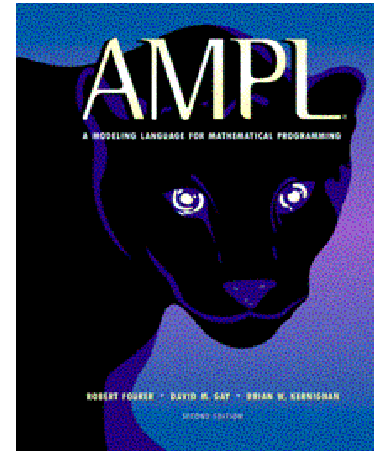
Attacking Hard Mixed-Integer Optimization Problems through the AMPL Modeling Language



Robert Fourer

AMPL Optimization LLC

www.ampl.com — 773-336-AMPL

Industrial Engineering & Management Sciences
Northwestern University

INFORMS Conference on O.R. Practice

Orlando — April 18-20, 2010 — Track T9

AMPL: Work Scheduling Example

Cover demands for workers

- Each "shift" requires a certain number of employees
- Each employee works a certain "schedule" of shifts
- *Each schedule that is worked by anyone must be worked by a fixed minimum number*

Minimize total workers needed

- Which schedules are used?
- How many work each of schedule?

AMPL

Algebraic modeling language: symbolic data

```
set SHIFTS;                  # shifts

param Nsched;                # number of schedules;
set SCHEDS = 1..Nsched;      # set of schedules

set SHIFT_LIST {SCHEDS} within SHIFTS;

param rate {SCHEDS} >= 0;         # pay rates
param required {SHIFTS} >= 0;     # staffing requirements

param least_assign >= 0;          # min workers on any schedule used
```

AMPL

Algebraic modeling language: symbolic model

```
var Work {SCHEDS} >= 0 integer;
var Use  {SCHEDS} >= 0 binary;

minimize Total_Cost:
   sum {j in SCHEDS} rate[j] * Work[j];

subject to Shift_Needs {i in SHIFTS}:
   sum {j in SCHEDS: i in SHIFT_LIST[j]} Work[j] >= required[i];

subject to Least_Use1 {j in SCHEDS}:
   least_assign * Use[j] <= Work[j];

subject to Least_Use2 {j in SCHEDS}:
   Work[j] <= (max {i in SHIFT_LIST[j]} required[i]) * Use[j];
```

AMPL

Explicit data independent of symbolic model

```
set SHIFTS := Mon1 Tue1 Wed1 Thu1 Fri1 Sat1
              Mon2 Tue2 Wed2 Thu2 Fri2 Sat2
              Mon3 Tue3 Wed3 Thu3 Fri3 ;

param Nsched := 126 ;

set SHIFT_LIST[1] := Mon1 Tue1 Wed1 Thu1 Fri1 ;
set SHIFT_LIST[2] := Mon1 Tue1 Wed1 Thu1 Fri2 ;
set SHIFT_LIST[3] := Mon1 Tue1 Wed1 Thu1 Fri3 ;
set SHIFT_LIST[4] := Mon1 Tue1 Wed1 Thu1 Sat1 ;
set SHIFT_LIST[5] := Mon1 Tue1 Wed1 Thu1 Sat2 ;  .......

param required :=  Mon1 100  Mon2 78  Mon3 52
                   Tue1 100  Tue2 78  Tue3 52
                   Wed1 100  Wed2 78  Wed3 52
                   Thu1 100  Thu2 78  Thu3 52
                   Fri1 100  Fri2 78  Fri3 52
                   Sat1 100  Sat2 78 ;
```

AMPL

Solver independent of model & data

```
ampl: model sched1.mod;
ampl: data sched.dat;

ampl: let least_assign := 7;

ampl: option solver cplex;
ampl: solve;

CPLEX 12.1.0: optimal integer solution; objective 266
473 MIP simplex iterations
72 branch-and-bound nodes

ampl: option omit_zero_rows 1, display_1col 0;
ampl: display Work;

Work [*] :=
  2 12    16 14    29  7    53  7    91 17   112  9   122 29
  3  7    18  7    37 21    78 21   100 19   116 20   124  7
  6 10    20  7    41  8    82 21   109  7   118 16
;
```

AMPL

Language independent of solver

```
ampl: option solver gurobi;
ampl: solve;

Gurobi 3.0.0: optimal solution; objective 266
396 simplex iterations
3 branch-and-cut nodes

ampl: display Work;

Work [*] :=
  1 29     37 35     84 18     91 17    101 11    112 18    118 17
 21 36     71  7     89 18     95  7    109 10    116  7    124 36
;
```

Topics

1: Look at the details

2: Know when to quit

3: Multiprocess

4: Tune

5: Reformulate

6: “Cheat” on the method

7: “Cheat” on the data

1: Look at the Details

Log lines

Preprocessing

Postprocessing

Log Lines

CPLEX

```
ampl: option cplex_options 'mipdisplay 2 mipinterval 100';
ampl: solve;

CPLEX 11.2.0: mipdisplay 2
mipinterval 100

        Nodes                                        Cuts/
   Node  Left    Objective  IInf  Best Integer    Best Node  ItCnt     Gap

      0     0     265.6000    18                   265.6000     38
      0     0     265.6000    22                   Cuts: 12     46
      0     0     265.6000    21                 MIRcuts: 1     54
*     0+    0                      10348.0000      265.6000     54   97.43%
      0     2     265.6000    18   10348.0000      265.6000     54   97.43%
    100   102     267.0000     6   10348.0000      265.6000    537   97.43%
*   119   118     integral     0     273.0000      265.6000    882    2.71%
    200   199     265.7500    19     273.0000      265.6000   1633    2.71%
*   244   215     integral     0     268.0000      265.6000   2744    0.90%
    300   271     266.5000    14     268.0000      265.6000   3093    0.90%
*   344+    1                        266.0000      265.6000   3323    0.15%
```

Log Lines *(cont'd)*

Work

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
	0	0	265.6000	18		265.6000	38	
	0	0	265.6000	22		Cuts: 12	46	
	0	0	265.6000	21		MIRcuts: 1	54	
*	0+	0			10348.0000	265.6000	54	97.43%
	0	2	265.6000	18	10348.0000	265.6000	54	97.43%
	100	102	267.0000	6	10348.0000	265.6000	537	97.43%
*	119	118	integral	0	273.0000	265.6000	882	2.71%
	200	199	265.7500	19	273.0000	265.6000	1633	2.71%
*	244	215	integral	0	268.0000	265.6000	2744	0.90%
	300	271	266.5000	14	268.0000	265.6000	3093	0.90%
*	344+	1			266.0000	265.6000	3323	0.15%

