Stochastic integer programming based algorithms for adaptable open block surgery scheduling

Camilo Mancilla Robert H. Storer
Lehigh University

April 22, 2012

Abstract

We develop algorithms for adaptable schedule problems with patient waiting time, surgeon waiting time, OR idle time and overtime costs. Open block surgery scheduling of multiple surgeons operating in multiple operating rooms (ORs) motivates the work. We investigate creating an "adaptable" schedule of surgeries under knowledge that this schedule will change (be rescheduled) during execution due to uncertain surgery times and other disruptions. The problem we address consists of finding an initial "day before" OR schedule (surgery starting times, surgery sequence in each OR and surgery sequence for each surgeon), and is formulated as a multi-stage stochastic integer program (with the initial schedule determined in phase 1) using sample average approximation. A small (due to complexity) computational study is conducted. We find optimal initial schedules under simple right shift rescheduling policies, and further demonstrate that more complex rescheduling policies (i.e. recourse) can be effective in improving performance given a variety of initial schedules

Stochastic integer program based algorithms for adaptable open block surgery scheduling

1 Introduction

The daily schedule for a hospital's operating rooms is typically finalized the day prior to surgery. This schedule includes the operating

room (OR) to which each surgery is assigned, the surgeon performing the procedure, and its planned starting time. Once the schedule is finalized, patients are called and told when to report to the hospital the next day. In many hospitals, surgeons perform all of their surgeries for the day in a single operating room. One disadvantage of this approach is that the surgeon is idle during the setup phase before, and the cleanup phase after each surgery. To avoid surgeon idle time, some hospitals have moved to an "open block" scheduling approach in which the surgeons move between different ORs to perform their surgeries. If scheduled carefully, open block scheduling can indeed significantly reduce surgeon idle time (Batun et al. (2011)). In this paper we address open block scheduling via stochastic programming assuming that the durations of each surgery, setup, and clean-up are random variables with known distributions.

Decisions made the day prior to surgery include 1) the sequence of surgeries for each surgeon, 2) the sequence of surgeries in each OR, and 3) the scheduled starting time of each surgery. One goal of this paper is to create methods to generate this initial schedule and to gain insight into the importance of this schedule on system performance.

On the day of surgery as the schedule is being executed, further "recourse" decisions are also necessary. In the simplest case, starting times are delayed past their scheduled time to compensate for the actual outcomes of surgery times and other disturbances. It may also be possible to move a surgery from its scheduled OR to another OR, a practice we have observed in several hospital OR suites. This more complicated recourse action may have an associated cost. Determining how to execute this more complex recourse is an additional problem to be addressed. A second goal of this paper is to create methods to execute a rescheduling policy that allows moving surgeries to a different OR and determine the degree to which this more complex recourse can be beneficial.

Finally, there will be interaction between the initial "day before" schedule and the recourse policies used to "reschedule" during execution. A final goal of this paper is to investigate how to produce initial schedules that work well in conjunction with a particular recourse/rescheduling policy.

Rescheduling. In practice the daily surgery schedule is inevitably changed during execution as the actual durations of surgeries are observed and as emergency surgeries and other potential disruptions arise. One common way to revise the schedule, or "reschedule", dur-

ing execution is to simply continue with the original OR assignments and surgeon surgery sequence but delay the starting times as required. In this paper we assume that surgeries cannot be started earlier than specified in the original schedule because the patient will not yet be ready for surgery (it may be helpful to think of the scheduled starting time as the earliest possible starting time). The simple policy that maintains the original OR assignments and sequences but delays surgeries as needed to accommodate disruptions is called the "right shift" rescheduling policy. An alternative to the simple "right shift" rescheduling policy is a policy that reassigns surgeries to different ORs by removing the surgery from its scheduled OR and inserting it into the sequence of another OR. Note that the starting times of all surgeries can never be earlier than the scheduled starting time established the day before (this also implies that the sequence in which the surgeon performs his/her surgeries does not change either). This more complicated rescheduling policy must include details specifying how to change the schedule based on the current status of the system. Moving a surgery to a different OR may result in additional costs due to communication, transportation, staffing, etc. This new schedule can be generated through a re-optimization procedure from a local or global perspective. The set of actions executed to produce revised schedules during execution are called rescheduling policies. For a thorough review of the rescheduling literature please see Vieira et al. (2003).

Previous literature on stochastic surgery scheduling implicitly assumes a "right shif" rescheduling policy Batun et al. (2011), Mancilla and Storer (2009),Mancilla and Storer (2010),Erdogan and Denton (2009). All of these papers use two-stage stochastic integer programming to find initial schedules. In the first stage, the initial schedule is computed so as to minimize expected cost. In the second stage, new start and finish times are computed for each surgery in each scenario by simply delaying (or "right shifting") starting times as necessary to accommodate the observed processing times. The advantage of assuming a "right shift" rescheduling policy (from an optimization viewpoint) is that the recourse in stage two is relatively easy to compute. If the only action allowed during schedule execution is to delay starting times, then these methods provide schedules that are robust in that the uncertainty in processing times is explicitly considered in the optimization. However an initial schedule developed under the assumption of a simple rescheduling policy (e.g. right shifting) may perform poorly if a more complex rescheduling policy is actually in

use. Two questions arise, 1) Can the more complex rescheduling policy render the initial schedule less important? and 2) Can we find initial schedules that perform better under the more complex recourse policy?

Adaptable Scheduling. The problem of finding an optimal initial schedule given (1) processing time distributions and (2) a specified rescheduling policy has been previously called the robust scheduling problem (Wu et al. (1993)). We have found that the term "robust scheduling" causes significant confusion. The field of "robust optimization" in which one seeks solutions to optimization problems that are insensitive to uncertainties in problem data is well established. Thus robust scheduling is often misconstrued as finding schedules that are similarly insensitive. For this reason we prefer to use the term "adaptable scheduling". In adaptable scheduling the initial schedule must take into account the rescheduling policy that is being used. Previous research in stochastic scheduling invariably assumes a simple "right shift" rescheduling policy when finding the initial schedule. In this paper we also consider the problem of finding an initial schedule under more complex rescheduling policies. Further, we investigate whether or not the more complex rescheduling policy mitigates the impact of the initial schedule (that is, with the more complex rescheduling policy, the initial schedule may be less importance in determining schedule performance). In adaptable scheduling one would ideally incorporate the more complex and realistic rescheduling policy directly into the recourse. The result will be a detailed multi-stage stochastic integer program (MSSP) with an optimization problem to be solved at each stage. The schedule found in stage one would be the "adaptable schedule" and the decisions found at subsequent stages would give the optimal (non-anticipative) rescheduling policies over the various scenarios. However, solving this multi-stage stochastic program (MSSP) is clearly very difficult due to its complexity as is discussed subsequently.

