An Updated Mixed Integer Programming Library: MIPLIB 3.

Robert E. Bixby
Sebastian Ceria
Cassandra M. McZeal
Martin W.P. Savelsbergh

February 1998

TR98-03

An Updated Mixed Integer Programming Library: MIPLIB 3.0

Robert E. Bixby

Department of Computational and Applied Mathematics
Rice University
Houston, TX 77251-1892
CPLEX Optimization, Inc.
bixby@caam.rice.edu

Sebastián Ceria

School of Business
Columbia University
New York, NY 10027-6902
sebas@cumparsita.gsb.columbia.edu

Cassandra M. McZeal

Department of Computational and Applied Mathematics
Rice University
Houston, TX 77251-1892
cmoore@caam.rice.edu

Martin W.P. Savelsbergh

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205.
Martin.Savelsbergh@isye.gatech.edu

1 Introduction

In response to the needs of researchers for access to challenging mixed integer programs, Bixby et al. [1] created MIPLIB, an electronically available library of both pure and mixed integer programs, most of which arise from real-world applications.

Since its introduction, MIPLIB has become a standard test set for comparing the performance of mixed integer optimization codes. Its availability has provided an important stimulus for researchers in this very active area. As technology has progressed, however, there have been significant improvements in state-of-the-art optimizers and computing machinery. Consequently, several instances have become too easy, and a need has emerged for more difficult instances. Also, it has been observed that certain types of problems are overrepresented in MIPLIB and others underrepresented. These considerations have prompted the present update.

Since mixed integer programming is such an active research area, and the performance of optimizers keeps improving, we anticipate that this update will not be the last. Subsequent updates are planned on a yearly basis. We encourage both researchers and practitioners in integer programming to submit real-world instances for consideration and possible inclusion in MIPLIB.

This note describes the MIPLIB update. We have added several new problems and deleted some existing ones. In addition, we have included, for each problem, certain auxiliary information describing the structure of the constraint matrix. The purpose of this information is to identify constraint classes that may be useful in the various phases of problem solving, such as preprocessing, constraint generation, and branching.

2 Problems

2.1 Problem Additions

Twenty-nine new problems have been added to MIPLIB: *10teams, arki001, blend2, dano3mip, danoint, fast0507, fiber, gesa2, gesa2_o, gesa3, gesa3_o, gt2, harp2, markshare1, markshare2, mas74, mas76, mitre, mkc, nw04, pp08a, pp08aCUTS, pk1, qnet1, qnet1_o, rout, seymour, swath, and vpm2*. Most of the new problems arose from real-world applications.

The *10teams* model is a sports scheduling model, describing the allocation of teams to time slots in the English football league. The primary objective is feasibility. The problem *arki001* describes raw-material extraction in batches to fit production processes. The problems *danoint* and *dano3mip* resulted from telecommunications applications. Problem *dano3mip* deals with ATM network layout (design). The application is described in more detail in "Computational experience with a difficult mixed-integer multi-commodity flow problem," by O. Günlük and D. Bienstock, which appeared in Mathematical Programming, **68** (1995), pp. 213-238.

The *fast0507* model is a set covering problem arising from a crew-scheduling application for a railway company. The model *fiber* is a fiber routing problem. The models *gesa2_o* and *gesa3_o* are general integer programs arising from the optimization of electricity generation in the Balearic Islands in Spain. The problems *gesa2* and *gesa3* are the same as for the *"_o"* versions, except that surrogate knapsacks, linking resources at the different islands, have been added. The LP and IP values are the same as the *"_o"* problems, but as integer problems they appear to be easier to solve.

The model *gt2* is a general integer program arising from a routing application, where the objective is to determine the number of trucks of a certain type that will be needed to satisfy "client demands." The problem *nw04* arises from a airline set partitioning application. The models *markshare1* and *markshare2* are market-sharing problems. Market-sharing problems are examples of goal programming problems. The models are described in "A class of hard small 0-1 programs." IPCO Conference Proceedings, 1998. The *mkc* model is a multiple knapsack problem with color constraints. Further details can be found in "The Multiple Knapsack Problem with Color Constraints", IBM Research Report 21138, T. J. Watson Research Center, Yorktown Heights, NY 10598.

