Scheduling Doctors to Clinical and Surgical Time Slots: A Column Generation Approach

Craig Froehle and Michael Magazine
University of Cincinnati
Kipp Martin
University of Chicago

With help from:
Linda Kromer
Omkar Saha
Cary Wise
University of Cincinnati

October 11, 2009
Outline

Problem Introduction

System Architecture

Key Concepts

Model Formulation

Column Generation
The Problem

Develop an application to take:

- Requests for surgical spaces (a cardiology surgery, at Location 1, in the AM, two days a week, two weeks per planning horizon)
- Requests for clinical spaces (an ENT clinic, at Location 2, all day, Monday and Wednesday, every week)
- Doctor requests (I play golf every Tuesday afternoon)

Find a schedule:

- Match available surgery and clinic rooms at the various locations with the space requests
- Match doctor requests as closely as possible to the space request assignments
The Problem

- Administrative assistants for each specialty need to sit at their desktop machines and enter the space and doctor request information for the given specialty.
- The requests across all specialties must be gathered and stored somewhere.
- An optimization model must be built and solved that finds a feasible allocation of space requests to available rooms and doctors to meet the requests.
- The results need to be returned to the desktops of each administrative assistant in order to schedule the doctors.
- The system must be run and used every month by people that know nothing about optimization.
Figure: Entering the Data.
Figure: Calling the Scheduling Application
System Architecture

- Excel (Client User Form)
- Visual Basic (Create SQL requests from User Form)
- MySQL (Relational Data Base)
- COIN-OR Bcp Solver
- COIN-OR OS
- HTTP / Tomcat
- JDBC (The database drivers)
- Java (Web Services)
Key Concepts

▶ **Space**: a room or set of rooms (pod) that can service either a clinic or surgery visit – each space has a capacity

▶ **Space Request**: a specification of

  ▶ location
  ▶ type – surgery or clinic
  ▶ specialty
  ▶ weeks
  ▶ days
  ▶ shifts (AM or PM)
  ▶ number of spaces
Key Concepts

Assignment: A complete specification weeks, days, shifts, and spaces that satisfy a request for a clinic or surgery space at a location

<table>
<thead>
<tr>
<th>Request</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liberty</td>
<td>Liberty</td>
</tr>
<tr>
<td>Surgery</td>
<td>Liberty</td>
</tr>
<tr>
<td>ENT</td>
<td>ENT</td>
</tr>
<tr>
<td>one week</td>
<td>second week</td>
</tr>
<tr>
<td>two days</td>
<td>Wed, Fri</td>
</tr>
<tr>
<td>either shift</td>
<td>AM (Wed), PM (Fri)</td>
</tr>
<tr>
<td>2 spaces</td>
<td>Room 212</td>
</tr>
</tbody>
</table>

Request Slot: a specification of location, week, day, shift, space(room) ID, request type (surgery or clinic). As assignment is a specification request slots.
Example of Request and Request Slots

A request:

ENT, Surgical Space, Liberty, A.M., 3 days, 2 weeks

This request generates an assignment that specifies the following request slots:

ENT, Liberty, Surgical, Storz, Week 1, M, AM
ENT, Liberty, Surgical, Storz, Week 1, W, AM
ENT, Liberty, Surgical, Storz, Week 1, F, AM
ENT, Liberty, Surgical, Stryker, Week 2, M, AM
ENT, Liberty, Surgical, Stryker, Week 2, W, AM
ENT, Liberty, Surgical, Stryker, Week 2, F, AM

Note: We are not worried about doctors right now.
Model Formulation

Indexes:

- $i$ – index of requests
- $j$ – index for assignments
- $k$ – index of request slots

Parameters:

- $n$ – number of requests
- $m$ – number of slots
- $d$ – number of dummy slots (every location has a dummy surgery room and a dummy clinic room)
- $S_i$ – an index set of assignments for request $i$
- $c_k$ – capacity of room in slot $k$
- $a_{ijk}$ – number of spaces allocated to request slot $k$ in assignment $j$ of request $i$
Model Formulation – Space Requests Only

Variables:

- \( x_{ij} \) – 1 if assignment \( j \) is selected for request \( i \)
- \( z_k \) – number of spaces allocated to dummy slot \( k \), each location has a “dummy” space for each time slot (day, week, shift).

