 8]. The second implementation (*cutplane*) is a cutting plane algorithm using Gomory cuts. Both routines calls linear programming routines in OPERA TB to solve relaxed subproblems.

Table 3: Routines for network programming.

Function	**Description**
gsearch	Searching all reachable nodes in a network. Stack approach.
gsearchq	Searching all reachable nodes in a network. Queue approach.
mintree	Finds the minimum spanning tree of an undirected graph.
dijkstra	Shortest path using Dijkstra's algorithm.
labelcor	Shortest path using a label correcting algorithm.
modlabel	Shortest path using a modified label correcting algorithm.
maxflow	Solving maximum flow problems using the Ford-Fulkerson augmenting path method.
salesman	Symmetric traveling salesman problem (TSP) solver using Lagrangian relaxation and the subgradient method with the Polyak rule II.
travelng	Solve TSP problems with branch-and-bound. Calls *salesman*.
NWsimplx	A network simplex algorithm for solution of minimum cost network flow problems (MCNFP).

Balas method for binary integer programs restricted to integer coefficients is implemented in the routine *balas* [10].

To illustrate the use of dynamic programming, the routine *dbknap* solving knapsack problems and the routine *dbinvent* solving deterministic inventory problems are implemented. Included is also an algorithm (*ksrelax*) adopting Lagrangian relaxation for the solution of knapsack problems.

4 Example and Demonstration Files

There are example files available for all implemented algorithms. Most are simple test problems implemented with the purpose to show how to call a solver algorithm. For a detailed description of the example and demonstration files, see [11] and [15].

For linear programs, routines that illustrate cycling, redundancy in constraints and perturbation theory are included. The Klee-Minty test example (*exKleeM*), see Bazaraa et al. [2, page 376], illustrates the exponential convergence of the simplex method for different number of variables. If running the simplex method using the original Dantzig selection rule, it visits all other vertices before it convergences to the optimal one.

5 Computational Results

We have made tests to compare the efficiency of different solvers on medium size LP problems. The OPERA TB solver *lpSolve*, two algorithms implemented in the solver *linprog* from Optimization Toolbox 2.0 [5] and the commercial Fortran solvers MINOS and QPOPT are compared. In all test cases the solvers converge to the same solution. The results are presented in Table 4, Table 5 and Table 6. The problem dimensions and all elements in (1) are chosen randomly. Since the simplex algorithm in *linprog* does not return the number of iterations as output, these figures could not be presented. *lpSolve* has been run with two selection rules; Bland's cycling prevention rule and the minimum cost rule.

The results in Table 4 show that problems with about 200 variables and 150 inequality constraints are solved by *lpSolve* fast and efficient. When comparing elapsed computational time for 20 problems, it is clear that *lpSolve* is much faster then the corresponding simplex algorithm implemented in the *linprog* solver. In fact *lpSolve*, with the minimum cost selection rule, is more than five times faster, a remarkable difference. *lpSolve* is also more than twice as fast as the other algorithm imple-

Table 4: Computational results on randomly generated medium size (order 200) LP problems for four different routines. *Iter* is the number of iterations and *Time* is the elapsed time in seconds on a Dell Latitude CPi 266XT running Matlab 5.3. The *lpS* solver is the TOMLAB *lpSolve*, and it is run with both Bland's selection rule (iterations It_b, time T_b) and with the minimum cost selection rule (iterations It_m, time T_m). The *linprog* solver in the Optimization Toolbox 2.0 implements two different algorithms, a medium-scale simplex algorithm (time T_m) and a large-scale primal-dual interior-point method (iterations It_l, time T_l). The number of variables, n, the number of inequality constraints, m, the objective function coefficients, the linear matrix and the right hand side are chosen randomly. The last row shows the mean value of each column.

