Assigning People in Practice

Robert Fourer
Department of Industrial Engineering and Management Sciences
Northwestern University
Evanston, IL 60208-3119, U.S.A.
4er@iems.northwestern.edu

CORS/INFORMS International Meeting
Banff, May 16-19, 2004

WC02: Optimization Modelling in Practice II
Outline

Classical assignment
- Assigning professor to offices
- Adjusting the results

Modified assignment
- Assigning students to project groups
- Modeling the complications

“Balanced” assignment
- Tests of formulations using sample data
- Scaling up to full data
**Classical Assignment**

Given
- $P$, a set of people
- $Q$, a set of places
- $c_{ij}$, cost of assigning person $i$ to place $j$

Define
- $X_{ij} = 1$ if person $i$ is assigned to place $j$
  - $= 0$ otherwise

Minimize

$$\sum_{p \in P} \sum_{q \in Q} c_{pq} X_{pq}$$

Subject to

- $\sum_{q \in Q} X_{pq} = 1$, for each $p \in P$
- $\sum_{p \in P} X_{pq} \leq 1$, for each $q \in Q$
- $X_{pq} \geq 0$, for each $p \in P$ and $q \in Q$
... same, but in AMPL

```ampl
set P;  # people
set Q;  # places

param c {P,Q} > 0;

var X {P,Q} binary;

minimize Z: sum {p in P} sum {q in Q} c[p,q] * X[p,q];

subject to P1 {p in P}: sum {q in Q} X[p,q] = 1;
subject to Q1 {q in Q}: sum {p in P} X[p,q] <= 1;
```
... same, but more readable

\[
\begin{align*}
\text{set} & \text{ PEOPLE;} \\
\text{set} & \text{ PLACES;} \\
\text{param pref } \{\text{PEOPLE, PLACES}\} & > 0; \\
\text{var Assign } \{\text{PEOPLE, PLACES}\} & \text{ binary;} \\
\text{minimize TotalPref:} \\
& \quad \text{sum } \{p \text{ in PEOPLE}\} \text{ sum } \{q \text{ in PLACES}\} \text{ pref}[p,q] \times \text{Assign}[p,q]; \\
\text{subj to OnePlacePerPerson } \{p \text{ in PEOPLE}\}: \\
& \quad \text{sum } \{q \text{ in PLACES}\} \text{ Assign}[p,q] = 1; \\
\text{subj to OnePersonPerPlace } \{q \text{ in PLACES}\}: \\
& \quad \text{sum } \{p \text{ in PEOPLE}\} \text{ Assign}[p,q] \leq 1;
\end{align*}
\]
## Data for Professors and Offices

```plaintext
set PEOPLE := Bassok Coullard Frey Hazen Hopp Hurter
             Jones Mehrotra Rieders Rath Rubenstein Spearman
             Sun Tamhane Thompson Zazanis ;

set PLACES := 1021 1049 1053 1055 1083 1087
              2009 2019 2053 2083 2087
              3021 3041 3083 3087
              4083 4087 ;

param pref:  1021 1049 1053 1055 1083 1087 2009 2019 2053 2083 2087 :=
             Bassok       7    7    7    7    7    7    7    6    5
             Coullard    11   14   13   12   16   15   10   11    9    8   7
             Frey         4    4    4    3    4    4    4    1    4    4
             Hazen       17   14   13   12   16   16    6   11    9    7   8
             Hopp        15   16   17    4   10   11    5   12   13    8   9
             Hurter      17   15   14   16   11   10    4   13    9    7   8
             Jones        5    4   14   15   16   17    1   11   10   12  13
             Mehrotra    17   14   15    9    7    8   10   11   12    3    4
             Rieders     12   17   16   15   14   13    7    8   11   10   9
             .......
```
A First Assignment

```
ampl: model offices.mod;
ampl: data offices.dat;
ampl: solve;
MINOS 5.5: optimal solution found.
128 iterations, objective 49
ampl: display Assign;
Assign [*,*] (tr)
# $2 = Coullard
# $6 = Hurter
:    Bassok  '$2'       Frey          Hazen    Hopp  '$6'     Jones :=
 1021    0     0    1             0             0    0   -6.38388e-17
 1049    0     0    0             0             0    0    0
 1053    0     0    0             0             0    0    0
 1055    0     0    0             0             1    0    0
 1083    0     0   -7.20534e-17   0             0    0    0
 1087    0     0    8.21457e-18   0             0    0    0
 2009    0     0    0             0             0    0    0
 2019    1     0    0             0             0    0    0
 2053    0     0   -5.56242e-17   0             0    0    0  .......
```
(displayed legibly)

```
ampl: option display_1col 10000, omit_zero_rows 1;
ampl: option display_eps .000001;

ampl: display {p in PEOPLE, q in PLACES} pref[p,q] * Assign[p,q];
pref[p,q]*Assign[p,q] :=

Bassok  2019  7
Coullard 4083  2
Frey     1021  4
Hazen    3083  3
Hopp     1055  4
Hurter   3041  1
Jones    3021  3
Mehrotra 2083  3
Rath     1053  2
Rieders  3087  3
Rubenstein 1049  1
Spearman 2009  1
Sun      4087  1
Tamhane  1087  7
Thompson 2053  2
Zazanis  2087  5
```
A Seniority-Weighted Assignment