Log Lines *(cont'd)*

Lower bounds

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
	0	0	265.6000	18		265.6000	38	
	0	0	265.6000	22		Cuts: 12	46	
	0	0	265.6000	21		MIRcuts: 1	54	
*	0+	0			10348.0000	265.6000	54	97.43%
	0	2	265.6000	18	10348.0000	265.6000	54	97.43%
	100	102	267.0000	6	10348.0000	265.6000	537	97.43%
*	119	118	integral	0	273.0000	265.6000	882	2.71%
	200	199	265.7500	19	273.0000	265.6000	1633	2.71%
*	244	215	integral	0	268.0000	265.6000	2744	0.90%
	300	271	266.5000	14	268.0000	265.6000	3093	0.90%
*	344+	1			266.0000	265.6000	3323	0.15%

Log Lines *(cont'd)*

Upper bounds

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
	0	0	265.6000	18		265.6000	38	
	0	0	265.6000	22		Cuts: 12	46	
	0	0	265.6000	21		MIRcuts: 1	54	
*	0+	0			10348.0000	265.6000	54	97.43%
	0	2	265.6000	18	10348.0000	265.6000	54	97.43%
	100	102	267.0000	6	10348.0000	265.6000	537	97.43%
*	119	118	integral	0	273.0000	265.6000	882	2.71%
	200	199	265.7500	19	273.0000	265.6000	1633	2.71%
*	244	215	integral	0	268.0000	265.6000	2744	0.90%
	300	271	266.5000	14	268.0000	265.6000	3093	0.90%
*	344+	1			266.0000	265.6000	3323	0.15%

Log Lines *(cont'd)*

Gap

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
	0	0	265.6000	18		265.6000	38	
	0	0	265.6000	22		Cuts: 12	46	
	0	0	265.6000	21		MIRcuts: 1	54	
*	0+	0			10348.0000	265.6000	54	97.43%
	0	2	265.6000	18	10348.0000	265.6000	54	97.43%
	100	102	267.0000	6	10348.0000	265.6000	537	97.43%
*	119	118	integral	0	273.0000	265.6000	882	2.71%
	200	199	265.7500	19	273.0000	265.6000	1633	2.71%
*	244	215	integral	0	268.0000	265.6000	2744	0.90%
	300	271	266.5000	14	268.0000	265.6000	3093	0.90%
*	344+	1			266.0000	265.6000	3323	0.15%

Preprocessing

Gurobi

```
ampl: option gurobi_options 'outlev 1 logfreq 1 timing 1';
ampl: option presolve 10;
ampl: option show_stats 1;

ampl: solve;

Presolve eliminates 159622 constraints and 755655 variables.
Adjusted problem:
385720 variables:
       384720 binary variables
       1000 linear variables
317322 constraints, all linear; 13961712 nonzeros
1 linear objective; 308776 nonzeros.

Gurobi 3.0.0:
Optimize a model with 317322 Rows, 385720 Columns and 13961712 NonZeros

Presolve removed 8335 rows and 41141 columns (presolve time = 2s) ...
Presolve removed 11605 rows and 46958 columns (presolve time = 12s) ...
Presolve removed 11605 rows and 52750 columns (presolve time = 12s) ...
Presolve removed 11605 rows and 56770 columns (presolve time = 13s) ...
Presolve removed 12944 rows and 58110 columns (presolve time = 16s) ...
```

Preprocessing *(cont'd)*

Gurobi

```
Presolve removed 218544 rows and 58411 columns (presolve time = 16s) ...
Presolve removed 218544 rows and 58411 columns (presolve time = 18s) ...
Presolve removed 218544 rows and 58411 columns (presolve time = 22s) ...
Presolve removed 218544 rows and 58411 columns (presolve time = 27s) ...
Presolve removed 219031 rows and 58411 columns (presolve time = 31s) ...
Presolve removed 222957 rows and 189567 columns (presolve time = 31s) ...
Presolve removed 264766 rows and 292409 columns (presolve time = 32s) ...
Presolve removed 270648 rows and 318250 columns (presolve time = 33s) ...
Presolve removed 285755 rows and 318284 columns (presolve time = 35s) ...
Presolve removed 285781 rows and 318284 columns (presolve time = 36s) ...
Presolve removed 289794 rows and 342963 columns (presolve time = 37s) ...
Presolve removed 297321 rows and 346355 columns (presolve time = 38s) ...
Presolve removed 300053 rows and 355682 columns (presolve time = 39s) ...
Presolve removed 300815 rows and 357734 columns (presolve time = 40s) ...
Presolve removed 301076 rows and 358221 columns (presolve time = 41s) ...
Presolve removed 301276 rows and 358621 columns

Presolve time: 42.87s

Presolved: 16046 Rows, 27099 Columns, 281126 Nonzeros
```

Preprocessing *(cont'd)*

Gurobi

```
Objective GCD is 0.001

Found heuristic solution: objective 136.2260000
Found heuristic solution: objective 29.8440000

Root relaxation: objective 1.592000e+00, 7204 iterations, 0.12 seconds


    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0                  0     1.5920000  1.59200  0.0%     -   43s

Explored 0 nodes (7204 simplex iterations) in 43.31 seconds
Thread count was 8 (of 8 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 1.5920000000e+00, best bound 1.5920000000e+00, gap 0.0%
```

Postprocessing

Gurobi

```
Explored 10161 nodes (609669 simplex iterations) in 146.98 seconds

Thread count was 8 (of 8 available processors)

Node limit reached
Best objective 1.200000000e+01, best bound 8.00000000e+00, gap 33.3333%

Optimize a model with 4112 Rows, 2439 Columns and 15791 NonZeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   2.350000e+02   0.000000e+00      0s
      35    1.2000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 35 iterations and 0.00 seconds
Optimal objective  1.200000000e+01

Gurobi 3.0.0: node limit; objective 12
609669 simplex iterations
10161 branch-and-cut nodes
plus 35 simplex iterations for intbasis
```

2: Know When to Quit

Difficult situations

- Work scheduling without log lines
- Work scheduling with log lines
- Balanced assignment: a complete failure

Can we stop early?

- In these cases: good bet
- In general: not so clear

Work Scheduling Without Log Lines

A day of CPLEX output

```
ampl: model sched1.mod;
ampl: data sched.dat;

ampl: let least_assign := 19;

ampl: option solver cplex;
ampl: option cplex_options 'branch 1';
ampl: solve;

252 variables:
        126 binary variables
        126 integer variables
269 constraints, all linear; 1134 nonzeros
1 linear objective; 126 nonzeros.

CPLEX 12.1.0: branch 1
```

Work Scheduling With Log Lines

CPLEX starts promisingly . . .

