Stochastic Sequencing and Scheduling of an Operating Room

Camilo Mancilla and Robert H. Storer

Lehigh University, Department of Industrial and Systems Engineering, November 14, 2009

1 abstract

We develop algorithms for a stochastic appointment sequencing and scheduling problem with waiting time, idle time, and overtime costs. Scheduling surgeries in an operating room motivates the work. The problem is formulated as an integer stochastic program using sample average approximation. A heuristic solution approach based on Benders' decomposition is developed and compared to exact methods and to previously proposed approaches. Extensive computational testing based on real data shows that the proposed methods produce good results compared to previous approaches. In addition we prove that the finite scenario sample average approximation problem is NP-complete.

2 Introduction

Operating Rooms (OR's) account for about 40% of the total revenue in a hospital and operate at 69% of their capacity ([1]). Thus there is significant opportunity for improvement, both in terms of utilization costs, and coordination with other hospital resources since the OR schedule drives many if not most hospital activities. In this paper a method is developed to help the OR Manager determine the sequence and scheduled starting times of surgeries in a single OR to minimize a weighted combination of costs that include patient waiting times, OR idle time and staff overtime. It is assumed that the duration of surgeries are random variables with known joint distribution. The marginal distributions of surgery durations are not assumed identical, but rather vary (based on the surgeon, type of surgical procedure, patient attributes, etc.). The method will be used to make operational decisions on a daily basis and thus must produce solutions in a timely fashion; on the order of a few minutes.

The problem can be decomposed into two parts. The first is to determine the sequence in which the surgeries will be performed. Given a sequence one must next determine the amount of time to allocate to each surgery, or equivalently assign a scheduled starting time to each surgery. This second problem (which we call the scheduling problem) has been studied previously under the name stochastic appointment scheduling. Previous approaches to this problem include using convolutions to compute starting times ([2]), sample average approximation and the L-Shape Algorithm ([3]), and heuristics ([4]). We will approach this problem using sample average approximation and linear programming in a similar fashion to one presented recently by [5].

Given reasonably efficient methods to solve the scheduling problem, we next develop an algorithm for the sequencing problem. According to [6] the sequencing problem is still an open question.

OR Managers typically book a fixed amount of time for each surgery based on previously recorded times for the surgeon and type of procedure. A standard rule of thumb is to assume the mean duration of the previous K surgeries for the surgeon in question and type of procedure. The performance measures considered include waiting time (when the actual starting time is greater than the scheduled starting time, the patient must wait), idle time (when the previous surgery finishes before the next scheduled starting time, the OR will be idle), and overtime (if the total time to complete all surgeries exceeds the scheduled work day). We assume a separate cost (per unit time) for each surgery for both waiting and idle times and an overtime cost. We also explicitly consider the randomness of the surgery times, thus the objective is to find a sequence and schedule of starting times to minimize total expected cost where expectation is taken with respect to the joint distribution of surgery times. In testing, the data used to generate surgery time scenarios in the sample average approximation is based on data provided by a large regional Hospital.

The main idea of the proposed method is based on a Benders' decomposition scheme. The master problem is used to find sequences and the sub-problems are the scheduling problems (stochastic linear programs) as discussed above. The master problem in the Benders' approach becomes extremely hard to solve as cuts are added, thus we turn to heuristics to approximate its solution and generate promising sequences.

The remainder of the article is organized as follows. In the next section a brief review of the literature related to the sequencing and scheduling problems and stochastic integer programming is provided. In Section 3, the model is described and formulated. In Section 4 new complexity results for the sequencing problem are presented. In Section 5 algorithms are proposed to solve the sequencing and scheduling problems. In section 6 a method is proposed to choose the number of scenarios. In Section 7 computational results are presented. In Section 8 conclusions and future research directions are discussed.

3 Literature Review

Relevant previous work can be divided in two categories; Stochastic Appointment Scheduling and Stochastic Integer Programming. We begin with previous work on the Stochastic Appointment Scheduling Problem. [2] found the optimal sequence when there are only 2 surgeries (convex order) and then showed that this criteria does not guarantee optimality when the number of surgeries is greater than 2. [7] assumed job durations were iid random variables following a Coxian (phase type) distribution. Since durations were iid, sequencing was irrelevant. He assumed costs for waiting time and total completion time. He developed an efficient numerical procedure to calculate mean job flow times then solved for the optimal scheduled starting times using non-linear programming. For examples with up to 10 jobs, he showed that even though job durations are iid, the optimal starting times are not equally spaced. [5] formulated the sequencing and scheduling problem as an integer stochastic program and then proposed simple heuristics to determine the sequence. Once a sequence is given, the schedule of starting times were found using a sample average approximation (scenario based) approach. The resulting scheduling problem is a linear stochastic program which they solved by an L-shaped algorithm described in [3]. To determine a sequence they proposed three

methods; sort by variance of duration, sort by mean of duration, and sort by coefficient of variation of duration. These simple heuristics were also compared to a simple interchange heuristic. They report results with real surgery time data and up to 12 surgeries (jobs). They also assumed equal penalty costs across surgeries for waiting and idle time. They found that sort by variance of duration gave the best results.

[8] assumed that job durations follow the Exponential distribution with different means and that patient arrivals can only be scheduled at finite times (every ten minutes). Their objective function included wait, idle and overtime costs. Given these assumptions, a queuing theory approach was used to calculate the objective function for a given schedule of starting times. They further proved that the objective function was multi-modular with respect to a neighborhood that can move the start time of jobs one interval earlier or later. This result guaranteed that a local search algorithm in this neighborhood will find the optimal solution. In [9] and [10] the authors also assumed discrete scheduled starting times (at 10 minute intervals over 3 hours) and included penalties for waiting time and overtime. They assumed three classes of patients where durations were iid within class but differed between classes. They used Phase Type and Lognormal distributions to model the three duration distributions. Given a sequence, they proposed a gradient based algorithm to find the optimal schedule of starting times based on submodularity properties of the objective function. They proposed an all-pairs swap based steepest decent local search heuristic to find a sequence. They stop the search after a fixed number of iterations or when a local minimum is found. They report testing with simulated data for cases with 4 and 6 customers and conclude that the heuristics produced good results in terms of iterations and optimality gap when compared with exhaustive enumeration.

The approach to the problem taken in this paper is Stochastic Integer Programming thus previous approaches to similar problems are relevant. There is a rich literature on Stochastic Integer Programming. In [11] an exhaustive review of methods for solving Stochastic Programming problems with integer variables was given. For the type of problem we address, there are several methods that might seem to apply. Our problem has both integer variables (for sequencing) and continuous variables (for scheduled starting times) in the first stage, continuous variables in the recourse function (waiting and idle times), and complete recourse. For finite scenario problems, [12] proposed the Integer L-Shaped Method, an algorithm that is suitable for solving Stochastic Programs where the first stage variables are binary and the recourse cost is easily computable. The method uses Benders' Decomposition combined with cuts that differ from traditional Benders' cuts. Another approach based on scenario decomposition was proposed in [13]. After decomposing the problem by scenarios, they then solve a problem with relaxed non-anticipativity constraints to get a lower bound within a branch and bound scheme. Finally there is the widely used Benders' decomposition approach. In Benders' approach one may decompose by fixing the integer variables or by fixing the set of first stage decisions. The problem we address is such that if the integer (sequencing) variables are fixed, the continuous (scheduling) variables can be computed with relative ease by solving a Linear Program using an interior point method. This makes Benders' decomposition particularly attractive for our problem. We also experimented briefly with the Integer L-shaped method but found that it offered no advantage over the Benders' approach in this case. Scenario decomposition is not appropriate for our problem since solutions to the single scenario problems provide no useful

information about the overall solution. This is because in any scenario subproblem, the starting times are simply set equal to the finish time of the previous job always resulting in zero cost.

The current literature also distinguishes between the infinite scenario problem and the finite scenario problem. For the infinite scenario case, two methodologies are potentially useful. In [14] the authors proposed an algorithm for solving the sample average approximation (finite scenario problem). They apply this procedure many times until stopping criteria related to statistical bounds are fulfilled. The other method aimed at the infinite scenario case is Stochastic Branch and Bound [15]. This method partitions the integer feasible space and computes statistical upper and lower bounds, then uses these bounds in the same way traditional branch and bound uses upper and lower bounds to find the true optimal value with probability one. Solving our sample average approximation problem turns out to be very time consuming, thus neither of these infinite scenario approaches are practically viable in our case.

