Approaches to Near-Optimally Solving Mixed-Integer Programs

Robert Fourer
Industrial Engineering & Management Sciences
Northwestern University, Evanston, IL, USA
AMPL Optimization LLC
4er@northwestern.edu — 4er@AMPL.com

What a Pivot — Workshop Honoring Bob Bixby’s 65th Birthday
Erlangen, Germany, 26-28 September 2010
Outline

Breaking up
- Work scheduling
- Balanced dinner assignment
- Progressive party assignment

Cutting off
- Paint chip cutting
- Balanced team assignment

Throwing out
- Roll cutting
Work Scheduling

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?
Work Scheduling

Model using zero-one variables

\[
\begin{align*}
\text{var } & \text{Work} \{\text{SCHEDS}\} \geq 0 \text{ integer}; \\
\text{var } & \text{Use} \{\text{SCHEDS}\} \geq 0 \text{ binary}; \\
\text{minimize } & \text{Total\_Cost:} \\
& \quad \text{sum } \{j \text{ in SCHEDS}\} \text{ Work}[j]; \\
\text{subject to } & \text{Shift\_Needs } \{i \text{ in SHIFTS}\}: \\
& \quad \text{sum } \{j \text{ in SCHEDS}: i \text{ in SHIFT\_LIST}[j]\} \text{ Work}[j] \geq \text{required}[i]; \\
\text{subject to } & \text{Least\_Use1 } \{j \text{ in SCHEDS}\}: \\
& \quad \text{Work}[j] \geq \text{least\_assign} \times \text{Use}[j]; \\
\text{subject to } & \text{Least\_Use2 } \{j \text{ in SCHEDS}\}: \\
& \quad \text{Work}[j] \leq (\text{max } \{i \text{ in SHIFT\_LIST}[j]\} \text{ required}[i]) \times \text{Use}[j];
\end{align*}
\]
Breaking Up 1

Work Scheduling

Test data

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 ;
set SHIFT_LIST[6] := Mon1 Tue1 Wed1 Thu2 Fri2 ;
set SHIFT_LIST[7] := Mon1 Tue1 Wed1 Thu2 Fri3 ;

......

param required :=
                     Mon1 100 Mon2 78 Mon3 52
                     Tue1 100 Tue2 78 Tue3 52
                     Wed1 100 Wed2 78 Wed3 52
Work Scheduling

Branch & bound

<table>
<thead>
<tr>
<th>least_assign</th>
<th>nodes</th>
<th>iterations</th>
<th>seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1345687</td>
<td>10113022</td>
<td>115</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td>&gt; 30000</td>
</tr>
<tr>
<td>18</td>
<td>15870199</td>
<td>125799234</td>
<td>1566</td>
</tr>
<tr>
<td>19</td>
<td>206355833</td>
<td>1619459036</td>
<td>11747</td>
</tr>
<tr>
<td>20</td>
<td>232603</td>
<td>1105751</td>
<td>19</td>
</tr>
<tr>
<td>21</td>
<td>273837</td>
<td>1262181</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>96277</td>
<td>533727</td>
<td>10</td>
</tr>
<tr>
<td>23</td>
<td>129899</td>
<td>632361</td>
<td>10</td>
</tr>
<tr>
<td>24</td>
<td>99489</td>
<td>483954</td>
<td>8</td>
</tr>
</tbody>
</table>

Optimum of relaxation is always 265.6

<= 16: optimum of MIP is 266

>= 20: optimum is integral with Work variables relaxed
Work Scheduling

Two-step 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

Typical run of indirect approach

```
ampl: model sched1.mod;
ampl: data sched.dat;
ampl: let least_assign := 17;
ampl: option solver gurobi;
ampl: let {j in SCHEDS} Work[j].relax := 1;
ampl: solve;
Gurobi 1.1.3: optimal solution; objective 266.5
7898786 simplex iterations;
1556653 branch-and-cut nodes

ampl: fix {j in SCHEDS} Use[j];
ampl: let {j in SCHEDS} Work[j].relax := 0;
ampl: solve;
Gurobi 1.1.3: optimal solution; objective 267
4 simplex iterations;
0 branch-and-cut nodes
```
Breaking Up 1

Work Scheduling

Two-step approach

<table>
<thead>
<tr>
<th>least_assign</th>
<th>nodes</th>
<th>iterations</th>
<th>seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>71924</td>
<td>285172</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>1556653</td>
<td>7898786</td>
<td>120</td>
</tr>
<tr>
<td>18</td>
<td>5538287</td>
<td>33278060</td>
<td>305</td>
</tr>
<tr>
<td>19</td>
<td>6866450</td>
<td>47120495</td>
<td>388</td>
</tr>
<tr>
<td>20</td>
<td>117970</td>
<td>440182</td>
<td>9</td>
</tr>
<tr>
<td>21</td>
<td>76873</td>
<td>299338</td>
<td>7</td>
</tr>
<tr>
<td>22</td>
<td>61727</td>
<td>259012</td>
<td>5</td>
</tr>
<tr>
<td>23</td>
<td>111721</td>
<td>392251</td>
<td>8</td>
</tr>
<tr>
<td>24</td>
<td>82152</td>
<td>292187</td>
<td>6</td>
</tr>
</tbody>
</table>

In this example . . .

- step 2 always trivially easy
- step 2 objective always rounds up step 1 objective

. . . hence optimal
Work Scheduling: More on Case “17”

**CPLEX 12.1**
- Direct: 465,596,558 nodes, 112,013 seconds
- Indirect: 6,886,122 nodes, 617 seconds

**Gurobi 3.0 beta**
- Direct: 1,330,555,419 nodes, 69,945 seconds
- Indirect: 6,354,683 nodes, 299 seconds
Balanced Dinner Assignment

Setting

- meeting of employees from around the world at New York offices of a Wall Street firm

Given

- title, location, department, sex, for each of about 1000 people

Assign

- these people to around 25 dinner groups

So that

- the groups are as “diverse” as possible
Minimum “Variation” Model

set PEOPLE;  # individuals to be assigned
set CATEG;
param type {PEOPLE,CATEG} symbolic default "";
set TYPES {k in CATEG} = setof {i in PEOPLE} type[i,k];

# categories by which people are classified;
# type of each person in each category

param numberGrps integer > 0;
param minInGrp integer > 0;
param maxInGrp integer >= minInGrp;