n	m	*lpS*	*lpS*	*minos*	*qpopt*	*linprog*	*lpS*	*lpS*	*minos*	*qpopt*	*linprog*	*linprog*
		It_b	It_m	*Iter*	*Iter*	It_l	T_b	T_m	*Time*	*Time*	T_m	T_l
228	132	32	10	17	12	23	13.48	4.10	0.28	0.18	40.13	11.97
191	164	20	9	9	10	19	13.31	5.81	0.28	0.14	23.59	13.22
212	155	63	16	30	16	19	37.47	9.28	0.33	0.19	33.48	12.72
185	158	53	25	16	16	19	33.08	15.47	0.27	0.17	23.83	11.91
222	168	35	12	22	12	21	25.21	8.43	0.33	0.19	39.62	17.30
207	162	10	8	6	7	22	6.41	5.06	0.29	0.14	31.30	15.97
229	130	42	12	21	19	21	17.20	4.81	0.30	0.22	42.20	10.89
213	136	56	6	21	6	20	25.31	2.59	0.29	0.16	34.82	10.88
227	146	95	19	33	20	23	50.40	9.92	0.31	0.22	43.94	15.32
192	150	25	6	13	5	17	13.89	3.17	0.29	0.14	26.92	10.42
195	155	12	8	9	7	22	7.01	4.60	0.27	0.16	26.80	14.45
221	160	30	12	10	11	22	19.54	7.66	0.31	0.19	40.30	17.82
183	144	61	9	9	10	21	31.37	4.48	0.25	0.17	23.04	11.25
200	165	19	10	15	14	19	12.98	6.68	0.31	0.21	28.52	15.17
199	137	16	6	7	5	20	7.20	2.62	0.26	0.15	30.91	10.74
188	154	18	8	9	7	17	10.51	4.56	0.31	0.15	26.16	11.35
202	159	25	13	18	11	17	15.79	8.12	0.29	0.18	31.85	13.15
223	155	103	16	20	17	24	62.72	9.54	0.33	0.22	40.19	20.04
196	121	17	7	16	6	19	5.89	2.40	0.24	0.16	26.63	8.55
202	133	47	10	12	12	21	20.17	4.22	0.25	0.18	29.97	11.46
206	149	39	11	16	11	20	21.45	6.18	0.29	0.18	32.21	13.23

mented in *linprog*, a primal-dual interior-point method aimed for large-scale problems [5]. There is a penalty about a factor of three to choose Bland's rule to prevent cycling in *lpSolve*. The solvers written in Fortran, MINOS and QPOPT, of course run much faster, but the iteration count show that *lpSolve* converges as fast as QPOPT and slightly better than MINOS. The speed-up is a factor of 35 when running QPOPT using the MEX-file interface.

In Table 5 a similar test is shown, running 20 problems with about 100 variables and 50 inequality constraints. The picture is the same, but the time difference, a factor of five, between *lpSolve* and the Fortran solvers are not so striking for these lower dimensional problems. *lpSolve* is now more than nine times faster than the simplex algorithm in *linprog* and twice as fast as the primal-dual interior-point method in *linprog*.

A similar test on larger dense problems, running 20 problems with about 500 variables and 240 inequality constraints, shows no benefit in using the primal-dual interior-point method in *linprog*, see Table 6. In that test *lpSolve* is more than five times faster, and 15 times faster than the simplex algorithm in *linprog*. Still it is about 35 times faster to use the MEX-file interfaces. In conclusion, for dense problems the commercial LP solvers in Optimization Toolbox offers no advantage compared to the TOMLAB solvers. It is clear that if speed is critical, the availability of Fortran solvers callable from Matlab using the MEX-file interfaces in TOMLAB is very important.