4 Problem Statement

We assume a finite set of jobs (surgeries) with durations that are random variables. We assume these job durations have a known joint distribution, and are independent of the position in the sequence to which the job is assigned. Only one job may be performed at a time, and overtime is incurred when job processing extends past a deadline representing the length of the work day. Two sets of decisions must be made, first the sequence to perform the jobs must be determined, then a starting time must be assigned to each job. The starting time can be thought of as the time the patient is scheduled to arrive to the OR, thus a job may not begin before its scheduled starting time. The objective function consists of three components, waiting time (the time a patient must wait between his/her scheduled starting time and actual starting time), idle time (the time the OR is idle while waiting for the next patient to arrive) and overtime. Given a sequence, starting times for each surgery (job), and the duration distributions, the expected waiting time and idle time before each job and the over time can be estimated by averaging over a sample of scenarios. The objective function is a weighted linear combination of these three expected costs. Note that waiting and idle costs may be different for each job. This problem has been modeled as a two stage stochastic program with binary and continuous variables in the first stage decisions in [5]. They incorporated the processing time uncertainty into the model using a sample average approximation (i.e. scenario based) approach. The binary variables define which job (surgery) should be placed in the i^{th} position in the sequence. The starting times and the binary variables are all included in the first

stage decisions. This problem can be formulated as shown below. This model is similar to [5].

$$\text{minimize: } \sum_{k=1}^{K} \frac{1}{K} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} c_j^w w_{i,j}^k + c_j^s s_{i,j}^k + c^l l^k \right) \tag{1}$$

$$\text{s.t.: } t_i - t_{i+1} - \sum_{j=1}^{n} w_{i+1,j}^k + \sum_{j=1}^{n} s_{i,j}^k + \sum_{j=1}^{n} w_{i,j}^k = -\sum_{j=1}^{n} z_j^k x_{ij} \qquad i,k \tag{2}$$

$$t_n + \sum_{j=1}^{n} w_{n,j}^k - l^k + g^k = -\sum_{j=1}^{n} z_j^k x_{nj} + d \qquad k \tag{3}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j \tag{4}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i \tag{5}$$

$$s_{i,j}^k \leq M_1 x_{ij} \qquad i,j,k \tag{6}$$

$$w_{i,j}^k \leq M_2^i x_{ij} \qquad i,j,k \tag{7}$$

$$w_{i,j}^k \geq 0 \, , s_{i,j}^k \geq 0 \, \forall (i,j,k) \in (I,J,K) \; l^k \geq 0, \; g^k \geq 0 \, \forall k \in K, \; x_{ij} \in \{0,1\}$$

Indices and Sets

J	Jobs to be scheduled j=1,...,n.
I	Positions in the sequence i=1,...,n.
K	Scenarios to be considered k=1,....,K.

Parameters

c_j^w	waiting time penalty for surgery j.
c_j^s	idle time penalty for surgery j.
c^l	overtime penalty.
d	time beyond which overtime is incurred.
M_1, M_2^i	are sufficiently large numbers.
z_i^k	duration of surgery i in scenario k.

Variables

t_i	scheduled starting time for the surgery in position i.
w_{ij}^k	waiting time in scenario k when surgery j is in position i.
s_{ij}^k	idle time in scenario k when surgery j is in position i.
l^k	total time to complete all jobs (makespan) in scenario k.
g^k	slack variable that measures the earliness with respect to time d.
x_{ij}	a binary variable denoting the assignment of surgery j to position i.

Constraints

(2.2),(2.3)	define the waiting and idle time for every surgery and scenario.
(2.4),(2.5)	assure each surgery is assigned to one position in the sequence.
(2.6),(2.7)	are logical constraints that force waiting and idle times to be zero.

4.1 Properties of the formulation

Once the sequence is fixed the waiting and idle times and tardiness can be computed for every scenario as a function of the scheduled starting times as shown below ([5]). In the stochastic programming framework the computation of these variables can be seen as the recourse function. Further, any sequence will yield a finite objective function value therefore we have complete recourse. Thus we will only need optimality cuts to solve this problem using Benders' decomposition.

$$w_i^k = max\left\{w_{i-1}^k + z_{i-1}^k - t_i + t_{i-1}, 0\right\}, \quad i = \{2, ..., n\} \tag{8}$$

$$s_i^k = max\left\{-w_i^k - z_i^k - t_i + t_{i+1}, 0\right\}, \quad i = \{1, ..., n-1\} \tag{9}$$

$$l^k = max\{w_n^k + z_n^k + t_n - d, 0\} \tag{10}$$

The waiting time and scheduled starting time of the first surgery in the sequence are both assumed to be zero.

4.2 Strengthening the MIP Formulation

If we apply traditional branch and bound to the problem formulated in section 3, the weakness of the formulation caused by the big M constraints will negatively affect performance. Since we will compare the performance of our algorithms to branch and bound, it is important to strengthen the MIP formulation to the extent possible. Through straightforward but lengthy analysis we found values for M_1 and M_2^i that preserve optimality. Proof of the validity of the results may be found in appendix A. Here we simply state the results.

We set M_1 that appears in the slack time constraints (6) as follows:

$$M_1 = \max_{i \in I}\{max_{k \in K}\{z_i^k\} - min_{k \in K}\{z_i^k\}\}$$

We set the M_2^i values that appear in the waiting time constraints (7) as follows:

$$M_2^i = \sum_{j=1}^{i-1} \delta_j$$

where δ_j corresponds to the jth largest value in terms of $\max_{k \in K} z_r^k - \min_{k \in K} z_r^k$. With these "big M" values, the formulation can be somewhat tightened.

5 Problem Complexity

Previous authors ([5], [6]) have speculated that the sample average approximation sequencing and scheduling problem (SAA-SSP) is NP-Complete, but to the best of our knowledge the question is still open. In this section we first prove SAA-SSP with equal idle cost and equal waiting cost (EC) and sufficiently many scenarios is NP-Complete. Next we show that SAA-SSP with unequal costs and 2 scenarios is NP-Complete. The first proof uses concepts similar to those in [16]. The second proof follows easily from the first.

For problem EC, we first state the feasibility problem.

Sample average approximation sequencing and scheduling problem (SAA-SSP). Given a collection of jobs I indexed by i, with task lengths in scenario k given by $z_i^k \geq 0$, and a budget B of schedule cost, does there exist a sequence and schedule for I whose cost does not exceed B ?

In order to prove the complexity of this problem, we construct a polynomial transformation to the 3-Partition problem. The 3-Partition problem as defined below is know to be NP-Complete [16].

Definition 1. *3-Partition: Given positive integers n, R, and a set of integers A =*$\{a_1, a_2, ..., a_{3n}\}$ *with* $\sum_{i=1}^{3n} a_i = nR$ *and* $\frac{R}{4} < a_i < \frac{R}{2}$ *for* $1 \leq i \leq 3n$*, does there exist a partition* $\langle A_1, A_2, ., ...A_n\rangle$ *of A into 3-elements sets such that, for each i* $\sum_{a \in A_i} a = R$ *?*

Theorem 2. *The SAA-SSP problem is NP-Complete.*

We show that 3-partition polynomially reduces to a given sample average approximation sequencing scheduling problem (SAA-SSP). The idea is to create an instance of SAA-SSP with 4n jobs and K_1+K_2=K= $\left\lceil \frac{20nR+50n^2R^2}{4+10nR} \right\rceil$ scenarios, in which the first 3n jobs have durations related with the partition A, while the remaining n jobs have a convenient duration such that scheduling leads to the solution of the 3-Partition problem. The cost penalties are chosen as $c_i^s = 4nK$, c_i^w=1 $\forall i \in \{1, .., 4n\}$. The budget of the schedule cost is B=$\frac{5K_1nR}{2K}$. We define 2 sets of jobs, G and D:

- G: jobs g_i in G are defined for 1$\leq$ i $\leq$ 3n. Jobs g_i have duration a_i in scenarios 1,..,K_1 and duration zero for scenarios K_1+1,..,K.
- D: jobs d_i in D are defined for 3n+1$\leq$ i $\leq$ 4n. Jobs d_i have duration H for scenarios 1,..,K_1 and H+R for the scenarios K_1+1,..,K.

The number of scenarios K=$K_1 + K_2$, where K_1= $\left\lceil \frac{10nR}{2+5nR} \right\rceil$ and K_2= $\left\lceil \frac{50n^2R^2}{4+10nR} \right\rceil$.

Intuitively, the main idea of the proof can be seen in Figure 1. First the idle cost is set high enough that the starting times will always result in no idle time in an optimal solution. Next, if the schedule is not the "3-Partition schedule" as shown in Figure 1, then there will be waiting time penalties for "D" jobs in the K_2 scenarios. We make K_2 large enough that this waiting time causes the budget to be exceeded.

Lemma 3. *The optimal scheduled starting times for the set of 4n jobs with cost penalties* c_i^s*=4nK,* c_i^w*=1* $\forall i \in \{1, .., m\}$ *is given by:* $t_i^* = min_{k \in K}\{t_{i-1}^* + w_{i-1}^k + z_{i-1}^k\}$ *(that is the schedule contains no idle time).*

For a detailed proof of this lemma (which is based simply on setting the idle costs high enough), see appendix B.

Lemma 4. *For any given sequence of problem SAA-SSP, the optimal schedule leads to a solution that has an integer valued objective function.*

Proof. By lemma 3 we know that the starting times follow a recursive formula that depends on waiting time and durations of jobs earlier in the sequence and we know that we can express waiting time using (8). Since, the durations

are integers, and the integers are closed under addition and subtraction, we can conclude that all waiting times are integer (and all idle times are zero). Since waiting costs are also integer, the objective function value of the optimal solution to the scheduling problem is an integer. □

The proof of theorem 1 will have two parts, first we prove that the solution of the 3-Partition problem defines all the optimal solutions of the SAA-SSP. Second, we show that the optimal solution of the SAA-SSP finds the perfect 3-Partition if there is one.

We will start by showing that the solution of the 3-Partition problem leads to a feasible solution of SAA-SSP. Assume there is a partition $\langle A_1, A_2, ..., A_n \rangle$ that fulfills the condition $A_i = \{a_{t(i,1)}, a_{t(i,2)}, a_{t(i,3)}\} \sum_{j=1}^{3} a_{t(i,j)} = R$, $1 \leq i \leq n$. We create the jobs G and D and then construct the desired sequence and schedule as shown in Figure 1.