\[
\begin{align*}
\text{min} & \quad \sum_k z_k \\
\text{s.t.} & \quad \sum_{i=1}^{n} \sum_{j \in S_i} a_{ijk} x_{ij} \leq c_k, \quad k = 1, \ldots, m \\
& \quad \sum_{i=1}^{n} \sum_{j \in S_i} a_{ijk} x_{ij} \leq z_k, \quad k = m + 1, \ldots, m + d \\
& \quad \sum_{j \in S_i} x_{ij} = 1, \quad i = 1, \ldots, n \\
& \quad x_{ij} \in \{0, 1\}, \quad j \in S_i, \quad i = 1, \ldots, n
\end{align*}
\]
Key Doc Concept 1 – a **doc slot**: a specification of location, week, day, shift, request type (surgery or clinic)

**Example Doc Slot**: Liberty, clinic, second week, Tuesday, AM, cardiology

A *doc slot* is like a *request slot* except that it **does not** have a room specification, but **does have** a specialty.
Slot Summary

Request Slot:

- week
- day
- shift
- location
- clinic or surgery
- room

Doc Slot:

- week
- day
- shift
- location
- clinic or surgery
- specialty
Slot Summary

Slots Continued:
Time Slot:

- week
- day
- shift
Model Formulation – Now Add Docs

Key Doc Concept: a **doc feasibility vector**: a list of indexes of **doc slots** a doctor *is available* to fill. Note: The doc slot might correspond to clinic, it might correspond to surgery.

Each doc has a feasibility vector.

The feasible vector is found for each doc by looping over the doc requests for that doc.
Model Formulation – Building Doc Feasibility Vector

- Education
- Research
- Academic
- Unavailable (specific location or all)
- Specific Clinic
- Specific Surgery

So doc requests have two flavors: when the doc cannot do a clinic or surgery (education, research, academic, unavailable) and when the doc would like to do either a clinic or surgery.
Model Formulation – – Building Doc Feasibility Vector

For Specific Clinic Request Any Location – eliminate all doc slots for surgery in that time slot at every location. Do not eliminate any doc slots for clinics in any location and makes these variables more desirable in the objective function.

For Specific Clinic Request Specific Location – eliminate all doc slots for surgery in that time slot and every location. Eliminate all doc slots for clinics in locations other than the specified location.

Do a similar thing for Specific Surgery Requests.
Model Formulation – With Docs

Additional Indexes:

\( h \) – index of docs, \( h = 1, \ldots, q \)

\( l \) – index of doc slots, \( l = 1, \ldots, p \)

Additional Parameters:

\( q \) – number of docs

\( p \) – number of doc slots

\( b_{hl} \) is 1 if doc slot \( l \) is an index in the doc \( h \) feasibility vector

(Actually we do not define a variable \( y_{hl} \) when \( b_{hl} = 0 \))

\( \theta_l \) is an index set of (specialty, request slot) pairs that map onto
doc slot \( l \), \( l = 1, \ldots, p \)
Example: (Specialty, Request slot) pairs

(ENT, Liberty, surgery, second week, Tuesday, AM, Pod 1)
(ENT, Liberty, surgery, second week, Tuesday, AM, Pod 2)

map onto doc slot

(ENT, Liberty, surgery, second week, Tuesday, AM)

implemented using C++ STL Container Map.
So for each assignment $j$ in request $i$ (the request determines the specialty) we sum up over all the ($specialty$, $k$) pairs that map to doc slot $l$

$$\alpha_{ijl} = \sum_{k \in \theta_l} a_{ijk}$$
Model Formulation – With Docs

Additional Variables:

\( y_{hl} \) – 1 if doc \( h \) assigned \( doc \ slot \ l \), 0 otherwise

Additional Constraints:

\[
\sum_{i=1}^{n} \sum_{j \in S_i} \alpha_{ijl} x_{ij} \leq \sum_{h=1}^{q} b_{hl} y_{hl}, \quad l = 1, \ldots, p
\]  

(6)
Finally, one more a constraint. In each *time slot* a doc can only have assignment. Let *t* index time slots.

**Additional Constraints:**

\[
\sum_{l=1}^{p} d_{tl} y_{hl} \leq 1, \quad t = 1, \ldots, T, \quad h = 1, \ldots, q
\]  

(7)

where \(d_{tl} = 1\) if time slot \(t\) coincides with *doc slot* \(l\).