# number of groups; bounds on size of groups


Thanks also to Collette Coullard.
Breaking Up 2

(variables and objective)

var Assign {i in PEOPLE, j in 1..numberGrps} binary;
    # assignments of people to groups

var MinType {k in CATEG, t in TYPES[k]}
    <= floor (card {i in PEOPLE: type[i,k] = t} / numberGrps);

var MaxType {k in CATEG, t in TYPES[k]}
    >= ceil (card {i in PEOPLE: type[i,k] = t} / numberGrps);
    # min/max of each type over all groups

minimize TotalVariation:
    sum {k in CATEG, t in TYPES[k]}
        (MaxType[k,t] - MinType[k,t]);
    # Sum of variation over all types
Breaking Up 2

\textit{(constraints)}

\begin{verbatim}
subj to AssignAll \{i in PEOPLE\}:
    sum \{j in 1..numberGrps\} Assign[i,j] = 1;

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

subj to MinTypeDefn
    \{j in 1..numberGrps, k in CATEG, t in TYPES[k]\}:
    MinType[k,t] <= sum \{i in PEOPLE: type[i,k] = t\} Assign[i,j];

subj to MaxTypeDefn
    \{j in 1..numberGrps, k in CATEG, t in TYPES[k]\}:
    MaxType[k,t] >= sum \{i in PEOPLE: type[i,k] = t\} Assign[i,j];

# Defining constraints for
# min and max type variables
\end{verbatim}
**Breaking Up 2**

**Solving for Minimum Variation**

1054 variables:
- 1000 binary variables
- 54 linear variables

560 constraints, all linear; 12200 nonzeros

1 linear objective; 54 nonzeros.

**CPLEX 3.0:**

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Left</th>
<th>Objective</th>
<th>IInf</th>
<th>Best Integer</th>
<th>Cuts/Best Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>17.0000</td>
<td>299</td>
<td>17.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>17.0000</td>
<td>322</td>
<td>17.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>17.0000</td>
<td>332</td>
<td>17.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>17.0000</td>
<td>328</td>
<td>17.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>17.0000</td>
<td>329</td>
<td>17.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>17.0000</td>
<td>329</td>
<td>17.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>17.0000</td>
<td>339</td>
<td>17.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>70</td>
<td>70</td>
<td>17.0000</td>
<td>344</td>
<td>17.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>17.0000</td>
<td>342</td>
<td>17.0000</td>
<td>17.0000</td>
</tr>
</tbody>
</table>

......
### Breaking Up 2
(continued)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Objective</th>
<th>IInf</th>
<th>Best Integer</th>
<th>Cuts/Best Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>Left</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>250</td>
<td>43.6818</td>
<td>74</td>
<td>17.0000</td>
</tr>
<tr>
<td>260</td>
<td>260</td>
<td>46.5000</td>
<td>58</td>
<td>17.0000</td>
</tr>
<tr>
<td>*</td>
<td>265</td>
<td>47.0000</td>
<td>0</td>
<td>47.0000</td>
</tr>
<tr>
<td>270</td>
<td>266</td>
<td>17.0000</td>
<td>314</td>
<td>47.0000</td>
</tr>
<tr>
<td>280</td>
<td>276</td>
<td>17.0000</td>
<td>351</td>
<td>47.0000</td>
</tr>
<tr>
<td>290</td>
<td>286</td>
<td>17.0000</td>
<td>340</td>
<td>47.0000</td>
</tr>
<tr>
<td>300</td>
<td>296</td>
<td>17.0000</td>
<td>337</td>
<td>47.0000</td>
</tr>
<tr>
<td>310</td>
<td>306</td>
<td>17.0000</td>
<td>341</td>
<td>47.0000</td>
</tr>
<tr>
<td>.....</td>
<td></td>
<td>21.5208</td>
<td>243</td>
<td>47.0000</td>
</tr>
<tr>
<td>630</td>
<td>609</td>
<td>23.3028</td>
<td>244</td>
<td>47.0000</td>
</tr>
<tr>
<td>650</td>
<td>626</td>
<td>17.3796</td>
<td>269</td>
<td>47.0000</td>
</tr>
<tr>
<td>660</td>
<td>636</td>
<td>17.7981</td>
<td>271</td>
<td>47.0000</td>
</tr>
<tr>
<td>*</td>
<td>666</td>
<td>19.0000</td>
<td>0</td>
<td>19.0000</td>
</tr>
<tr>
<td>670</td>
<td>440</td>
<td>17.0000</td>
<td>147</td>
<td>19.0000</td>
</tr>
<tr>
<td>680</td>
<td>446</td>
<td>17.0714</td>
<td>213</td>
<td>19.0000</td>
</tr>
<tr>
<td>690</td>
<td>454</td>
<td>17.5000</td>
<td>186</td>
<td>19.0000</td>
</tr>
</tbody>
</table>
## Breaking Up 2
(concluded)

<table>
<thead>
<tr>
<th>Node</th>
<th>Left</th>
<th>Objective</th>
<th>IInf</th>
<th>Best Integer</th>
<th>Cuts/Best Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>461</td>
<td>17.1364</td>
<td>268</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>710</td>
<td>468</td>
<td>17.3117</td>
<td>267</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>720</td>
<td>475</td>
<td>17.0000</td>
<td>211</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>730</td>
<td>484</td>
<td>17.2652</td>
<td>226</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>740</td>
<td>490</td>
<td>17.0000</td>
<td>106</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>750</td>
<td>497</td>
<td>17.0000</td>
<td>24</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>*</td>
<td>752</td>
<td>0</td>
<td>17.0000</td>
<td>0</td>
<td>17.0000</td>
</tr>
</tbody>
</table>

Times (seconds):
- Input = 0.266667
- Solve = 864.733
- Output = 0.166667

**CPLEX 3.0**: optimal integer solution; objective 17
- 45621 simplex iterations
- 752 branch-and-bound nodes
Scaling Up

Real model was more complicated

- Rooms hold from 20–25 to 50–55 people
- Must avoid isolating assignments:
  - a person is “isolated” in a group that contains no one from the same location with the same or “adjacent” title