Table 5: Computational results on randomly generated medium size (order 100) LP problems for four different routines. See the comments in Table 4.

n	m	*lpS*	*lpS*	*minos*	*qpopt*	*linprog*	*lpS*	*lpS*	*minos*	*qpopt*	*linprog*	*linprog*
		It_b	It_m	$Iter$	$Iter$	It_l	T_b	T_m	$Time$	$Time$	T_m	T_l
128	32	37	12	10	11	16	1.83	0.49	0.30	0.20	8.67	1.40
129	60	8	10	10	9	18	0.79	0.98	0.13	0.10	8.61	2.02
125	45	8	9	16	7	14	0.56	0.60	0.14	0.12	7.53	1.21
81	65	27	5	7	4	13	2.79	0.58	0.11	0.09	3.09	1.33
102	40	25	9	12	8	13	1.27	0.53	0.13	0.11	4.76	0.90
96	33	13	7	6	8	12	0.56	0.37	0.12	0.11	4.24	0.75
110	61	29	10	9	9	16	2.70	1.00	0.13	0.11	5.85	1.67
113	27	25	8	9	8	11	0.82	0.36	0.13	0.11	6.00	0.71
127	58	16	9	13	8	15	1.44	0.85	0.14	0.12	7.95	1.78
85	58	10	7	7	7	14	0.93	0.65	0.14	0.09	3.23	1.30
103	31	15	7	9	6	12	0.62	0.36	0.13	0.11	4.90	0.73
101	41	22	9	11	9	12	1.19	0.55	0.14	0.10	4.73	0.89
83	41	9	6	7	7	13	0.54	0.41	0.13	0.11	3.14	0.90
118	39	28	9	8	8	13	1.37	0.53	0.14	0.12	6.63	1.00
92	33	13	8	8	7	12	0.59	0.40	0.11	0.11	3.90	0.74
110	46	21	7	15	6	14	1.29	0.52	0.13	0.12	5.75	1.20
82	65	25	6	6	5	15	2.60	0.69	0.14	0.10	3.03	1.51
104	29	6	6	10	4	11	0.33	0.33	0.12	0.11	5.02	0.66
83	48	28	8	10	10	13	1.77	0.60	0.12	0.11	3.12	0.98
90	50	8	4	4	3	12	0.63	0.35	0.12	0.10	3.68	1.07
103	45	19	8	9	7	13	1.23	0.56	0.14	0.11	5.19	1.14

Table 6: Computational results on randomly generated medium size (order 500) LP problems for four different routines. See the comments in Table 4.

n	m	*lpS*	*lpS*	*minos*	*qpopt*	*linprog*	*lpS*	*lpS*	*minos*	*qpopt*	*linprog*	*linprog*
		It_b	It_m	$Iter$	$Iter$	It_l	T_b	T_m	$Time$	$Time$	T_m	T_l
528	232	35	7	7	6	29	55.96	10.56	0.85	0.44	432.88	126.80
482	252	33	9	7	8	25	64.43	16.72	0.85	0.45	338.89	118.86
503	251	72	15	38	17	36	140.04	28.44	1.01	0.76	373.71	173.26
507	259	142	18	46	27	29	301.34	37.12	1.19	1.06	386.26	150.92
487	240	48	17	33	19	27	83.67	28.96	1.00	0.76	341.00	117.99
506	251	46	8	11	8	24	89.16	14.63	0.88	0.49	384.73	118.40
504	256	35	9	16	8	37	72.69	17.91	0.97	0.54	383.51	185.99
489	255	36	28	27	28	27	72.48	55.97	1.05	1.05	347.75	132.25
514	228	9	4	4	3	33	13.10	5.45	0.80	0.33	393.39	135.56
524	245	64	11	27	14	29	116.62	19.27	1.00	0.71	424.76	139.50
506	255	112	22	28	23	23	226.55	43.64	0.99	0.94	382.81	116.94
497	224	50	11	14	12	31	74.42	15.89	0.85	0.59	357.15	124.36
482	249	27	16	17	20	31	50.37	29.37	0.88	0.76	328.22	142.15
485	249	18	6	21	5	20	33.23	10.34	1.07	0.37	335.43	94.40
509	223	84	22	35	17	36	121.11	31.23	0.92	0.76	385.42	143.92
506	224	38	12	11	14	34	55.76	17.13	0.84	0.67	378.93	136.69
511	241	115	10	36	9	27	202.22	16.74	1.03	0.50	382.81	125.95
497	230	78	23	43	12	27	121.27	35.28	1.00	0.58	356.62	110.28
514	226	84	21	42	26	32	125.63	30.94	1.03	1.06	398.97	129.22
511	268	59	10	30	9	29	139.28	22.61	1.06	0.54	379.70	165.16
503	243	59	14	25	14	29	107.97	24.41	0.96	0.67	374.65	134.43

