1 Introduction

Staff scheduling in healthcare continues to be a vexing problem that consumes valuable managerial resources and whose complexity can impact the quality of care delivered and the job satisfaction of its highly skilled labor force. []

Wide range of staff scheduling problems whose details depend on the specific healthcare delivery subsystem we are talking about. Examples: recovery room nurses, floor nurses, lab technicians, pharmacists and pharmacy technicians, surgical techs, ED nurses and techs, transporters.

Introduce motivation for tactical tour scheduling problems via real projects. Value of being able to produce realistic multi-week schedules to illustrate how new scheduling practices might actually work.

For many systems, the kernel of the problem is a tour scheduling problem in which staffing levels needed vary both by time of day and day of week.

Introduce project website containing Pyomo implementation of this model, motivation for doing this, relationship to OBSched

2 Tour scheduling problems

2.1 Scheduling environment

Features of tour scheduling problems including notions of planning cycle, periods, required staffing levels, shifts, tours, tour types,

In [**?**], intra-tour start time flexibility was implicitly modeled using start time windows and associated control constraints. That same approach can be used in this multi-week model. However, it significantly complicates notation to include start time windows and since the focus of this paper is on multi-week modeling, we will restrict our attention to the case of no intra-tour start time flexibility. The Pyomo implementation available at the project website includes intra-tour start time flexibility.

To explicitly represent a tour we need of list of days worked, including the shift start times and lengths over the planning cycle. For example, consider the following very simple scenario for a two week planning cycle:

- only allowable tour type calls for five eight-hour shifts per week, M-F,
- within each tour, all shifts must start at 7a or they must all start at 3p,
- no restrictions on the number of weekend days worked over the planning cycle.

In this case, there are only two possible tours:

	Week 1							Week 2						
Tour Variable	Su	M	T	W	U	F	Sa	Su	M	T	W	U	F	Sa
x_1	x	7a	7a	7a	7a	7a	x	x	7a	7a	7a	7a	7a	x
x_2	x	3p	3p	3p	3p	3p	x	x	3p	3p	3p	3p	3p	x

In this case, the solution to the tour scheduling problem is fully specified by the number of people working Tour 1 and the number working Tour 2 and we only need two tour variables (x_1 and x_2) in our model. Of course, the number of tour variables can become enormous as we increase our scheduling options such as additional tour types, shift start times and shift lengths. We will explore this issue in the next section. Here, we want to show that there are other ways we could model this very simple scenario in terms of the variables we choose to use. For example, instead of explicit tour variables, we could use one variable to represent the number of people working a specific days worked pattern and another to represent the number of people working each shift. This is called an *implicitmodeling* approach [].

	Week 1							Week 2						
Days Worked Variable	Su	M	T	W	U	F	Sa	Su	M	T	W	U	F	Sa
y_1	x	On	On	On	On	On	x	x	On	On	On	On	On	x

Shift Variable	Shift Start Time
s_1	7a
s_2	3p

To specify a valid solution to our tour scheduling problem, we need values for s_1, s_2 and y_1. For this toy problem, it is easy to see the correspondence between variables in the two approaches. For given values of x_1 and x_2, $y_1 = x_1 + x_2$, $s_1 = x_1$, and $s_2 = x_2$. While the explicit approach only used two variables, the implicit approach required three variables. However, as we start to model more realistic problems, it will become clear that the implicit approach requires far fewer variables than its explicit counterpart. This reduction will come at a cost in terms of additional constraints as well as in the need for a tour construction post-processing phase. In terms of additional constraints, note that in the implicit approach we must ensure that $y_1 = s_1 + s_2$ if we are going to be able to translate a solution to the implicit problem to an explicit set of tours.

2.2 Implicit modeling of tours

Implicit modeling for tour scheduling problems is not new.

Review one week implicit tour scheduling models including my AOR 2004 paper.

