

1
2
3
4
5
6
7
8

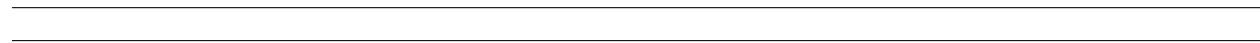
SUPPLEMENTARY INFORMATION FOR:

An integrated platform for intuitive mathematical programming modeling using L^AT_EX

Charalampos P. Triantafyllidis^{a,b}, Lazaros G. Papageorgiou ^{a,*}

^a*Centre for Process Systems Engineering, Department of Chemical Engineering,
UCL (University College London), UK*

^b*Smith School of Enterprise and the Environment,
University of Oxford, UK*



*Corresponding Author

Email addresses:

`h.triantafyllidis@ucl.ac.uk`, `charalampos.triantafyllidis@smithschool.ox.ac.uk` (Charalampos P. Triantafyllidis), `l.papageorgiou@ucl.ac.uk` (Lazaros G. Papageorgiou)

9 1. Library of modeling examples

10 A collection of well-known problems alongside some more executive problems is given, so the
11 user gets a rigid idea of how problems of different formulations and requirements can be represented
12 inside the platform in an admissible manner.

13 The first two problems, the Transportation problem and the Traveling Salesman problem, are
14 well-known in theoretical computer science with applications that span numerous fields. All the
15 formulations presented below (as raw L^AT_EX code of course) were the ones also used as input for
16 the platform exactly as shown here.

17 1.1. The Transportation problem

18 We wish to find optimum transportation schedule keeping cost of transport minimized. A
19 number of sources distribute a product to a number of destinations and we wish to optimize
20 the effective cost while satisfying the required given demands (Schrijver, 2002). The well-known
21 modeling approach for this problem is given below:

$$\begin{aligned} & \text{minimize} && \sum_{i,j} (c_{i,j} x_{i,j}) \\ & \text{subject to:} && \sum_j (x_{i,j}) \leq a_i \quad \forall i \\ & && \sum_i (x_{i,j}) \geq b_j \quad \forall j \\ & && x \in R_+ \end{aligned}$$

22 This problem is the simplest included, as only two sets of constraints are given and the single set
23 of variables belongs to (positive) real numbers. Once we input this problem to the platform with
24 the associated data-set we get the following parser output:

```
25 -Reading the variable's types and symbols...
26 The optimization problem has :
27 (0) binary set(s) of variable(s)
28 (0) set(s) of integer variable(s) and
29 (1) set(s) of positive real variable(s)
30 (0) set(s) of non-negative real variable(s)
31 -Reading the objective function definition...
32 The optimization sense is: MINIMIZE
33 ***Parameter 'c' detected in the constraint***
34 **Variable 'x' detected in the objective function with index set: {i,
35 j}**
36 Found the remaining 1 variables definitions in the constraints!
37 -Reading the constraints set (1)...
38 ***Parameter 'a' detected in the constraint***
39 -Reading the constraints set (2)...
40 ***Parameter 'b' detected in the constraint***
41 Parsing the model terminated successfully... Now printing the Pyomo equivalent
42 model...
43
```

45 We can see that the parser correctly identified the following:

- 46 • the sets and types of variables
- 47 • the type of the optimization

- the variables existing in the objective function and the parameters
- an on-the-fly detection of parameters a, b while scanning through the sets of constraints.

The .py model file was generated as follows:

```

51 #####
52 # Define imports of packages:
53 #####
54 from pyomo.environ import *
55 from pyomo.opt import SolverFactory
56 model = AbstractModel()
57 #####
58 # Define variables , parameters and sets:
59 #####
60 model.i = Set(dimen=1)
61 model.j = Set(dimen=1)
62 #####
63 # New parameter detected:
64 model.c = Param(model.i,model.j, initialize = 0)
65 #####
66 model.x = Var(model.i,model.j, domain=NonNegativeReals)
67 #####
68 # Define objective function:
69 #####
70 def obj_expression(model):
71     model.F = sum(model.c[i,j]*model.x[i,j] for i in model.i for j in model.j)
72     return model.F
73 model.OBJ = Objective(rule=obj_expression, sense = minimize)
74 #####
75 # Define constraints:
76 #####
77 #####
78 #####
79 # New parameter detected:
80 model.a = Param(model.i, initialize = 0)
81 #####
82 def axb_constraint_rule_1(model,i):
83     model.C_1= sum(model.x[i,j] for j in model.j) <= model.a[i]
84     return model.C_1
85 model.AxbConstraint_1=Constraint(model.i,rule=axb_constraint_rule_1)
86 #####
87 # New parameter detected:
88 model.b = Param(model.j, initialize = 0)
89 #####
90 def axb_constraint_rule_2(model,j):
91     model.C_2= sum(model.x[i,j] for i in model.i) >= model.b[j]
92     return model.C_2
93 model.AxbConstraint_2=Constraint(model.j,rule=axb_constraint_rule_2)
94 #####
95 # Now link the .data file to the model:
96 #####
97 if __name__ == '__main__':
98     opt = SolverFactory('cplex')
99     opt.options['threads'] = 12
100    opt.options['parallel'] = -1