Problem was too big

- Aggregate people who match in all categories (986 people, but only 287 different kinds)
- Solve first for title and location only, then for refinement to department and sex
- Stop at first feasible solution to title-location problem
Full “Title-Location” Model

set PEOPLE ordered;

param title {PEOPLE} symbolic;
param loc {PEOPLE} symbolic;

set TITLE ordered;
    check {i in PEOPLE}: title[i] in TITLE;

set LOC = setof {i in PEOPLE} loc[i];

set TYPE2 = setof {i in PEOPLE} (title[i], loc[i]);
param number2 {(i1, i2) in TYPE2} =
    card {i in PEOPLE: title[i]=i1 and loc[i]=i2};

set REST ordered;

param loDine {REST} integer > 10;
param hiDine {j in REST} integer >= loDine[j];

param loCap := sum {j in REST} loDine[j];
param hiCap := sum {j in REST} hiDine[j];

param loFudge := ceil ((loCap less card {PEOPLE}) / card {REST});
param hiFudge := ceil ((card {PEOPLE} less hiCap) / card {REST});
param frac2title {i1 in TITLE} 
    = sum {(i1,i2) in TYPE2} number2[i1,i2] / card {PEOPLE};

param frac2loc {i2 in LOC} 
    = sum {(i1,i2) in TYPE2} number2[i1,i2] / card {PEOPLE};

param expDine {j in REST} 
    = if loFudge > 0 then loDine[j] else 
     if hiFudge > 0 then hiDine[j] else (loDine[j] + hiDine[j]) / 2;

param loTargetTitle {i1 in TITLE, j in REST} := 
    floor (round (frac2title[i1] * expDine[j], 6));

param hiTargetTitle {i1 in TITLE, j in REST} := 
    ceil (round (frac2title[i1] * expDine[j], 6));

param loTargetLoc {i2 in LOC, j in REST} := 
    floor (round (frac2loc[i2] * expDine[j], 6));

param hiTargetLoc {i2 in LOC, j in REST} := 
    ceil (round (frac2loc[i2] * expDine[j], 6));
Breaking Up 2

(variables, objective, assign constraints)

```
var Assign2 {TYPE2,REST} integer >= 0;
var Dev2Title {TITLE} >= 0;
var Dev2Loc {LOC} >= 0;

minimize Deviation:
    sum {i1 in TITLE} Dev2Title[i1] + sum {i2 in LOC} Dev2Loc[i2];

subject to Assign2Type {(i1,i2) in TYPE2}:
    sum {j in REST} Assign2[i1,i2,j] = number2[i1,i2];

subject to Assign2Rest {j in REST}:
    loDine[j] - loFudge
    <= sum {(i1,i2) in TYPE2} Assign2[i1,i2,j]
    <= hiDine[j] + hiFudge;
```
Breaking Up 2

*(constraints to define “variation”)*

subject to Lo2TitleDefn \{i1 in TITLE, j in REST\}:
\[
Dev2Title[i1] \geq \\
loTargetTitle[i1,j] - \sum \{(i1,i2) in TYPE2\} \text{Assign2}[i1,i2,j];
\]

subject to Hi2TitleDefn \{i1 in TITLE, j in REST\}:
\[
Dev2Title[i1] \geq \\
\sum \{(i1,i2) in TYPE2\} \text{Assign2}[i1,i2,j] - \text{hiTargetTitle}[i1,j];
\]

subject to Lo2LocDefn \{i2 in LOC, j in REST\}:
\[
Dev2Loc[i2] \geq \\
loTargetLoc[i2,j] - \sum \{(i1,i2) in TYPE2\} \text{Assign2}[i1,i2,j];
\]

subject to Hi2LocDefn \{i2 in LOC, j in REST\}:
\[
Dev2Loc[i2] \geq \\
\sum \{(i1,i2) in TYPE2\} \text{Assign2}[i1,i2,j] - \text{hiTargetLoc}[i2,j];
\]
Breaking Up 2

(parameters for ruling out “isolation”)

```plaintext
set ADJACENT {i1 in TITLE} =
    (if i1 <> first(TITLE) then {prev(i1)} else {}) union
    (if i1 <> last(TITLE) then {next(i1)} else {});

set ISO = {(i1,i2) in TYPE2: (i2 <> "Unknown") and
    ((number2[i1,i2] >= 2) or
    (number2[i1,i2] = 1 and
     sum {ii1 in ADJACENT[i1]: (ii1,i2) in TYPE2}
     number2[ii1,i2] > 0))};

param give {ISO} default 2;
param giveTitle {TITLE} default 2;
param giveLoc {LOC} default 2;

param upperbnd {(i1,i2) in ISO, j in REST} =
    min (ceil((number2[i1,i2]/card {PEOPLE}) * hiDine[j]) + give[i1,i2],
    hiTargetTitle[i1,j] + giveTitle[i1],
    hiTargetLoc[i2,j] + giveLoc[i2],
    number2[i1,i2]);
```
Breaking Up 2

*(constraints to rule out “isolation”)*

\[
\begin{align*}
\text{var} & \quad \text{Lone } \{(i1,i2) \text{ in ISO, } j \text{ in REST}\} \text{ binary;} \\
\text{subj to Isolation1 } \{(i1,i2) \text{ in ISO, } j \text{ in REST}\}: \\
& \quad \text{Assign2}[i1,i2,j] \leq \text{upperbnd}[i1,i2,j] \times \text{Lone}[i1,i2,j]; \\
\text{subj to Isolation2a } \{(i1,i2) \text{ in ISO, } j \text{ in REST}\}: \\
& \quad \text{Assign2}[i1,i2,j] + \\
& \quad \quad \text{sum } \{i11 \text{ in ADJACENT}[i1]: (i11,i2) \text{ in TYPE2}\} \text{ Assign2}[i11,i2,j] \\
& \quad \quad \quad \geq 2 \times \text{Lone}[i1,i2,j]; \\
\text{subj to Isolation2b } \{(i1,i2) \text{ in ISO, } j \text{ in REST}\}: \\
& \quad \text{Assign2}[i1,i2,j] \geq \text{Lone}[i1,i2,j];
\end{align*}
\]
Success

First problem
- using OSL: 128 “supernodes”, 6.7 hours
- using CPLEX 2.1: took too long

Second problem
- using CPLEX 2.1: 864 nodes, 3.6 hours
- using OSL: 853 nodes, 4.3 hours

Finish
- Refine to individual assignments: a trivial LP
- Make table of assignments using AMPL printf command
- Ship table to client, who imports to database
Breaking Up 2

Solver Improvements

**CPLEX 3.0**
- First problem: 1200 nodes, 1.1 hours
- Second problem: 1021 nodes, 1.3 hours

**CPLEX 4.0**
- First problem: 517 nodes, 5.4 minutes
- Second problem: 1021 nodes, 21.8 minutes

**CPLEX 9.0**
- First problem: 560 nodes, 83.1 seconds
- Second problem: 0 nodes, 17.9 seconds
Solver Improvements

**CPLEX 12.1**
- First problem: 0 nodes, 9.5 seconds
- Second problem: 0 nodes, 1.5 seconds

**Gurobi 2.0**
- First problem: 0 nodes, 13.5 seconds
- Second problem: 0 nodes, 1.6 seconds
Progressive Party Assignment

Setting
- yacht club holding a party
- each boat has a certain crew size & guest capacity