2 Problem definition

In daily operating room scheduling there is a given set of surgeons each with a given set of surgeries to perform on a given day. The OR Manager must decide, the day before surgery, how to schedule these surgeries in an open block fashion. Specifically, he/she must decide:

- The sequence in which each surgeon will perform his/her set of surgeries (in practice this decision may be pre-specified by the surgeon, or in other cases the sequence may be decided by the scheduler).
- The sequence of surgeries in each OR.
- The planned (earliest) starting time for each surgery.

Since this initial schedule will change during execution, the OR manager seeks an initial adaptable schedule that minimizes the overall cost at the end of the day given that rescheduling will almost surely occur due to uncertainty in surgery times. In this paper, costs considered include (1) idle time of surgeons between surgeries, (2) patient waiting time between the scheduled and actual start time, (3) idle time of the operating rooms and staff, (4) overtime of the OR staff and (5) the number of OR room assignment changes made during rescheduling. Further it is assumed that each surgery is composed of three phases: (1) setup, (2) surgery, and (3) clean-up. The distribution of the duration of each phase is assumed known, and the presence of the surgeon is only required during the surgery phase. Further, all surgery, setup and clean-up times have unique distributions.

We begin by describing the full multi-stage stochastic program (MSSP) formulation. We then approximate the MSSP with a two-stage stochastic integer program where the first stage decisions provide the adaptable schedule and the second stage decisions approximate the rescheduling policies. In the following, we show a simple example that provides intuition as to why this approach might work, then we explain the approximation and finally we propose a Monte Carlo method to explicitly test its performance.

2.1 Motivating Example

Assume that we have to schedule 3 surgeries in 2 OR's and that we have only 3 scenarios as shown in Figure 1. In Figure 1 we show two different schedules. In schedule A surgery 3 is scheduled in OR 2. In schedule B surgery 3 is scheduled in OR 1. In both schedules surgery 3 is initially scheduled to start at 1:00. For each schedule, the left figure shows what will happen when right shift is applied after realizing the outcomes in each of the 3 scenarios (note that surgery 3 must remain in its original OR under right shift rescheduling). Under a right shift rescheduling policy, the optimal schedule will be schedule

A (surgery 3 in OR 2). Note that the average patient waiting time for schedule A is (0+20+20)/3 (thus the expected waiting cost is $13.33c^w$) while the average in the bottom schedule is (50+0+0)/3 (thus the expected waiting cost is $16.67c^w$) where c^w is the per unit time cost of patient waiting. However, if we treat this problem as a multi-stage stochastic problem, the final assignment of surgery 3 to an OR can be scenario dependent. The two right hand schedules show the final optimal schedule for each scenario under the assumption that we can move surgery 3 to a new OR. Under this assumption we see that in schedule A we would move surgery 3 to OR 1 in both scenarios 2 and 3. In schedule B surgery 3 would be moved to OR 2 in scenario 1. After moving surgeries in this fashion, there will be no delay in the starting times of surgery 3 in either schedule in all scenarios (and thus the patient waiting cost will be zero in both cases). However, there may be costs associated with moving surgeries to another OR. Letting c^c represent the cost of moving a surgery to another room, we see that the top schedule incurs two moves (scenarios 2 and 3) while the bottom schedule only requires one move (scenario 1) and thus is the better choice under this more complex rescheduling policy. (Note: this analysis requires that c^c is much less than c^w to be precise). The key point of this example is that the optimal initial schedule is different under the right shift rescheduling policy than under a more complex rescheduling policy in which shifting ORs is allowed. Thus an initial schedule should take into account both the uncertainty in processing times and the rescheduling policy being used.

In the simple example the optimal solution is option B, and we could have figured this out by first computing the waiting times for every scenario after applying the rescheduling policy, and then counting the number of changes (move surgery 3 from OR 1 to OR 2 or vice versa).

2.2 Multiple Stage Integer Programming Formulation

In this section we describe a multi-stage stochastic integer programming model of the OR scheduling - rescheduling decision process discussed above. In this model the stages are defined by discrete points in time where we observe the system (for instance, every 30 minutes) and revise the schedule (using the rescheduling policy). In this model the decision variables are:

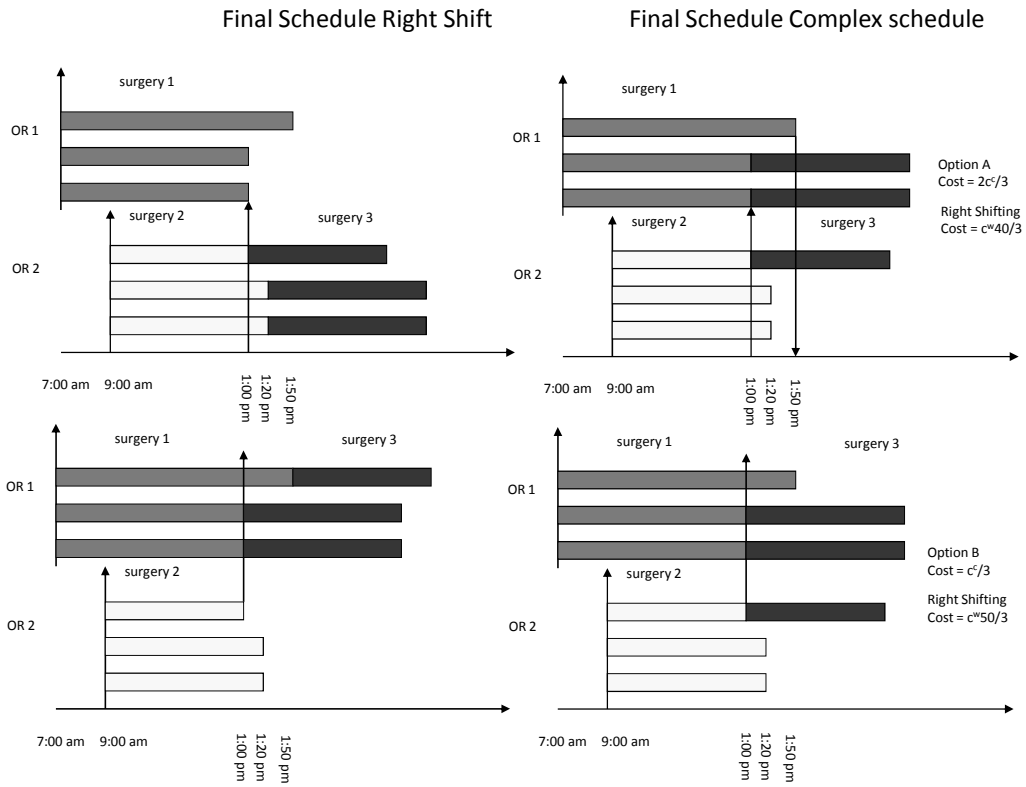

Figure 1: Small example.