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
*	0+	0			12112.0000		40	---
	0	0	265.6000	25	12112.0000	265.6000	40	97.81%
	0	0	265.6000	29	12112.0000	Cuts: 12	79	97.81%
	0	0	265.6000	24	12112.0000	MIRcuts: 4	87	97.81%
	0	0	265.6000	24	12112.0000	MIRcuts: 3	123	97.81%
	0	2	265.6000	24	12112.0000	265.6000	123	97.81%
*	12+	12			616.0000	265.6000	198	56.88%
*	12+	12			277.0000	265.6000	198	4.12%
*	560+	83			276.0000	265.6000	4897	3.77%
*	566+	54			269.0000	265.6000	5117	1.26%
	10000	2132	268.0000	9	269.0000	265.6000	157659	1.26%
	20000	4057	cutoff		269.0000	265.6000	327850	1.26%
	30000	5926	266.3333	12	269.0000	265.6000	507164	1.26%
	40000	7647	cutoff		269.0000	265.6000	690189	1.26%
	50000	9517	268.0000	23	269.0000	265.6000	884164	1.26%
	60000	11228	265.7500	25	269.0000	265.6000	1088023	1.26%
	70000	13361	268.0000	17	269.0000	265.6000	1276209	1.26%
	80000	15594	268.0000	17	269.0000	265.6000	1462445	1.26%

Work Scheduling With Log Lines *(cont'd)*

. . . tightens the bound . . .

Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
1160000	196770	268.0000	15	269.0000	266.2500	24998690	1.02%
1170000	197681	268.0000	6	269.0000	266.3333	25285264	0.99%
1180000	198093	268.0000	6	269.0000	266.5000	25567868	0.93%
1190000	198138	266.6667	24	269.0000	266.6667	25804095	0.87%
1200000	198604	267.2857	17	269.0000	266.6667	26063901	0.87%
Elapsed real time = 263.20 sec. (tree size = 66.83 MB)							
1210000	198978	cutoff		269.0000	267.0000	26306006	0.74%
1220000	198908	267.0000	9	269.0000	267.0000	26673281	0.74%
1230000	199278	cutoff		269.0000	267.0000	27003874	0.74%
1240000	199565	268.0000	10	269.0000	267.2500	27261518	0.65%
1250000	200080	268.0000	8	269.0000	267.3333	27521492	0.62%
1260000	200971	cutoff		269.0000	267.3333	27789325	0.62%
1270000	201601	cutoff		269.0000	267.5000	28036060	0.56%
1280000	202092	cutoff		269.0000	267.7500	28247750	0.46%
1290000	202643	268.0000	4	269.0000	268.0000	28449503	0.37%
1300000	203307	cutoff		269.0000	268.0000	28648240	0.37%
Elapsed real time = 286.48 sec. (tree size = 68.31 MB)							
1310000	203837	cutoff		269.0000	268.0000	28850298	0.37%
1320000	203880	cutoff		269.0000	268.0000	29058072	0.37%

Work Scheduling With Log Lines *(cont'd)*

. . . eventually stops expanding the search tree . . .

```
               Nodes                                 Best        Cuts/
     Node       Left     Objective  IInf          Integer    Best Node        ItCnt       Gap

 387790000   550587        cutoff                269.0000     268.0000  1.40870e+10   0.37%
 387800000   550593      268.0000     10         269.0000     268.0000  1.40874e+10   0.37%
Elapsed real time = 95728.25 sec. (tree size = 155.89 MB)
Nodefile size = 28.64 MB (19.83 MB after compression)
 387810000   550586        cutoff                269.0000     268.0000  1.40879e+10   0.37%
 387820000   550532        cutoff                269.0000     268.0000  1.40884e+10   0.37%
 387830000   550535        cutoff                269.0000     268.0000  1.40888e+10   0.37%
 387840000   550540        cutoff                269.0000     268.0000  1.40893e+10   0.37%
 387850000   550544      268.0000     18         269.0000     268.0000  1.40898e+10   0.37%
 387860000   550557      268.0000     13         269.0000     268.0000  1.40902e+10   0.37%
 387870000   550578      268.0000      7         269.0000     268.0000  1.40907e+10   0.37%
 387880000   550611      268.0000     17         269.0000     268.0000  1.40911e+10   0.37%
 387890000   550564      268.0000      6         269.0000     268.0000  1.40916e+10   0.37%
 387900000   550529      268.0000      5         269.0000     268.0000  1.40920e+10   0.37%
Elapsed real time = 95766.47 sec. (tree size = 155.87 MB)
Nodefile size = 28.64 MB (19.83 MB after compression)
 387910000   550448      268.0000     10         269.0000     268.0000  1.40923e+10   0.37%
 387920000   550458        cutoff                269.0000     268.0000  1.40928e+10   0.37%
 387930000   550436        cutoff                269.0000     268.0000  1.40932e+10   0.37%
 387940000   550390      268.0000      7         269.0000     268.0000  1.40936e+10   0.37%
 387950000   550334      268.0000      7         269.0000     268.0000  1.40940e+10   0.37%
```

Work Scheduling With Log Lines *(cont'd)*

. . . then takes "forever" to prove optimality

```
          Nodes                                 Best       Cuts/
   Node    Left   Objective  IInf            Integer   Best Node       ItCnt       Gap

 465550000   301    268.0000     6         269.0000    268.0000  1.72059e+10    0.37%
 465560000   249    268.0000    12         269.0000    268.0000  1.72063e+10    0.37%
 465570000   228      cutoff               269.0000    268.0000  1.72068e+10    0.37%
 465580000   100      cutoff               269.0000    268.0000  1.72072e+10    0.37%
 465590000    80    268.0000     6         269.0000    268.0000  1.72076e+10    0.37%

Flow cuts applied:  1
Gomory fractional cuts applied:  1

Root node processing (before b&c):
  Real time             =       0.03

Parallel b&c, 8 threads:
  Real time             =  112012.60
  Sync time (average)   =   23098.41
  Wait time (average)   =   64021.17
                          ---------
Total (root+branch&cut) = 112012.63 sec.

CPLEX 12.1.0: optimal integer solution; objective 269
-2147483648 MIP simplex iterations
465596558 branch-and-bound nodes
```

Work Scheduling With Log Lines

Gurobi starts promisingly . . .

```
Found heuristic solution: objective 408.0000000

Root relaxation: objective 2.656000e+02, 97 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  265.60000    0   15   408.00000  265.60000  34.9%     -    0s
H    0     0                        290.00000  265.60000  8.41%     -    0s
     0     0  266.00000    0   24   290.00000  266.00000  8.28%     -    0s
H    0     0                        275.00000  266.00000  3.27%     -    0s
     0     0  266.00000    0   21   275.00000  266.00000  3.27%     -    0s
     0     0  266.00000    0   21   275.00000  266.00000  3.27%     -    0s
     0     0  266.00000    0   12   275.00000  266.00000  3.27%     -    0s
     0     2  266.00000    0    7   275.00000  266.00000  3.27%     -    0s
H   42    50                        269.00000  266.00000  1.12%   8.6    0s
 26094 12393  268.00000   65   19   269.00000  266.00000  1.12%   6.7    3s
 61393 28900     cutoff   51        269.00000  266.00000  1.12%   6.6    6s
103339 48360  268.00000   43    7   269.00000  266.00000  1.12%   6.6    9s
139466 64591  268.00000   57    7   269.00000  266.00000  1.12%   6.6   12s
176702 80630     cutoff   71        269.00000  266.00000  1.12%   6.6   15s
217155 97936  268.00000   29   17   269.00000  266.00000  1.12%   6.6   18s
```

Work Scheduling With Log Lines *(cont'd)*

. . . tightens the bound . . .