Decisions
- choose a minimal number yachts as “hosts”
- assign each non-host crew to visit a host yacht
- . . . in each of 6 periods

Requirements
- no yacht’s capacity is exceeded
- no crew visits the same yacht more than once
- no two crews meet more than once
Progressive Party Problem

Parameters & variables

```plaintext
param B > 0, integer;
set BOATS := 1 .. B;

param capacity {BOATS} integer >= 0;
param crew {BOATS} integer > 0;
param guest_cap {i in BOATS} := capacity[i] less crew[i];

param T > 0, integer;
set TIMES := 1..T;

var Host {i in BOATS} binary;       # i is a host boat
var Visit {i in BOATS, j in BOATS, t in TIMES: i <> j} binary;
    # crew of j visits party on i at t
var Meet {i in BOATS, j in BOATS, t in TIMES: i < j} >= 0, <= 1;
    # crews of i and j meet at t
```

Progressive Party Problem

Host objective and constraints

minimize TotalHosts: sum {i in BOATS} Host[i];

    # minimize total host boats

set MUST_BE_HOST within BOATS;
subj to MustBeHost {i in MUST_BE_HOST}: Host[i] = 1;

    # some boats are designated host boats

set MUST_BE_GUEST within BOATS;
subj to MustBeGuest {i in MUST_BE_GUEST}: Host[i] = 0;

    # some boats (the virtual boats) are designated guest boats

param mincrew := min {j in BOATS} crew[j];
subj to NeverHost {i in BOATS: guest_cap[i] < mincrew}: Host[i] = 0;

    # boats with very limited guest capacity can never be hosts
Progressive Party Problem

Visit constraints

```plaintext
subj to PartyHost {i in BOATS, j in BOATS, t in TIMES: i <> j}:
    Visit[i,j,t] <= Host[i];
    # parties must occur on host boats

subj to Cap {i in BOATS, t in TIMES}:
    sum {j in BOATS: j <> i} crew[j] * Visit[i,j,t] <= guest_cap[i] * Host[i];
    # boats may not have more visitors than they can handle

subj to CrewHost {j in BOATS, t in TIMES}:
    Host[j] + sum {i in BOATS: i <> j} Visit[i,j,t] = 1;
    # every crew is either hosting or visiting a party

subj to VisitOnce {i in BOATS, j in BOATS: i <> j}:
    sum {t in TIMES} Visit[i,j,t] <= Host[i];
    # a crew may visit a host at most once
```
Progressive Party Problem

Meeting constraints

subj to Link {i in BOATS, 
    j in BOATS, jj in BOATS, t in TIMES: i <> j and i <> jj and j < jj}:
    Meet[j,jj,t] >= Visit[i,j,t] + Visit[i,jj,t] - 1;
    # meetings occur when two crews are on same host at same time

subj to MeetOnce {j in BOATS, jj in BOATS: j < jj}:
    sum {t in TIMES} Meet[j,jj,t] <= 1;
    # two crews may meet at most once
**Progressive Party Problem**

**Data**

```
param B := 42;
param T := 6;
param:  capacity  crew :=
  1      6      2
  2      8      2
  3     12      2
  4     12      2
  5     12      4
  6     12      4
  7     12      4
  .......
  37     6      4
  38     6      5
  39     9      7
  40     0      2
  41     0      3
  42     0      4;
set MUST_BE_HOST := 1 2 3;
set MUST_BE_GUEST := 40 41 42;
```
Direct Approach

ampl: solve;

Presolve eliminates 88078 constraints and 2892 variables.
Adjusted problem:
12648 variables:
    7482 binary variables
    5166 linear variables
131990 constraints, all linear; 410546 nonzeros
1 linear objective; 36 no

MIP Presolve eliminated 14317 rows and 721 columns.
MIP Presolve modified 6762 coefficients.
Reduced MIP has 117674 rows, 11928 columns, and 374126 nonzeros.
Reduced MIP has 7482 binaries, 0 generals, 0 SOSs, and 0 indicators.
Probing time = 0.03 sec.

MIP Presolve eliminated 978 rows and 0 columns.
MIP Presolve modified 1956 coefficients.
Reduced MIP has 116696 rows, 11928 columns, and 371192 nonzeros.
Reduced MIP has 7482 binaries, 0 generals, 0 SOSs, and 0 indicators.
Probing time = 0.03 sec.

Clique table members: 7283.
MIP emphasis: integer feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 8 threads.
Root relaxation solution time = 6.47 sec. nzeros.
### Breaking Up 3

#### Direct Approach (branching)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Cuts/</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>Left</td>
<td>Objective</td>
<td>IInf</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>12.2000</td>
<td>484</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>12.2000</td>
<td>312</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>12.2000</td>
<td>332</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>12.2000</td>
<td>256</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>12.2000</td>
<td>150</td>
</tr>
<tr>
<td>40</td>
<td>42</td>
<td>12.2222</td>
<td>280</td>
</tr>
<tr>
<td>80</td>
<td>82</td>
<td>12.3333</td>
<td>253</td>
</tr>
<tr>
<td>120</td>
<td>122</td>
<td>13.0000</td>
<td>285</td>
</tr>
<tr>
<td>160</td>
<td>162</td>
<td>13.0000</td>
<td>216</td>
</tr>
<tr>
<td>*</td>
<td>192+</td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>202</td>
<td>13.0000</td>
<td>167</td>
</tr>
<tr>
<td>240</td>
<td>242</td>
<td>13.0000</td>
<td>165</td>
</tr>
<tr>
<td>280</td>
<td>282</td>
<td>13.0000</td>
<td>274</td>
</tr>
<tr>
<td>320</td>
<td>322</td>
<td>13.0000</td>
<td>69</td>
</tr>
<tr>
<td>360</td>
<td>362</td>
<td>13.0000</td>
<td>65</td>
</tr>
<tr>
<td>*</td>
<td>367+</td>
<td>316</td>
<td></td>
</tr>
<tr>
<td>380</td>
<td>330</td>
<td>13.0000</td>
<td>147</td>
</tr>
<tr>
<td>400</td>
<td>350</td>
<td>13.0000</td>
<td>8</td>
</tr>
</tbody>
</table>
Direct Approach (results)

Clique cuts applied: 9
Cover cuts applied: 263
Implied bound cuts applied: 56
Zero-half cuts applied: 15

Root node processing (before b&c):
   Real time = 234.06
Parallel b&c, 8 threads:
   Real time = 377.09
   Sync time (average) = 38.18
   Wait time (average) = 168.18

Total (root+branch&cut) = 611.15 sec.