The jobs in G have been scheduled between the D jobs following the pattern shown in Figure 1. We set all the starting times at the beginning of each block resulting in no idle time. Recall that in scenarios $K_1 + 1, ..., K$ the G jobs all have zero duration. The schedule cost for this sequence can be computed in straightforward fashion to be,

$$z^* = \frac{1}{K} \sum_{k=1}^{K} (\sum_{g \in G} c_g^w w_g^k + c_g^s s_g^k + \sum_{d \in D} c_d^w w_d^k + c_d^s s_d^k) = \frac{K_1}{K} (nR + 2 \sum_{i=1}^{n} a_{t(i,1)} + \sum_{i=1}^{n} a_{t(i,2)})$$

Since, the a_i are bounded above by $\frac{R}{2}$ the schedule cost is bounded above by $\frac{5K_1 nR}{2K}$, therefore this sequence and schedule fulfills the budget B.

The second part of the proof shows that any sequence and schedule that does not exceed the budget B contains a perfect partition as shown in Figure 1. First, we will show that only a sequence and schedule following the pattern shown in figure 1 fulfills the budget B.

Claim 1: Any sequence that does not have 3 G jobs between 2 D jobs will exceed the budget. In order to show this claim we will divide the proof in cases. First, we will show the case when we have less than 3 G jobs between 2 D jobs. We will show that 2 G jobs between D jobs is suboptimal. These results can be extended in straightforward manner to the cases with 0 or 1 G job between D jobs. For this case we further divide into 4 cases where the starting time of the first D job is defined by the K_1 scenarios (Case 1.1) or K_2 scenarios (Case 1.2 and Case 1.3) or when the starting time is the same for all the scenarios (Case 1.4). The impact of every case is reflected in Figure 2. Since, the smallest ϵ is 1 (Lemma 3) we can compute the schedule cost for every case.

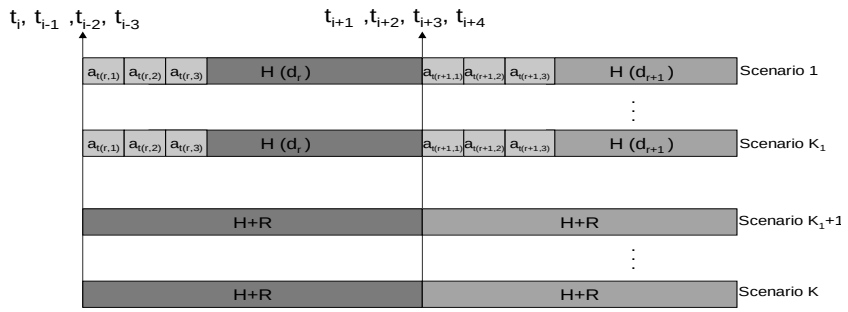

Figure 1: Constructed Sequence

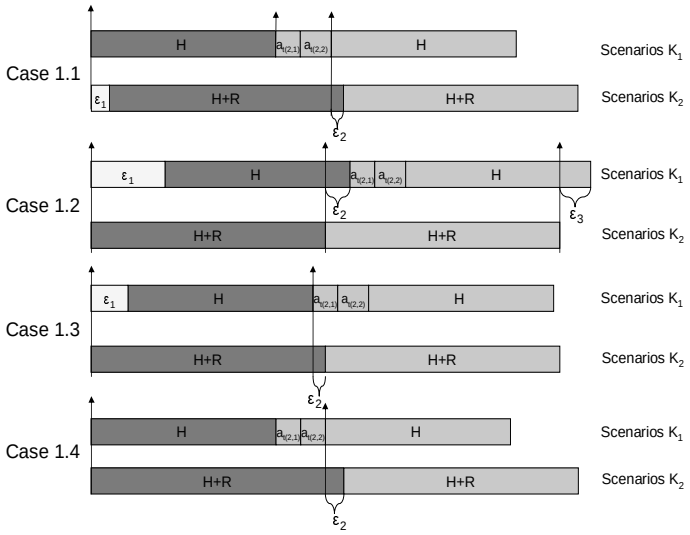

Figure 2: Different Cases

$$z = \begin{cases} W_S + K_2(\epsilon_1 + \epsilon_2) = W_S + \frac{50n^2R^2}{2+5nR} > \frac{5nR}{2+5nR} = B & \text{Case 1.1} \\ W_S + K_1(\epsilon_1 + \epsilon_2 + \epsilon_3) + K_2\epsilon_r > \frac{K_1}{2} = B & \text{Case 1.2} \\ W_S + K_2\epsilon_1 > \frac{K_1}{2} = B & \text{Case 1.3} \\ W_S + K_2\epsilon_2 > B & \text{Case 1.4} \end{cases}$$

Note, in the above equations we used the following equivalences.

$$B = \frac{5nRK_1}{2K} = \frac{K_1}{2} = \frac{5nR}{2+5nR}$$

For case 1.2 there must exist a job r with positive waiting time in the K_2 scenarios since otherwise there would be idle time.

Case 2 where we have more than three jobs between D jobs can be proven by similar arguments.

Claim 2: The sequence consisting of 3 G jobs followed by a D job is superior to a sequence with one D job followed by 3 G jobs. The duration of the jobs g are zero for scenarios K_1+1,..K thus the optimal sequence is to assign them at the beginning of the each block. Otherwise, these jobs will incur waiting time in scenarios K_2 causing the budget to be exceeded.

Given that every optimal sequence and schedule must follow the pattern in Figure 1, we next show that only schedules built on a perfect 3-Partition fulfill the budget. We show this by induction on the number of blocks.

First, we consider the case with only two blocks. If we have a sequence and schedule not built on a perfect partition we have 2 cases as shown in figure 3.

The cost of schedule (W) exceeds the budget B in each case as seen in the following.

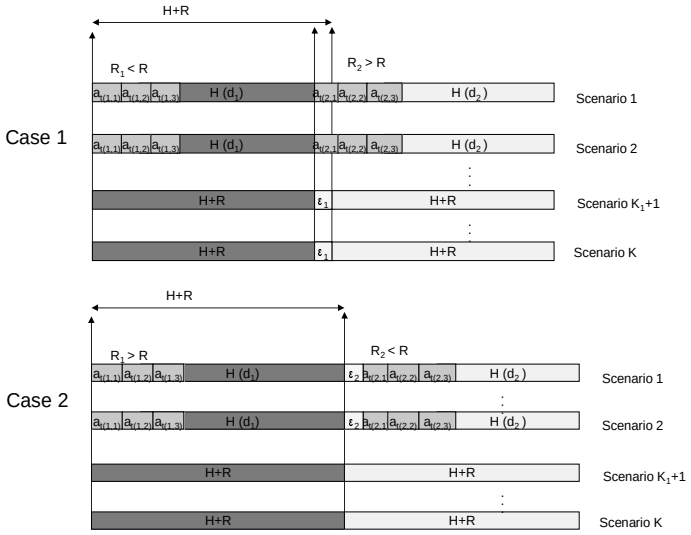

Figure 3: The Two Block Case

$$W = \begin{cases} W_{B_1} + K_1(3\epsilon_2 + 2a_{t(2,1)} + a_{t(2,2)} + R_2) & \text{if } R_2 < R \\ W_{B_1} + (2a_{t(2,1)} + a_{t(2,2)} + R_2 - 3\epsilon_1)K_1 + K_2\epsilon_1 & \text{if } R_2 > R. \end{cases}$$

where W_{B_1} is the waiting time incurred by the second and third job in the first block.

When the schedule is based on a perfect partition, that is $R_1 = R_2 = R$ the objective function value is $W_{B_1} + (2a_{t(2,1)} + a_{t(2,2)} + R_2)K_1$ which meets the budget. Therefore, the solution of the 3-Partition problem leads to the only schedule that meets the budget and thus is optimal for the SAA-SSP problem when we have two blocks. We have proven the 2 block case, and assuming that this true for the n block case, we will show that must be true for the case with n+1 blocks. We will compute the waiting time for the last block in the case n+1 blocks.

$$W_{n+1} = \begin{cases} K_1(3\epsilon_{n+1} + 2a_{t(n+1,1)} + a_{t(2,2)} + R_{n+1}) & \text{if } R_{n+1} < \text{R.} \\ K_1(2a_{t(n+1,1)} + a_{t(n+1,2)} + R_{n+1} - 3\epsilon_n) + K_2\epsilon_n & \text{if } R_{n+1} > \text{R.} \\ W_{B_1} + K_1(2a_{t(n+1,1)} + a_{t(n+1,2)} + R_{n+1}) & \text{if } R_{n+1} = \text{R.} \end{cases}$$

The schedule cost is the summation of the contribution of every block, therefore:

$$W = \sum_{i=1}^{n+1} W_i$$

By assumption, we know that the optimal schedule when we have n blocks is reached when we have the optimal 3-Partition. This implies that the best value for the first n blocks is reached when we have the solution of the 3-Partition for the first n blocks. This implies that $R_{n+1} = R$ because $\sum_{i=1}^{3(n+1)} a_i = (n+1)R$ by assumption. Therefore, the optimal value of the SAA-SSP is reached when we have the solution of the 3-Partition problem. With this we conclude the proof.

Theorem 5. *The Sample Average Approximation Sequencing and Scheduling Problem with two scenarios is NP-Complete when the idle cost and waiting cost are allowed to be different for every surgery.*

We will provide only a sketch of the proof for this theorem since the actual proof is very close to the previous one. We construct a set of jobs (D and G) exactly as in the previous proof but with different costs. In this case K_1=1 and $K_2 = 1$ and the waiting cost for the D jobs is set to $\frac{5nR}{2}$. The result is a scheduling problem with the same properties as in the previous proof, except with a slightly different sequence pattern than the one shown in figure 1. In this case, the optimal sequence will follow the pattern shown in figure 4. With the prescribed waiting cost for the D jobs, any wait caused in these jobs will cause the budget to be exceeded as in the previous proof.