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
508581	228275	267.20000	66	26	269.00000	266.00000	1.12%	6.4	39s
555160	247970	268.00000	60	13	269.00000	266.00000	1.12%	6.4	42s
593775	263820	268.00000	80	15	269.00000	266.00000	1.12%	6.4	45s
637257	280342	268.00000	65	7	269.00000	266.00000	1.12%	6.4	48s
679843	295729	268.00000	55	14	269.00000	266.00000	1.12%	6.4	51s
724464	315739	268.00000	72	6	269.00000	266.00000	1.12%	6.4	54s
764318	331586	cutoff	68		269.00000	266.00000	1.12%	6.4	57s
807548	348225	268.00000	43	22	269.00000	266.00000	1.12%	6.3	60s
850968	362477	cutoff	56		269.00000	266.00000	1.12%	6.3	63s
908917	374846	268.00000	53	18	269.00000	266.75000	0.84%	6.2	66s
981286	392065	268.00000	57	9	269.00000	268.00000	0.37%	6.1	69s
1052711	400737	cutoff	58		269.00000	268.00000	0.37%	6.2	72s
1121014	403413	cutoff	56		269.00000	268.00000	0.37%	6.2	75s
1194809	403685	268.00000	57	17	269.00000	268.00000	0.37%	6.3	78s
1271977	405824	268.00000	85	7	269.00000	268.00000	0.37%	6.3	81s
1321885	408302	cutoff	88		269.00000	268.00000	0.37%	6.3	84s
1392557	411299	268.00000	62	7	269.00000	268.00000	0.37%	6.3	87s
1462331	410872	cutoff	58		269.00000	268.00000	0.37%	6.4	90s
1525446	412952	268.00000	64	7	269.00000	268.00000	0.37%	6.4	93s

Work Scheduling With Log Lines *(cont'd)*

. . . eventually stops expanding the search tree . . .

Nodes Expl	Nodes Unexpl	Current Node Obj	Depth	IntInf	Objective Bounds Incumbent	BestBd	Gap	Work It/Node	Time
468100473	4333549	268.00000	63	6	269.00000	268.00000	0.37%	7.6	23514s
468158488	4333932	cutoff	55		269.00000	268.00000	0.37%	7.6	23517s
468217704	4334267	cutoff	51		269.00000	268.00000	0.37%	7.6	23520s
468270818	4334676	268.00000	43	7	269.00000	268.00000	0.37%	7.6	23523s
468331266	4334927	268.00000	47	11	269.00000	268.00000	0.37%	7.6	23526s
468392344	4335461	cutoff	46		269.00000	268.00000	0.37%	7.6	23529s
468451907	4335661	cutoff	57		269.00000	268.00000	0.37%	7.6	23532s
468512663	4335747	cutoff	71		269.00000	268.00000	0.37%	7.6	23535s
468573142	4336803	cutoff	52		269.00000	268.00000	0.37%	7.6	23538s
468634351	4337183	268.00000	68	11	269.00000	268.00000	0.37%	7.6	23541s
468691039	4337566	268.00000	70	5	269.00000	268.00000	0.37%	7.6	23544s
468748985	4336541	268.00000	62	16	269.00000	268.00000	0.37%	7.6	23547s
468805241	4335743	268.00000	65	18	269.00000	268.00000	0.37%	7.6	23550s
468866302	4335535	268.00000	66	9	269.00000	268.00000	0.37%	7.6	23553s
468926151	4334696	cutoff	65		269.00000	268.00000	0.37%	7.6	23556s
468985994	4334355	268.00000	63	14	269.00000	268.00000	0.37%	7.6	23559s
469044544	4333792	268.00000	52	9	269.00000	268.00000	0.37%	7.6	23562s
469097106	4333524	268.00000	72	8	269.00000	268.00000	0.37%	7.6	23565s
469156297	4332696	cutoff	83		269.00000	268.00000	0.37%	7.6	23568s

Work Scheduling With Log Lines *(cont'd)*

. . . *then takes "forever" to prove optimality*

```
        Nodes      |    Current Node       |     Objective Bounds       |    Work
    Expl Unexpl |  Obj    Depth  IntInf | Incumbent     BestBd    Gap | It/Node Time

1330292347  2062  268.00000   45    14    269.00000  268.00000  0.37%   7.6 69930s
1330348816  1624  268.00000   79    22    269.00000  268.00000  0.37%   7.6 69933s
1330391972   920  268.00000   76     5    269.00000  268.00000  0.37%   7.6 69936s
1330448520   803     cutoff   69          269.00000  268.00000  0.37%   7.6 69939s
1330505973   333  268.00000   49    11    269.00000  268.00000  0.37%   7.6 69942s

Cutting planes:
  Gomory: 1
  Implied bound: 2

Thread count was 8 (of 8 available processors)

Times (seconds):
Input =  0.001
Solve =  69944.7 (summed over threads)
Output = 0.012
Elapsed = 69944

Gurobi 3.0.0: optimal solution; objective 269
10114432447 simplex iterations
1330555419 branch-and-cut nodes
```

Balanced Assignment: Complete Failure

Starts well . . .

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
	0	0	0.0000	61		0.0000	99	
*	0+	0			232.0000	0.0000	99	100.00%
	0	0	0.0000	60	232.0000	Cuts: 55	174	100.00%
	0	0	0.0000	66	232.0000	Flowcuts: 17	250	100.00%
	0	0	0.0000	58	232.0000	Flowcuts: 9	300	100.00%
	0	0	0.0000	57	232.0000	Flowcuts: 13	326	100.00%
*	0+	0			230.0000	0.0000	326	100.00%
*	0+	0			216.0000	0.0000	326	100.00%
	0	2	0.0000	57	216.0000	0.0000	326	100.00%
*	440+	403			214.0000	0.0000	7938	100.00%
*	552+	339			212.0000	0.0000	10797	100.00%
	1000	556	69.9315	50	212.0000	0.0000	16491	100.00%
	2000	1332	42.8547	47	212.0000	0.0000	25669	100.00%
	3000	2276	81.6541	49	212.0000	5.0928	37332	97.60%
	4000	3214	77.9166	49	212.0000	5.1140	47933	97.59%
	5000	4160	71.0567	52	212.0000	6.4918	57582	96.94%
	6000	5089	97.3040	47	212.0000	7.8042	66662	96.32%
	7000	6021	158.4869	37	212.0000	9.3981	75348	95.57%
	8000	6942	157.5392	36	212.0000	11.2257	84237	94.70%

....