```

```
101 instance = model.create_instance(r'lp_transport.dat')
102 results = opt.solve(instance, tee=True)
103 results.write()
```

105 Then, by merging the Pyomo model with the data-set, CPLEX gives:

```
106 Welcome to IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 12.6.3.0
107 with Simplex, Mixed Integer & Barrier Optimizers
108 5725-A06 5725-A29 5724-Y48 5724-Y49 5724-Y54 5724-Y55 5655-Y21
109 Copyright IBM Corp. 1988, 2015. All Rights Reserved.
110
111 Type 'help' for a list of available commands.
112 Type 'help' followed by a command name for more
113 information on commands.
114
115 CPLEX> Logfile 'cplex.log' closed.
116 CPLEX> New value for parallel optimization mode: -1
117 CPLEX> New value for default parallel thread count: 12
118 Logfile 'cplex.log' open.
119 Read time = 0.00 sec. (0.00 ticks)
120 Objective sense      : Minimize
121 Variables            :      7
122 Objective nonzeros   :      6
123 Linear constraints    :      6 [Less: 2, Greater: 3, Equal: 1]
124   Nonzeros           :     13
125   RHS nonzeros       :      6
126
127 Variables            : Min LB: 0.0000000      Max UB: all infinite
128 Objective nonzeros   : Min   : 0.1260000      Max   : 0.2250000
129 Linear constraints    :
130   Nonzeros           : Min   : 1.0000000      Max   : 1.0000000
131   RHS nonzeros       : Min   : 1.0000000      Max   : 600.0000
132
133 CPLEX> Tried aggregator 1 time.
134 LP Presolve eliminated 1 rows and 1 columns.
135 Reduced LP has 5 rows, 6 columns, and 12 nonzeros.
136 Presolve time = 0.00 sec. (0.00 ticks)
137
138 Iteration log . . .
139 Iteration:      1   Dual objective      =           73.125000
140
141 Dual simplex - Optimal: Objective = 1.5367500000e+002
142 Solution time =   0.00 sec. Iterations = 4 (0)
143 Deterministic time = 0.01 ticks (10.12 ticks/sec)
```

145 The associated data-set for this example was :

```
146 set i := 1 2;
147 set j := 1 2 3;
148
149 param a:=
150 1 350
151 2 600;
152
153
154 param b:=
```

```

155 1 325
156 2 300
157 3 275;
158
159 param c:=
160 1 1 0.225
161 1 2 0.153
162 1 3 0.162
163 2 1 0.225
164 2 2 0.162
165 2 3 0.126 ;

```

167 1.2. The Traveling Salesman Problem

168 One of the most well-known and examined problems, stills being very difficult to solve as the
169 scale of the problem increases is the Traveling Salesman Problem ([Applegate et al., 2007](#)), ([Johnson,](#)
170 [1990](#)).

$$\begin{aligned}
& \text{minimize} && \sum_{i,j:i \neq j} (c_{i,j}x_{i,j}) \\
& \text{subject to:} && \\
& && \sum_{j:i \neq j} (x_{i,j}) = 1 && \forall i \\
& && \sum_{i:i \neq j} (x_{i,j}) = 1 && \forall j \\
& && u_i - u_j + n * x_{i,j} \leq n - 1 && \forall i \geq 2, i \neq j, j \leq n \\
& && u \in Z_+, x \in \{0, 1\}
\end{aligned}$$

171 The parser correctly converts the model and the output is given below:

```

172
173 -Reading the variable's types and symbols...
174 The optimization problem has :
175 (1) binary set(s) of variable(s)
176 (0) set(s) of integer variable(s) and
177 (0) set(s) of positive real variable(s)
178 (0) set(s) of non-negative real variable(s)
179 (1) set(s) of positive integer variable(s)
180 -Reading the objective function definition...
181 The optimization sense is: MINIMIZE
182 ***Parameter 'c' detected in the constraint***
183 **Variable 'x' detected in the objective function with index set: {i,
184 j}**
185 1 out of total 2 sets of variables were not detected in the objective
186 function...
187 Searching for the rest of the variables in the model...
188 Found set(s) of variable(s) u with index set {i}
189 Found the remaining 1 variables definitions in the constraints!
190 -Reading the constraints set (1)...
191 -Reading the constraints set (2)...
192 -Reading the constraints set (3)...
193 ***Parameter 'n' detected in the constraint***
194 Parsing the model terminated successfully... Now printing the Pyomo equivalent
195 model...