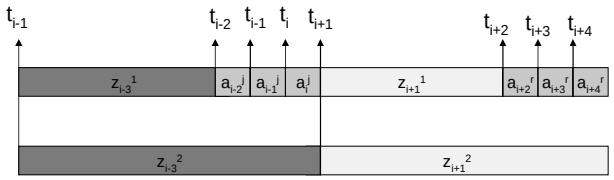

Figure 4: 2 Scenarios Case

6 Proposed Solution Methodology

The approach we propose for the SAA-SSP uses a heuristic method to find good solutions in a reasonable amount of time. The Master problem in our Benders' Decomposition is an integer program thus is difficult to solve and even more difficult when we add cuts in every iteration. Relaxing the side constraints results in an easy to solve assignment problem. We use this property to construct a heuristic to generate good feasible solutions to the master problem. The idea of solving the master problem heuristically has also been proposed by several authors such as [17] and [18].

6.1 Proposed Algorithm

The basic outline of our algorithm is as follows; start with an arbitrary sequence. Set the Upper Bound (UB)=∞ and Lower Bound (LB)=-∞.

1. Solve the LP scheduling subproblem for the current sequence yielding z^*, x^*, p^*.
2. Update the UB if z^* <UB.
3. Generate a Benders' cut from the optimal extreme point in the dual (p^*).
4. Use our simplified master problem heuristic to try to find a feasible solution to MP.
5. If the new sequence is the same as the previous sequence, restart the heuristic.
6. If (number of iterations $<$ max iterations) go to 1, else stop.

6.1.1 The LP Scheduling Subproblem

Given a sequence, finding the optimal scheduled starting times is known to be a linear program ([3], [5]). This is the LP subproblem in step one above. The size of this LP grows rapidly with the number of scenarios. [3] proposed solving this problem using the L-Shaped method. We developed a network flow like approach based on the fact that the dual is a network flow problem with side constraints. For problems with 2500 scenarios and 10 surgeries the network flow based method was superior to both the Standard L-Shaped Method and Standard Dual Simplex. However, further testing revealed that interior point methods were at least one order of magnitude faster than any of the other methods. Thus, we use the Cplex barrier algorithm to solve the subproblems within the algorithm. The LP subproblem which, for a given sequence, finds scheduled starting times that minimize total cost averaged over the scenarios is given below.

$$\min : \sum_{k=1}^{K} \frac{1}{K} \left(\sum_{i=1}^{n} c_i^w w_i^k + c_i^s s_i^k + c^l l^k \right)$$

$$\begin{aligned}
&\text{s.t.: } t_i - t_{i+1} - w_{i+1}^k + s_i^k + w_i^k && = -z_i^k \qquad \forall i \in I \, \forall k \in K \\
&t_n + w_n^k - l^k + g^k && = d - z_n^k \quad \forall k \in K \\
&w_i^k, s_i^k \geq 0 \, \forall (i,k) \in (I,K), \; l^k, g^k \geq 0 \, \forall k \in K, \; t_1 \geq 0
\end{aligned}$$

This is the dual of the scheduling problem,

$$\max : -\sum_{k=1}^{K} \sum_{i=1}^{N} f_i^k z_i^k - \sum_{k=1}^{K} f_n^k (d - z_n^k)$$

$$\begin{aligned}
\text{s.t.} \quad & \textstyle\sum_{k \in K} f_1^k \leq 0 \\
& \textstyle\sum_{k \in K} f_i^k - \sum_{k \in K} f_{i+1}^k = 0 && \forall i \in I \\
& f_{i+1}^k - f_i^k \leq \tfrac{1}{K} c_i^w && \forall k \in K \\
& f_i^k \leq \tfrac{1}{K} c_i^s && \forall i \in \{1, .., n-1\} \\
& -\tfrac{1}{K} c^l \leq f_n^k \leq 0 && \forall k \in K.
\end{aligned}$$

6.2 The Master Problem in Benders' Decomposition

We use a straightforward application of Benders' Decomposition on the mixed integer stochastic program defined in section 3. The master problem contains the integer sequencing variables (x_{ij}). The master problem formulation appears below.

Definition 6. *The master problem at iteration T in Benders' Decomposition.*

$$\text{minimize: } z_M = z$$

$$
\begin{aligned}
\text{s.t.:} \qquad \sum_{i=1}^{n} x_{ij} &= 1 \quad \forall j \in \{1..n\} \\
\sum_{j=1}^{n} x_{ij} &= 1 \quad \forall i \in \{1..n\} \\
\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^{t} x_{ij} &< z \quad t \in \{1,..,T\} \qquad (11) \\
z &< UB_T \\
x_{ij} &\in \{1,0\}
\end{aligned}
$$

Where

$$
x_{ij} = \begin{cases} 1 & \text{if we assign the surgery j to position i in the sequence} \\ 0 & \text{Otherwise} \end{cases}
$$

The constraints (11) are the Benders' optimality cuts. The coefficients of these constraints come from the inner product between the dual variables (f_i^k) and the surgery durations in the LP subproblem. These cuts contain information found from solving the LP subproblems and allow one to implicitly eliminate sequences that will not improve the objective function.

To assure convergence of the iterative Benders' solution approach, all that is needed is a feasible integer solution to the master problem at each iteration. However, since this problem is computationally hard to solve, we construct a simplified version of the master problem that is easy to solve, but does not guarantee a feasible solution to the master problem.

6.3 Simplified Master Problem

We remove the side constraints (Benders' cuts) and create a new objective function based on the coefficients of these constraints. The resulting simplified master problem (SMP) is an assignment problem and thus can be easily solved.

$$
\text{minimize: } z_{SMP} = \sum_{i=1}^{n} \sum_{j=1}^{n} \overline{a}_{ij} x_{ij}
$$

$$
\begin{aligned}
\text{s.t.:} \quad \sum_{i=1}^{n} x_{ij} &= 1 \; \forall j \in \{1..n\} \\
\sum_{j=1}^{n} x_{ij} &= 1 \; \forall i \in \{1..n\} \\
0 \leq x_{ij} &\leq 1
\end{aligned}
$$

The idea is to capture the information in the cut constraints in such a way that we can find a feasible solution to the master problem with reasonable reliability. We experimented with several methods for constructing these objective function coefficients $\overline{a}_{ij}$ from the optimality cut constraints. Based on this experimentation the best method was found to be $\overline{a}_{ij} = max_{t \in \{1..T\}}\{a_{ij}^t\}$. The desirability of using the max operator to aggregate the constraints is supported by the following proposition.

Proposition 7. *The optimal solution to the SMP is an upper bound of the master problem in Benders' Decomposition. Furthermore, if the optimal objective function value of the SMP is less than the current upper bound, the optimal solution is a feasible solution to the master problem.*

Proof. In the the first iteration (i.e. with one side constraint), it is clear that the simplified problem finds a feasible solution to the master problem. In subsequent iterations with more than one side constraint the master problem can be defined as follows,

$$\min_{\rho \in S} \max_{t \in T} \{F_t(\rho)\}$$

Where S is the set of sequences and F_t is the left hand side in constraint t for sequence ρ in the Master Problem. Given a sequence ρ, exactly n x_{ij}'s will be equal to 1. Let $a^t_{l_\rho}$ l_ρ=1,..,n be the coefficients in constraint t for the x_{ij}'s that are equal to 1 under sequence ρ.

$$F_t : S \mapsto \mathbb{R}$$
$$F_t(\rho) = \sum_{l_\rho=1}^{n} a^t_{l_\rho}$$

Thus the Master Problem can be written as the summation of coefficients, and in this form we can see that

$$\min_{\rho \in S} \max_{t \in T} \{\sum_{l_\rho=1}^{n} a^t_{l_\rho}\} \Rightarrow \min_{\rho \in S} \max_{t \in T} \{\sum_{l_\rho=1}^{n} a^t_{l_\rho}\} \leq \min_{\rho \in S} \{\sum_{l_\rho=1}^{n} \max_{t \in T} \{a^t_{l_\rho}\}\}$$

Thus the solution of the SMP is an upper bound to the master problem. □

If this upper bound is less than the current bound, we have a feasible solution to the Master Problem. Of course there is no guarantee that the upper bound provided by the SMP improves the overall upper bound. When $z_{SMP} \geq z_M$ two things can to happen, (1) we get a new sequence in which case we continue iterating or, (2) we get the same sequence in which case the algorithm will produce this same sequence on subsequent iterations (i.e. we are stuck). When the algorithm gets stuck we restart the algorithm from a new sequence using one of the methods described in the next section.

6.4 Restart Rules

Once the algorithm returns the same sequence for two consecutive iterations, it will continue to do so ad infinitum, therefore, we need to restart from a new sequence. Given the new restart sequence, we simply remove all previous Benders' cuts and start again. We tried three restart rules as discussed below.

6.4.1 Worst Case

This anti-cycling rule is based on finding a sequence "far way" from sequences visited since the last restart. To accomplish this we simple replace "minimize" with "maximize" in the SMP objective function.