The model *pp08aCUTS* is the same as *pp08a* except that flow cover inequalities have been added to strengthen the formulation. The problems *qnet* and *qnet_o* are line-leasing models. The problem *seymour* is a set-covering problem that arose from work related to the proof of the well-known 4-color Theorem from graph theory. It tries to find a "best" unavoidable set of reducible configurations. To our knowledge, this model has not been solved to provable optimality. The model *swath* arises from the defense industry and involves planning missions for radar surveillance. The model *vpm2* is a variation of the *vpm1* model already in MIPLIB.

2.2 Problem Statistics

The following tables give various statistics for the complete, revised set of MIPLIB problems.

In Table 1, the first column, **NAME**, contains the name of the model. The next two columns, **ROWS** and **COLS**, contain the number of rows (constraints), not including free rows, and the number of columns (variables) in the problem, respectively. The column **INT** specifies the number of variables that are restricted to integer values, and the **0/1** column specifies how many of these integer variables are binary. The column **CONT** specifies the number of variables that are continuous. The next two columns report the best-known integral solution and the optimal value with the integrality restrictions relaxed, respectively. Three entries in the **INT SOLN** column includes the qualifier **(not opt)** which indicates that the reported solution has not been proved to be optimal. The last column **LP SOLN** contains the solution to the linear programming relaxation for each problem.

Table 2 contains information regarding the origins of each problem. The first column is, again, **NAME**. The second column, **ORIGINATOR**, gives the name of the person (institution) from whom (which) the problem originated. The next column, **FORMULATOR**, gives the name of the person or organization responsible for formulating the model, and the last column, **DONATOR**, gives the name of the person or institution who contributed the problem.

2.3 Problem Deletions

As a part of the modifications to MIPLIB, twenty-five problems were deleted from the test set. One of the factors used to determine whether a problem was to be deleted was the ease with which the problem solved. The remainder of the deleted problems were chosen to reduce duplicity. These problems were each part of a set of problems exhibiting either similar form or behavior.

In order to determine which problems were "easy," all of the current MIPLIB models were run on a SPARC 10/41 using the CPLEX 3.0 mixed-integer optimizer[1]. The settings given below were used. They were meant to correspond to what one might consider the most straightforward implementation of a reasonable, LP-based branch-and-bound solver:

- No preprocessing.
- Branching variable selection (*largest infeasibility*): Take the variable with the largest integer infeasibility.
- Node selection (*best bound*): In a minimization problem, take as the next node to process the one with the smallest value for the solution of its LP relaxation.
- No cutting-plane generation.
- No heuristic generation of integral solutions.

Using these settings, the problems *air01, air02, bm23, cracpb1, diamond, lp4l, misc01, misc02, misc04, mod013, p0040, pipex, sample2, sentoy, stein9,* and *stein15* all solved in 11 seconds or less and were selected for deletion. The solution of every other problem took at least 25 seconds. The model *air06* was also removed since it solves as a linear program, requiring no branching.

[1]CPLEX is a trademark of CPLEX Optimization, Inc.

Various models were deleted to eliminate instances with very similar structures and similar computational properties:

bell3b, bell4 : The problems *bell3a, bell3b, bell4,* and *bell5* are all based upon the same underlying model. Both *bell3a* and *bell3b* are relatively easy; *bell4* and *bell5* seem about equally difficult, with *bell5* the more difficult of the two.

misc05 : The problems *misc05* and *misc06* have very similar solution characteristics, and the *"misc"* set seemed overrepresented.

p0291 : The *"p"* set also seemed overrepresented.

fixnet3, fixnet4 : The problems *fixnet3, fixnet4,* and *fixnet6* were all randomly generated and have very similar solution characteristics. From the three, we selected one to keep, more-or-less at random.

set1al, set1cl : Again, *set1al, set1cl,* and *set1ch* were all randomly generated and have similar solution characteristics. *set1ch* appears to be the most difficult and was retained.