```

197 The produced Pyomo model is as follows:

```
198
199 #####
200 # Define imports of packages:
201 #####
202 from pyomo.environ import *
203 from pyomo.opt import SolverFactory
204 model = AbstractModel()
205 #####
206 # Define variables, parameters and sets:
207 #####
208 model.i = Set(dimen=1)
209 model.j = Set(dimen=1)
210 #####
211 # New parameter detected:
212 model.c = Param(model.i,model.j, initialize = 0)
213 #####
214 model.x = Var(model.i,model.j, domain=Binary)
215 model.u = Var(model.i, domain=PositiveIntegers)
216 #####
217 # Define objective function:
218 #####
219 def obj_expression(model):
220     model.F = sum(model.c[i,j]*model.x[i,j] for i in model.i for j in model.j if
221         i!=j)
222     return model.F
223 model.OBJ = Objective(rule=obj_expression, sense = minimize)
224 #####
225 # Define constraints:
226 #####
227 def axb_constraint_rule_1(model,i):
228     model.C_1= sum(model.x[i,j] for j in model.j if i!=j) == 1
229     return model.C_1
230 model.AxbConstraint_1=Constraint(model.i,rule=axb_constraint_rule_1)
231 def axb_constraint_rule_2(model,j):
232     model.C_2= sum(model.x[i,j] for i in model.i if i!=j) == 1
233     return model.C_2
234 model.AxbConstraint_2=Constraint(model.j,rule=axb_constraint_rule_2)
235 #####
236 # New parameter detected:
237 model.n = Param(initialize = 0)
238 #####
239 def axb_constraint_rule_3(model,i,j):
240     if i>=2 and i!=j and j<=model.n:
241         model.C_3= model.u[i]-model.u[j]+model.n*model.x[i,j] <= model.n-1
242         return model.C_3
243     else:
244         return Constraint.Skip
245 model.AxbConstraint_3=Constraint(model.i,model.j,rule=axb_constraint_rule_3)
246 #####
247 # Now link the .data file to the model:
248 #####
249 if __name__ == '__main__':
250     opt = SolverFactory('cplex')
```

```

251 | opt.options[ 'threads' ] = 12
252 | opt.options[ 'parallel' ] = -1
253 | instance = model.create_instance( r'TSP.dat' )
254 | results = opt.solve(instance, tee=True)
255 | results.write()

```

257 Solving the model with the associated data-set gives:

```

258 | Welcome to IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 12.6.3.0
259 | with Simplex, Mixed Integer & Barrier Optimizers
260 | 5725-A06 5725-A29 5724-Y48 5724-Y49 5724-Y54 5724-Y55 5655-Y21
261 | Copyright IBM Corp. 1988, 2015. All Rights Reserved.
262 |
263 |
264 | Type 'help' for a list of available commands.
265 | Type 'help' followed by a command name for more
266 | information on commands.
267 |
268 | CPLEX> Logfile 'cplex.log' closed.
269 | CPLEX> New value for parallel optimization mode: -1
270 | Logfile 'cplex.log' open.
271 | CPLEX> New value for default parallel thread count: 12
272 | Read time = 0.00 sec. (0.00 ticks)
273 | Objective sense      : Minimize
274 | Variables            :      17  [Nneg: 1, Binary: 12, General Integer: 4]
275 | Objective nonzeros   :      12
276 | Linear constraints   :      18  [Less: 9, Equal: 9]
277 |   Nonzeros           :      52
278 |   RHS nonzeros       :      18
279 |
280 | Variables            : Min LB: 0.0000000      Max UB: 1.0000000
281 | Objective nonzeros   : Min   : 2.0000000      Max   : 100.00000
282 | Linear constraints   :
283 |   Nonzeros           : Min   : 1.0000000      Max   : 4.0000000
284 |   RHS nonzeros       : Min   : 1.0000000      Max   : 3.0000000
285 | CPLEX> Tried aggregator 1 time.
286 | MIP Presolve eliminated 4 rows and 2 columns.
287 | Reduced MIP has 14 rows, 15 columns, and 42 nonzeros.
288 | Reduced MIP has 12 binaries, 3 generals, 0 SOSs, and 0 indicators.
289 | Presolve time = 0.00 sec. (0.02 ticks)
290 | Probing time = 0.00 sec. (0.01 ticks)
291 | Tried aggregator 1 time.
292 | Reduced MIP has 14 rows, 15 columns, and 42 nonzeros.
293 | Reduced MIP has 12 binaries, 3 generals, 0 SOSs, and 0 indicators.
294 | Presolve time = 0.00 sec. (0.02 ticks)
295 | Probing time = 0.00 sec. (0.01 ticks)
296 | Clique table members: 11.
297 | MIP emphasis: balance optimality and feasibility.
298 | MIP search method: dynamic search.
299 | Parallel mode: opportunistic, using up to 12 threads.
300 | Root relaxation solution time = 0.00 sec. (0.02 ticks)
301 |
302 |           Nodes                               Cuts/
303 | Node  Left   Objective  IInf  Best Integer  Best Bound  ItCnt  Gap
304 |