param base >= 1;
param weight {PEOPLE} > 0;
param pref {PEOPLE,PLACES} > 0;
var Assign {PEOPLE,PLACES} binary;

minimize TotalPref:
  sum {p in PEOPLE} base^weight[p] * 
    sum {q in PLACES} pref[p,q] * Assign[p,q];

param base := 10 ;
param weight :=
  Bassok 1   Hopp 3   Rath 4   Sun 2
  Coullard 3 Hurter 4 Rieders 1 Tamhane 4
  Frey 4   Jones 4   Rubenstein 4 Thompson 4
  Hazen 3  Mehrotra 2 Spearman 2  Zazanis 2 ;
(results)

MINOS 5.5: optimal solution found.
128 iterations, objective 102330

ampl: display \{p in \text{PEOPLE}, q in \text{PLACES}\} \text{pref}[p,q] \times \text{Assign}[p,q];
\text{pref}[p,q]\times\text{Assign}[p,q] :=

|         |      |  
|---------|------|----------|
| Bassok  | 1087 | 7        |
| Coullard| 3087 | 3        |
| Frey    | 2053 | 1        |
| Hazen   | 3083 | 3        |
| Hopp    | 1055 | 4        |
| Hurter  | 4083 | 2        |
| Jones   | 2009 | 1        |
| Mehrotra| 1083 | 7        |
| Rath    | 1053 | 2        |
| Rieders | 3021 | 6        |
| Rubenstein| 1049 | 1       |
| Spearman| 2083 | 2        |
| Sun     | 2019 | 8        |
| Tamhane | 4087 | 1        |
| Thompson| 3041 | 1        |
| Zazanis | 2087 | 5        |
A Politically-Sensitive Assignment

set GIVEN within \{PEOPLE, PLACES\};

\ldots\ldots

subj to PoliticalDecisions \{(p,q) \text{ in GIVEN}\}:
\quad Assign[p,q] = 1;

set given := (Rubenstein,1049) (Rath,1053) (Frey,2019);
A More Equitable Assignment

\[
\begin{align*}
\text{param worst integer } & \leq \text{ card } \{\text{PLACES}\}; \\
\ldots & \\
\text{subj to NotTooAwful} \\
\{p \in \text{PEOPLE}, q \in \text{PLACES}: \; \text{pref}[p, q] > \text{worst}\}: \\
\text{Assign}[p, q] & = 0;
\end{align*}
\]

ampl: \texttt{let worst := 7;}
ampl: \texttt{solve;}
MINOS 5.5: optimal solution found.
46 iterations, objective 130830

ampl: \texttt{let worst := 6;}
ampl: \texttt{solve;}
MINOS 5.5: infeasible problem.
4 iterations
Observation #1