Times (seconds):
Input = 0.156
Solve = 611.963
Output = 0.125

CPLEX 12.2.0.0: optimal integer solution; objective 13
418678 MIP simplex iterations
420 branch-and-bound nodes

... results highly variable across settings and solvers
Multi-Step Approach

**Determine hosts**
- solve 1-period problem
- fix hosts
- fix 1st-period visits

**Determine visits: for** $t = 2, 3, \ldots$
- solve $t^{th}$-period problem
- fix $t^{th}$ period visits

\ldots hosts & previous $t-1$ periods already fixed
model partyKA.mod;
data partyKA.dat;

option solver cplexamp;
option cplex_options 'branch 1 startalg 1 subalg 1 mipemphasis 1 timing 1';
option show_stats 1;

# -------
let T := 1;
repeat {
    solve;
    if T = 1 then fix Host;
    if solve_result = "solved" then {
        let T := T + 1;
        fix {i in BOATS, j in BOATS: i <> j} Visit[i,j,T-1];
    }
    else break;
};
Breaking Up 3

Multi-Step Run *(periods 1 to 3)*

```plaintext
ampl: include partyKB.run

Reduced MIP has 983 rows, 1272 columns, and 4364 nonzeros.
Reduced MIP has 1272 binaries, 0 generals, 0 SOSs, and 0 indicators.
Solve =  0.249

CPLEX 12.2.0.0: optimal integer solution; objective 13
189 MIP simplex iterations
0 branch-and-bound nodes

Reduced MIP has 169 rows, 342 columns, and 997 nonzeros.
Reduced MIP has 342 binaries, 0 generals, 0 SOSs, and 0 indicators.
Solve =  0.063

CPLEX 12.2.0.0: optimal integer solution; objective 13
76 MIP simplex iterations
0 branch-and-bound nodes

Reduced MIP has 258 rows, 313 columns, and 1162 nonzeros.
Reduced MIP has 313 binaries, 0 generals, 0 SOSs, and 0 indicators.
Solve =  0.062

CPLEX 12.2.0.0: optimal integer solution; objective 13
77 MIP simplex iterations
0 branch-and-bound nodes
```
Multi-Step Run \textit{(periods 4 to 6)}

Reduced MIP has 319 rows, 284 columns, and 1284 nonzeros.  
Reduced MIP has 284 binaries, 0 generals, 0 SOSs, and 0 indicators.  
Solve = 0.093  
CPLEX 12.2.0.0: optimal integer solution; objective 13  
64 MIP simplex iterations  
0 branch-and-bound nodes  

Reduced MIP has 328 rows, 255 columns, and 1289 nonzeros.  
Reduced MIP has 255 binaries, 0 generals, 0 SOSs, and 0 indicators.  
Solve = 0.047  
CPLEX 12.2.0.0: optimal integer solution; objective 13  
65 MIP simplex iterations  
0 branch-and-bound nodes  

Reduced MIP has 327 rows, 226 columns, and 1264 nonzeros.  
Reduced MIP has 226 binaries, 0 generals, 0 SOSs, and 0 indicators.  
Solve = 0.031  
CPLEX 12.2.0.0: optimal integer solution; objective 13  
58 MIP simplex iterations  
0 branch-and-bound nodes


**Breaking Up 3**

**Multi-Step Run (periods 7 to 9)**

---

Reduced MIP has 281 rows, 197 columns, and 1103 nonzeros.  
Reduced MIP has 197 binaries, 0 generals, 0 SOSs, and 0 indicators.  
Solve = 0.094  
CPLEX 12.2.0.0: optimal integer solution; objective 13  
69 MIP simplex iterations  
0 branch-and-bound nodes

Reduced MIP has 232 rows, 168 columns, and 914 nonzeros.  
Reduced MIP has 168 binaries, 0 generals, 0 SOSs, and 0 indicators.  
Solve = 0.094  
CPLEX 12.2.0.0: optimal integer solution; objective 13  
126 MIP simplex iterations  
0 branch-and-bound nodes

Reduced MIP has 174 rows, 133 columns, and 672 nonzeros.  
Reduced MIP has 133 binaries, 0 generals, 0 SOSs, and 0 indicators.  
Solve = 0.187  
CPLEX 12.2.0.0: optimal integer solution; objective 13  
2009 MIP simplex iterations  
50 branch-and-bound nodes

---
Breaking Up 3

Multi-Step Run \textit{(no period 10)}

Reduced MIP has 120 rows, 102 columns, and 469 nonzeros.
Reduced MIP has 102 binaries, 0 generals, 0 SOSs, and 0 indicators.

Solve = 0.062

CPLEX 12.2.0.0: integer infeasible.
75 MIP simplex iterations
0 branch-and-bound nodes
Paint Chip Cutting

Produce paint chips from rolls of material

- Several “groups” (types) of chips
- Various numbers of “colors” per group
- Numerous “patterns” of groups on rolls

Costs proportional to numbers of

- Patterns cut
- Pattern changes
- Width changes
Chip Cutting

Model (variables & objective)

\[
\begin{align*}
\text{var } & \text{Cut } \{1..\text{nPats}\} \geq 0, \text{ integer}; \quad \# \text{ number of each pattern cut} \\
\text{var } & \text{PatternChange } \{1..\text{nPats}\} \text{ binary}; \quad \# \text{ 1 iff a pattern is used} \\
\text{var } & \text{WebChange } \{\text{WIDTHS}\} \text{ binary}; \quad \# \text{ 1 iff a width is used} \\
\text{minimize } & \text{Total_Cost:} \\
& \quad \text{sum } \{j \text{ in } 1..\text{nPats}\} \text{ cut_cost}[j] \times \text{Cut}[j] + \quad \\
& \quad \text{pattern_changeover_factor } \times \quad \quad \\
& \quad \text{sum } \{j \text{ in } 1..\text{nPats}\} \text{ change_cost}[j] \times \text{PatternChange}[j] + \quad \\
& \quad \text{web_change_factor } \times \quad \quad \\
& \quad \text{sum } \{w \text{ in } \text{WIDTHS}\} (\text{coat_change_cost} + \text{slit_change_cost}) \times \text{WebChange}[w];
\end{align*}
\]
Cutting Off 1

Chip Cutting

Model (constraints)

subject to SatisfyDemand \{g \text{ in GROUPS}\}:

\[ \text{sum } \{j \text{ in } 1..nPats\} \text{ number_of}[g,j] \times \text{Cut}[j] \geq \text{ncolors}[g]; \]

subject to DefinePatternChange \{j \text{ in } 1..nPats\}:

\[ \text{Cut}[j] \leq \text{maxuse}[j] \times \text{PatternChange}[j]; \]

subject to DefineWebChange \{j \text{ in } 1..nPats\}:

\[ \text{PatternChange}[j] \leq \text{WebChange}[\text{width}[j]]; \]

\[
\text{param maxuse } \{j \text{ in } 1..nPats\} := \\
\quad \max \{g \text{ in GROUPS: number_of}[g,j] > 0\} \text{ ncolors}[g] / \text{number_of}[g,j];
\]