```

```

305 *      0+   0                302.0000      0.0000
306     100.00%
307 *      0   0      integral    0      14.0000      14.0000      6
308     0.00%
309 Elapsed time = 0.00 sec. (0.19 ticks , tree = 0.00 MB, solutions = 2)
310
311 Root node processing (before b&c):
312   Real time           =    0.00 sec. (0.19 ticks)
313 Parallel b&c, 12 threads:
314   Real time           =    0.00 sec. (0.00 ticks)
315   Sync time (average) =    0.00 sec.
316   Wait time (average) =    0.00 sec.
317
318 Total (root+branch&cut) =    0.00 sec. (0.19 ticks)
319
320 Solution pool: 2 solutions saved.
321
322 MIP - Integer optimal solution: Objective = 1.4000000000e+001
323 Solution time =    0.00 sec. Iterations = 6 Nodes = 0
324 Deterministic time = 0.19 ticks (192.41 ticks/sec)

```

326 The data-set for this demonstration was:

```

327
328 set i := 1 2 3 4;
329 set j := 1 2 3 4;
330 param n:= 4 ;
331 param c:=
332     1 1  100
333     1 2   2
334     1 3  100
335     1 4  100
336     2 1  100
337     2 2  100
338     2 3   3
339     2 4  100
340     3 1  100
341     3 2  100
342     3 3  100
343     3 4   4
344     4 1   5
345     4 2  100
346     4 3  100
347     4 4  100 ;
348

```

350 1.3. Cancer Therapy

351 The model which was original published in (Knijnenburg et al., 2016) and attempts to binarize
352 continuous outputs for cancer research, via Binary-Integer Optimization, based on a Boolean-
353 Function Synthesis formulation:


```

minimize      -  $\sum_{n:y_n=0} (w_n y_{2n}) - \sum_{n:y_n=1} (w_n y_{2n})$ 
subject to:
               $sa_{p,k} + sb_{p,k} \leq 1$   $\forall p, k$ 
               $\sum_p (sa_{p,k} + sb_{p,k}) \leq M$   $\forall k$ 
               $g * t_{n,k} \leq \sum_{p:X_{n,p}=1} (1 - sb_{p,k}) + \sum_{p:X_{n,p}=0} (1 - sa_{p,k})$   $\forall n, k$ 
               $\sum_{p:X_{n,p}=1} (1 - sb_{p,k}) + \sum_{p:X_{n,p}=0} (1 - sa_{p,k}) \leq t_{n,k} + g - 1$   $\forall n, k$ 
               $y_{2n} \leq \sum_k (t_{n,k})$   $\forall n$ 
               $\sum_k (t_{n,k}) \leq d * y_{2n}$   $\forall n$ 
               $t, sa, sb, y_{2n} \in \{0, 1\}$ 

```

354 The parser gives:

```

355
356     -Reading the variable's types and symbols...
357     The optimization problem has :
358         (4) binary set(s) of variable(s)
359         (0) set(s) of integer variable(s) and
360         (0) set(s) of positive real variable(s)
361         (0) set(s) of non-negative real variable(s)
362     -Reading the objective function definition...
363     The optimization sense is: MINIMIZE
364     ***Parameter 'w' detected in the constraint***
365     **Variable 'y2' detected in the objective function with index set: {n
366 }**
367     3 out of total 4 sets of variables were not detected in the objective
368 function...
369     Searching for the rest of the variables in the model...
370         Found set(s) of variable(s) sa with index set {p,k}
371         Found set(s) of variable(s) sb with index set {p,k}
372         Found set(s) of variable(s) t with index set {n,k}
373     Found the remaining 3 variables definitions in the constraints!
374     -Reading the constraints set (1)...
375     -Reading the constraints set (2)...
376         ***Parameter 'M' detected in the constraint***
377     -Reading the constraints set (3)...
378         ***Parameter 'g' detected in the constraint***
379     -Reading the constraints set (4)...
380     -Reading the constraints set (5)...
381     -Reading the constraints set (6)...
382         ***Parameter 'd' detected in the constraint***
383 Parsing the model terminated successfully... Now printing the Pyomo equivalent
384 model...