First stage decision variables:

- The surgery sequence for each surgeon.
- The initial sequence of surgeries in each OR.
- The scheduled starting time of each surgery.

Subsequent stage decision variables:

- The revised sequence of surgeries in each OR.

In each stage after stage one, and for every branch in the scenario tree, a new sequence of surgeries for each OR is computed. This revised OR sequence is the result of observing the state of the system at the current stage and then solving a multi-stage stochastic IP from "now until the end".

Note that the surgery sequence for each surgeon and the scheduled starting times for each surgery are decided and fixed in the first stage. This is the part of the schedule that must be decided ahead of time (e.g. the day before) so that the patients know when to report to the hospital. In subsequent stages, after observing outcomes up to the current stage, the OR manager may decide to move one or more surgeries to different ORs. He or she must also decide where in the OR-sequence this surgery should be inserted. Also note that the surgery cannot start before its original scheduled starting time as decided in stage 1.

We assume that each stage is separated by 30 minutes. That is, every 30 minutes, we observe the state of the system and make rescheduling decisions that may move surgeries to different ORs and positions in the OR sequence. As mentioned previously, when a surgery is moved, a fixed cost penalty is assessed.

Formulating this problem in the form of a multi-stage stochastic integer program is complicated because the state definitions depend on the entire path and all decisions made up until the current stage (i.e the problem is stagewise dependent). Thus for example, one would need variables with 20 indexes to define the system state in stage 20. Further, solving this problem to optimality is problematic due to the extremely rapid explosion of the state space (i.e. curse of dimensionality). In order to define the multi-stage problem clearly, we provide a dynamic programming formulation in Appendix A, however, we do not attempt to solve this problem.

In the next section we formulate a two-stage approximation based on relaxing the non-anticipativity constraints in the multi-stage prob-

lem. We conclude that even this (much simplified) approximation to the multi-stage problem is extremely difficult to solve, thus requiring further simplification. To illustrate the point, an example problem with 2 surgeons, 3 surgeries per surgeon, 3 ORs, and 10 scenarios could not be solved in over five days of computation.

3 Approximation to the multi-stage stochastic problem

The basic idea of our approximation scheme is to find an initial schedule that considers that changes may occur during execution but that these changes come at a cost. Since, it impossible to solve the multi-stage stochastic program in reasonable time, we will approximate this problem as a two-stage stochastic integer program. The first stage decisions (starting times of each surgery, surgeon surgery sequences and initial OR sequences) are made at time zero (and correspond to the scheduling decisions that are made the day before surgery). The second stage (recourse) decisions are the rescheduling actions that change the OR schedule to accommodate disruptions. In the approximation, these stage two recourse actions are made with the assumption of perfect information. This approximation can be interpreted as making the stage one decisions the day before without knowledge of random variable outcomes while the stage two decisions are made at the beginning of the day (after stage one decisions are fixed) assuming perfect information of outcomes. One way to justify this approximation is that it is a relaxation of the multi-stage formulation in which the non-anticipativity constraints are relaxed for scenarios after time 0. After relaxing we will get K different stage two solutions (one for each scenario) representing the best rescheduling for each scenario given the initial schedule. The schedule found in stage one is now hopefully an "adaptable schedule" that works well given the presence of disruptions and rescheduling actions.

4 Approximation applied to the assignment of surgeries to multiple ORs

The mathematical formulation of the two stage approximation is given below. We call this the SAPIA (sample average perfect information

approximation) problem. This model is a two stage stochastic program with integer variables in each stage. We note here that even this two stage approximation is difficult to solve in reasonable time and we will later describe further approximations required to produce solutions in reasonable time.

$$\min z_A = \sum_{k\in K} \frac{1}{K} \left(\sum_{i\in I_s, s\in S} c^w w_{is}^k + \sum_{j\in J} c^{pw} pw_j^k + \sum_{r\in R} c^s s_r^k + \sum_{r\in R} c^o O^k + \sum_{i\in I, s\in S, r\in R} c^c u_{isr}^k \right)$$

$$\begin{aligned}
\text{s.t.}\quad & x_{1jr}^0 (pre_j^k + post_j^k + sur_j^k) \le C_{jr}^k && j,k,r && (2)\\
& -M(2 - x_{i-1sr}^k - x_{iqr}^k) + C_{sr}^k + (pre_q^k + post_q^k + sur_q^k) \le C_{qr}^k && s,q,r,k,i>2 && (3)\\
& -M(3 - y_{iq}^s - y_{i+1g}^s - \sum_{m\le n} x_{mgr}) + \sum_{r\in R} C_{qr}^k - post_q^k + sp_g^k \le C_{gr}^k && g,q,r && (4)\\
& -M(2 - y_{iq}^s - y_{i+1g}^s) - \sum_{r\in R} C_{qr}^k + post_q^k + \sum_{r\in R} C_{gr}^k - sp_g^k \le w_{is}^k && k,i<n_s,q,g,s && (5)\\
& C_{jr}^k - d_r \le O_r^k && k && (6)\\
& st_j + pw_j^k \ge C_j^k - (pre_j^k + sur_j^k + post_j^k) - M(1 - y_{pj}) && j,p && (7)\\
& st_j \le C_j^k - (pre_j^k + sur_j^k + post_j^k) + M(1 - y_{pj}) && j,p && (8)\\
& C_{jr}^k \le M \sum_{i\le n} x_{ijr} && j,r,k && (9)\\
& \sum_{p\in I_s} y_{pj}^s = 1 && j,s && (10)\\
& \sum_{j\in J_s} y_{pj}^s = 1 && p,s && (11)\\
& x_{irs}^0 - x_{irs}^k \le u_{irs}^k && i,s,r,k && (12)\\
& x_{irs}^k - x_{irs}^0 \le u_{irs}^k && i,s,r,k && (13)\\
& \sum_{s\in S} x_{irs}^0 \le 1 && i,r && (14)\\
& \sum_{r\in R} x_{irs}^0 \le 1 && i,s && (15)\\
& \sum_{r\in R} \sum_{i\in I_s} x_{irs}^0 = |I_s| && s && (16)\\
& \sum_{s\in S} x_{irs}^k \le 1 && i,r,k && (17)\\
& \sum_{r\in R} x_{irs}^k \le 1 && i,s,k && (18)\\
& \sum_{r\in R} \sum_{i\in I_s} x_{irs}^0 = |I_s| && s,k && (19)\\
& x_{ijr} \in \{0,1\}\ \forall (i,j,r) \in (J,J,R)
\end{aligned}$$

Indices and Sets

n is the number of procedures to be scheduled
n_s is the number of procedures for surgeon s
K is the number of scenarios

j indexes the set of procedures: from 1,...,n.
I is the set of positions for the OR sequence.
J's indexes the procedures for surgeon s.
I's indexes the positions for surgery sequence for surgeon s.
p indexes the position in the surgery sequence for a particular surgeon.
r indexes the set operating rooms: r=1,..,R
k indexes scenarios: k=1,....,K.