\# upper limit on \text{Cut}[j]  

\[ \ldots \text{very long solve times} \]
Chip Cutting

Model (restricted)

subject to DefinePatternChange \{ j in 1..nPats \}:
\[ \text{Cut}[j] \leq \text{maxuse}[j] \times \text{PatternChange}[j] \];

subject to MinPatternUse \{ j in 1..nPats \}:
\[ \text{Cut}[j] \geq \text{ceil}(\text{minuse}[j]) \times \text{PatternChange}[j] \];

param minuse \{ j in 1..nPats \} :=
\[ \min \{ g \in \text{GROUPS} : \text{number_of}[g,j] > 0 \} \frac{\text{ncolors}[g]}{\text{number_of}[g,j]} \];
# if you use a pattern at all,
# use it to cut all colors of at least one group

... not necessarily optimal, but ...
Cutting Off 1

Chip Cutting

Sample data

param: GROUPS: ncolors slitwidth cutoff paint finish substrate :=

<table>
<thead>
<tr>
<th>grp1</th>
<th>8</th>
<th>3.8125</th>
<th>1.75</th>
<th>latex</th>
<th>flat</th>
<th>P40</th>
</tr>
</thead>
<tbody>
<tr>
<td>grp2</td>
<td>3</td>
<td>3.9375</td>
<td>1.75</td>
<td>latex</td>
<td>flat</td>
<td>P40</td>
</tr>
<tr>
<td>grp3</td>
<td>32</td>
<td>1.6875</td>
<td>1.00</td>
<td>latex</td>
<td>flat</td>
<td>P40</td>
</tr>
<tr>
<td>grp4</td>
<td>4</td>
<td>1.8125</td>
<td>1.00</td>
<td>latex</td>
<td>flat</td>
<td>P40</td>
</tr>
<tr>
<td>grp5</td>
<td>3</td>
<td>1.75</td>
<td>1.00</td>
<td>latex</td>
<td>flat</td>
<td>P40</td>
</tr>
<tr>
<td>grp6</td>
<td>2</td>
<td>1.75</td>
<td>1.00</td>
<td>latex</td>
<td>semi_gloss</td>
<td>P40</td>
</tr>
<tr>
<td>grp7</td>
<td>3</td>
<td>1.875</td>
<td>1.00</td>
<td>latex</td>
<td>flat</td>
<td>P40</td>
</tr>
<tr>
<td>grp8</td>
<td>1</td>
<td>1.875</td>
<td>1.00</td>
<td>latex</td>
<td>gloss</td>
<td>P40</td>
</tr>
</tbody>
</table>

param orderqty := 588500;

param spoilage_factor := .15;
Results

Without restriction

- 1812 rows, 1807 columns, 5976 nonzeros
- 7,115,951 simplex iterations
- 221,368 branch-and-bound nodes
- 14,620.4 seconds

With restriction

- 2402 rows, 1656 columns, 7091 nonzeros
- 230,667 simplex iterations
- 9,892 branch-and-bound nodes
- 501.55 seconds

Objective value

- Same in both cases
Results *(today)*

**Without restriction**
- 1724 rows, 1719 columns, 5800 nonzeros
- 49,831 simplex iterations
- 3,157 branch-and-bound nodes
- 4.867 seconds

**With restriction**
- 2344 rows, 1598 columns, 6982 nonzeros
- 21,598 simplex iterations
- 568 branch-and-bound nodes
- 2.872 seconds

*(Gurobi 1.1.3, 8 processors)*
Results *(today, harder case)*

**Without restriction**
- 4019 rows, 4009 columns, 15198 nonzeros
- 60,122 simplex iterations
- 1,955 branch-and-bound nodes
- 20.626 seconds

**With restriction**
- 5667 rows, 4394 columns, 18464 nonzeros
- 14,468 simplex iterations
- 150 branch-and-bound nodes
- 5.464 seconds

*(Gurobi 1.1.3, 8 processors)*
Balanced Team Assignment

Same idea, different formulation

- Class example of where branch-and-bound fails
  - steadily growing tree
  - terrible initial lower bound
  - gap scarcely grows

Partition people into groups

- diversity measured by several characteristics
- each characteristic has several values

Make groups as diverse as possible

- count “overlaps” for each person in their assigned group
  - for each other in group, count # of matching characteristics
  - sum over all others in group
- minimize sum of overlaps
Balanced Assignment

Test data

- 26 people
- 4 characteristics (4, 4, 4, 2 values)
- 5 groups

**CPLEX 11.2.0:**

Reduced MIP has 161 rows, 265 columns, and 3725 nonzeros.
Reduced MIP has 130 binaries, 0 generals, 0 SOSs, and 0 indicators.

Clique table members: 26.

MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.

Parallel mode: none, using 1 thread.

Root relaxation solution time = -0.00 sec.
Balanced Assignment

Active start . . .