Use a small assignment model to generate assignments

Then go with the one you prefer
Observation #2

*Generate the assignments for yourself*

*Announce only the assignment you choose*
Modified Assignment

Given

- Students and projects
- Preferences of students for projects
- Subgroups of students wanting the same project
- List of students who have cars

Assign

- 3 or 4 students per project
- At least one car per project
- Students in each subgroup to the same project

... with preference to students not in subgroups
Student Data

```plaintext
set STU ordered;
param car {STU} binary;

param ngroup integer >= 0;
set GRP = 1..ngroup;

set MEM {GRP} ordered by STU;
    check {g1 in GRP, g2 in g1+1..ngroup}:
        card (MEM[g1] inter MEM[g2]) = 0;

set SAMEGRP = union {g in GRP}
    {s1 in MEM[g], s2 in MEM[g]: ord(s1) < ord(s2)};
```
Project Data

set PRJ;
param cars_needed {PRJ} integer >= 0;
param min_team {PRJ} integer >= 0;
param max_team {p in PRJ} integer >= min_team[p];

param rank {STU,PRJ} integer >= 0, <= card {PRJ};
    check {(s1,s2) in SAMEGRP, p in PRJ}:
        rank[s1,p] = rank[s2,p];
Objective

\begin{verbatim}
var Assign \{STU,PRJ\} binary;

set GROUPED = union \{g in GRP\} MEM[g];

param group_weight \geq 1;

minimize Total_Rank:
    \sum \{s in STU, p in PRJ\} \text{rank}[s,p] \times \text{Assign}[s,p] \times 
    (\text{if } s \text{ in GROUPED then group}_{weight} \text{ else } 1);
\end{verbatim}
General Constraints

subject to Assign_Students {s in STU}:
    sum {p in PRJ} Assign[s,p] = 1;

subject to Assign_Projects {p in PRJ}:
    min_team[p] <= sum {s in STU} Assign[s,p] <= max_team[p];

subject to Enough_Cars {p in PRJ}:
    sum {s in STU} car[s] * Assign[s,p] >= cars_needed[p];

subject to Preserve_Groups {(s1,s2) in SAMEGRP, p in PRJ}:
    Assign[s1,p] = Assign[s2,p];
Ad Hoc Constraints

param cutoff >= 1, <= card {PRJ};
subject to Not_Too_Bad
    {s in STU, p in PRJ: rank[s,p] > cutoff}:
        Assign[s,p] = 0;

set PRJ_PREF within {STU,PRJ};
subject to Project_Preference {(s,p) in PRJ_PREF}:
        Assign[s,p] = 1;
Project and Student Data

param: PRJ: cars_needed min_team max_team :=

  "Ameritech" 1 4 4
  "DSC" 1 4 4
  "Motorola" 1 4 4
  "NMH" 1 4 4
  "S&C Elec" 1 4 4
  "TreeHouse" 1 4 4
  "UPS" 1 4 4;

param: STU: car :=

  Bhandari_Elsa 0
  Black_Andrew 1
  Croke_Michael 0
  Ellis_Mary_Beth 1
  Fernandez_Jason 0
  Friedlander_Jeffrey 1
  Gambell_Anthony 1
  Iwase_Yoshinori 0
  Katen_Philip 1
  .......
Subgroup and Rank Data