```

386 The Pyomo output of the parser is:

```

387
388 #####
389 # Define imports of packages:
390 #####

```

```

391 from pyomo.environ import *
392 from pyomo.opt import SolverFactory
393 model = AbstractModel()
394 #####
395 # Define variables , parameters and sets :
396 #####
397 model.n = Set(dimen=1)
398 #####
399 # New parameter detected:
400 model.w = Param(model.n, initialize = 0)
401 #####
402 model.y2 = Var(model.n, domain=Binary)
403 #####
404 # New indexing parameter detected...
405 model.y = Param(model.n, initialize = 0)
406 #####
407 model.p = Set(dimen=1)
408 model.k = Set(dimen=1)
409 model.sa = Var(model.p, model.k, domain=Binary)
410 model.sb = Var(model.p, model.k, domain=Binary)
411 model.t = Var(model.n, model.k, domain=Binary)
412 #####
413 # Define objective function:
414 #####
415 def obj_expression(model):
416     model.F = sum(-model.w[n]*model.y2[n] for n in model.n if model.y[n]==0)-sum
417     (model.w[n]*model.y2[n] for n in model.n if model.y[n]==1)
418     return model.F
419 model.OBJ = Objective(rule=obj_expression , sense = minimize)
420 #####
421 # Define constraints:
422 #####
423 def axb_constraint_rule_1(model,p,k):
424     model.C_1= model.sa[p,k]+model.sb[p,k] <= 1
425     return model.C_1
426 model.AxbConstraint_1=Constraint(model.p,model.k,rule=axb_constraint_rule_1)
427 #####
428 # New parameter detected:
429 model.M = Param(initialize = 0)
430 #####
431 def axb_constraint_rule_2(model,k):
432     model.C_2= sum(model.sa[p,k]+model.sb[p,k] for p in model.p) <= model.M
433     return model.C_2
434 model.AxbConstraint_2=Constraint(model.k,rule=axb_constraint_rule_2)
435 #####
436 # New parameter detected:
437 model.g = Param(initialize = 0)
438 #####
439 #####
440 # New indexing parameter detected...
441 model.X = Param(model.n,model.p, initialize = 0)
442 #####
443 def axb_constraint_rule_3(model,n,k):
444     model.C_3= model.g*model.t[n,k] <= sum(1-model.sb[p,k] for p in model.p if

```

```

445     model.X[n,p]==1)+sum(1-model.sa[p,k] for p in model.p if model.X[n,p]==0)
446     return model.C_3
447 model.AxbConstraint_3=Constraint(model.n,model.k,rule=axb_constraint_rule_3)
448 def axb_constraint_rule_4(model,n,k):
449     model.C_4= sum(1-model.sb[p,k] for p in model.p if model.X[n,p]==1)+sum(1-
450     model.sa[p,k] for p in model.p if model.X[n,p]==0) <= model.t[n,k]+model.g
451     -1
452     return model.C_4
453 model.AxbConstraint_4=Constraint(model.n,model.k,rule=axb_constraint_rule_4)
454 def axb_constraint_rule_5(model,n):
455     model.C_5= model.y2[n] <= sum(model.t[n,k] for k in model.k)
456     return model.C_5
457 model.AxbConstraint_5=Constraint(model.n,rule=axb_constraint_rule_5)
458 #####
459 # New parameter detected:
460 model.d = Param(initialize = 0)
461 #####
462 def axb_constraint_rule_6(model,n):
463     model.C_6= sum(model.t[n,k] for k in model.k) <= model.d*model.y2[n]
464     return model.C_6
465 model.AxbConstraint_6=Constraint(model.n,rule=axb_constraint_rule_6)
466 #####
467 # Now link the .data file to the model:
468 #####
469 if __name__ == '__main__':
470     opt = SolverFactory('cplex')
471     opt.options['threads'] = 12
472     opt.options['parallel'] = -1
473     instance = model.create_instance(r'LOBICO.dat')
474     results = opt.solve(instance, tee=True)
475     results.write()

```

477 CPLEX converged as follows:

```

478 Welcome to IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 12.6.3.0
479 with Simplex, Mixed Integer & Barrier Optimizers
480 5725-A06 5725-A29 5724-Y48 5724-Y49 5724-Y54 5724-Y55 5655-Y21
481 Copyright IBM Corp. 1988, 2015. All Rights Reserved.
482
483
484 Type 'help' for a list of available commands.
485 Type 'help' followed by a command name for more
486 information on commands.
487
488 CPLEX> Logfile 'cplex.log' closed.
489 CPLEX> New value for parallel optimization mode: -1
490 CPLEX> New value for default parallel thread count: 12
491 Logfile 'cplex.log' open.
492 Read time = 0.03 sec. (2.36 ticks)
493 Objective sense      : Minimize
494 Variables            :    2167  [Nneg: 1,  Binary: 2166]
495 Objective nonzeros   :     642
496 Linear constraints    :    3975  [Less: 3974,  Equal: 1]
497   Nonzeros           :   160981
498   RHS nonzeros       :     2691