2.4 Constraint Classification

One of the most important techniques to successfully solve mixed integer programs is reformulation. Reformulation techniques, such as preprocessing and cut generation, are incorporated in almost all state-of-the-art mixed integer optimizers. Many reformulation techniques are based on specific structures that may be embedded in the constraint matrix. For example, generation of lifted cover inequalities can only be done if the constraint matrix contains knapsack inequalities. Therefore, knowledge about the structure of the constraint matrix is useful in determining whether certain reformulation techniques can be applied. In view of the above, we have decided to provide some basic information about the constraint matrix of the problems in the MIPLIB. For each of the problems in the MIPLIB information is given about the types of constraints that appear in the constraint matrix. In defining a constraint type we use the symbol x to denote binary variables and the symbol y to denote general integer and continuous variables. Each constraint type will be an equivalence class with respect to complementing binary variables, i.e., if a constraint with term $a_j x_j$ belongs to a given type, then the constraint with $a_j x_j$ replaced by $a_j(1 - x_j)$ also belongs to that class. Consequently, the most general constraint that can appear in a mixed integer program can be represented as follows

$$\sum_{j \in B} a_j x_j + \sum_{j \in I \cup C} a_j y_j \;\square\; b,$$

where B, I and C denote the sets of binary, integer, and continuous variables, respectively, a_j is positive for $j \in B$, a_j is nonzero for $j \in I \cup C$, and $\square$ is the sense of the constraint, either $\geq$ or $=$.

We distinguish the types of constraints given in Table 3.

Note that

$$\sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j \leq 1 - |N^-|$$

is also considered to be a packing constraint, since complementing the variables with a negative coefficient gives

$$\sum_{j \in N^+} x_j + \sum_{j \in N^-} \overline{x}_j \leq 1$$

Table 4 contains information regarding the different types of constraints occuring in each problem as well as the actual number of constraints of each type. For example, the row corresponding to problem *p0033* has entries 11 and 4 in columns K and S, respectively, indicating that eleven knapsack and four special ordered set constraints are present.

3 MIPLIB Availability

The MIPLIB library is accessible via the internet. The MIPLIB page can be found at *http://www.caam.rice.edu/ bixby/miplib/miplib.html.* The library can also be obtained via anonymous ftp. The library and related files are in the directory */pub/people/bixby/miplib* on the machine *ftp.caam.rice.edu.*

4 MIPLIB submissions

As noted earlier, we encourage both researchers and practitioners in integer programming to submit real-world instances to the authors for consideration and possible inclusion in MIPLIB. The library will be updated at least once each year after review of the submitted problems. In order to submit a problem, the MPS file containing the problem should be transferred into the directory *pub/people/bixby/incoming/bixplex* on the machine *ftp.caam.rice.edu.* The submission should be accompanied by an e-mail message to *miplib@caam.rice.edu* containing information about the origin of the instance, the optimal solution (if known), and the way the optimal solution was obtained.

References

[1] Bixby, R. E., E. A. Boyd, and R. R. Indovina. 1992. MIPLIB: A Test Set of Mixed Integer Programming Problems. SIAM News 25:2 (March).