<table>
<thead>
<tr>
<th>Group</th>
<th>Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bhandari_Elsa Vargas_Lorena Wise_David</td>
</tr>
<tr>
<td>2</td>
<td>Friedland_Jeffrey Katen_Philip Kemp_Charles Pain_Lucas</td>
</tr>
<tr>
<td>3</td>
<td>Ellis_Mary_Beth Xu_Ping</td>
</tr>
<tr>
<td>4</td>
<td>Kim_Linda Pan_Shaio-Tien Subudhayan_Suppachok Lee_Danny</td>
</tr>
<tr>
<td>5</td>
<td>Gambell_Anthony McCune_Christopher McCune_Jason</td>
</tr>
<tr>
<td>6</td>
<td>Kim_Rita Black_Andrew Shemluck_Matt Fernandez_Jason</td>
</tr>
<tr>
<td>7</td>
<td>Sit_Danny Wang_Jensen</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rank</th>
<th>Companies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;Ameritech&quot; &quot;DSC&quot; &quot;Motorola&quot; &quot;NMH&quot; &quot;S&amp;C Elec&quot; &quot;TreeHouse&quot; &quot;UPS&quot;</td>
</tr>
<tr>
<td>Bhandari_Elsa</td>
<td>1 4 5 6 2 7 3</td>
</tr>
<tr>
<td>Black_Andrew</td>
<td>7 3 2 6 4 5 1</td>
</tr>
<tr>
<td>Croke_Michael</td>
<td>5 1 2 6 3 7 4</td>
</tr>
<tr>
<td>Ellis_Mary_Beth</td>
<td>7 2 1 3 5 6 4</td>
</tr>
<tr>
<td>Fernandez_Jason</td>
<td>7 3 2 6 4 5 1</td>
</tr>
<tr>
<td>Friedlander_Jeffrey</td>
<td>7 2 5 3 4 1 6</td>
</tr>
<tr>
<td>Gambell_Anthony</td>
<td>1 7 6 3 4 2 5</td>
</tr>
<tr>
<td>Iwase_Yoshinori</td>
<td>4 5 1 7 2 6 3</td>
</tr>
</tbody>
</table>
Miscellaneous Data

```
param group_weight 3 ;

param cutoff := 4 ;

set PRJ_PREF := "McCune_Christopher" Ameritech ;
```
Solution

ampl: option display_1col 10000, omit_zero_rows 1;
ampl: option display_eps .000001;

ampl: solve;
MINOS 5.5: optimal solution found.
13 iterations, objective 101

ampl: display {p in PRJ, s in STU} Assign[s,p];
Assign[s,p] :=
Ameritech Bhandari_Elsa 0.333333
Ameritech Gambell_Anthony 1
Ameritech McCune_Christopher 1
Ameritech McCune_Jason 1
Ameritech Vargas_Lorena 0.333333
Ameritech Wise_David 0.333333
DSC Bhandari_Elsa 0.666667
DSC Black_Andrew 0.25
DSC Croke_Michael 1
DSC Fernandez_Jason 0.25
........
(with integer variables)

CPLEX 9.0.0: optimal integer solution; objective 116
22 MIP simplex iterations
0 branch-and-bound nodes

ampl: display {p in PRJ, s in STU} rank[s,p] * Assign[s,p];
rank[s,p]*Assign[s,p] :=
Ameritech Gambell_Anthony 1
Ameritech Iwase_Yoshinori 4
Ameritech McCune_Christopher 1
Ameritech McCune_Jason 1
DSC Bhandari_Elsa 4
DSC Croke_Michael 1
DSC Vargas_Lorena 4
DSC Wise_David 4
NMH King_Nancy 1
NMH Mehawich_Michael 1
NMH Starr_Cathy 1
NMH Terrell_Eric 3
.......
Observation #3

Assignment problems are seldom linear programs

They require discrete optimization technologies
Observation #4

Assignment models make intensive use of sets

Their modeling language formulations make extensive use of set features
“Balanced” 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,
➢ but no one is unduly “isolated”
Plan of Attack

Year 1

- Dump it on a (human) database administrator
- Apply some ad hoc heuristics, by hand

Year 2

- Hire a consultant (me), to:
  - build some optimization models
  - test simple models on small subsets of data
  - scale up to more complex models on the full data

Year 3, 4, 5, . . .