Recent multi-week implicit tour scheduling paper

Also review implicit modeling of breaks and other stuff

2.3 Problem size explosion

3 Related Work

If I haven't already covered it above

4 Overview of an Implicit Multi-Week Model

As depressingly illustrated in Section 2.3, multi-week tour scheduling problems can get massively large. Since even the number of explicit multi-week days worked patterns can get large as the number of weeks increases, our approach is to model these days worked patterns implicitly by focusing on only explicitly modeling the patterns of weekend days worked. This is driven by the observation that it is often the weekends that must be treated in some special way due to the general undesirability of working on weekends as well as the often different required staffing levels seen on weekends. While drastically reducing the total number of variables required, we need to introduce several additional types of variables as well as many control constraints to ensure that a solution to an instance of our implicit model corresponds with a feasible and optimal solution to the equivalent tour scheduling model.

5 Definitions and notation

The notation and definitions presented in this section, are very closely aligned with the implementation of the model in Pyomo.

Planning cycle related terms...

Planning cycle parameters

$$\begin{aligned} n_W &= \text{number of weeks in the planning cycle} \\ n_P &= \text{number of periods per day} \\ n_C &= 7 n_W n_P \text{ number of periods in planning cycle} \end{aligned}$$

Define a number of sets of indices.

$$\begin{aligned} \mathbb{P} &= \{1,2,\dots n_P\} \\ \mathbb{D} &= \{1,2,\dots 7\} \\ \mathbb{W} &= \{1,2,\dots n_W\} \end{aligned}$$

So, each period in the planning cycle is defined by a tuple $(i,d,w) \in \mathbb{B}$ where $\mathbb{B} = \mathbb{P} \times \mathbb{D} \times \mathbb{W}$.

Shift length and tour type parameters

$$\begin{aligned} n_K &= \text{number of different shift lengths} \\ \mathbb{K} &= \{1,2,\dots n_K\} \\ l_k &= k\text{'th shift length in periods, for } k \in \mathbb{K} \end{aligned}$$

$$\begin{aligned} n_T &= \text{number of different tour types} \\ \mathbb{T} &= \{1,2,\dots n_T\} \\ L(i) &= \text{set of shift length indices allowed for tour type } i\text{, for } i \in \mathbb{T} \end{aligned}$$

$$\mathrm{A}_{ijkt} = \begin{cases} 1 & \text{1 if a shift of length } k \text{ starting in period } i \text{ of day } j \text{ for tour type } t \text{ is allowed,} \\ & \text{for } i \in \mathbb{P},\ j \in \mathbb{D},\ k \in \mathbb{K},\ t \in \mathbb{T}. \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbb{A} = \{(i,j,k,t) | A_{ijkt} = 1\}$$

is the set of *allowable shift start times*.

Shift and tour type variables

Instead of explicitly representing tours with one variable per tour, we implicitly represent tours using a number of building block variables whose values are coordinated by numerous control constraints to ensure that valid tours can be constructed from these variables.

Shift variables represent a single shift with a start day and time, shift length, in a specific week and corresponding to a certain tour type.

S_{ijkwt} = number of shifts of length k starting in period i of day j in week w for tour type t, for $(i,j,k,t) \in \mathbb{A}, t \in \mathbb{T}$.

Tour type variables represent the number of people assigned to each tour type in each start time period.

T_{it} = number of tours of tour type t assigned to start time period i for $i \in \mathbb{P}, t \in \mathbb{T}$.

Daily tour type variables represent the number of people assigned to each tour type in each start time period and working on a given day in a given week. Note that these variables are not shift length specific.

DT_{itjw} = number of tours of tour type t assigned to start time period i and working day j in week w, for $i \in \mathbb{P}, j \in \mathbb{D}, t \in \mathbb{T}, w \in \mathbb{W}$.

Daily shift tour type variables represent the number of people assigned to each tour type in each start time period and working a shift of a given length on a given day in a given week.