6.4.2 Perturbation

This anti-cycling rule tries to slightly perturb the sequence in order to escape the cycle.

$$\text{minimize: } z_{A_r} = \sum_{i=1}^{n} \sum_{j=1}^{n} (\overline{a_{ij}} + a_{min} U(0,1)) x_{ij}$$

$$\begin{aligned} \text{s.t.:} \quad & \sum_{i=1}^{n} x_{ij} = 1 \; \forall j \in \{1..n\} \\ & \sum_{j=1}^{n} x_{ij} = 1 \; \forall i \in \{1..n\} \\ & 0 \leq x_{ij} \leq 1 \end{aligned}$$

where $\overline{a_{ij}} = max_{t \in \{1..T\}}\{a_{ij}^t\}$ and $a_{min} = min_{i,j}\overline{a_{ij}}$ and U(0,1) represents pseudo-randomly generated Uniform(0,1) deviates. The value of a_{min} was chosen so that the sequence will not change much. Unfortunately, this rule does not guarantee a new restart sequence difference from all previous restart sequences. Thus we may repeat a previous iteration. To avoid this we developed the next method.

6.4.3 Memory Random Restart

This anti-cycling rule is based on finding a sequence different from all previous restart sequences. We store all restart sequences $R_1, .., R_q$ where q is the number of times that we have restarted the algorithm. The idea of this restart rule is to guarantee that we are not going to cycle between restarting points and therefore the algorithm will visit at least one new sequence each iteration. We formulated the following feasibility IP (M_q).

$$\begin{aligned} \text{s.t.:} \quad & \sum_{i=1}^{n} x_{ij} = 1 && \forall j \in \{1..n\} \\ & \sum_{j=1}^{n} x_{ij} = 1 && \forall i \in \{1..n\} \\ & \sum_{x_i^r=1} x_i - \sum_{x_i^r=0} x_i \leq \psi && \forall r \in \{R_1, .., R_q\} \\ & x_{ij} \in \{1, 0\} \end{aligned}$$

where ψ is random number generated from IntUniform(0,n-1). We use Cplex to find a feasible solution. Cplex finds a feasible solution extremely quickly as it turns out, so that the impact on the algorithm's execution time is minimal. The constraints that contain the previous restart points guarantee that we will not restart the next iteration from these previous sequences. Further, the random number ψ serves as a kind of distance from the set of restarting points where when ψ=n-1 we might obtain a feasible sequence for M_q that differs from restart sequences R_r in at most 2 elements. When ψ=0 the new restart sequence will differ from every previous restart point in all the elements.

7 Accuracy of the Finite Sample Average Approximation

The ultimate goal is to solve the infinite scenario stochastic programming problem. The method proposed in this research attempts to solve a finite scenario problem. Further, the method does not guarantee an optimal solution to this finite scenario problem. In this section we will try to evaluate how our algorithm will perform on the infinite scenario problem. [19] explain a way to compute statistical upper and lower bounds on the optimal solution to the infinite scenario case based on external sample techniques. Unfortunately, this method requires solving a finite scenario problem many many times and thus is computational prohibitive in our case, even for a small number of scenarios. We therefore designed a simpler experiment to quantify the performance of our algorithm on the infinite scenario case. There are two main issues. The first issue is sampling error, that is: "How well is the infinite scenario objective

function approximated by the finite scenario (sample average) objective function?" The second issue is: "How does the run time of the algorithm affect performance with respect to the infinite scenario problem?" There is a basic tradeoff we need to evaluate. For a fixed computation time allowance, how many scenarios should we use? If we use many scenarios, we will only have time to generate a few candidate sequences thus limiting our ability to "optimize". On the other hand with few scenarios, we can generate many sequences, and perhaps even solve the finite scenario problem to optimality. However, the optimal solution to the finite scenario problem may be a poor solution to the infinite scenario problem when the number of scenarios is small. To quantify this tradeoff we constructed the following experiment. We generated 50 test problems with 10 jobs each. For each of these 50 problems we generated finite scenario instances with 50, 100, 250, and 500 scenarios. For each of these 400 instances we then ran the algorithm for 20,40,60,80,100,120,140 seconds (the experiments were conducted in the same computational conditions detailed in section 7). For each run of the algorithm we saved the best ten sequences where "best" is with respect to the number of scenarios used in the current run. This resulted in the generation of 7x4x10=280 (not necessarily all different) total sequences for each of the 50 test problems. For each of these sequences, we solved the LP scheduling sub-problem with 10,000 scenarios and reported the best sequence S^*_{10000} and objective function z^*_{10000} found. This serves as our approximation to an overall best solution to the infinite scenario problem for each of the 50 test cases. Our first set of results is aimed at understanding the sampling error. The basic question we asked is: "How often does the sequence our algorithm thinks is best turn out to be the best sequence in the 10,000 scenario case?". For each of the "10 best" sequences returned by the algorithm, the plots below show the cumulative frequency for which that solution was best for 10,000 scenarios. For example figure 5 shows that for 500 scenarios, the solution judged to be the best of the top ten by the algorithm was indeed the best of the top ten 40% of the time. By examining the plot we see that 50 and 100 scenarios incur significant sampling error since each of the top ten solutions has essentially the same probability of being best for the 10,000 scenario case. With 500 scenarios, we see that solutions that are good for the finite scenario problem also tend to be good for the 10,000 scenario problem. We can conclude that 100 scenarios does not provide a sufficient approximation to the 10,000 scenario case. 500 scenarios seems to provide a reasonable approximation while 250 is borderline.

The second set of results demonstrates the tradeoff between the number of scenarios and the number of sequences generated. For each number of scenarios and run times we take the best sequence (according to the algorithm), evaluate it with 10,000 scenarios using the LP scheduling sub-problem, then compute the percent error from z^*_{10000}. The percent error is averaged over the 50 test instances and shown in the plot below. Figure 6 shows that 250 scenarios has the best performance overall. The 50 and 100 scenario cases do not perform well for any run time because, as shown previously, they do not approximate the 10,000 scenario problem well. After 140 seconds, the 500 scenario case has "caught up with" the 250 scenario case. If one were to run the algorithm for more than 140 seconds, 500 scenarios would likely be the better choice.

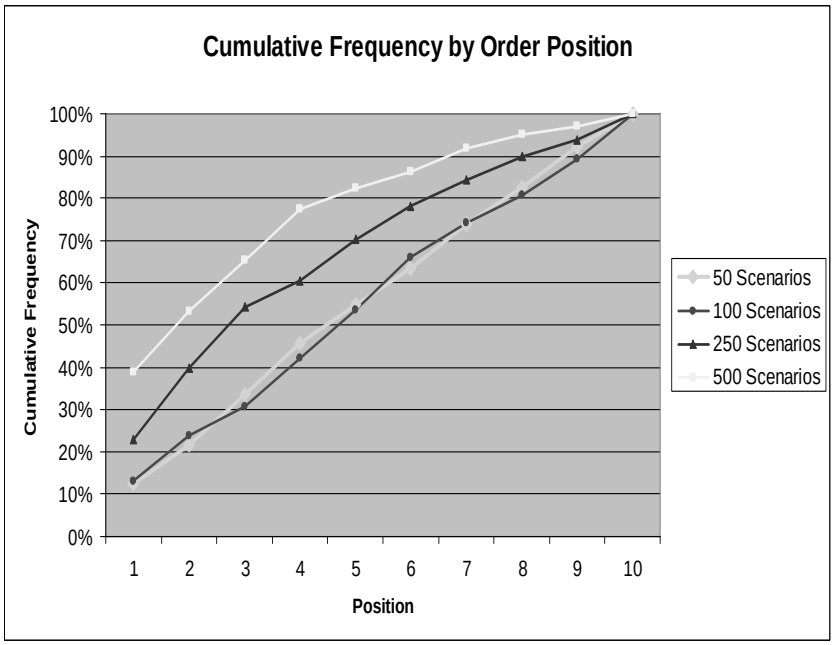

Figure 5: Cumulative Frequency by Order Position

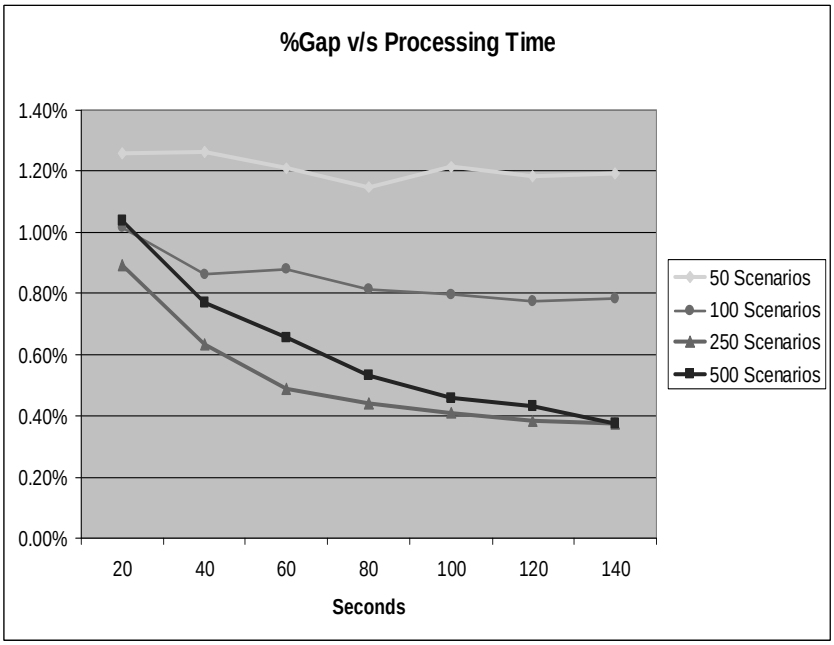

Figure 6: Percent Error Results

8 Computational Experience for the Finite Scenario Case

To quantify the performance of our algorithms for solving the finite scenario problem we performed two sets of experiments. In the first we compared solutions from the algorithms with the optimal solution on smaller problems. In the second we compared algorithm performance with the two simple benchmark heuristics over a broader set of test problems. The experiments were conducted on a Pentium Xeon 3.0 GHz (x2) with a 4000 (MB) server in the Coral Lab at Lehigh University.