```

```

499
500 Variables           : Min LB: 0.0000000      Max UB: 1.000000
501 Objective nonzeros  : Min   : 1.000000e-005    Max   : 0.03049700
502 Linear constraints  :
503   Nonzeros          : Min   : 1.000000      Max   : 60.00000
504   RHS nonzeros      : Min   : 1.000000      Max   : 60.00000
505 CPLEX> Tried aggregator 1 time.
506 MIP Presolve eliminated 1958 rows and 3 columns.
507 MIP Presolve added 1106 rows and 0 columns.
508 MIP Presolve modified 78324 coefficients.
509 Reduced MIP has 3123 rows, 2164 columns, and 82967 nonzeros.
510 Reduced MIP has 2164 binaries, 0 generals, 0 SOSs, and 0 indicators.
511 Presolve time = 0.19 sec. (348.06 ticks)
512 Found incumbent of value 0.000008 after 0.25 sec. (471.51 ticks)
513 Probing time = 0.00 sec. (7.15 ticks)
514 Tried aggregator 1 time.
515 Reduced MIP has 3123 rows, 2164 columns, and 82967 nonzeros.
516 Reduced MIP has 2164 binaries, 0 generals, 0 SOSs, and 0 indicators.
517 Presolve time = 0.05 sec. (106.61 ticks)
518 Probing time = 0.02 sec. (7.22 ticks)
519 Clique table members: 4098.
520 MIP emphasis: balance optimality and feasibility.
521 MIP search method: dynamic search.
522 Parallel mode: opportunistic, using up to 12 threads.
523 Root relaxation solution time = 0.28 sec. (345.09 ticks)
524
525      Nodes
526      Node  Left      Objective  IInf  Best Integer      Cuts/
527                                     Best Bound      ItCnt      Gap
528 *      0+      0              -0.5507  1644      0.0000      -1.0000      1549      —
529      0      0              -0.5507  1644      0.0000      -0.5507      1549      —
530 *      0+      0              -0.1103      -0.5507
531      399.45%
532 *      0+      0              -0.2646      -0.5507
533      108.13%
534      0      0              -0.5442  1608      -0.2646      Cuts: 133      1601
535      105.68%
536      0      0              -0.5410  1605      -0.2646      Cuts: 206      1631
537      104.46%
538      0      0              -0.5101  1843      -0.2646      Cuts: 23      1857
539      92.78%
540      0      0              cutoff      -0.2646      -0.2646      1915
541      0.00%
542 Elapsed time = 2.28 sec. (2770.31 ticks, tree = 0.01 MB, solutions = 3)
543
544 Zero-half cuts applied: 262
545 Lift and project cuts applied: 2
546 Gomory fractional cuts applied: 7
547
548 Root node processing (before b&c):
549   Real time           = 2.28 sec. (2770.42 ticks)
550 Parallel b&c, 12 threads:
551   Real time           = 0.00 sec. (0.00 ticks)
552   Sync time (average) = 0.00 sec.

```

```
553 | Wait time (average) = 0.00 sec.  
554 | _____  
555 | Total (root+branch&cut) = 2.28 sec. (2770.42 ticks)  
556 |  
557 | Solution pool: 3 solutions saved.  
558 |  
559 | MIP - Integer optimal solution: Objective = -2.6459700000e-001  
560 | Solution time = 2.28 sec. Iterations = 1915 Nodes = 0  
561 | Deterministic time = 2770.43 ticks (1214.57 ticks/sec)  
562 |
```

563 The data file can be found online using the manuscript given links, and as being very large it was
564 omitted here. The value of the objective function for the supplied example was verified to be correct
565 by comparing it with the equivalent MATLAB model that the authors developed and distributed.

566 **2. Other modeling examples**

567 Examples of MILP and MIQP formulations which were successfully parsed by the platform are
 568 given below.

569 *2.1. Optimal Process Plant Layout*

570 Designing the layout of a chemical plant involves decisions concerning the spatial allocation of
 571 equipment items and the required connections among them. This problem is known as the *layout*
 572 *problem* and the model we used below was originally presented in (Papageorgiou and Rotstein,
 573 1998):