Table 1: Problem Statistics

NAME	ROWS	COLS	INT	0/1	CONT	INT SOLN	LP SOLN
10teams	230	2025	1800	ALL	225	924	917
air03	124	10757	10757	ALL	0	340160	338864.25
air04	823	8904	8904	ALL	0	56137	55535.436
air05	426	7195	7195	ALL	0	26374	25877.609
arki001	1048	1388	538	415	850	7580813.0459(not opt)	7579599.80787
bell3a	123	133	71	39	62	878430.32	862578.64
bell5	91	104	58	30	46	8966406.49	8608417.95
blend2	274	353	264	231	89	7.598985	6.9156751140
cap6000	2176	6000	6000	ALL	0	-2451377	-2451537.325
dano3mip	3202	13873	552	ALL	13321	728.1111(not opt)	576.23162474
danoint	664	521	56	ALL	465	65.67	62.637280418
dcmulti	290	548	75	ALL	473	188182	183975.5397
dsbmip	1182	1886	192	160	1694	-305.19817501	-305.19817501
egout	98	141	55	ALL	86	568.101	149.589
enigma	21	100	100	ALL	0	0.0	0.0
fast0507	507	63009	63009	ALL	0	174	172.14556668
fiber	363	1298	1254	ALL	44	405935.18000	156082.51759
fixnet6	478	878	378	ALL	500	3983	1200.88
flugpl	18	18	11	0	7	1201500	1167185.73
gen	780	870	150	144	720	112313	112130.0
gesa2	1392	1224	408	240	816	25779856.372	25476489.678
gesa2_o	1248	1224	720	384	504	25779856.372	25476489.678
gesa3	1368	1152	384	216	768	27991042.648	27833632.451
gesa3_o	1224	1152	672	336	480	27991042.648	27833632.451
gt2	29	188	188	24	0	21166.000	13460.233074
harp2	112	2993	2993	ALL	0	-73899798.00	-74353341.502
khb05250	101	1350	24	ALL	1326	106940226	95919464.0
l152lav	97	1989	1989	ALL	0	4722	4656.36
lseu	28	89	89	ALL	0	1120	834.68
markshare1	6	62	50	ALL	12	1	0
markshare2	7	74	60	ALL	14	1	0
mas74	13	151	150	ALL	1	11801.1857	10482.795280
mas76	12	151	150	ALL	1	40005.0541	38893.903641
misc03	96	160	159	ALL	1	3360	1910.0
misc06	820	1808	112	ALL	1696	12850.8607	12841.69
misc07	212	260	259	ALL	1	2810	1415.0
mitre	2054	10724	10724	ALL	0	115155	114740.51848
mod008	6	319	319	ALL	0	307	290.93
mod010	146	2655	2655	ALL	0	6548	6532.08
mod011	4480	10958	96	ALL	10862	-54558535	-62121982.552
modglob	291	422	98	ALL	324	20740508	20430947.0
mkc	3411	5325	5323	ALL	2	-553.75(not opt)	-611.85000000
noswot	182	128	100	75	28	-43	-43.0
nw04	36	87482	87482	ALL	0	16862	16310.66667
p0033	16	33	33	ALL	0	3089	2520.57
p0201	133	201	201	ALL	0	7615	6875.0
p0282	241	282	282	ALL	0	258411	176867.50
p0548	176	548	548	ALL	0	8691	315.29
p2756	755	2756	2756	ALL	0	3124	2688.75
pk1	45	86	55	ALL	31	11.0	0.0
pp08a	136	240	64	ALL	176	7350.0	2748.3452381
pp08aCUTS	246	240	64	ALL	176	7350.0	5480.6061563
qiu	1192	840	48	ALL	792	-132.873137	-931.638857
qnet1	503	1541	1417	1288	124	16029.692681	14274.102667
qnet1_o	456	1541	1417	1288	124	16029.692681	12095.571667
rentacar	6803	9557	55	ALL	9502	30356761	28806137.644
rgn	24	180	100	ALL	80	82.1999	48.7999
rout	291	556	315	300	241	1077.56	981.86428571

Table 1: Problem Statistics (cont.)