Balanced Assignment: Failure *(cont'd)*

. . . bogs down without much further progress

```
          Nodes                                           Cuts/
   Node    Left   Objective  IInf  Best Integer    Best Node    ItCnt     Gap
 ..................
 6244000 5769420    91.8882    46      212.0000      55.4261  37227229   73.86%
 6245000 5770348   123.4752    34      212.0000      55.4272  37233744   73.86%
 6246000 5771270    63.5603    48      212.0000      55.4289  37239584   73.85%
 6247000 5772192   106.5663    43      212.0000      55.4294  37245120   73.85%
 6248000 5773112    64.0217    47      212.0000      55.4308  37251128   73.85%
 6249000 5774034   181.2576    31      212.0000      55.4310  37257940   73.85%
 6250000 5774954   119.4546    35      212.0000      55.4320  37263877   73.85%
Elapsed time = 9116.25 sec. (tree size = 1616.65 MB)
Nodefile size = 1488.81 MB (685.88 MB after compression)
 6251000 5775885   182.0327    29      212.0000      55.4328  37270210   73.85%
 6252000 5776807   140.1960    39      212.0000      55.4330  37275647   73.85%
 6253000 5777720    91.9423    43      212.0000      55.4346  37281516   73.85%
 6254000 5778648   127.8185    35      212.0000      55.4355  37286884   73.85%
8 flow-cover cuts
2 Gomory cuts
1 zero-half cut
9 mixed-integer rounding cuts
CPLEX 11.2.0: ran out of memory.
```

Stopping Early

1st example

- optimum is either 269 or 268
- solution with 269 is known
- isn't that good enough?

2nd example

- gap is still huge, but . . .
- solution with 212 seems likely to be optimal

. . . we'll return to these

The trouble with this reasoning

- not robust over a range of similar problems
- could be wrong when gap is large

3: Multiprocess

Predictable effects

- Process branch-and-bound nodes faster

Unpredictable effects

- Build a different tree

Work scheduling examples . . .

least_assign = 21 *(Gurobi 3.0 beta)*

1 thread

- 578661 nodes   213.322 seconds   0.369 sec / 1000 nodes

8 threads

- 132441 nodes   10.805 seconds   0.082 sec / 1000 nodes

Speedup

- ***4.4*** on nodes
- ***4.5*** on time/node

least_assign = 20 *(Gurobi 3.0 beta)*

1 thread

- 276903 nodes 93.374 seconds 0.337 sec / 1000 nodes

8 threads

- 186677 nodes 12.992 seconds 0.070 sec / 1000 nodes

Speedup

- ***1.5*** on nodes
- ***4.8*** on time/node

least_assign = 20 *(CPLEX 12.1.0)*

1 thread

- 99342 nodes 54.766 seconds 0.551 sec / 1000 nodes

8 threads "deterministic"

- 163140 nodes 16.847 seconds 0.103 sec / 1000 nodes

8 threads "opportunistic"

- 62130 nodes 4.918 seconds 0.079 sec / 1000 nodes
- 165574 nodes 13.706 seconds 0.083 sec / 1000 nodes
- 153602 nodes 12.870 seconds 0.084 sec / 1000 nodes
- 310312 nodes 19.184 seconds 0.062 sec / 1000 nodes
- 599794 nodes 51.592 seconds 0.086 sec / 1000 nodes
- 154022 nodes 10.760 seconds 0.070 sec / 1000 nodes
- 75747 nodes 5.597 seconds 0.074 sec / 1000 nodes
- 383213 nodes 42.240 seconds 0.110 sec / 1000 nodes

4: Tune

Default settings

Tuning run

Improved settings

Is tuning worthwhile?

Default Settings

CPLEX 11.1 run

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
	0	0	265.6000	22		265.6000	38	
	0	0	265.6000	22		Cuts: 13	72	
	0	0	265.6000	27		MIRcuts: 1	91	
*	0+	0			10348.0000	265.6000	91	97.43%
	0	2	265.6000	25	10348.0000	265.6000	91	97.43%
*	138	128	integral	0	289.0000	265.6000	3272	8.10%
*	140+	122			285.0000	265.6000	3292	6.81%
*	236	175	integral	0	283.0000	265.6000	4737	6.15%
*	318	168	integral	0	280.0000	265.6000	6175	5.14%
*	418+	140			272.0000	265.6000	8205	2.35%
*	418+	99			269.0000	265.6000	8205	1.26%
*	529+	120			267.0000	265.6000	9524	0.52%
*	85629+	1			266.0000	265.6000	1408458	0.15%

159.2 seconds

Tuning Run

CPLEX tuning option

```
ampl: option cplex_options 'tunefile t.out tunedisplay 2 tunetime 240';
ampl: solve;

CPLEX 11.1.0: tunefile t.out
tunedisplay 2
tunetime 240

Tuning on problem 'c252v269i126o126'

Test 'defaults':
   Integer optimal solution.
   Time =  155.31 sec.  Objective = 266   Best bound = 266
Tuning progress: 13%

Test 'short1':
   Time limit exceeded.
   Time =   15.55 sec.  Objective = 267   Best bound = 265.6
Tuning progress: 14%
```

Tuning Run *(cont'd)*

CPLEX tuning option (cont'd)

```
Test 'short_test2':
CPX_PARAM_CUTPASS  1
   Time limit exceeded.
   Time =   15.53 sec.  Objective = 267   Best bound = 265.6
Tuning progress: 16%

Test 'short_test3':
CPX_PARAM_FRACCUTS  2
   Time limit exceeded.
   Time =   15.55 sec.  Objective = 267   Best bound = 265.6
Tuning progress: 17%

Test 'short_test4':
CPX_PARAM_FRACPASS  10
CPX_PARAM_FRACCAND  10000
   Time limit exceeded.
   Time =   15.55 sec.  Objective = 267   Best bound = 265.6
Tuning progress: 18%

.....
```

Tuning Run *(cont'd)*

CPLEX tuning option (cont'd)

```
Test 'short_test8':
CPX_PARAM_HEURFREQ  100
CPX_PARAM_CUTSFACTOR  100
CPX_PARAM_BTTOL  -1070056790
   Integer optimal solution.
   Time =    9.11 sec.  Objective = 266   Best bound = 266
Tuning progress: 49%

.....

Test 'short_test10':
CPX_PARAM_RINSHEUR  20
CPX_PARAM_HEURFREQ  3
CPX_PARAM_CUTSFACTOR  20
   Integer optimal solution.
   Time =    8.84 sec.  Objective = 266   Best bound = 266
Tuning progress: 53%
.....
```