$$\begin{aligned}
 &\text{minimize} && \sum_{i,j:i \neq j} (C_{i,j}D_{i,j}) \\
 &\text{subject to:} && \\
 & && D_{i,j} = R_{i,j} + LF_{i,j} + A_{i,j} + B_{i,j} && \forall i < |i|, j \geq i + 1 \\
 & && l_i = a_i o_i + b_i (1 - o_i) && \forall i \\
 & && d_i = a_i + b_i - l_i && \forall i \\
 & && x_i \geq \frac{l_i}{2} && \forall i \\
 & && y_i \geq \frac{d_i}{2} && \forall i \\
 & && R_{i,j} - LF_{i,j} = x_i - x_j && \forall i < |i|, j \geq i + 1 \\
 & && A_{i,j} - B_{i,j} = y_i - y_j && \forall i < |i|, j \geq i + 1 \\
 & && x_i - x_j + M(E1_{i,j} + E2_{i,j}) \geq \frac{l_i}{2} + \frac{l_j}{2} && \forall i < |i|, j \geq i + 1 \\
 & && x_j - x_i + M(1 - E1_{i,j} + E2_{i,j}) \geq \frac{l_i}{2} + \frac{l_j}{2} && \forall i < |i|, j \geq i + 1 \\
 & && y_i - y_j + M(1 + E1_{i,j} - E2_{i,j}) \geq \frac{d_i}{2} + \frac{d_j}{2} && \forall i < |i|, j \geq i + 1 \\
 & && y_j - y_i + M(2 - E1_{i,j} - E2_{i,j}) \geq \frac{d_i}{2} + \frac{d_j}{2} && \forall i < |i|, j \geq i + 1 \\
 & && l, LF, d, x, y, A, B, D, R \in R_+, o, E1, E2 \in \{0, 1\}
 \end{aligned}$$

574 The parser output is the following one:

```

575 -Reading the variable's types and symbols...
576 The optimization problem has :
577 (3) binary set(s) of variable(s)
578 (0) set(s) of integer variable(s) and
579 (9) set(s) of positive real variable(s)
580 (0) set(s) of non-negative real variable(s)
581 -Reading the objective function definition...
582 The optimization sense is: MINIMIZE
583 ***Parameter 'C' detected in the constraint***
584 **Variable 'D' detected in the objective function with index set: {i,
585 j}**
586 11 out of total 12 sets of variables were not detected in the
587 objective function...
588 Searching for the rest of the variables in the model...
589 Found set(s) of variable(s) LF with index set {i, j}
590 Found set(s) of variable(s) A with index set {i, j}
591 Found set(s) of variable(s) R with index set {i, j}
592 Found set(s) of variable(s) l with index set {i}
593 Found set(s) of variable(s) o with index set {i}
594 Found set(s) of variable(s) d with index set {i}
595 Found set(s) of variable(s) x with index set {i}
596

```

```
597         Found set(s) of variable(s) y with index set {i}
598         Found set(s) of variable(s) B with index set {i,j}
599         Found set(s) of variable(s) E1 with index set {i,j}
600         Found set(s) of variable(s) E2 with index set {i,j}
601     Found the remaining 11 variables definitions in the constraints!
602     -Reading the constraints set (1)...
603     -Reading the constraints set (2)...
604         ***Parameter 'a' detected in the constraint***
605         ***Parameter 'b' detected in the constraint***
606     -Reading the constraints set (3)...
607     -Reading the constraints set (4)...
608     -Reading the constraints set (5)...
609     -Reading the constraints set (6)...
610     -Reading the constraints set (7)...
611     -Reading the constraints set (8)...
612         ***Parameter 'M' detected in the constraint***
613     -Reading the constraints set (9)...
614     -Reading the constraints set (10)...
615     -Reading the constraints set (11)...
616 Parsing the model terminated successfully... Now printing the Pyomo equivalent
617 model...
```

619 *2.2. Community detection via modularity metric optimization*

620 The detection of community structure has been used to reveal the relationships between individ-
621 ual objects and their groupings in networks. A mathematical programming approach to identify the
622 optimal community structures in complex networks based on the maximization of a network mod-
623 ularity metric for partitioning a network into modules is formulated as a mixed integer quadratic
624 programming (MIQP) model. This problem is known as *Modularity Optimization* and the model
625 we used below was originally presented in (Xu et al., 2007):

$$\begin{aligned}
&\text{maximize} && \sum_m \left(\frac{L_m}{\text{Lambda}} - \left(\frac{D_m}{2\text{Lambda}} \right)^2 \right) \\
&\text{subject to:} && \\
&&& \sum_m (Y_{n,m}) = 1 && \forall n \\
&&& X_{n,e,m} \leq Y_{n,m} && \forall m, (n, e) \in LK \\
&&& X_{n,e,m} \leq Y_{e,m} && \forall m, (n, e) \in LK \\
&&& L_m = \sum_{n,e:(n,e) \in LK} (X_{n,e,m}) && \forall m \\
&&& D_m = \sum_n (\text{delta}_n Y_{n,m}) && \forall m \\
&&& E_m \leq E_{m-1} \quad \forall m > 1 \\
&&& \sum_{n,e:(n,e) \in LK} (X_{n,e,m}) \geq a * E_m && \forall m \\
&&& \sum_{n,e:(n,e) \in LK} (X_{n,e,m}) \leq b * E_m && \forall m \\
&&& L_m - L_k \leq \text{epsilon} + b(1 - E_k) && \forall m, k > m \\
&&& L_k - L_m \leq \text{epsilon} + b(1 - E_k) && \forall m, k > m \\
&&& L, D \in R_+, E, X, Y \in \{0, 1\}
\end{aligned}$$