Parameters

c^w surgeon idle time cost per unit time.
c_{pw} patient waiting time cost per unit time.
c^o overtime cost per unit time.
c^s idle time cost per unit time.
c^c cost of changing one OR assignment.
d_r time beyond which overtime is incurred in O.R. r
M is sufficiently large "big M" number.
pre_j^k duration of setup for procedure j in scenario k.
sur_j^k duration of surgery for procedure j in scenario k.
$post_j^k$ duration of cleanup for procedure j in scenario k.
sp_j^k sumation of surgery and cleanup for j in scenario k.

Variables

W_{is}^k Surgeon waiting time before the procedure in position i of the sequence for surgeon s in scenario k.
pw_j^k patient waiting time for surgery j in scenario k.
I_r^k Total OR idle time in scenario k for OR r.
C_{jr}^k Completion time of procedure j in OR r in scenario k.
O_r^k Overtime time in OR r in scenario k.
x_{irs}^k a binary variable denoting the assignment of surgeon s in position i in OR r in scenario k.
x_{irs}^0 a binary variable denoting the assignment of surgery j in position i in OR r in the original schedule (first stage variable).
y_{pj}^s a binary variable denoting the assignment of surgery j in position p in surgeon sequence s.
u_{irs}^k a binary variable denoting the number of differences between the final OR assignment in scenario k and the original OR schedule x^0.

Constraints

(2), (3) define the completion time based on OR sequences.
(4) define completion time based on surgery sequence, the second

surgery of surgeon s should start after the first surgery is completed.
(5) computes the waiting time for every surgeon.
(6) computes the overtime for every OR.
(7) computes the patient waiting time.
(8) guarantees that every surgery should start after its scheduled starting time.
(9) sets to zero the completion time of surgery j if this was not assigned to OR r.
(10), (11) assignment constraints that guarantees that every surgery is assigned to one OR.
(12), (13) assignment constraints that guarantees that every surgery in the surgery sequence gets assigned.
(14), (15) computes the OR schedule changes relative to the original schedule defined at time 0.
This problem is a two-stage stochastic integer program where the first stage and the second stage have both continuous and binary variables. During experimentation we discovered that the problem behaves differently based on the value of the "cost changing ORs".

Case 1: The right shift or high cost of changing ORs case:

When the cost of changing ORs is high enough, the OR manager will never change the initial OR assignments. This is equivalent to adopting the simpler "right shift" rescheduling policy. Thus solving the two stage problem with a "high cost of changing ORs" will produce the optimal initial schedule under the right shift rescheduling policy. In this case the initial OR schedule is of greater importance since no future room changes are possible. To solve this problem we use an approach similar to that proposed by Mancilla and Storer (2010) where the problem is decomposed by OR sequence. The idea is to solve a subset of problems defined by the OR sequence in each OR. Once we have fixed the OR sequence for every OR we will have a stochastic MIP that will search for the best surgery sequence for every surgeon. This sub-problem is manageable for Cplex as was found in Mancilla and Storer (2010). To clarify, we define vector a^r as the "OR sequence for OR r". For instance, if a^r is (A,B,C,C,0,0,0) then the first surgery in room r is from surgeon A, the second from surgeon B, then 2 from surgeon C. The search space of the vectors a^r is large (there are(S!)I possible OR sequences) but we can take advantage of various fast lower bounding techniques to quickly eliminate all but a few viable OR sequences. The basic outline of the algorithm is as following,

OR-Sequence decomposition algorithm

Set the upper bound (UB) to ∞ .

1. Initialize S as the finite set of all OR sequences.
   Initialize s_i as a promising OR sequence and remove from S.
   Solve the corresponding SAA MIP and initialize the UB.
2. Remove a new OR sequence (s_i) from S if S is not empty, otherwise end.
3. Solve the LP relaxation of the mean value problem for (s_i). If the solution is less than the UB go to 4. Otherwise go to 2.
4. Solve the sequencing sub-problem SAA for s_i. If the solution is less than the current UB update it, otherwise go to 2.

The idea behind this algorithm is that only a few sub-problem solutions will be close to optimal, while the vast majority of the OR sequences will be terrible. Since the LP relaxation of the mean value problem provides an easily computable lower bound, it can be used to very quickly eliminate all but a few OR sequences. In this decomposition approach we assume that the maximum number of surgeries per OR is 4 in order to minimize the number of possible OR sequences.

Initial Testing for Case 1

As was found in Mancilla and Storer (2010) the ratio between surgery and setup plus clean up times has an impact on the processing time of the algorithm. We define a parameter B_s for each surgeon s as follows:

$$B_s = \frac{\sum_{j=1}^{n} \sum_{k=1}^{K} sur_j^k}{\sum_{j=1}^{n} \sum_{k=1}^{K} pre_j^k + post_j^k}$$

Note that we only consider surgeries that belong to surgeon s in this equation. If, for example, the B_s parameter for a particular surgeon s is one, then it might be reasonable to have 2 ORs available since the surgeon is only required for half the time (on average) of each surgery. The parameters B_s are one of the factors we investigate in our computational experiment. Other factors include the number of scenarios and the cost ratio as shown in the table below. The cost ratio is defined as surgeon waiting cost divided by OR idle cost. Test problems all consisted of 3 surgeons each with 4 surgeries scheduled in 4 ORs. Duration distributions were all lognormal with means and variances created as discussed in Mancilla and Storer (2010). We ran 5 different sets of data. The "high cost ratio" corresponds to a cost

ratio of 2 and "low cost ratio" is 1. We say that the B_s parameters are "similar" when the coefficient of variation (computed based on the distributions for each surgeon) is below 0.1 and high otherwise. Table 2 shows the algorithm run time in seconds for the various problems tested. Processing times vary in the 1 to 3 hour range. The run time increases with the number of scenarios as one would expect and decreases when the cost ratio is high. Processing time decreases when the cost ratio is high because the bound obtained by the mean value problem is tighter when the cost ration is high, therefore the algorithm does not have to fully explore as many subproblems.

Table 1: Experiment Design

Factor	Possible Value
Number of Scenarios	100 , 150
B Parameter	Similar, Different
Cost ratio	High , Low

Table 2: Processing time in seconds

Factors	Scenarios	
B Parameter, Cost Ratio	100	150
Similar , Low	5568	9259
Similar , High	4890	10352
Different, Low	4433	11449
Different, High	3563	8682

Table 2 shows the time required to solve each of the cases in our experiment.