- Re-run with new complications
Minimum “Sameness” Model

set PEOPLE; # individuals to be assigned
set CATEG;
param type {PEOPLE,CATEG} symbolic;
    # categories by which people are classified;
    # type of each person in each category
set SAMETYPE = {i1 in PEOPLE, i2 in PEOPLE diff {i1},
k in CATEG: type[i1,k] = type[i2,k]};
    # set of triples (i1,i2,k) such that individuals
    # i1 and i2 have the same type in category k
param numberGrps integer > 0;
param minInGrp integer > 0;
param maxInGrp integer >= minInGrp;
    # number of groups; bounds on size of groups
(quadratic objective)

```plaintext
var Assign {i in PEOPLE, j in 1..numberGrps} binary;

# Assign[i,j] is 1 if and only if
# person i is assigned to group j

minimize TotalSameness:
    sum {(i1,i2,k) in SAMETYPE, j in 1..numberGrps} Assign[i1,j] * Assign[i2,j];

# Product of variables is 1 iff both are 1

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

# Each person assigned to one group

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

# Each group has an acceptable size
```
(linearized objectives)

Simple Linearization

minimize TotalSameness:
  \[ \sum_{j \in GRP} \sum_{(i1,i2,k) \in SAMETYPE} \text{Same}[i1,i2,j]; \]
subj to SameDefn
  \{i1 \in PEOPLE, i2 \in PEOPLE, j \in 1..\text{numberGrps}\}:
  \text{Same}[i1,i2,j] \geq \text{Assign}[i1,j] + \text{Assign}[i2,j] - 1;

Concise Linearization

minimize TotalSameness:
  \[ \sum_{j \in GRP} \sum_{i \in PEOPLE} \text{Sameness}[i,j]; \]
subj to SamenessDefn \{i \in PEOPLE, j \in GRP\}:
  \text{Sameness}[i,j] \geq \sum_{(i,i2,k) \in SAMETYPE} \text{Assign}[i2,j]
  \quad - \text{maxSameness} \times (1 - \text{Assign}[i,j]);
Solving as Continuous Quadratic

100 people, 10 groups

ampl: solve;

1000 variables, all nonlinear
110 constraints, all linear; 2000 nonzeros
1 nonlinear objective; 1000 linear nonzeros.

MINOS 5.4: ignoring integrality of 1000 variables

MINOS times:
read: 11.35
solve: 279.73 excluding minos setup: 279.67
write: 0.02
total: 291.10

MINOS 5.4: optimal solution found.
349 iterations, objective 1744

. . . all variables turn out integer !!!
Solving as Continuous Quadratic

100 people, 10 groups (more recent run)

```
AMPL: solve;
1000 variables, all nonlinear
110 constraints, all linear; 2000 nonzeros
1 nonlinear objective; 1000 linear nonzeros.
MINOS 5.5: ignoring integrality of 1000 variables
MINOS times:
read: 0.29
solve: 3.10 excluding minos setup: 3.10
write: 0.00
total: 3.39
MINOS 5.5: optimal solution found.
279 iterations, objective 1714

... all variables turn out integer !!!
```
Solving as Integer Quadratic

ampl: solve;

1000 variables, all nonlinear
110 constraints, all linear; 2000 nonzeros
1 nonlinear objective; 1000 nonzeros.

.........

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>73900</td>
<td>73196</td>
<td>1573.1903</td>
<td>383</td>
<td>1724.0000</td>
<td>1343.6630</td>
<td>454152</td>
<td>22.06%</td>
</tr>
<tr>
<td>74000</td>
<td>73296</td>
<td>1695.0051</td>
<td>119</td>
<td>1724.0000</td>
<td>1343.6630</td>
<td>455487</td>
<td>22.06%</td>
</tr>
<tr>
<td>* 74000+72612</td>
<td>0</td>
<td>1720.0000</td>
<td>1343.6630</td>
<td>455487</td>
<td>21.88%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Times (seconds):
Input = 0.981
Solve = 6458.19
Output = 0.411

CPLEX 9.0.0: feasible integer solution; objective 1720
455487 MIP simplex iterations
74000 branch-and-bound nodes

... is this convex ???
**Solving the Simple Linearization**

ampl: solve;

96520 variables:
  1000 binary variables
  95520 linear variables
95630 constraints, all linear; 288560 nonzeros
1 linear objective; 95520 nonzeros.