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Node</th>
<th>Left</th>
<th>Objective</th>
<th>IInf</th>
<th>Best Integer</th>
<th>Cuts/</th>
<th>Best Node</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0000</td>
<td></td>
<td>0.0000</td>
<td></td>
<td>99</td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
<td>0</td>
<td></td>
<td></td>
<td>232.0000</td>
<td></td>
<td>0.0000</td>
<td></td>
<td>99 100.00%</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>60</td>
<td>232.0000</td>
<td></td>
<td>Cuts: 55</td>
<td>174</td>
<td>100.00%</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>66</td>
<td>232.0000</td>
<td></td>
<td>Flowcuts: 17</td>
<td>250</td>
<td>100.00%</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>58</td>
<td>232.0000</td>
<td></td>
<td>Flowcuts: 9</td>
<td>300</td>
<td>100.00%</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>57</td>
<td>232.0000</td>
<td></td>
<td>Flowcuts: 13</td>
<td>326</td>
<td>100.00%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>230.0000</td>
<td></td>
<td>0.0000</td>
<td></td>
<td>326 100.00%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>216.0000</td>
<td></td>
<td>0.0000</td>
<td></td>
<td>326 100.00%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td>57</td>
<td>216.0000</td>
<td></td>
<td>0.0000</td>
<td></td>
<td>326 100.00%</td>
</tr>
<tr>
<td></td>
<td>440+</td>
<td>403</td>
<td></td>
<td></td>
<td>214.0000</td>
<td></td>
<td>0.0000</td>
<td></td>
<td>7938 100.00%</td>
</tr>
<tr>
<td></td>
<td>552+</td>
<td>339</td>
<td></td>
<td></td>
<td>212.0000</td>
<td></td>
<td>0.0000</td>
<td></td>
<td>10797 100.00%</td>
</tr>
<tr>
<td>1000</td>
<td>556</td>
<td></td>
<td>69.9315</td>
<td>50</td>
<td>212.0000</td>
<td></td>
<td>0.0000</td>
<td></td>
<td>16491 100.00%</td>
</tr>
<tr>
<td>2000</td>
<td>1332</td>
<td></td>
<td>42.8547</td>
<td>47</td>
<td>212.0000</td>
<td></td>
<td>0.0000</td>
<td></td>
<td>25669 100.00%</td>
</tr>
<tr>
<td>3000</td>
<td>2276</td>
<td></td>
<td>81.6541</td>
<td>49</td>
<td>212.0000</td>
<td></td>
<td>5.9928</td>
<td>3732</td>
<td>97.60%</td>
</tr>
<tr>
<td>4000</td>
<td>3214</td>
<td></td>
<td>77.9166</td>
<td>49</td>
<td>212.0000</td>
<td></td>
<td>5.1140</td>
<td>47933</td>
<td>97.59%</td>
</tr>
<tr>
<td>5000</td>
<td>4160</td>
<td></td>
<td>71.0567</td>
<td>52</td>
<td>212.0000</td>
<td></td>
<td>6.4918</td>
<td>57582</td>
<td>96.94%</td>
</tr>
<tr>
<td>6000</td>
<td>5089</td>
<td></td>
<td>97.3040</td>
<td>47</td>
<td>212.0000</td>
<td></td>
<td>7.8042</td>
<td>66662</td>
<td>96.32%</td>
</tr>
<tr>
<td>7000</td>
<td>6021</td>
<td></td>
<td>158.4869</td>
<td>37</td>
<td>212.0000</td>
<td></td>
<td>9.3981</td>
<td>75348</td>
<td>95.57%</td>
</tr>
<tr>
<td>8000</td>
<td>6942</td>
<td></td>
<td>157.5392</td>
<td>36</td>
<td>212.0000</td>
<td></td>
<td>11.2257</td>
<td>84237</td>
<td>94.70%</td>
</tr>
</tbody>
</table>

.................
Balanced Assignment

... bogs down completely

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Node</th>
<th>Left</th>
<th>Objective</th>
<th>IInf</th>
<th>Best Integer</th>
<th>Cuts/</th>
<th>Best Node</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6244000</td>
<td>5769420</td>
<td>91.8882</td>
<td>46</td>
<td>212.0000</td>
<td>55.4261</td>
<td>37227229</td>
<td>73.86%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6245000</td>
<td>5770348</td>
<td>123.4752</td>
<td>34</td>
<td>212.0000</td>
<td>55.4272</td>
<td>3723744</td>
<td>73.86%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6246000</td>
<td>5771270</td>
<td>63.5603</td>
<td>48</td>
<td>212.0000</td>
<td>55.4289</td>
<td>37239584</td>
<td>73.85%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6247000</td>
<td>5772192</td>
<td>106.5663</td>
<td>43</td>
<td>212.0000</td>
<td>55.4294</td>
<td>37245120</td>
<td>73.85%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6248000</td>
<td>5773112</td>
<td>64.0217</td>
<td>47</td>
<td>212.0000</td>
<td>55.4308</td>
<td>37251128</td>
<td>73.85%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6249000</td>
<td>5774034</td>
<td>181.2576</td>
<td>31</td>
<td>212.0000</td>
<td>55.4310</td>
<td>37257940</td>
<td>73.85%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6250000</td>
<td>5774954</td>
<td>119.4546</td>
<td>35</td>
<td>212.0000</td>
<td>55.4320</td>
<td>37263877</td>
<td>73.85%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6251000</td>
<td>5775885</td>
<td>182.0327</td>
<td>29</td>
<td>212.0000</td>
<td>55.4328</td>
<td>37270210</td>
<td>73.85%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6252000</td>
<td>5776807</td>
<td>140.1960</td>
<td>39</td>
<td>212.0000</td>
<td>55.4330</td>
<td>37275647</td>
<td>73.85%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6253000</td>
<td>5777720</td>
<td>91.9423</td>
<td>43</td>
<td>212.0000</td>
<td>55.4346</td>
<td>37281516</td>
<td>73.85%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6254000</td>
<td>5778648</td>
<td>127.8185</td>
<td>35</td>
<td>212.0000</td>
<td>55.4355</td>
<td>37286884</td>
<td>73.85%</td>
<td></td>
</tr>
</tbody>
</table>

Elapsed time = 9116.25 sec. (tree size = 1616.65 MB)
Nodefile size = 1488.81 MB (685.88 MB after compression)

8 flow-cover cuts
2 Gomory cuts
1 zero-half cut
9 mixed-integer rounding cuts
CPLEX 11.2.0: ran out of memory.
Balanced Assignment

Definition of overlap for person $i$

\[
\text{minimize TotalOverlap:} \\
\quad \sum_{i \in \text{PEOPLE}} \text{Overlap}[i]; \\
\text{subj to OverlapDefn } \{i \in \text{PEOPLE}, j \in 1..\text{numberGrps}\}: \\
\quad \text{Overlap}[i] \geq \\
\quad \quad \sum_{i2 \in \text{PEOPLE} \text{ diff } \{i\}: \text{title}[i2] = \text{title}[i]} \text{Assign}[i2,j] + \\
\quad \quad \sum_{i2 \in \text{PEOPLE} \text{ diff } \{i\}: \text{loc}[i2] = \text{loc}[i]} \text{Assign}[i2,j] + \\
\quad \quad \sum_{i2 \in \text{PEOPLE} \text{ diff } \{i\}: \text{dept}[i2] = \text{dept}[i]} \text{Assign}[i2,j] + \\
\quad \quad \sum_{i2 \in \text{PEOPLE} \text{ diff } \{i\}: \text{sex}[i2] = \text{sex}[i]} \text{Assign}[i2,j] \\
\quad \quad \quad - \max\text{Overlap}[i] \times (1 - \text{Assign}[i,j]);
\]

- $\max\text{Overlap}[i]$ must be $\geq$ greatest overlap possible
- Smaller values give stronger b&b lower bounds
  * theoretically correct: $4 \times (\max\text{InGrp}-1) \rightarrow 0.0$
  * empirically justified: $1 \times (\max\text{InGrp}-1) \rightarrow 156.8$
Background