Case 2: Low cost of changing ORs (complex rescheduling) case:
When the "cost of changing ORs" is low, the OR manager will periodically observe the current state of the schedule, and will make rescheduling adjustments in order to diminish the impact of disruptions. Thus it is possible that every scenario may end up with a different OR sequence after the OR manager has executed the rescheduling policy. The mathematical formulation of our approximate problem (SAPIA) is a two-stage stochastic integer program with integer recourse. Since we have integer recourse it might be reasonable to use

decomposition techniques such as scenario decomposition proposed in Caroe and Schultz (1997). We implemented this decomposition technique but the processing times were unacceptably long. Next we decided to divide the problem into different steps. The first step is the selection of each surgeon's surgery sequence. Once the surgery sequence for each surgeon is fixed we next compute the initial OR sequence and finally the scheduled starting times. The outline of this procedure is as follows:

Surgery-OR-Starting time decomposition algorithm

1. Find an initial surgery sequence for each surgeon (could be obtained from the high cost of change case, or from sort by variance, or from surgeon preferences, etc).
2. Solve every scenario independently in order to find the optimal final OR sequence for each scenario.
3. Solve the Initial "minimum distance OR-sequence" problem (see below) to get the initial OR-Sequence.
4. Compute the starting times (using the optimal starting time algorithm explained below) assuming the final OR-assignment for every scenario found in step 3.

The idea behind this heuristic is to find a good (adaptable) OR-sequence and starting times given the sequence for each surgeon. We will also investigate different ways to assign the surgeon's surgery sequence and report which methods work best. In the next section we describe how we find the initial "minimum distance OR-sequence".

Initial Minimum Distance OR-Sequence.

Once we have fixed each surgeon's surgery sequence we compute the optimal OR-sequence for every scenario by solving k (easy to solve) integer programming problems. We may end up with as many as k different OR-sequences therefore we need a procedure to find a single initial OR-sequence given the optimal OR-sequence for each scenario. That is, we seek an initial OR sequence that is in some sense close to the various OR sequences in each scenario. We propose to solve the following optimization problem,

$$\min Z^c = \frac{1}{K} \sum_{k \in K} \sum_{i \in I} \sum_{r \in OR} \sum_{s \in S} c^c u^k_{irs}$$

$$
\begin{array}{ll}
x^0_{irs} - x^k_{irs} \leq u^k_{irs} & i, s, r, k \\
x^k_{irs} - x^0_{irs} \leq u^k_{irs} & i, s, r, k \\
\sum_{s \in S} x^0_{irs} \leq 1 & i, r \\
\sum_{r \in OR} x^0_{irs} \leq 1 & i, s \\
\sum_{r \in OR} \sum_{i \in I_s} x^0_{irs} = |I_s| & s \\
\sum_{s \in S} x^k_{irs} \leq 1 & i, r, k \\
\sum_{r \in OR} x^k_{irs} \leq 1 & i, s, k \\
\sum_{r \in OR} \sum_{i \in I_s} x^0_{irs} = |I_s| & s, k \\
x^k_{irs}, x^0_{irs} \in \{0, 1\} &
\end{array}
$$

The variables used in this problem are the same those we define in SAPIA with the addition of x^k variables that are obtained by solving the k scenarios problems separately. The idea is simply to find the initial OR sequence that minimizes the total number of room changes over the k scenarios assuming that the k optimal scenario based OR sequences approximate the final schedule of the real problem (after rescheduling). That is, we find the initial OR sequence from which we can get to the k optimal final OR sequences with the minimum number of room changes.

Optimal starting times given the OR-sequence. Once we have fixed (1) the surgery-sequence for each surgeon, (2) The (stage 1) initial OR sequence for each OR, and (3) the (stage 2) OR-sequences for every scenario (that is, all the integer variables are fixed) the SAPIA becomes a two stage stochastic linear programming problem that can be solved fairly quickly for the initial scheduled starting times. Once we have computed the starting times, the objective function can be easily computed from simple linear equations. For example the patient waiting time is given by:

$$pw^k_j = C^k_j - pre^k_j - sur^k_j - post^k_j - st^*_j$$

Other components of the objective function are computed in a similar way.

4.1 Further justification of the two-stage approximation

In this section we present two results that help understand why the SAPIA approximation problem might be useful in finding promising solutions for the multi-stage problem.

Claim: The objective function value of the optimal solution to the SAPIA approximation problem (z_S) is a lower bound to the optimal objective function value to the problem under a right-shift rescheduling policy (z_{RS}). ($z_S \leq z_{RS}$)

This is true because the solution to problem RS is one of many possible solutions to the SAPIA problem. Problem RS is a special case of the SAPIA problem in which no rescheduling moves are allowed.

Claim: The optimal objective function value of the (SAPIA) approximation problem is a lower bound for the multi-stage objective function z_{MS}. Further, ($z_S \leq z_{MS} \leq z_{RS}$)

1. ($z_S \leq z_{MS}$) because SAPIA is a relaxation of the multi-stage problem.
2. ($z_{MS} \leq z_{RS}$) because the right shift solution is a special case of the multistage problem.

5 Algorithm Summary

In this section we present a summary of the models we have developed to help clarify the situation.

Multi-stage model: Our first model was a multi-stage stochastic integer program that finds the sequence of surgeries for each surgeon, the OR sequence for each OR, and the starting times for each surgery. The multi-stage model also incorporates a complex rescheduling policy in which the state of the schedule is observed at each stage and an optimization algorithm is run that allows surgeries to be moved to other ORs during schedule execution (and inserted into that OR's sequence). There is little hope of solving this model to optimality.

Right shift rescheduling special case of the multistage model: If we assume the simple right shift rescheduling policy, the multi-stage model can be reduced to a 2-stage problem. By extending the decomposition methods from our previous work on parallel OR scheduling (Mancilla and Storer (2010)), we can solve this problem to optimality for a reasonable number of scenarios as shown in section 4.

SAPIA (sample average perfect information approximation) model: This model includes the more complex rescheduling policy but relaxes the non-anticipativity constraints in the multi-stage model. The result is a 2-stage approximation to the multi-stage model. In the first stage (representing decisions required the day prior to surgery) we must specify each surgeon's sequence, the scheduled starting time of each surgery, and an initial OR sequence for each OR. In the second stage, we assume all random variable outcomes are revealed, and that we are allowed to change the OR sequences (but not the surgeon sequences or starting times). However each change made to the OR sequence in stage 2 incurs a fixed cost (the "cost of changing ORs"). This model is very difficult to solve in reasonable time. We also developed progressive hedging algorithm methods to try to solve this problem, but were not successful.