NAME	ROWS	COLS	INT	0/1	CONT	INT SOLN	LP SOLN
set1ch	492	712	240	ALL	472	54537.75	32007.73
seymour	4944	1372	1372	ALL	0	423(not opt)	403.84647413
stein27	118	27	27	ALL	0	18	13.0
stein45	331	45	45	ALL	0	30	22.0
swath	884	6805	6724	ALL	81	497.603(not opt)	334.4968581
vpm1	234	378	168	ALL	210	20	15.4167
vpm2	234	378	168	ALL	210	13.75	9.8892645972

Table 2: Problem Origins

NAME	ORIGINATOR	FORMULATOR	DONATOR
10teams	Dash Associates		Bob Daniel
air03			Greg Astfalk
air04			Greg Astfalk
air05			Greg Astfalk
arki001	Avesta-Sheffield, Sweden	Nils Holmberg	Arne Stolbjerg Drud
bell3a	William Cook	William Cook	William Cook
bell5	William Cook	William Cook	William Cook
blend2	Dash Associates		Bob Daniel
cap6000	Karla Hoffman, Manfred Padberg	Telecommunications Corporation	Karla Hoffman
dano3mip	Bell Communications Research		Daniel Bienstock
danoint	Columbia's Center for Telecommunications Research		Daniel Bienstock
dcmulti	Jeremy Shapiro	Jeremy Shapiro	Jonathan Eckstein
dsbmip			John J. Forrest
egout	Etienne Loute	Laurence A. Wolsey	Martin W.P. Savelsbergh
enigma	Harlan Crowder	Harlan Crowder	E. Andrew Boyd
fast0507	Italian Railway Company	Pier Luigi Guida	Sebastián Ceria
fiber	US West	Youngho Lee	Martin W.P. Savelsbergh
fixnet6		T. J. Van Roy	Martin W.P. Savelsbergh
flugpl	Harvey M. Wagner	John W. Gregory	E. Andrew Boyd
gesa2	Spanish Electricity	Laurence A. Wolsey	Sebastián Ceria
gesa2_o	Spanish Electricity	GESA (Consulting Company)	Sebastián Ceria
gesa3	Spanish Electricity	Laurence A. Wolsey	Sebastián Ceria
gesa3_o	Spanish Electricity	GESA (Consulting Company)	Sebastián Ceria
gen		Laurence A. Wolsey	Martin W.P. Savelsbergh
gt2			Sebastián Ceria
harp2			Martin W.P. Savelsbergh
khb05250	Kuhn-Hamburger	Laurence A. Wolsey	Martin W.P. Savelsbergh
l152lav	Harlan Crowder	Harlan Crowder	John W. Gregory
lseu	C. E. Lemke, K. Spielberg	Ellis L. Johnson, Uwe H. Suhl	John J. Forrest
markshare1	Gerard Cornuejols Milind Dawande	Gerard Cornuejols Milind Dawande	Milind Dawande
markshare2	Gerard Cornuejols Milind Dawande	Gerard Cornuejols Milind Dawande	Milind Dawande
mas74	Undisclosed	Undisclosed	Jonathan Eckstein
mas76	Undisclosed	Undisclosed	Jonathan Eckstein
misc03			Greg Astfalk
misc06			Greg Astfalk
misc07			Greg Astfalk
mitre			Martin W.P. Savelsbergh
mkc	Jayant Kalagnanam Milind Dawande	Jayant Kalagnanam Milind Dawande	Jayant Kalagnanam Milind Dawande
mod008	IBM France	IBM France	John J. Forrest
mod010	IBM Yorktown Hts	IBM Yorktown Hts	John J. Forrest
mod011	Uwe H. Suhl	Uwe H. Suhl	John J. Forrest
modglob	Y. Smeers	Laurence A. Wolsey	Martin W.P. Savelsbergh

Table 2: Problem Origins (cont.)