Tuning Run *(cont'd)*

CPLEX tuning option (cont'd)

```
Test 'short_test8':
CPX_PARAM_HEURFREQ  100
CPX_PARAM_CUTSFACTOR  100
CPX_PARAM_BTTOL  -1070056790
   Integer optimal solution.
   Time =    9.11 sec.  Objective = 266   Best bound = 266
Tuning progress: 49%

.....

Test 'short_test24':
CPX_PARAM_CUTPASS  1
CPX_PARAM_HEURFREQ  -1
CPX_PARAM_PROBE  -1
CPX_PARAM_VARSEL  4
   Integer optimal solution.
   Time =    6.16 sec.  Objective = 266   Best bound = 266
Tuning progress: 95%

.....
```

Tuning Run *(cont'd)*

CPLEX tuning option (conclusion)

```
Tuning finished.
4 settings written to tunefile file "t.out“

ampl: option cplex_options 'paramfile t.out mipdisplay 2 mipinterval 1000';

ampl: solve;

CPLEX 11.1.0: cutpass = 1
heurfreq = -1
probe = -1
varsel = 4
```

Improved Settings

CPLEX 11.1 run

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
	0	0	265.6000	22		265.6000	38	
	0	0	265.6000	22		Cuts: 13	72	
	0	2	265.6000	22		265.6000	72	
*	101	100	integral	0	322.0000	265.6000	1074	17.52%
*	162	138	integral	0	295.0000	265.6000	1621	9.97%
*	164	137	integral	0	291.0000	265.6000	1641	8.73%
*	229	187	integral	0	284.0000	265.6000	2540	6.48%
*	286	239	integral	0	283.0000	265.6000	3129	6.15%
*	352	292	integral	0	281.0000	265.6000	4397	5.48%
*	715	393	integral	0	272.0000	265.6000	10512	2.35%
*	1584	645	integral	0	269.0000	265.6000	26976	1.26%
*	3429	1545	integral	0	268.0000	265.6000	60651	0.90%
*	9877	0	integral	0	266.0000	265.6000	165097	0.15%

6.45 seconds

Improved Settings *(cont'd)*

Results using mixed-integer formulation

- With defaults: 159.2 sec, 85629 nodes, 1408458 iters
- After tuning: 6.45 sec, 9878 nodes, 165097 iters

Settings for tuned run

- `cutpass = 1`
- `heurfreq = -1`
- `probe = -1`
- `varsel = 4`

. . . same result with default of probe = 0

Is Tuning Worthwhile?

Yes, sometimes

- Default settings are not always so good

But . . .

- Chance may play a role
- No settings may be consistently better

And you could always . . .

Try Another Solver

Gurobi 1.1 run (with default settings)

```
Found heuristic solution: objective 408.0000
Root relaxation: objective 2.656000e+02, 42 iterations

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  265.6000    0   23  408.0000  265.6000  34.9%     -    0s
H    0     0                      280.0000  265.6000  5.14%     -    0s
     0     0  266.0000    0   21  280.0000  266.0000  5.00%     -    0s
     0     0  266.0000    0   21  280.0000  266.0000  5.00%     -    0s
     0     0  266.0000    0    6  280.0000  266.0000  5.00%     -    0s
     0     2  266.0000    0    6  280.0000  266.0000  5.00%     -    0s
H   56    52                      272.0000  266.0000  2.21%  14.7    0s
H  443   159                      271.0000  266.0000  1.85%   6.6    0s
   833   319  266.0000   31    7  271.0000  266.0000  1.85%   8.5    1s
H  912    69                      267.0000  266.0000  0.37%   8.8    1s
  2311   743    cutoff   99       267.0000  266.0000  0.37%   9.4    2s
H 4022     3                      266.0000  266.0000  0.00%   8.5    2s
```

2.83 seconds (2 threads)

Try Another Tuner

Paper at this year's CPAIOR conference

- Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, **Automated Configuration of Mixed Integer Programming Solvers**
- We study the application of an automated algorithm configuration procedure to different MIP solvers, instance types and optimization objectives. We show that this fully-automated process yields substantial improvements to the performance of three MIP solvers: CPLEX, GUROBI, and LPSOLVE. Although our method can be used "out of the box" without any domain knowledge specific to MIP, we show that it outperforms the CPLEX special-purpose automated tuning tool.
- `www.cs.ubc.ca/labs/beta/Projects/MIP-Config/`

5: Reformulate

Balanced assignment revisited

- Previous failed run
- Tighter formulations
- Successful run

Reconceived formulations

Failed Run

Active start . . .

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
	0	0	0.0000	61		0.0000	99	
*	0+	0			232.0000	0.0000	99	100.00%
	0	0	0.0000	60	232.0000	Cuts: 55	174	100.00%
	0	0	0.0000	66	232.0000	Flowcuts: 17	250	100.00%
	0	0	0.0000	58	232.0000	Flowcuts: 9	300	100.00%
	0	0	0.0000	57	232.0000	Flowcuts: 13	326	100.00%
*	0+	0			230.0000	0.0000	326	100.00%
*	0+	0			216.0000	0.0000	326	100.00%
	0	2	0.0000	57	216.0000	0.0000	326	100.00%
*	440+	403			214.0000	0.0000	7938	100.00%
*	552+	339			212.0000	0.0000	10797	100.00%
	1000	556	69.9315	50	212.0000	0.0000	16491	100.00%
	2000	1332	42.8547	47	212.0000	0.0000	25669	100.00%
	3000	2276	81.6541	49	212.0000	5.0928	37332	97.60%
	4000	3214	77.9166	49	212.0000	5.1140	47933	97.59%
	5000	4160	71.0567	52	212.0000	6.4918	57582	96.94%
	6000	5089	97.3040	47	212.0000	7.8042	66662	96.32%
	7000	6021	158.4869	37	212.0000	9.3981	75348	95.57%
	8000	6942	157.5392	36	212.0000	11.2257	84237	94.70%

....

Failed Run *(cont'd)*

. . . bogs down completely

```
           Nodes                                          Cuts/
    Node    Left  Objective  IInf  Best Integer     Best Node     ItCnt      Gap
 ..................
 6244000 5769420    91.8882    46      212.0000       55.4261  37227229   73.86%
 6245000 5770348   123.4752    34      212.0000       55.4272  37233744   73.86%
 6246000 5771270    63.5603    48      212.0000       55.4289  37239584   73.85%
 6247000 5772192   106.5663    43      212.0000       55.4294  37245120   73.85%
 6248000 5773112    64.0217    47      212.0000       55.4308  37251128   73.85%
 6249000 5774034   181.2576    31      212.0000       55.4310  37257940   73.85%
 6250000 5774954   119.4546    35      212.0000       55.4320  37263877   73.85%
Elapsed time = 9116.25 sec. (tree size = 1616.65 MB)
Nodefile size = 1488.81 MB (685.88 MB after compression)
 6251000 5775885   182.0327    29      212.0000       55.4328  37270210   73.85%
 6252000 5776807   140.1960    39      212.0000       55.4330  37275647   73.85%
 6253000 5777720    91.9423    43      212.0000       55.4346  37281516   73.85%
 6254000 5778648   127.8185    35      212.0000       55.4355  37286884   73.85%
8 flow-cover cuts
2 Gomory cuts
1 zero-half cut
9 mixed-integer rounding cuts
CPLEX 11.2.0: ran out of memory.
```