SAPIA-surgeon sequences fixed model (SAPIA-SSF): If we assume the sequence of surgeries for each surgeon is known in advance, the SAPIA problem simplifies. In many cases surgeons prefer to set the sequence themselves based on procedure difficulty, patient preference, etc. We may also be able to test various ways to produce surgeon sequences in a first step, then solve for OR sequences and starting times using this model. Even after fixing the surgeon sequences, this SAPIA-SSF problem is still difficult to solve to optimality. We developed an approximate method based on scenario decomposition. We first found optimal (but different) OR sequences for each scenario separately. We then solved an optimization problem that finds the "minimum distance" initial OR sequence. This "minimum distance" initial OR sequence is the one that minimizes the total number of OR changes required to get to the final OR sequence in each scenario.

Optimal starting time model: Once both the surgeons sequences and initial OR sequences are determined, we must still find the scheduled starting time for each surgery. When both surgeon and OR sequences are fixed, the SAPIA reduces to a two stage stochastic linear programming problem that can be solved quickly by Cplex to produce optimal starting times for each surgery.

6 Evaluating the solutions by Monte Carlo simulation of the rescheduling policy

In order to evaluate the quality of the initial schedules the proposed methods produce, we develop a simulated environment that simulates the initial schedule under the more complex rescheduling policy. Given distributions of setup, cleanup and surgery times and the various cost factors, our algorithm produces initial surgeon sequences, OR sequences and starting times. We then evaluate this "adaptable" schedule's performance using the simulation of the more complex rescheduling policy. The simulation simulates the actions of the OR manager by observing the current state of the system every 30 (simulated) minutes and then solving a rescheduling optimization problem which gives a revised schedule. The simulation thus estimates the final cost after schedule execution and rescheduling. The simulation is repeated many times and the average total cost is reported. Obviously the processing time scenarios used in the simulation are random, and are different (independently generated) from those used to solve the original stochastic scheduling problem. Next we explain how we constructed this simulated environment in more detail.

The disruptions that we will consider are:

- Duration of surgery, setup and cleanup times.
- Emergency arrivals (indirectly).

The simulation of the dynamic (complex) rescheduling policy requires us to define the interval at which we observe the system, and how we will perform the rescheduling each period given the updated information on the state of the system. We assume that we observe the schedule execution (i.e. state of the system) every 30 minutes, and make a rescheduling decision by solving an optimization problem as defined below. When the system is observed at time t, we assume the duration of surgeries (or setups or cleanups) completed before t are known with certainty and that the distribution of surgeries that have not started by time t are those given in the problem statement. For durations started before t but not yet completed, we use the conditional distribution at time t given the initial distribution and actual starting time. As discussed previously, the actual optimization problem we would like to solve at each period is a multi-stage stochastic

program (from period t to the end). In our simulation we approximate this using the mean value problem. For durations currently unknown, we simply plug in (conditional) distribution means. The result is an integer program that can be solved reasonably quickly. Solving this problem produces a revised OR-sequence for each OR at time t for future surgeries. These OR sequences are followed up to the next decision epoch. We also save the revised OR sequences made at each time period and use them to compute final costs. There is an additional important detail in this computation. Suppose a surgery is moved from OR 1 to OR 2 at time t = 30, then moved back to its original spot in OR 1 at time t = 60. In this case we do not charge an "OR change cost". Further, suppose a surgery is moved from OR 1 to OR 2 at time t=30, then moved from OR 2 to OR 3 at time t = 60, and that the surgery is ultimately executed in OR 3. In this case we charge for only a single "OR change".

We approximate emergency arrivals, not with additional surgeries to be scheduled, but by assuming that they contribute to the duration of an already scheduled surgery. Bimodal distributions for the scheduled surgeries are used for this purpose. The idea is to construct this new distribution based on the probability that a particular surgery is interrupted by an emergency arrival. The assumption is that the probability of being interrupted increases as the surgery mean increase. Suppose on a given day that we expect an average of λ emergency arrivals per hour. We then compute the probability that surgery j will be interrupted by an emergency as a function of its mean duration MS_j. Assuming Poisson arrivals, the probability that surgery j is interrupted is = 1 - $\exp(\lambda \frac{MS_j}{N})$ where N is the number of ORs. Finally we can construct a bimodal distribution where the mixture coefficient is the probability of being interrupted, the first unimodal distribution is the surgery distribution and the second unimodal distribution is the sum of the scheduled plus emergency surgery durations. In our experiments we assume emergencies arrive at a rate of one every two hours.

Finally we note that we also simulate initial schedules under the right shift rescheduling policy again using independently generated scenarios. This simulation is a straightforward exercise.

7 Computational Experience

In this section we present experiments designed to evaluate the scheduling methods discussed previously. In the following list we summarize these experiments, then provide more detail on the experiments followed by results. We created 4 problem instances for testing.

- We test 5 different methods for creating the initial schedule (which includes surgeon sequences, initial OR sequences and starting times)
- Each of the five methods is evaluated by simulation under both the simple right shift and the complex rescheduling policies.
- For each of the five methods and both rescheduling policies, evaluations are made at many different levels of the "cost of changing ORs" parameter.

Problem Instances: All four problem instances were generated as discussed in section 4. In the first instance we assumed that the cost of patient waiting time was very low (0.01 USD/minute) compared to the other cost coefficients which were: surgeon idle time (3 USD/minute), OR staff idle time (3 USD/minute), and overtime (4.5 USD/minute). The second problem instance was the same as the first except that the cost of patient waiting time was increased by an order of magnitude to 0.1 USD/minute. The third problem instance included emergency arrivals modeled by the bimodal distributions described previously and the cost of patient waiting time was set to 5 USD/minute. The fourth problem instance included emergency arrivals and set the cost of patient waiting time to 0.01 USD/minute.

Scheduling Methods. We implemented five different scheduling methods as described in this section. The methods were selected in an attempt to compare our ideas to methods similar to what a typical OR manager might do. These methods are used to generate the initial schedule (initial surgeon sequences, OR sequences and scheduled starting times). These initial schedules are then evaluated by the simulation discussed previously under both rescheduling policies.

- **Case 1: "RS" Scheduling.**As discussed previously, if we assume a right shift rescheduling policy, the multi stage problem reduces to a two stage problem that we can solve using our decomposition approach. In this case we use this method to gen-

erate the initial surgeon sequences, OR sequences, and starting times.