For example, the *doc slot* 

ENT, Liberty, clinic, second week, Tuesday, AM

corresponds with *time slot*

second week, Tuesday, AM
Model Formulation – With Docs

Critical ideas:

1. Do NOT solve (1)-(7) for all columns.

2. Solve a restriction of (1)-(7).

3. Generate all $y$ variables up front

4. Generate $x$ variables with column generation
Column Generation

The are less than 100 thousand rows. Not a big deal.

What about variables? Let’s just think about a single request and how many assignments might be possible to satisfy the single request.

The request is for either shift, every day, every week and there are two rooms available at the requested location that can handle this request. How many columns do we need in the model to handle this?

There are 25 days in the planning horizon and each day there are 4 choices (2 shifts and 2 room). The number of assignments is

\[ 4^{25} = 2^{50} \approx 10^{15} \]
CCH – Number of Assignments

Let:

- $m$ – number of weeks requested
- $n$ – number of days requested
- $s$ – number of spaces that satisfy the request

\[
\left(\frac{5!}{m!(5-m)!}\right) \left(\frac{5!}{n!(5-n)!}\right)^m (2^n)^m (s^n)^m
\]

A request for either shift, every day, every week with two rooms available at the requested location that can handle the request corresponds to $m = 5$, $n = 5$, and $s = 2$. 
Column Generation

The column generation makes heavy use of object oriented programming. We have a class for each dimension:

- Location class
- Request Type class (surgery or clinic)
- Week class
- Day class
- Room class
- Shift class (AM or PM or Both)

We organize these classes in a hierarchy.
Column Generation

The Class structure used for pricing:

```
CapacityConstraint Class
constraint_cost(SpaceRequest *req, double *dual)

Location Class
location_cost(SpaceRequest *req, double *dual)

RequestType Class
requesttype_cost(SpaceRequest *req, double *dual)

Week Class
week_cost(SpaceRequest *req, double *dual)

Day Class
day_cost(SpaceRequest *req, double *dual)

Room Class
room_cost(SpaceRequest *req, double *dual)

Shift Class
shift_cost(SpaceRequest *req, double *dual)
```
Amount of work: Recall pricing a request for a location that has two rooms that can handle the request. The request is for either shift, every day, every week and would require pricing out $2^{50}$.

$$4 + 4 + 4 + ... + 4 + 4 = 4 \times 25 = 100 \text{ operations}$$

Order Matters: Working with these objects is like working with Lego blocks. I can put them together different ways. For example, in the current setup, I can schedule an ortho surgery in a different room at Liberty every day of the week. What if I want surgery in the same room every day?
The following assignments:

Assignment 776, Liberty, Surgery, Stryker, 1, M, AM, 1
Assignment 776, Liberty, Surgery, Storz, 1, T, AM, 1
Assignment 777, Liberty, Surgery, Storz, 1, M, AM, 1
Assignment 777, Liberty, Surgery, Stryker, 1, T, AM, 1

satisfy the requests:

Request 776, Ortho, Surgery, Liberty, AM, M, T, Week 1
Request 777, ENT, Surgery, Liberty, AM, M, T, Week 1
The following assignments:

Assignment 776, Liberty, Surgery, Stryker, 1, M, AM, 1
Assignment 776, Liberty, Surgery, Storz, 1, T, AM, 1
Assignment 777, Liberty, Surgery, Storz, 1, M, AM, 1
Assignment 777, Liberty, Surgery, Stryker, 1, T, AM, 1

are not allowed for the requests:

Request 776, Ortho, Surgery, Liberty, AM, M, T, Week 1
Request 777, ENT, Surgery, Liberty, AM, M, T, Week 1
### Column Generation Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Potential Columns</th>
<th>Root Node Columns</th>
<th>Total Generated</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9,154</td>
<td>62</td>
<td>172</td>
<td>117</td>
</tr>
<tr>
<td>2</td>
<td>9,893</td>
<td>29</td>
<td>145</td>
<td>123</td>
</tr>
<tr>
<td>3</td>
<td>20,978,496,312</td>
<td>502</td>
<td>3,092</td>
<td>1,341</td>
</tr>
</tbody>
</table>

**A Key Takeaway:** With Bcp you have **CONTROL** over the branch-and-bound tree in terms of adding cuts or columns.