Tighter Formulation 1

Definition of overlap for person i

```
minimize TotalOverlap:
   sum {i in PEOPLE} Overlap[i];

subj to OverlapDefn {i in PEOPLE, j in 1..numberGrps}:
   Overlap[i] >=
      sum {i2 in PEOPLE diff {i}: title[i2] = title[i]} Assign[i2,j] +
      sum {i2 in PEOPLE diff {i}: loc[i2] = loc[i]} Assign[i2,j] +
      sum {i2 in PEOPLE diff {i}: dept[i2] = dept[i]} Assign[i2,j] +
      sum {i2 in PEOPLE diff {i}: sex[i2] = sex[i]} Assign[i2,j]
         - maxOverlap[i] * (1 - Assign[i,j]);
```

- `maxOverlap[i]` must be ≥ greatest overlap possible
- Smaller values give stronger b&b lower bounds
  - theoretically correct: `4 * (maxInGrp-1)` → `0.0`
  - empirically justified: `1 * (maxInGrp-1)` → `156.8 (26.0%)`

Tighter Formulation 2

Symmetry constraint (a)

```
subject to BreakSymm1
   {i in FIRST_PEOPLE, j in ord(i,FIRST_PEOPLE)+1..numberGrps}:
      Assign[i,j] = 0;
```

- choose the first (`numberGrps-1`) people in some way
- assign the *i*th person to one of the first *i* groups

Tighter Formulation 2 *(cont'd)*

Symmetry constraint (b)

```
set TYPES = setof {i in PEOPLE} (title[i],loc[i],dept[i],sex[i]);
set TYPEpeople {(t1,t2,t3,t4) in TYPES} =
   {i in PEOPLE: title[i]=t1 and loc[i]=t2 and
                 dept[i]=t3 and sex[i]=t4} ordered by PEOPLE;

subject to BreakSymm2 {(t1,t2,t3,t4) in TYPES,
   pnum in 1..card(TYPEpeople[t1,t2,t3,t4])-1, j in 1..numberGrps, k in 1..j-1}:
      Assign[member(pnum+1,TYPEpeople[t1,t2,t3,t4]),k]
         <= 1 - Assign[member(pnum,TYPEpeople[t1,t2,t3,t4]),j];
```

- identify "types" of people
  who are identical in all four characteristics
- order the people of each type, and order the groups
- with each type, assign
  higher-numbered people to higher-numbered groups

Tighter Formulation 2 *(cont'd)*

Symmetry strategies

- `BreakSymm1` increases the b&b lower bound a bit
- `BreakSymm2` does not increase the lower bound
- CPLEX's `symmetry` directive is more effective
  - set `symmetry=5` for greatest symmetry-breaking effort

Tighter Formulation 3

Group size limits

```
subj to GroupSize {j in 1..numberGrps}:
   minInGrp <= sum {i in PEOPLE} Assign[i,j] <= maxInGrp;
```

- `miningrp` must be smaller than group size average
- `maxingrp` must be larger than group size average

- Tighter limits give stronger b&b lower bounds
  - `floor(card(PEOPLE)/numberGrps) - 1`
    `ceil (card(PEOPLE)/numberGrps) + 1` → `156.8 (26.0%)`
  - `floor(card(PEOPLE)/numberGrps)`
    `ceil (card(PEOPLE)/numberGrps)` → `177.6 (16.2%)`

Tighter Formulation 3 *(cont'd)*

Group sizes

```
param minInGrp := floor (card(PEOPLE)/numberGrps);
param nMinInGrp := numberGrps - card{PEOPLE} mod numberGrps;

subj to GroupSizeMin {j in 1..nMinInGrp}:
   sum {i in PEOPLE} Assign[i,j] = minInGrp;

subj to GroupSizeMax {j in nMinInGrp+1..numberGrps}:
   sum {i in PEOPLE} Assign[i,j] = minInGrp + 1;
```

- Compute exact sizes of all groups
  - min, max sizes $\rightarrow$ 177.6 (16.2%)
  - exact sizes $\rightarrow$ 183.36 (13.5%)

Example 3

Balanced Assignment

Incorporating enhancements . . .

```
ampl: model gs1f.mod;
ampl: data gs1b.dat;
ampl: option solver cplex;
ampl: option cplex_options 'symmetry 5 mipdisplay 2 mipinterval 1000';
ampl: solve;

MIP Presolve eliminated 54 rows and 0 columns.
MIP Presolve modified 2636 coefficients.
Reduced MIP has 197 rows, 156 columns, and 2585 nonzeros.
Reduced MIP has 130 binaries, 0 generals, 0 SOSs, and 0 indicators.
Clique table members: 62.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: none, using 1 thread.
Root relaxation solution time =    0.03 sec.

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer     Best Node    ItCnt     Gap

*     0+    0                          252.0000                      0     ---
      0     0      183.3626   134      252.0000      183.3626      262   27.24%
.......
```

Successful Run

Much more promising start . . .

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
	0	0	189.1865	100	252.0000	Cuts: 49	445	24.93%
	0	0	189.7246	96	252.0000	Cuts: 12	558	24.71%
*	0+	0			240.0000	189.7246	558	20.95%
	0	0	189.7964	96	240.0000	ZeroHalf: 5	664	20.92%
	0	0	189.8864	97	240.0000	ZeroHalf: 8	782	20.88%
	0	0	189.9590	96	240.0000	ZeroHalf: 6	1002	20.85%
	0	0	189.9768	100	240.0000	ZeroHalf: 7	1166	20.84%
	0	0	189.9769	99	240.0000	ZeroHalf: 4	1184	20.84%
*	0+	0			220.0000	189.9769	1203	13.65%
*	0+	0			216.0000	189.9769	1203	12.05%
	0	2	192.8299	78	216.0000	192.8299	1203	10.73%
*	100+	80			212.0000	193.0563	6092	8.94%
	1000	479	200.3732	83	212.0000	195.6130	36233	7.73%
	2000	1242	205.1626	64	212.0000	195.9832	65307	7.56%
	3000	2103	205.8520	59	212.0000	196.4174	93546	7.35%
	4000	2946	205.5224	57	212.0000	196.8495	120479	7.15%
	5000	3790	201.5651	53	212.0000	197.1664	145209	7.00%
	6000	4624	210.5546	34	212.0000	197.4648	169658	6.86%
	7000	5468	201.2841	60	212.0000	197.6005	195286	6.79%

....