- **Case 2: "SBV-Opt" scheduling.** In this case we first find the sequence for each surgeon using the sort by variance heuristic (sequenced from smallest to largest variance). Next we determine the sequence in each OR using our SAPIA-SSF model and "minimum distance sequence" approximation algorithm. Finally we find the starting times by solving the optimal starting time algorithm (the two-stage stochastic LP that results when both the surgeon and OR sequences are fixed in the SAPIA problem).
- **Case 3: "SBV-Manual".** This is an attempt to schedule in a manner similar to what an OR manager might do "by hand". First the surgeon sequences are set by the sort by variance heuristic as in case 2. The OR sequences are then found by solving the mean value relaxation of the SAPIA-SSF. Given the surgeon sequences and the mean values for each surgery one can imagine an OR manager trying to fit the surgeries into a Gantt chart. This is our attempt at approximating an OR manager moving these "puzzle pieces" around in a Gantt chart to find a good OR sequence. Once the OR sequence is set we also used a more ad hoc method to find the surgery starting times (although we could have solved the 2 stage stochastic LP as in case 2). Here we imagine that each surgeon's cases were all scheduled in a single OR. We then solve for the starting times using the single OR scheduling method described in Mancilla and Storer (2009).Finally we make sure that there is no overlap of surgeries in the mean value schedule in each OR. If overlap is found, starting times are delayed to eliminate the overlap.
- **Case 4: "SBM-Opt".**The only difference between case 4 and case 2 is that we set the sequence for each surgeon by sorting by mean surgery time from largest to smallest. Since many surgeons prefer to schedule their more complex surgeries first, this represents what might happen when surgeons decide their own sequence.
- **Case 5: "SBM-Manager".** The only difference between case 5 and case 3 is that we set the sequence for each surgeon by sorting by mean surgery time from largest to smallest.

Evaluating Schedules by Simulation. We evaluated each of the five methods by simulating the schedules under both the simple right

shift and more complex rescheduling policies. To simulate the more complex policy, we used the method described in section 4. We used 250 replicates of this simulation making sure that the scenarios in the simulation were generated independently from those used in the algorithms. To simulate performance under the simple right shift policy we used the same 250 scenarios. Note that in the three figures below in which results are presented, one can easily distinguish between cases where right shift and complex rescheduling are used. When there is no change in performance over the various levels of "cost of changing ORs" (i.e. a horizontal line) we know that right shift rescheduling is being used.

Cost of Changing ORs. For each combination of problem instance, scheduling method, and rescheduling policy we used 9 different "cost of changing ORs" parameter values. The value of the more complex rescheduling policy depends to a large extent on how easy and cheap it is to change rooms during schedule execution. We used values of 0, 15, 30, 45, 60, 75, and 90 USD as the cost of moving a single surgery to another OR.

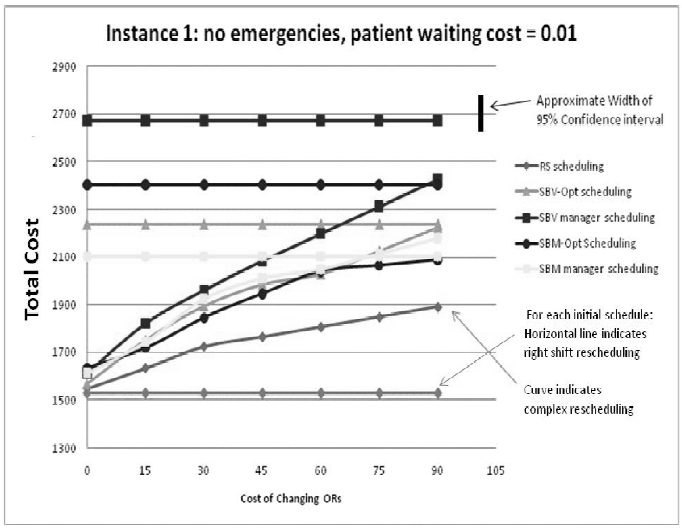

Figure 2: Instance 1: No emergencies, Patient waiting cost=0.01.

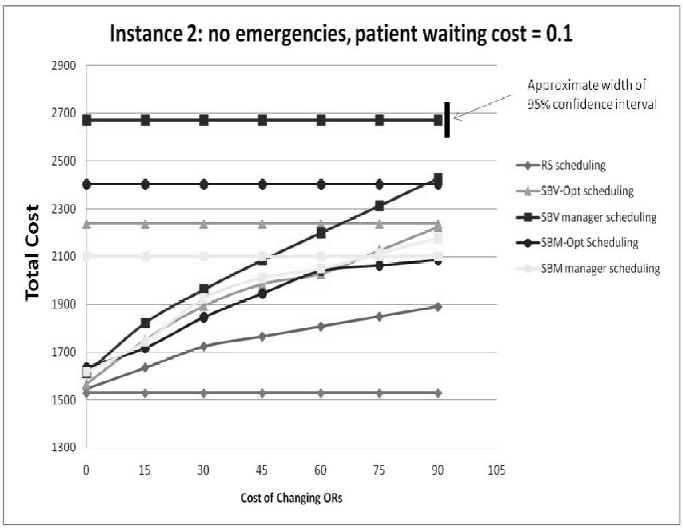

Figure 3: Instance 1: No emergencies, Patient waiting cost=0.1.

Based on the results presented in the figures, we can draw some conclusions.

- Under the right shift rescheduling policy, the initial schedule is very important. In this case using the proposed method to find an initial schedule can cut cost by up to 50%.
- If sub-optimal or arbitrary methods are used to find the initial surgeon sequences, one can still achieve much better performance by implementing the more complex rescheduling policy, especially when the cost of changing ORs is low. In the non-emergency cases with low cost of changing ORs, the initial schedule appears to be unimportant since complex rescheduling can "fix the problems" in the initial schedule.
- As cost of changing ORs increases, and in the case with emergencies, there is benefit to using the better (adaptable) initial scheduling methods even under complex rescheduling.
- When the patient waiting cost is low, scheduled starting times tend to be earlier reserving more time and thus flexibility. Thus in this case, we observe greater benefit from the more complex rescheduling policy.