CPLEX 3.0:

....... 

... *wait forever with no solution* !!!
Solving the Concise Linearization

ampl: solve;

2000 variables:
  1000 binary variables
  1000 linear variables
1110 constraints, all linear; 99520 nonzeros
1 linear objective; 1000 nonzeros.

CPLEX 3.0:
No MIP presolve or aggregator reductions.
Elapsed time = 30.30 sec.

........

... now branch-and-bound begins →
(continued)

<table>
<thead>
<tr>
<th>Node</th>
<th>Left</th>
<th>Objective</th>
<th>IInf</th>
<th>Best Integer</th>
<th>Best Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.0000</td>
<td>274</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>78.0862</td>
<td>300</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>123.6578</td>
<td>288</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>208.7162</td>
<td>271</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>348.8889</td>
<td>241</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>447.9946</td>
<td>219</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>260</td>
<td>260</td>
<td>1416.4902</td>
<td>53</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>280</td>
<td>280</td>
<td>1561.0237</td>
<td>34</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>300</td>
<td>1757.6146</td>
<td>8</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>305</td>
<td>1792.0000</td>
<td>0</td>
<td>1792.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>320</td>
<td>316</td>
<td>32.0996</td>
<td>310</td>
<td>1792.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

... continues for a long time with no improvement
Applying a Greedy Heuristic

```ampl
param conflict; param min_conflict; param min_group;
for {p in PEOPLE} {
    let min_conflict := Infinity;
    for {j in 1..numberGrps} {
        let conflict := sum {(p,i,k) in SAMETYPE} Assign[i,j];
        if conflict < min_conflict then {
            let min_conflict := conflict;
            let min_group := j;
        }
    }
    let Assign[p,min_group] := 1;
}

ampl: include balAssignGreedy.run;
TotalSameness = 1762
```
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.
(variables and objective)

```plaintext
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
```
(constraints)

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
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>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>0</td>
<td>0</td>
<td>17.0000</td>
<td>299</td>
<td>17.0000</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>17.0000</td>
<td>322</td>
<td>17.0000</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>17.0000</td>
<td>332</td>
<td>17.0000</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>17.0000</td>
<td>328</td>
<td>17.0000</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>17.0000</td>
<td>329</td>
<td>17.0000</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>17.0000</td>
<td>329</td>
<td>17.0000</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>17.0000</td>
<td>339</td>
<td>17.0000</td>
</tr>
<tr>
<td>70</td>
<td>70</td>
<td>17.0000</td>
<td>344</td>
<td>17.0000</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>17.0000</td>
<td>342</td>
<td>17.0000</td>
</tr>
</tbody>
</table>

........
### (continued)