DST_{itkjw} = number of tours of tour type t assigned to start time period i and working shift length k on day j in week w, for $i \in \mathbb{P}, t \in \mathbb{T}, kt \in \mathbb{K}, j \in \mathbb{D}, w \in \mathbb{W}$.

```
def DailyShiftWorked_index_rule(M):
    return [(i,t,k,j,w) for i in M.WINDOWS
                        for t in M.activeTT
                        for k in M.tt_length_x[t]
                        for j in M.DAYS
                        for w in M.WEEKS
                        if (i,t,j) in M.okDailyTourType]



model_phase1.DailyShiftWorked_index = Set(dimen=5,initialize=DailyShiftWorked_index_rule)


model_phase1.DailyShiftWorked = Var(model_phase1.DailyShiftWorked_index, within=NonNegativel

model_phase1.okDailyTourType = model_phase1.okTourType * model_phase1.DAYS


#    /* DailyTourType[i,t,d] Number of employees working tour type t
#       starting in window i and working day d in week w*/

def DailyTourType_index_rule(M):
    return [(i,t,j,w) for i in M.WINDOWS
                      for t in M.activeTT
                      for j in M.DAYS
                      for w in M.WEEKS
                      if (i,t,j) in M.okDailyTourType]



model_phase1.DailyTourType_index = Set(dimen=4,initialize=DailyTourType_index_rule)

model_phase1.DailyTourType = Var(model_phase1.DailyTourType_index, within=NonNegativeInteger
```

To smooth the postpartum occupancy across days of the week, we introduce two, non-negative, dummy variables, M^U and M^L. These are used to bound the mean occupancy across the week from above and below – see Equation (2). Then we minimize the difference between the bounds.

Model: OB-SMOOTH
Minimize

$$M^U - M^L \tag{1}$$

Subject to

$$0 \leq M^L \leq \mu^4_{i,j} \leq M^U \quad (i = 1,2,\ldots P,\ j = 1,2,\ldots 7) \tag{2}$$

$$L^a_{i,j} \leq \Lambda^a_{i,j} \leq U^a_{i,j} \quad (a = 5,6, i = 1,2,\ldots P,\ j = 1,2,\ldots 7) \tag{3}$$

$$L^a_{\cdot,j} \leq \sum_{i=1}^{P} \Lambda^a_{i,j} \leq U^a_{\cdot,j} \quad (a = 5,6,\ j = 1,2,\ldots 7) \tag{4}$$

$$L^a \leq \sum_{i=1}^{P} \sum_{j=1}^{7} \Lambda^a_{i,j} \leq U^a \quad (a = 5,6) \tag{5}$$

Equations (**??**)-(**??**) for arrival rates (6)

Equation (**??**) for discharge rates (7)

Equation (12) for conservation of flow (8)

Equation (**??**) for mean occupancy (9)

Equations (**??**)-(**??**) for variance of occupancy (10)

$$\Lambda^a_{i,j} \geq 0 \text{ and integer}, \quad (a = 5,6, i = 1,2,\ldots P,\ j = 1,2,\ldots 7) \tag{11}$$

Equation (3) sets upper and lower bounds on the number of scheduled inductions and scheduled cesareans by time period of day and day of week. Equations (4) and (5) set upper and lower bounds on the number of scheduled inductions and scheduled cesareans by day of week and for the entire week, respectively. The remaining constraints constitute the patient flow portion of the model. Note that the only decision variables are $\Lambda^5_{i,j}$, the number of scheduled inductions, and $\Lambda^6_{i,j}$, the number of scheduled cesareans. The model OB-SMOOTH is a standard mixed integer linear program.

5.1 Computational experiments

Solve realistic problems. No competitive models to which to compare to other than reporting the size of comparable explicit tour scheduling models. I guess I could compare to the 5/7 model for the restricted environment for which it was designed.

6 Just some templates to use

$$\lambda^{h,s(h,m)}_{i,j} = D^{h,r(h,m)}_{i,j} \quad (i = 1,2,\ldots P,\ j = 1,2,\ldots 7, h \in \mathscr{H}, m = 2,3\ldots n_h). \tag{12}$$

7 Conclusions and Future Work

BLAH, BLAH

Table 1: Patient care units

Unit #	Unit Name	Abbr.
1	Labor & Delivery	LD
2	Recovery room	R
3	Cesarean section procedure area	C
4	Postpartum unit	PP