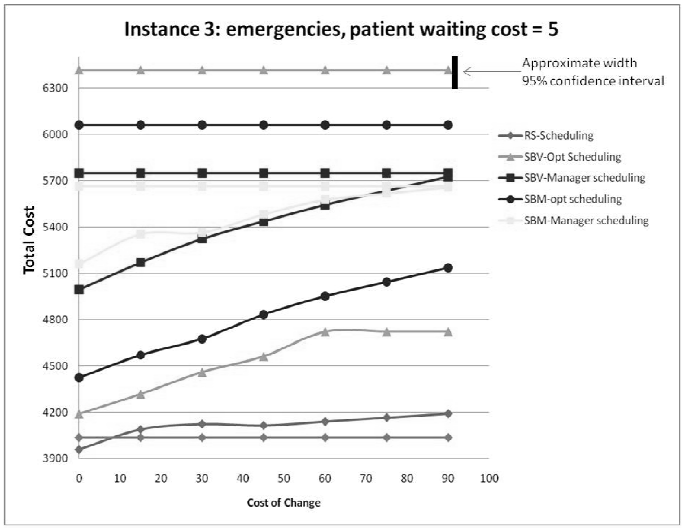

Figure 4: Instance 3: Emergencies, Patient waiting cost=5

- Given that surgeon sequences are specified in advance, there appears to be an advantage to using the proposed methods for finding the OR sequences and starting times over the "manager" methods meant to approximate what an OR manager might do by hand. The advantage is clearest in the case with emergencies, and when right shift rescheduling is used.
- There is no clear winner between sort by variance and sort by mean for this problem.
- By far the best initial scheduling method in our experiments was right shift scheduling. This method produces an initial schedule under the assumption of right shift rescheduling. Given this initial schedule, the more complex rescheduling policy was not helpful unless the cost of changing ORs was zero. Even in this case the benefit was marginal. Looking at this in a different way, if one uses our right shift method to produce the entire initial schedule, there appears to be little need for the more complex rescheduling procedure.
- If one were able to solve the full blown multi-stage stochastic program, one could certainly do better, and benefit from the more complex rescheduling policy. However, this problem seems to be quite intractable.

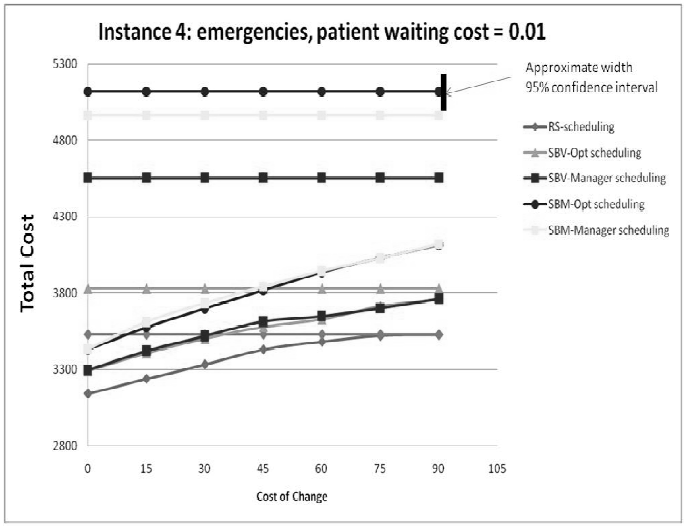

Figure 5: Instance 4: Emergencies, Patient waiting cost=0.01

8 Conclusions

In this paper we use stochastic programming methods to investigate adaptable scheduling in the case of open block scheduling in operating rooms. The adaptable scheduling problem, first addressed in the 1990's and called "robust scheduling" seeks good initial schedules under the assumption that rescheduling policies more complex than simple right shifting may be in use. We first formulated the problem as a multi-stage stochastic integer problem and then proposed approximation models and algorithms that help find reasonable solutions for adaptable scheduling. To solve the adaptable scheduling problem under simple right shift rescheduling we proposed a new decomposition algorithm and were able to solve the problem to optimality on small (but realistically sized in the realm of OR scheduling) problems. We also created a new rescheduling policy based on solving mean value versions of the multi-stage problem every 30 minutes. We then implemented a simulation model with this rescheduling policy embedded, and used it to test various scheduling methods. The main conclusions seem to be that (1) building near optimal initial schedules under the

assumption of right shift rescheduling seems to produce excellent initial schedules, and (2) When sub optimal methods are used to build the initial schedule (e.g. when each surgeon specifies his/her own sequence), significant benefit can be achieved be using the more complex rescheduling policy which seems to be able to compensate by adjusting the schedule during exection.

A Multi stage formulation of the adaptatible scheduling problem

Formulating this problem in the form of a multi-stage stochastic integer program is extremely complicated because the state definitions depend on the entire path and all decisions made up until the current stage. Thus for example, one would need variables with 20 indexes to define the system state in stage 20. Further, solving this problem to optimality is problematic due to the extremely rapid explosion of the state space (i.e. curse of dimensionality). In order to define the multi-stage problem, we provide the following multi-stage formulation using a general form provided in Shapiro et al. (2009),

$$\min_{x_0, x_1, \ldots x_T} \mathrm{E}\left[f_0(x_0) + f_1(x_1(\Xi_{[1]}), \Xi_1) + ... + f_T(x_T(\Xi_{[T]}), \Xi_T)\right]$$

$$\text{s.t.} x_0 \in X_0, x_i(\Xi_{[i]}) \in X_i(x_{i-1}(\Xi_{[i-1]}), \Xi_i), \text{i=2,..T}$$

Where x_0 are the first stage decisions: surgery sequence for each surgeon, initial sequence of surgeries in each OR and scheduled starting time of each surgery. The $x_1, ...x_T$ are the decisions executed after time 0, (in our case we make a decision every 30 minutes therefore the index can be translated to time by simply multiplying by 30 minutes). These decisions are the revised sequence of surgeries in each OR. The $\Xi' s$ represent the random variables that in our case are the surgery durations. $\Xi_{[1]}$ represents the surgeries that have been executed or are in progress up to the end stage 1. The Ξ_i are the surgeries that will be executed in the future and the uncertain portion of the surgeries that are being executed at the end of stage i-1. The functions $f_1, .., f_T$ are the cost incurred between every stage. The sets X_0 and X_i represent the possible actions in every stage. For instance, in stage i we cannot move to another OR the surgeries that started before stage i.

References

Batun, Sakine, Brian T Denton, Todd R Huschka, Andrew J Schaefer. 2011. The benefit of pooling operating rooms and parallel surgery processing under uncertainty. *Informs Journal on Computing* **23** 220–237.

Caroe, C, R Schultz. 1997. Dual decomposition in stochastic integer programming. *Operations Research Letters* **24** 37–45.

Erdogan, S A, B T Denton. 2009. Surgery planning and scheduling: A literature review. *Technical Report NSCU* .

Mancilla, C, R H Storer. 2009. Stochastic sequencing and scheduling an operation room. Tech. rep., Industrial and System Engineering, Lehigh University.

Mancilla, C, R H Storer. 2010. Stochastic sequencing in parallel ors. Tech. rep., Industrial and System Engineering, Lehigh University.

Shapiro, A, D Dentcheva, A Ruszczynski. 2009. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM.

Vieira, G, J Herrmann, E Lin. 2003. Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling* **6** 39–62.

Wu, David S, Robert H Storer, Pei-Chann Chang. 1993. One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers and Operation Research* **20(1)** 1–14.