Successful Run *(cont'd)*

. . . leads to successful conclusion

```
           Nodes                                          Cuts/
    Node    Left    Objective  IInf  Best Integer    Best Node      ItCnt     Gap

 30287000   8802       cutoff             212.0000    211.0000 416705257   0.47%
 30288000   7927       cutoff             212.0000    211.0000 416709767   0.47%
 30289000   7021   infeasible             212.0000    211.0000 416714199   0.47%
 30290000   6101   infeasible             212.0000    211.0000 416718973   0.47%
Elapsed time = 46415.00 sec. (tree size = 12.94 MB)
 30291000   5249       cutoff             212.0000    211.0000 416724639   0.47%
 30292000   4407   infeasible             212.0000    211.0000 416730198   0.47%
 30293000   3519   infeasible             212.0000    211.0000 416735118   0.47%
 30294000   2636       cutoff             212.0000    211.0000 416740781   0.47%
 30295000   1758   infeasible             212.0000    211.0000 416746255   0.47%
 30296000    863   infeasible             212.0000    211.0000 416748900   0.47%

3 cover cuts
8 implied bound cuts
23 mixed-integer rounding cuts
35 zero-half cuts
12 Gomory fractional cuts

CPLEX 11.2.0: optimal integer solution; objective 212
416751729 MIP simplex iterations
30296965 branch-and-bound nodes
```

Reconceived Formulation

Different variables

- Min of each type in any group
- Max of each type in any group

Different objective

- Sum of (max – min) over all types

6: "Cheat" on the method

Work scheduling revisited . . .

Cover demands for workers

- Each "shift" requires a certain number of employees
- Each employee works a certain "schedule" of shifts
- *Each schedule that is worked by anyone must be worked by a fixed minimum number*

Minimize total workers needed

- Which schedules are used? `Use[j]` vars
- How many work each of schedule? `Work[j]` vars

Work Scheduling Revisited

Direct approach

- Apply branch-and-bound to whole problem
- Branch "up" first

Indirect approach

- Step 1: Relax integrality of **`Work`** variables
  Solve for zero-one **`Use`** variables
- Step 2: Fix **`Use`** variables
  Solve for integer **`Work`** variables

. . . not necessarily optimal, but . . .

Work Scheduling Revisited

Typical run of indirect approach

```
ampl: model sched1.mod; data sched.dat;

ampl: let least_assign := 16;

ampl: option solver cplex;
ampl: option cplex_options 'branch 1';

ampl: let {j in SCHEDS} Work[j].relax := 1;

ampl: solve;

CPLEX 11.2.0: optimal integer solution; objective 265.6
870496 MIP simplex iterations
55911 branch-and-bound nodes

ampl: fix {j in SCHEDS} Use[j];
ampl: let {j in SCHEDS} Work[j].relax := 0;

ampl: solve;

CPLEX 11.2.0: optimal integer solution; objective 266
24 MIP simplex iterations
4 branch-and-bound nodes
```

Work Scheduling: Hard Case

CPLEX 12.1

- Direct: 465,596,558 nodes, 112013 seconds
- Indirect: 6,886,122 nodes, 617 seconds

Gurobi 3.0 beta

- Direct: 1,330,555,419 nodes, 69945 seconds
- Indirect: 6,354,683 nodes, 299 seconds

Observations

- step 1 gives fractional solution
- step 2 trivially easy and rounds up step 1 objective

. . . hence optimal

7: “Cheat” on the data

Cut large “raw” rolls into smaller ones

- ❖ All raw rolls the same width
- ❖ Various smaller widths ordered
- ❖ Varying numbers of widths ordered

Minimize total raw rolls cut

- ❖ By generating patterns during optimization
- ❖ By enumerating patterns in advance

Roll Cutting

Cutting model

```
set WIDTHS;                                # set of widths to be cut
param orders {WIDTHS} > 0;                 # number of each width to be cut

param nPAT integer >= 0;                   # number of patterns
param nbr {WIDTHS,1..nPAT} integer >= 0;   # rolls of width i in pattern j

var Cut {1..nPAT} integer >= 0;            # rolls cut using each pattern

minimize Number:

   sum {j in 1..nPAT} Cut[j];              # total raw rolls cut

subject to Fill {i in WIDTHS}:

   sum {j in 1..nPAT} nbr[i,j] * Cut[j] >= orders[i];

                                           # for each width,
                                           # rolls cut meet orders
```

Roll Cutting *(cont'd)*

Pattern generation model

```
param roll_width > 0;
param price {WIDTHS} default 0.0;

var Use {WIDTHS} integer >= 0;

minimize Reduced_Cost:

   1 - sum {i in WIDTHS} price[i] * Use[i];

subj to Width_Limit:

   sum {i in WIDTHS} i * Use[i] <= roll_width;
```

Roll Cutting *(cont'd)*

Pattern generation script

```
repeat {

   solve Cutting_Opt;

   let {i in WIDTHS} price[i] := Fill[i].dual;

   solve Pattern_Gen;

   if Reduced_Cost < -0.00001 then {
      let nPAT := nPAT + 1;
      let {i in WIDTHS} nbr[i,nPAT] := Use[i];
      }

   else break;

   };
```

Roll Cutting *(cont'd)*

Pattern enumeration script

```
repeat {
   if curr_sum + curr_width <= roll_width then {
      let pattern[curr_width] := floor((roll_width-curr_sum)/curr_width);
      let curr_sum := curr_sum + pattern[curr_width] * curr_width;
      }
   if curr_width != last(WIDTHS) then
      let curr_width := next(curr_width,WIDTHS);
   else {
      let nPAT := nPAT + 1;
      let {w in WIDTHS} nbr[w,nPAT] := pattern[w];
      let curr_sum := curr_sum - pattern[last(WIDTHS)] * last(WIDTHS);
      let pattern[last(WIDTHS)] := 0;
      let curr_width := min {w in WIDTHS: pattern[w] > 0} w;
      if curr_width < Infinity then {
         let curr_sum := curr_sum - curr_width;
         let pattern[curr_width] := pattern[curr_width] - 1;
         let curr_width := next(curr_width,WIDTHS);
         }
      else break;
      }
   }
```

Roll Cutting *(cont'd)*

Sample data

```
param roll_width := 172 ;

param: WIDTHS: orders :=

       25.000     5
       24.750    73
       18.000    14
       17.500     4
       15.500    23
       15.375     5
       13.875    29
       12.500    87
       12.250     9
       12.000    31
       10.250     6
       10.125    14
       10.000    43
        8.750    15
        8.500    21
        7.750     5 ;
```

. . . Robert W. Haessler, "Selection and Design of Heuristic Procedures for Solving Roll Trim Problems" Management Science 34 (1988) 1460–1471, Table 2

Roll Cutting Results

Patterns generated during optimization (Gilmore-Gomory procedure)

- 32.80 rolls in continuous relaxation
- 40 rolls rounded up to integer
- 34 rolls solving IP using generated patterns

All patterns enumerated in advance

- 27,338,021 non-dominated patterns — too big

Every 100th pattern saved

- 273,380 patterns
- 33 rolls solving IP using enumerated patterns
- 50 seconds: b&b heuristic solves at root (no cuts)

. . . takes much longer to generate than solve