6 Conclusions and Further Work

The toolbox OPERA TB is a powerful tool for computer based learning of operations research and mathematical programming. OPERA TB includes a large set of standard algorithms and example files for linear and discrete optimization. For small and medium size optimization research problems, OPERA TB is a valuable research tool. Use of sparse matrices is possible when running MINOS using the MEX-file interface.

OPERA TB is to be extended in several directions. More routines for network programming and integer programming is to be included. Interfaces to AMPL and CUTE should be developed for network and integer programs. The sparse matrix handling in Matlab will be exploited to increase the efficiency of the implemented solvers and to extend the possible use of commercial solvers from Matlab. An interface to XPRESS/MP will be implemented in the near future. Mixed integer nonlinear optimization (MINLP) solvers are also to be included.

Acknowledgements

We would like to thank Prof. Philip E. Gill for giving us access to the commercial codes NPSOL, NPOPT, NLSSOL, LPOPT, QPOPT and LSSOL, and Prof. Michael Saunders for giving us access to MINOS, and making important comments on the paper. We also like to thank Samuel Huang, who has voluntary provided code that extends some of the algorithms as well as corresponding example files.

References

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

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

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

[4] Mary Ann Branch and Andy Grace. *Optimization Toolbox User's Guide*. 24 Prime Park Way, Natick, MA 01760-1500, 1996.

[5] Thomas Coleman, Mary Ann Branch, and Andy Grace. *Optimization Toolbox User's Guide*. 24 Prime Park Way, Natick, MA 01760-1500, 1999. Third Printing Revised for Version 2 (Release 11).

[6] Erik Dotzauer and Kenneth Holmström. The TOMLAB Graphical User Interface for Nonlinear Programming. *Advanced Modeling and Optimization*, 1(2):9–16, 1999.

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

[8] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. *User's Guide for NPSOL (Version 4.0): A Fortran package for nonlinear programming*. Department of Operations Research, Stanford University, Stanford, CA, 1986. SOL 86-2.

[9] 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.

[10] Kaj Holmberg. Heltalsprogrammering och dynamisk programmering och flöden i nätverk och kombinatorisk optimering. Technical report, Division of Optimization Theory, Linköping University, Linköping, Sweden, 1988-1993.

[11] Kenneth Holmström. OPERA TB 1.0 - A Matlab Toolbox for Optimization Algorithms in Operations Research. Technical Report IMa-TOM-1997-01, Department of Mathematics and Physics, Mälardalen University, Sweden, 1997.

[12] Kenneth Holmström. TOMLAB - An Environment for Solving Optimization Problems in Matlab. In M. Olsson, editor, *Proceedings for the Nordic Matlab Conference '97, October 27-28*, Stockholm, Sweden, 1997. Computer Solutions Europe AB.

[13] Kenneth Holmström. The TOMLAB Optimization Environment in Matlab. *Advanced Modeling and Optimization*, 1(1):47–69, 1999.

[14] Kenneth Holmström and Mattias Björkman. The TOMLAB NLPLIB Toolbox for Nonlinear Programming. *Advanced Modeling and Optimization*, 1(1):70–86, 1999.

[15] Kenneth Holmström, Mattias Björkman, and Erik Dotzauer. TOMLAB v1.0 User's Guide. Technical Report IMa-TOM-1999-01, Department of Mathematics and Physics, Mälardalen University, Sweden, 1999.

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

[17] Bruce A. Murtagh and Michael A. Saunders. MINOS 5.4 USER'S GUIDE. Technical Report SOL 83-20R, Revised Feb. 1995, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California 94305-4022, 1995.

[18] 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.

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