<table>
<thead>
<tr>
<th>Nodes</th>
<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>250</td>
<td>250</td>
<td>250</td>
<td>43.6818</td>
<td>74</td>
<td>17.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>260</td>
<td>260</td>
<td>260</td>
<td>46.5000</td>
<td>58</td>
<td>17.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>*</td>
<td>265</td>
<td>263</td>
<td>47.0000</td>
<td>0</td>
<td>47.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>270</td>
<td>266</td>
<td>266</td>
<td>17.0000</td>
<td>314</td>
<td>47.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>280</td>
<td>276</td>
<td>266</td>
<td>17.0000</td>
<td>351</td>
<td>47.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>290</td>
<td>286</td>
<td>266</td>
<td>17.0000</td>
<td>340</td>
<td>47.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>300</td>
<td>296</td>
<td>286</td>
<td>17.0000</td>
<td>337</td>
<td>47.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>310</td>
<td>306</td>
<td>296</td>
<td>17.0000</td>
<td>341</td>
<td>47.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>....</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>630</td>
<td>609</td>
<td>609</td>
<td>21.5208</td>
<td>243</td>
<td>47.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>640</td>
<td>618</td>
<td>618</td>
<td>23.3028</td>
<td>244</td>
<td>47.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>650</td>
<td>626</td>
<td>626</td>
<td>17.3796</td>
<td>269</td>
<td>47.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>660</td>
<td>636</td>
<td>636</td>
<td>17.7981</td>
<td>271</td>
<td>47.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>*</td>
<td>666</td>
<td>440</td>
<td>19.0000</td>
<td>0</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>670</td>
<td>440</td>
<td>440</td>
<td>17.0000</td>
<td>147</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>680</td>
<td>446</td>
<td>446</td>
<td>17.0714</td>
<td>213</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>690</td>
<td>454</td>
<td>454</td>
<td>17.5000</td>
<td>186</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
</tbody>
</table>
(concluded)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Nodes</th>
<th>Objective</th>
<th>IInf</th>
<th>Best Integer</th>
<th>Cuts/</th>
<th>Best Node</th>
<th>Times (seconds):</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Input = 0.266667</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Solve = 864.733</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Output = 0.166667</td>
</tr>
<tr>
<td>Node</td>
<td>Left</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>700</td>
<td>461</td>
<td>17.1364</td>
<td>268</td>
<td>19.0000</td>
<td>17.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>710</td>
<td>468</td>
<td>17.3117</td>
<td>267</td>
<td>19.0000</td>
<td>17.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>720</td>
<td>475</td>
<td>17.0000</td>
<td>211</td>
<td>19.0000</td>
<td>17.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>730</td>
<td>484</td>
<td>17.2652</td>
<td>226</td>
<td>19.0000</td>
<td>17.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>740</td>
<td>490</td>
<td>17.0000</td>
<td>106</td>
<td>19.0000</td>
<td>17.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>750</td>
<td>497</td>
<td>17.0000</td>
<td>24</td>
<td>19.0000</td>
<td>17.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 752</td>
<td>0</td>
<td>17.0000</td>
<td>0</td>
<td>17.0000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CPLEX 3.0: optimal integer solution; objective 17
45621 simplex iterations
752 branch-and-bound nodes
Solving for Minimum Variation

1054 variables:
- 1000 binary variables
- 54 linear variables

560 constraints, all linear; 12200 nonzeros
1 linear objective; 54 nonzeros.

CPLEX 9.0.0:
Clique table members: 100
MIP emphasis: balance optimality and feasibility

<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>0</td>
<td>17.0000</td>
<td>212</td>
<td></td>
<td>17.0000</td>
</tr>
<tr>
<td>1</td>
<td>17.0000</td>
<td>228</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>17.0000</td>
<td>0</td>
<td>19.0000</td>
<td>Fract: 49</td>
</tr>
<tr>
<td>1</td>
<td>17.0000</td>
<td>183</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>2</td>
<td>17.0000</td>
<td>146</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>3</td>
<td>17.0000</td>
<td>170</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>4</td>
<td>17.0000</td>
<td>153</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>5</td>
<td>17.0000</td>
<td>103</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>6</td>
<td>17.0000</td>
<td>81</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>7</td>
<td>17.0000</td>
<td>72</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
</tbody>
</table>
Solving for Minimum Variation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>17.0000</th>
<th></th>
<th>19.0000</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
<td>17.0000</td>
<td>88</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>17.0000</td>
<td>69</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>17.0000</td>
<td>69</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>17.0000</td>
<td>77</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>17.0000</td>
<td>73</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>17.0000</td>
<td>68</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>17.0000</td>
<td>99</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>17.0000</td>
<td>98</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>17.0000</td>
<td>128</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
<td>17.0000</td>
<td>140</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>17.0000</td>
<td>95</td>
<td>19.0000</td>
<td>17.0000</td>
</tr>
<tr>
<td>*</td>
<td>19</td>
<td></td>
<td></td>
<td>17.0000</td>
<td></td>
</tr>
</tbody>
</table>