NAME	ORIGINATOR	FORMULATOR	DONATOR
noswot		Linus E. Schrage	John W. Gregory
nw04	Northwest Airlines		Karla Hoffman
p0033	CJP set		E. Andrew Boyd
p0201	CJP set		E. Andrew Boyd
p0282	CJP set		E. Andrew Boyd
p0548	CJP set	Ellis L. Johnson	E. Andrew Boyd
p2756	CJP set	Ellis L. Johnson	E. Andrew Boyd
pk1		Pinar Keskinocak	Sebastián Ceria
pp08a			Martin W.P. Savelsbergh
pp08aCUTS			Martin W.P. Savelsbergh
qiu	Yu-Ping Chiu	Yu-Ping Chiu	Jonathan Eckstein
qnet1	BASF	Laurence A. Wolsey	Sebastián Ceria
qnet1_o	BASF	BASF	Sebastián Ceria
rentacar			John J. Forrest
rgn	Linus E. Schrage	Laurence A. Wolsey	Martin W.P. Savelsbergh
rout	S. Graves	Hernan Abeledo	Sebastián Ceria
set1ch		Laurence A. Wolsey	Martin W.P. Savelsbergh
seymour			Paul Seymour
stein27	George L. Nemhauser	John W. Gregory	E. Andrew Boyd
stein45	George L. Nemhauser	John W. Gregory	E. Andrew Boyd
swath	Undisclosed	David Panton	David Panton
vpm1		Laurence A. Wolsey	Martin W.P. Savelsbergh
vpm2		Laurence A. Wolsey	Martin W.P. Savelsbergh

Table 3: Constraint types

G	General	$\sum_{j\in B} a_j x_j + \sum_{j\in I\cup C} a_j y_j \square b$
K	Knapsack	$\sum_{j\in B} a_j x_j \leq b$
E	Equality knapsack	$\sum_{j\in B} a_j x_j = b$
F	Facility location	$\sum_{j\in B} a_j x_j \square a_k x_k$
I	Invariant knapsack	$\sum_{j\in B} x_j \square k (k \neq 1 \wedge k \neq \lvert B\rvert - 1)$
P	Packing	$\sum_{j\in B} x_j \leq 1$
C	Covering	$\sum_{j\in B} x_j \geq 1$
S	Special ordered set	$\sum_{j\in B} x_j = 1$
U	variable Upper bound	$a_j y_j \square a_k x_k$
L	variable Lower bound	$a_j y_j \geq a_k x_k$

Table 4: Constraint types

NAME	G	K	E	F	I	P	C	S	U	L
10teams	110					40		80		
air03								124		
air04								823		
air05								426		
arki001	1027	13			6	2				
bell3a	101					22				
bell5	76					15				
blend2	93				79		9		88	
cap6000		7				2046		123		
dano3mip	2518								606	30
danoint	256								392	
dcmulti	165					22	10		45	45
dsbmip	1142							40		
egout	43								55	
enigma			1					20		
fast0507						3	504			
fiber	44							90		
fixnet6	100								378	
flugpl	18									
gen	318	24							432	6
gesa2	912					14			288	144
gesa2_o	624					48	144		288	144
gesa3	936					48			264	120
gesa3_o	672					48	120		264	120
gt2	27					2				
harp2		30		9				73		
khb05250	77								24	
l152lav		1						95		
lseu		11				17				
markshare1	6									
markshare2	7									
mas74	12				1					
mas76	11				1					
misc03	1	3		2	30	3	31	5		
misc06	820									
misc07	1	3		2	42	3	127	7		
mitre		383			1148		140	383		
mkc	2			24	24	3361				
mod008		6								
mod010		1						144		
mod011	4400							16	64	
modglob	95								196	
noswot	137								20	25
nw04								36		
p0033		11				4				
p0201		33			54	26	20			
p0282		64				177				
p0548		112				64				
p2756		403				352				
pk1	45									

Table 4: Constraint types

NAME	G	K	E	F	I	P	C	S	U	L
pp08a	72								64	
pp08aCUTS	182								64	
qiu	664								528	
qnet1	178			32		4	4	48		45
qnet1_o	152			32				48		32
rentacar	6674								55	55
rgn	20					4				
rout	47					14			230	
set1ch	252								240	
seymour						285	4659			
stein27					1		117			
stein45					1	1	329			
swath	381				80			423		
vpm1	66								168	
vpm2	66								168	