8.1 Comparison with Optimal Solutions

In this section, we compared the solutions found by our memory restart algorithm to the optimal solution found by Cplex using branch and bound. We generated two categories of test problems, those with equal costs and those with different costs. It is worth noting that the cases with equal cost were significantly easier for Cplex to solve to optimality than were the cases where the costs were different. We generated 10 instances with 10 scenarios and 10 instances with 100 scenarios in each category. In the next section we discuss in detail how we generated a large suite of test problems by varying a variety of factors. In this section we selected instances so as to span a wide range of these factors.

To find optimal solutions we used Cplex 10.2 to solve the strengthened IP formulation discussed in subsection 3.2. Within Cplex the "MIP emphasis" parameter was set to "optimality". We ran our memory restart algorithm for 1000 and 4000 iterations for each test problem. Tables 1 and 2 show the results of this experiment. The run time results are the average run time per problem instance. Note that in some cases Cplex took several days to find the optimal solution. It is interesting to note that the optimality gap seems to be smaller for the problems with more scenarios. One possible explanation for this behavior is that as the number of scenarios increases, the number of iterations between restarts of our algorithm also increases (see Table 8).

#Inst.	#Surgeries	Scenarios	Iter.	Opt. Gap (%)	ABenders(sec)	Cplex (hours)
10	10	10	1000	2.20	7.3	0.88
10	10	10	4000	0.84	47.3	0.88
10	10	100	1000	0.90	44	126.73
10	10	100	4000	0.48	183.2	126.73

Table 1: Optimality Gap Different Cost Case

8.2 Comparison with Benchmark Heuristics

8.2.1 Benchmark Heuristics

We implemented two simple heuristics for comparison purposes. The first is the "sort by variance" heuristic proposed by [5]. This simply sequences jobs from smallest to largest variance. This heuristic can be expected to work fairly

#Inst.	#Surgeries	Scenarios	Iter.	Opt. Gap (%)	ABenders(sec)	Cplex (hours)
10	10	10	1000	5.38	7.3	0.078
10	10	10	4000	3.65	47.3	0.078
10	10	100	1000	1.00	44	13.67
10	10	100	4000	0.80	183.2	13.67

Table 2: Optimality Gap Equal Cost Case

well in the equal costs case (i.e. costs are equal across surgeries), but there is no reason to expect it to work well in the unequal costs case. The second benchmark heuristic was a simple perturbation heuristic based on sort by variance (this kind of heuristics were proposed by [20]). A randomly generated perturbation was added to each job's variance, then jobs were sequenced from smallest to largest perturbed variance. The perturbations were generated from 2U$(-\theta, \theta)$ where θ=$\frac{\max_{j\in J}\sigma_j - \min_{j\in J}\sigma_j}{|J|}$. We generated the same number of sequences that were generated by the Benders' based heuristics and reported the best one found. Since computation time is dominated by the time required to solve the LP subproblems, the run times were roughly equivalent.

8.2.2 Test Problems

We created a set of test problems based on factors that might affect algorithm performance. The factors chosen were: number of surgeries, presence of overtime cost, wait and idle cost structure , number of scenarios, and surgery time distribution. The number of iterations was fixed at 1000 (we study the affects of number of iterations in the second experiment to follow).

Factor	Possible Value
Number of Surgeries	10, 15, 20
Overtime Cost	Yes , No
Cost Structure	equal waiting and equal idle costs, different costs
Number of Scenarios	10, 50, 100, 250, 500
Surgery Distribution	$(\mu, \sigma_i)_1, (\mu, \sigma)_2, (\mu_i, \sigma)_3, (\mu_i, \sigma_i)_4$

Table 3: Experiment Design

Please, note that the indices 1,2,3,4 in the possible values of the factor Surgery Distribution are used as the labels of Data in Figure 7.

The means and variances of surgery duration distributions were based on real data from a local hospital. We then generated simulated surgery times from truncate normal distributions with parameters reflected in the real data. In table 3 under surgery distribution the symbol μ,σ means all surgery durations were generated using the same mean and standard deviation (186 , 66). The symbol μ_i,σ means that σ was set at 66 and μ_i was set based on the coefficient

of variation generated from a uniform(0.21,1.05)distribution.

The symbol μ,σ_i means that μ was set at 186 then σ_i was set based on the coefficient of variation generated from a uniform(0.21,1.05) distribution.

The symbol μ_i,σ_i means that μ_i was first generated from uniform(90,300) distribution then σ_i was set based on a coefficient of variation generated from a uniform(0.21,1.05) distribution.

For the cost factor, in the equal cost case we generated a single waiting cost and idle cost each from a uniform (20,150) distribution. These two costs were then applied to every surgery. In the unequal cost case individual idle and waiting costs were generated for each surgery, again from a uniform (20,150) distribution.

When overtime is included, the over time cost is set to 1.5 times the average of the waiting cost. The deadline was set equal to the sum (over surgeries) of the average (over scenarios) duration plus one standard deviation (over scenarios) of this sum.

We created a full factorial experimental design and performed 5 replicates for each combination of factor levels. This resulted in 1200 instances for each of the five algorithms tested: Approximate Benders' Decomposition(with the three different restart rules), sort by variance, and perturbed sort by variance with 1000 iterations. We generated 1000 (not necessarily unique) sequences for each algorithm (except sort be variance), solved the LP sub-problem for each to get the objective function, and report the best solution found. Thus for each problem instance we have five solutions, one for each heuristic. We take the best solution of these five, then compute the percent gap from this best solution for each heuristic for each problem instance. The overall results appear in tables 4 and 5.

Algorithm v/s Scenarios	10	50	100	250	500	% Gap
Approximate Benders' (RS: Perturbation)	4.7	2.5	1.5	1.0	0.7	2.1
Approximate Benders' (RS: Worst Case)	3.5	1.8	0.7	0.5	0.4	1.4
Approximate Benders' (RS: Memory Random)	1.2	1.2	0.3	0.2	0.3	0.7
Sort by Variance	32.1	17.0	7.3	4.9	2.9	12.9
Perturbed Sort by Variance	17.0	10.3	3.6	3.2	1.7	7.2

Table 4: Average percent gap over the best solution found v/s scenarios (equal cost case)

Algorithm v/s Scenarios	10	50	100	250	500	% Gap
Approximate Benders' (RS: Perturbation)	5.9	2.9	2.0	1.6	1.4	2.8
Approximate Benders' (RS: Worst Case)	3.9	1.9	1.0	0.7	0.7	1.6
Approximate Benders' (RS: Memory Random)	1.2	1.0	0.5	0.5	0.5	0.7
Sort by Variance	28.5	15.1	12.7	11.8	10.3	15.7
Perturbed Sort by Variance	13.7	7.5	5.5	4.9	3.6	7.1

Table 5: Average percent gap over the best solution found v/s scenarios (different cost case)

The results show that all three Benders' based algorithms significantly out perform the simpler heuristics. We performed an analysis of variance and graphed interaction plots to see how algorithm performance was affected by the experimental factors. The interaction plots which appear in figure 7 show that the performance of each algorithm was reasonably uniform over the different factors.

Tables 4 and 5 show the algorithm performance as the number of scenarios varies. It is interesting to note that as the number of scenarios increases, the difference in performance decreases. In particular for the equal costs case, the simple sort by variance heuristic performs quite well compared with the other methods. Since the ultimate goal is to solve the infinite scenario problem, it would seem that sort by variance is an effective heuristic in the case of equal costs. When costs are not equal, significant improvement over sort by variance is possible.

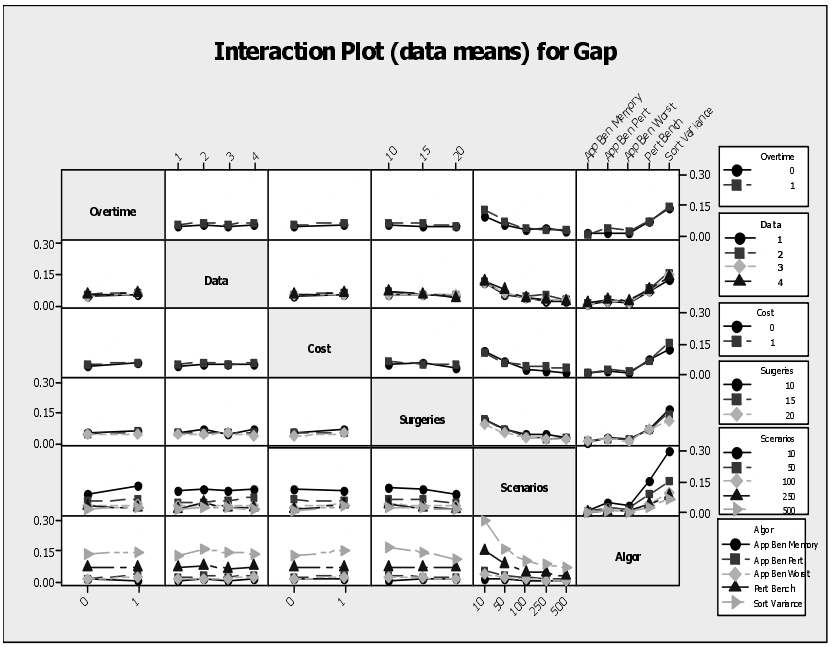

Figure 7: Interaction plots for experiment 1

The previous experiment fixed the number of iterations at 1000. We conducted a second experiment which examined the performance of the three algorithms over a varying number of iterations. In this experiment the factors chosen were: number of surgeries, number of scenarios and number of iterations (1000, 2000, 3000,4000). We further assume no overtime cost, the distributions of the job durations are from the case where σ_i and μ_i have no restrictions, and the cost coefficient are not equal. The results of the analysis of variance show that all main effects (algorithm, iterations, scenarios, and number of surgeries) are statistically significant. In particular the "memory" restart heuristic was (statistically) significantly better than the other two. Interestingly, no two factor or higher interactions were significant implying that algorithm performance behaved in a very uniform manner across the other factors. From this we conclude that the memory heuristic appears to be the best in a statistical sense in all cases, although the actual difference in performance is fairly small. The interaction plots in figure 8 illustrate these results. We also report the average run times for the memory algorithm and the average number of restarts in Table 7.