Balanced Assignment

Group size limits

\[
\text{subj to GroupSize } \{ j \text{ in } 1..\text{numberGrps}\}:
\begin{align*}
\text{minInGrp} & \leq \text{sum } \{ i \text{ in PEOPLE} \} \text{ Assign}[i,j] \leq \text{maxInGrp};
\end{align*}
\]

- \text{minInGrp} must be smaller than group size average
- \text{maxInGrp} must be larger than group size average
- Tighter limits give stronger b&b lower bounds

\[
\begin{align*}
\text{floor}(\text{card(PEOPLE)}/\text{numberGrps}) - 1 \\
\text{ceil}(\text{card(PEOPLE)}/\text{numberGrps}) + 1 & \rightarrow 156.8 \\
\text{floor}(\text{card(PEOPLE)}/\text{numberGrps}) \\
\text{ceil}(\text{card(PEOPLE)}/\text{numberGrps}) & \rightarrow 177.6
\end{align*}
\]
Balanced Assignment

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;

➤ Specify exact sizes of all groups
➤ Exact sizes give stronger b&b lower bounds

* min & max sizes for every g  →  177.6
* exact sizes  →  183.36
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.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Cuts/</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Node</td>
<td>Left</td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

.......
Balanced Assignment

**Much more promising start . . .**

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Cuts/</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>Best Node</td>
<td>IInf</td>
<td>Objective</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>189.1865</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>189.7246</td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
<td>0</td>
<td>240.0000</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>189.7964</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>189.8864</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>189.9590</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>189.9768</td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
<td>0</td>
<td>220.0000</td>
</tr>
<tr>
<td>*</td>
<td>0+</td>
<td>0</td>
<td>216.0000</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>192.8299</td>
<td>78</td>
</tr>
<tr>
<td>*</td>
<td>100+</td>
<td>80</td>
<td>212.0000</td>
</tr>
<tr>
<td>1000</td>
<td>479</td>
<td>200.3732</td>
<td>83</td>
</tr>
<tr>
<td>2000</td>
<td>1242</td>
<td>205.1626</td>
<td>64</td>
</tr>
<tr>
<td>3000</td>
<td>2103</td>
<td>205.8520</td>
<td>59</td>
</tr>
<tr>
<td>4000</td>
<td>2946</td>
<td>205.5224</td>
<td>57</td>
</tr>
<tr>
<td>5000</td>
<td>3790</td>
<td>201.5651</td>
<td>53</td>
</tr>
<tr>
<td>6000</td>
<td>4624</td>
<td>210.5546</td>
<td>34</td>
</tr>
<tr>
<td>7000</td>
<td>5468</td>
<td>201.2841</td>
<td>60</td>
</tr>
</tbody>
</table>

Cutting Off 2
Balanced Assignment

... leads to successful conclusion

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Objective</th>
<th>IInf</th>
<th>Best Integer</th>
<th>Cuts/</th>
<th>Best Node</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>Left</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30287000</td>
<td>8802</td>
<td>cutoff</td>
<td>212.0000</td>
<td>211.0000</td>
<td>416705257</td>
<td>0.47%</td>
<td></td>
</tr>
<tr>
<td>30288000</td>
<td>7927</td>
<td>cutoff</td>
<td>212.0000</td>
<td>211.0000</td>
<td>416709767</td>
<td>0.47%</td>
<td></td>
</tr>
<tr>
<td>30289000</td>
<td>7021</td>
<td>infeasible</td>
<td>212.0000</td>
<td>211.0000</td>
<td>416714199</td>
<td>0.47%</td>
<td></td>
</tr>
<tr>
<td>30290000</td>
<td>6101</td>
<td>infeasible</td>
<td>212.0000</td>
<td>211.0000</td>
<td>416718973</td>
<td>0.47%</td>
<td></td>
</tr>
<tr>
<td>30291000</td>
<td>5249</td>
<td>cutoff</td>
<td>212.0000</td>
<td>211.0000</td>
<td>416724639</td>
<td>0.47%</td>
<td></td>
</tr>
<tr>
<td>30292000</td>
<td>4407</td>
<td>infeasible</td>
<td>212.0000</td>
<td>211.0000</td>
<td>416730198</td>
<td>0.47%</td>
<td></td>
</tr>
<tr>
<td>30293000</td>
<td>3519</td>
<td>infeasible</td>
<td>212.0000</td>
<td>211.0000</td>
<td>416735118</td>
<td>0.47%</td>
<td></td>
</tr>
<tr>
<td>30294000</td>
<td>2636</td>
<td>cutoff</td>
<td>212.0000</td>
<td>211.0000</td>
<td>416740781</td>
<td>0.47%</td>
<td></td>
</tr>
<tr>
<td>30295000</td>
<td>1758</td>
<td>infeasible</td>
<td>212.0000</td>
<td>211.0000</td>
<td>416746255</td>
<td>0.47%</td>
<td></td>
</tr>
<tr>
<td>30296000</td>
<td>863</td>
<td>infeasible</td>
<td>212.0000</td>
<td>211.0000</td>
<td>416748900</td>
<td>0.47%</td>
<td></td>
</tr>
</tbody>
</table>

Elapsed time = 46415.00 sec. (tree size = 12.94 MB)

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

Robert Fourer, Approaches to Near-Optimally Solving Mixed-Integer Programs
Bixby Workshop, Erlangen, Germany—26-28 September 2010
Roll Cutting

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

- Solve the pattern-choice MIP using either of . . .
  * patterns generated by the Gilmore-Gomory method (for solving the relaxation)
  * all nondominated patterns
**Throwing Out**

**Roll Cutting**

**Cutting model**

```plaintext
set WIDTHS;
param orders {WIDTHS} > 0;
param nPAT integer >= 0;
param nbr {WIDTHS,1..nPAT} integer >= 0;
var Cut {1..nPAT} integer >= 0;

minimize Number:
    sum {j in 1..nPAT} Cut[j];

subject to Fill {i in WIDTHS}:
    sum {j in 1..nPAT} nbr[i,j] * Cut[j] >= orders[i];

# set of widths to be cut
# number of each width to be cut
# number of patterns
# rolls of width i in pattern j
# rolls cut using each pattern
# total raw rolls cut
# for each width,
# rolls cut meet orders
```
Throwing Out

Roll Cutting

Pattern generation model

```plaintext
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;
```
Throwing Out

Roll Cutting

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;
};
```
Throwing Out

Roll Cutting

Pattern enumeration script

```plaintext
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

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 ;
```

Throwing Out

Roll Cutting

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