Gomory fractional cuts applied: 10

Times (seconds):
Input = 0.02
Solve = 16.844
Output = 0.02

CPLEX 9.0.0: optimal integer solution; objective 17
5624 MIP simplex iterations
19 branch-and-bound nodes
## Summary of Results

<table>
<thead>
<tr>
<th></th>
<th>Total same-ness</th>
<th>Max variation</th>
<th>Total variation</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greedy</td>
<td>1762</td>
<td>3</td>
<td>45</td>
<td>45 seconds</td>
</tr>
<tr>
<td>Quadratic</td>
<td>1744</td>
<td>2</td>
<td>39</td>
<td>4.7 min</td>
</tr>
<tr>
<td>Min total variation</td>
<td>1706</td>
<td>1</td>
<td>17</td>
<td>14.4 min</td>
</tr>
<tr>
<td><strong>New</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quadr continuous</td>
<td>1752</td>
<td>2</td>
<td>44</td>
<td>4.07 sec</td>
</tr>
<tr>
<td>Quadr integer</td>
<td>1720</td>
<td>2</td>
<td>27</td>
<td>107 min</td>
</tr>
<tr>
<td>Min total variation</td>
<td>1706</td>
<td>1</td>
<td>17</td>
<td>16.8 sec</td>
</tr>
</tbody>
</table>
Scaling Up

_Model is 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 is 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 “Location-Rank” 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});
(variables)

\[
\text{param frac2title \{i1 in TITLE}\}
\quad = \frac{\text{sum \{(i1,i2) in TYPE2\} number2[i1,i2]}}{\text{card \{PEOPLE\}}};
\]

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

\[
\text{param expDine \{j in REST\}
\quad = \begin{cases} 
\text{loDine}[j] & \text{if } \text{loFudge} > 0 \\
\text{hiDine}[j] & \text{if } \text{hiFudge} > 0 \\
\frac{\text{loDine}[j] + \text{hiDine}[j]}{2} & \text{otherwise}
\end{cases};
\]

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

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

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

\[
\text{param hiTargetLoc \{i2 in LOC, j in REST\} :=
\quad \text{ceil (round (frac2loc[i2] * expDine[j], 6))};}
\]
(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;
```
(constraints to define “variation”)

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

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

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

subject to Hi2LocDefn \{i2 \in LOC, j \in REST\}:
   \text{Dev2Loc}[i2] \geq \sum \{(i1,i2) \in \text{TYPE2}\} \text{Assign2}[i1,i2,j] - \text{hiTargetLoc}[i2,j];
(parameters for ruling out “isolation”)

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]);
(constraints to rule out “isolation”)

```
var Lone {(i1,i2) in ISO, j in REST} binary;

subj to Isolation1 {(i1,i2) in ISO, j in REST}:
  Assign2[i1,i2,j] <= upperbnd[i1,i2,j] * Lone[i1,i2,j];

subj to Isolation2a {(i1,i2) in ISO, j in REST}:
  Assign2[i1,i2,j] +
    sum {ii1 in ADJACENT[i1]: (ii1,i2) in TYPE2} Assign2[ii1,i2,j]
  >= 2 * Lone[i1,i2,j];

subj to Isolation2b {(i1,i2) in ISO, j in REST}:
  Assign2[i1,i2,j] >= Lone[i1,i2,j];
```
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
Observation #5

Clients can invent and change the rules as they wish

Assignment of people is a social, not physical, problem
“Oh, we forgot to mention . . .”

One more complication

- No group may have only 1 woman

Not a problem, though

- Women are between 18% and 22% of every group in solution already sent!

. . . client’s ad hoc solutions must have been pretty bad
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
More Recent Cases

*Balanced series of assignments*

*Sequence of workshop assignments*

*Balanced class seat assignments*
Observation #6

Subsequent problems may get harder
But they may just as well get easier