Factor	Possible Value
Number of Surgeries	10, 15, 20
Number of Scenarios	10, 50, 100, 250, 500
Number of Iterations	1000, 2000, 3000, 4000

Table 6: Design for experiment 2

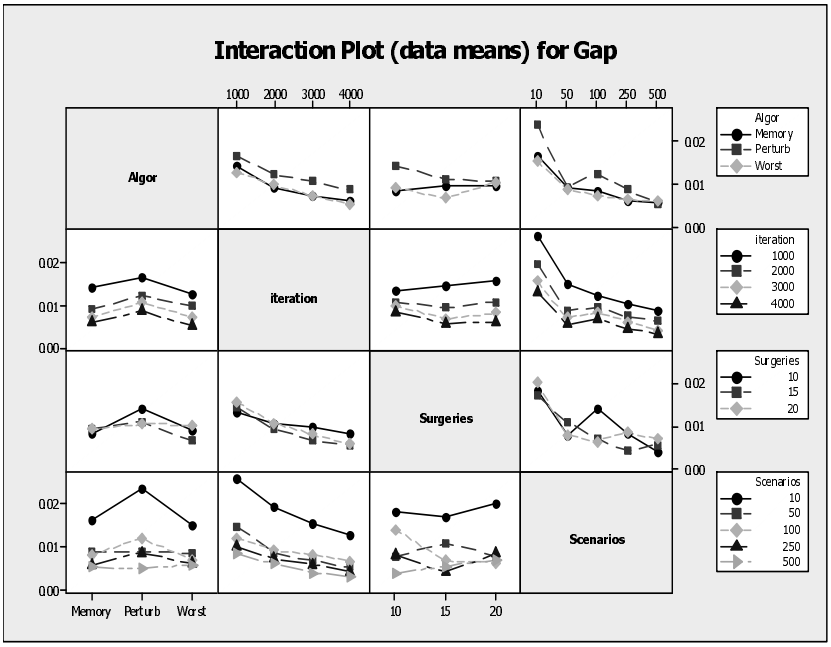

Figure 8: Interaction plots for experiment 2

	Average processing time (secs.)				
$\|J\|$ v/s $\|K\|$	10	50	100	250	500
10	5.8 - 7.3	18.4 - 18.9	34.7 - 39.0	97.4 - 112.6	200.2 - 234.9
15	8.0 - 9.4	28.0 - 29.9	69.7 - 67.3	247.0 - 204.3	382.3 - 435.0
20	10.5 - 12.9	40.9 - 56.9	99.5 - 104.2	282.5 - 315.9	694.7 - 771.8

Table 7: For memory algorithm over 1000 iterations (x - y) x: no overtime cost y: overtime cost

	Average restarts				
$\|J\|$ v/s $\|K\|$	10	50	100	250	500
10	42.6 - 89.7	31.5 - 49.5	33.3 - 50.2	32.6 - 49.4	33.1 - 47.8
15	26.1 - 64.2	17.1 - 33.5	16.5 - 32.0	16.6 - 31.3	16.7 - 32.0
20	18.7 - 60.8	11.1 - 24.5	10.3 - 24.6	10.0 - 22.7	10.1 - 16.4

Table 8: For memory algorithm over 1000 iterations (x - y) x: no overtime cost y: overtime cost

9 Conclusions

In this paper we developed new sequencing algorithms for the stochastic appointment scheduling problem. Few papers have addressed this problem, the lone exception being [5]. Denton tested some simple heuristics and showed that "sort by variance" was the best of those tested. In this paper we developed new algorithms based on Benders' decomposition which perform significantly better than sort by variance and a perturbed sort by variance heuristic, especially in the different cost case. To be fair, the proposed algorithms utilize much more computing time (a couple of minutes) than sort by variance, but roughly the same time as perturbed sort by variance. The algorithm run times are more than sufficient for implementation on real problems. Of the three Benders' based algorithms, the "memory restart" method provides the best results (statistically), but the difference between the three methods is fairly small from a practical standpoint. The results further showed that the relative performance of the three algorithms is uniform across the several factors used to create problem instances with different characteristics in testing. It is worth noting that the general approach used to create our heuristics may work well for other problems where the master problem without Benders' cuts is easy to solve, but the problem with cuts is hard. This approach seems particularly amenable to stochastic sequencing problems where the master problem (before Benders' cuts) is an assignment problem. A final contribution of this paper is a formal proof that the sequencing problem is NP-Complete. While this has been alluded to by several authors, the question, to the best of our knowledge, was previously open.

10 Appendix A: Details on Strengthening the Formulation from Section 3.2

Proposition 8. *If $c_i^w > 0$, $c_i^s > 0 \; \forall i \in \{1, ...n\}$ and $c^l \geq 0$ the values of the s_i^k at the optimal solution will have this upper bound:*

$$\overline{S} = F_s(t^{max}) = \max_{i \in I}\{\max_{k \in K}\{z_i^k\} - \min_{k \in K}\{z_i^k\}\}$$

Where F_s is a function that gives the maximum value of the idle time over of the possible surgeries when the starting times are fixed at the maximum duration scenario for every surgery.

Proof. First, it is known that any feasible solution is an upper bound for a minimization problem. Therefore, if the starting times are fixed by a some criteria, the value of the objective function will be an upper bound. Thus, it will define the worst case assignment of the starting times in terms of idle time. This assignment is to schedule time for every surgery at the maximum duration scenario. Therefore, the starting times will be defined by this formula.

$$t_i^{max} = \sum_{j=1}^{i-1} z_j^* \tag{12}$$

where z_j^* is the longest duration of surgery j. It is clear that with this starting time definition the waiting time for all surgeries and for all scenarios is zero. Please, see the figure 9. In order to prove that $\overline{S}$ is an upper bound for all the

optimal idle times (given a sequence) we first prove that:

$$s_i^k \leq \overline{s_i^k} \tag{13}$$

where s_i^k is the idle time for scenario k in surgery i at the optimal solution and $\overline{s_i^k}$ is the idle time found with the starting time definition.

The idle times for every scenario can be computer using the following formula for the case of t^{max}.

$$\overline{s_i^k} = max\{0 + z_i^* - z_i^k, 0\} = z_i^* - z_i^k \tag{14}$$

In the case where the optimal starting times are used.

$$s_i^k = max\{-w_i^k - t_i + t_{i+1} - z_i^k, 0\} \tag{15}$$

So, by contradiction. Assume that

$$s_i^k > \overline{s_i^k} \tag{16}$$

This implies that

$$-w_i^k - t_i + t_{i+1} - z_i^k > z_i^* - z_i^k \tag{17}$$

Simplifying terms

$$-w_i^k - t_i + t_{i+1} > z_i^* \Rightarrow t_{i+1} - t_i > z_i^* \tag{18}$$

But, this is a contradiction because the book time for surgery i can be reduced and thus the objective function will be reduced. With the last result the upper bound can be computed.

$$s_i^k \leq \overline{s_i^k} \leq \max_{i \in \{1,..,n\}} \overline{s_i^k} = \overline{S} \tag{19}$$

□

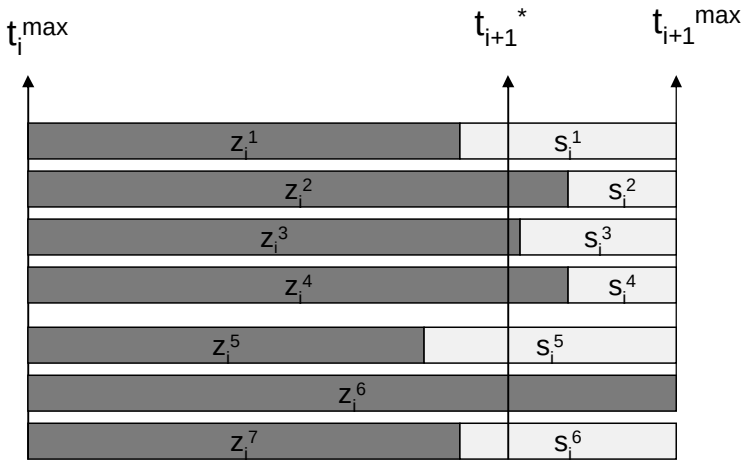

Figure 9: Starting Times at zero waiting time

Proposition 9. *If $c_i^w > 0$, $c_i^s > 0 \; \forall i \in \{1, ...n\}$ and $c^l \geq 0$ the upper bound $\overline{S}$ found in proposition 8 is valid regardless of the sequence.*

Proof. The arguments used in the demonstration of proposition 8 are not related with the sequence therefore the results can be extended to any sequence. □

With the previous results the M_1 (that correspond to the constraints (7) in the problem) will be set following this formula.

$$M_1 = \max_{i \in I}\{\max_{k \in K}\{z_i^k\} - \min_{k \in K}\{z_i^k\}\} \tag{20}$$

For the waiting time case the upper bound is not trivial because these variables are related to the previous surgeries in the sequence. However, the bound can be found with a similar approach.