626 The parser has to identify on-the-fly an array of parameters as shown below:

```

627 -Reading the variable's types and symbols...
628
629 The optimization problem has :
630 (3) binary set(s) of variable(s)
631 (0) set(s) of integer variable(s) and
632 (2) set(s) of positive real variable(s)
633 (0) set(s) of non-negative real variable(s)
634 -Reading the objective function definition...
635 The optimization sense is: MAXIMIZE
636
637 Quadratic Programming Objective function detected!!!
638
639 ***Parameter 'Lambda' detected in the constraint***
640 **Variable 'L' detected in the objective function with index set: {m
641 }**
642 **Variable 'D' detected in the objective function with index set: {m
643 }**
644 3 out of total 5 sets of variables were not detected in the objective
645 function...
646 Searching for the rest of the variables in the model...
647 Found set(s) of variable(s) Y with index set {n,m}
648 Found set(s) of variable(s) X with index set {n,e,m}
649 Found set(s) of variable(s) E with index set {m}
650 Found the remaining 3 variables definitions in the constraints!

```



```

651 -Reading the constraints set (1)...
652 -Reading the constraints set (2)...
653 -Reading the constraints set (3)...
654 -Reading the constraints set (4)...
655 -Reading the constraints set (5)...
656     ***Parameter 'delta' detected in the constraint***
657 -Reading the constraints set (6)...
658 -Reading the constraints set (7)...
659     ***Parameter 'a' detected in the constraint***
660 -Reading the constraints set (8)...
661     ***Parameter 'b' detected in the constraint***
662 -Reading the constraints set (9)...
663     ***Parameter 'epsilon' detected in the constraint***
664 -Reading the constraints set (10)...
665 Parsing the model terminated successfully... Now printing the Pyomo equivalent
666 model ...

```

Model name / Step	Conversion from L ^A T _E X	Pyomo model generation	Solver (CPLEX)
LP Transportation problem	0.048	0.034	0.00
Traveling Salesman Problem (TSP)	0.049	0.026	0.02
Cancel Therapy	0.060	4.520	4.00
Optimal Process Plant Layout	0.109	0.151	1.14
Community Detection via Modularity	0.054	0.192	0.86

Table 1: Cpu-time (in seconds) for the modeling steps. The computational environment used was: Intel Core i5 8250U @1.6Ghz , 8GB RAM 2.4 Ghz, Windows 10 x64, Python 3.6

668 References

- 669 Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J., 2007. The Traveling Salesman Problem: A Com-
670 putational Study (Princeton Series in Applied Mathematics). Princeton University Press, Princeton, NJ,
671 USA.
- 672 Johnson, D.S., 1990. Local optimization and the Traveling Salesman Problem. Springer Berlin Heidelberg,
673 Berlin, Heidelberg. pp. 446–461.
- 674 Knijnenburg, T.A., Klau, G.W., Iorio, F., Garnett, M.J., McDermott, U., Shmulevich, I., Wessels, L.F.A.,
675 2016. Logic models to predict continuous outputs based on binary inputs with an application to per-
676 sonalized cancer therapy. Scientific Reports 6, 36812. URL: <http://dx.doi.org/10.1038/srep36812>,
677 doi:10.1038/srep36812.
- 678 Papageorgiou, L.G., Rotstein, G.E., 1998. Continuous-domain mathematical models for optimal process
679 plant layout. Industrial & Engineering Chemistry Research 37, 3631–3639. URL: <http://dx.doi.org/10.1021/ie980146v>, doi:10.1021/ie980146v, arXiv:<http://dx.doi.org/10.1021/ie980146v>.
- 680 Schrijver, A., 2002. On the history of the transportation and maximum flow problems. Mathematical
681 Programming 91, 437–445.
- 682 Xu, G., Tsoka, S., Papageorgiou, L.G., 2007. Finding community structures in complex networks using
683 mixed integer optimisation. The European Physical Journal B 60, 231–239. URL: <https://doi.org/10.1140/epjb/e2007-00331-0>, doi:10.1140/epjb/e2007-00331-0.