Lemma 10. *If $c_i^w > 0$, $c_i^s > 0 \; \forall i \in \{1, ...n\}$ and $c^l \geq 0$ the values of the w_i^k at the optimal solution will have this upper bound:*

$$\overline{W} = F_w(t^{min})$$

Where F_w is a function that gives the maximum value of the waiting time over the possible surgeries when the starting times are fixed by this criteria. Please, see the figure 10

$$t_i^{min} = \min_{k \in K}\{t_{i-1}^{min} + \overline{w_{i-1}^k} + z_{i-1}^k\} \tag{21}$$

Proof. We will proof by contradiction, using the fact that any seeting of starting times is a feasible solution for the scheduling problem.

Assume that:

$$\overline{w_i^k} < w_i^k \tag{22}$$

Where $\overline{w_i^k}$ is the waiting time for surgery i in scenario k when we set the starting times as defined in 21 (no idle time case) and w_i^k is the optimal waiting time. So, using the equation 4.1 it can be show that

$$\overline{w_{i-1}^k} + z_{i-1}^k - t_i^{min} + t_{i-1}^{min} < w_{i-1}^k + z_{i-1}^k - t_i^* + t_{i-1}^* \tag{23}$$

This implies that:

$$\overline{w_{i-1}^k} - t_i^{min} + t_{i-1}^{min} < w_{i-1}^k - t_i^* + t_{i-1}^* \tag{24}$$

$$\overline{w_{i-1}^k} - w_{i-1}^k < \Delta t^{min} - \Delta t^* \tag{25}$$

The book time assign by t^{min} for every surgery is always less than the book time assigned at the optimal solution because it is choosing the minimum duration time. Therefore, this implies that

$$\overline{w_{i-1}^k} - w_{i-1}^k < 0 \tag{26}$$

Another property of t^{min} is that

$$\overline{s_i^k} = 0 \; \forall (i,k) \in \{I, K\}$$

At the same time, it can be show that

$$\overline{l^k} \leq l^k \; \forall k \in K$$

This result comes from the fact that t^{min} leaves the schedule with idle time therefore it is the minimum overtime cost that can be achieved. So, if we add the previous results by scenario and surgeries.

$$\sum_{k \in K} ((\sum_{i \in I} \overline{w_i^k} - w_i^k - s_i^k) + \overline{l^k} - l^k) < 0$$

This result can be reordered and penalized by waiting, idle and overtime cost.

$$\sum_{k \in K} (\sum_{i \in I} c_i^w \overline{w_i^k} + c^l \overline{l^k}) < \sum_{k \in K} ((\sum_{i \in I} c_i^k w_i^k + c_i^s s_i^k) + c^l l^k) < 0$$

But, this is only the objective function when it used t^{min} and the optimal t's (t^*).

$$z(t^{min}) < z^*(t^*)$$

This is a contradiction, because $z(t^{min})$ is an upper bound of the scheduling problem. □

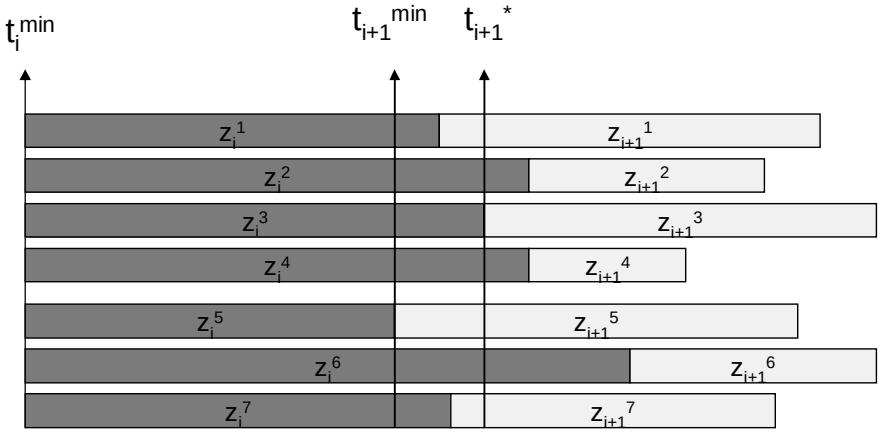

Figure 10: Stating Times at zero idle time

As, it was already mentioned the waiting times computed using t^{min} will depend on the sequence chosen.

Lemma 11. *If the starting time are fixed using t^{min} the maximum waiting for surgery i (w_i^{max}) will follow this formula:*

$$\overline{w_i^k} = \max_{k \in K} \{\overline{w_{i-1}^k} + z_{i-1}^k\} - \min_{k \in K} \{\overline{w_{i-1}^k} + z_{i-1}^k\} \tag{27}$$

Proof. It is known that

$$w_i^k = \max\{w_{i-1}^k + z_{i-1}^k - t_i + t_{i-1}, 0\} \tag{28}$$

For the particular case where it is t^{min} all the idle times will be zero and the waiting times will be greater than zero. If the starting times are replaced by the formula 21.

$$\overline{w_i^k} = \overline{w_{i-1}^k} + z_{i-1}^k + t_{i-1} - \min_{k\in K}\{t_{i-1} + \overline{w_{i-1}^k} + z_{i-1}^k\} \tag{29}$$

It can be canceled t_{i-1}.

$$\overline{w_i^k} = \overline{w_{i-1}^k} + z_{i-1}^k - \min_{k\in K}\{\overline{w_{i-1}^k} + z_{i-1}^k\} \tag{30}$$

Since, we aim to find the maximum value of $\overline{w_i^k}$.

$$\overline{w_i^k} = \max_{k\in K}\{\overline{w_{i-1}^k} + z_{i-1}^k\} - \min_{k\in K}\{\overline{w_{i-1}^k} + z_{i-1}^k\} \tag{31}$$

This is the result of the lemma. □

Proposition 12. *If the starting time are fixed using t^{min} the maximum waiting for surgery i (w_i^{max}) will follow this formula:*

$$w_i^{max} = \max_{k\in K}\{\sum_{j=1}^{i-1} z_j^k\} - \min_{t\in K}\{\sum_{r=1}^{i-1} z_r^t\}\ \forall i \in \{2, ..n\} \tag{32}$$

Proof. We use proof by construction. For the case i=2, Lemma can be used 11 it was assumed $w_1^k = 0$ for all k. For a better understanding please see Figure 11. For the case i, if it is used Lemma 11.

$$\overline{w_i^k} = \max_{k\in K}\{\overline{w_{i-1}^k} + z_{i-1}^k\} - \min_{k\in K}\{\overline{w_{i-1}^k} + z_{i-1}^k\} \tag{33}$$

Then we can use the definition of waiting time used in 4.1. Therefore,

$$\overline{w_i^k} = \max_{k\in K}\{\overline{w_{i-2}^k} + z_{i-2} - \min_{r\in K}\{\overline{w_{i-2}^k} + z_{i-2}^k\} + z_{i-1}^k\} - \min_{k\in K}\{\overline{w_{i-2}^k} + z_{i-2} - \min_{r\in K}\{\overline{w_{i-2}^k} + z_{i-2}^k\} + z_{i-1}^k\} \tag{34}$$

But, it is equal to

$$\overline{w_i^k} = \max_{k\in K}\{\overline{w_{i-2}^k} + z_{i-2}^k + z_{i-1}^k\} - \min_{k\in K}\{\overline{w_{i-2}^k} + z_{i-2}^k + z_{i-1}^k\} \tag{35}$$

So, if we continue to apply the same procedure we will obtain the desired formula.

□

With Proposition 12 an upper bound can be obtained that will be useful in order to compute M_1^i. Unfortunately, this problem is hard to solve, but leads to another bound that is easier to compute.

According to proposition 12 the maximum waiting time for job i (surgery) depends only on the previous jobs, therefore an upper bound can be found by computing a bound for every surgery. This leads to the next proposition.

Proposition 13. *The maximum waiting time for job i has an upper bound $\widehat{W}$. The value of $\widehat{W}$ is*

$$\widehat{W} = \sum_{j=1}^{i-1} \delta_j \tag{36}$$

where δ_j is the job (surgery) that is the jth biggest value in terms of $\max_{k\in K} z_r^k - \min_{k\in K} z_r^k$. This bound is an approximation to the bound proposed in Proposition 12. Thus, we have the following relationship $\overline{W} \leq \widehat{W}$

Proof. The argument for this proposition comes from this bound.

$$w_i^{max} = \max_{k\in K}\{\sum_{j=1}^{i-1} z_j^k\} - \min_{t\in K}\{\sum_{r=1}^{i-1} z_r^t\} \leq \sum_{j=1}^{i-1} \max_{k\in K}\{z_j^k\} - \min_{t\in K}\{z_j^k\} \tag{37}$$

the desired upper bound can be obtained ($\widehat{W}$) when the quantities $max_{k\in K}\{z_j^k\} - min_{t\in K}\{z_j^k\}$ are sorted in descending order. □

From the previous results the definition of $M_2's$ can be stated.

$$M_2^i = \sum_{j=1}^{i-1} \delta_j \tag{38}$$

where δ_j corresponds to the jth biggest value in terms of $\max_{k\in K} z_r^k - \min_{k\in K} z_r^k$.

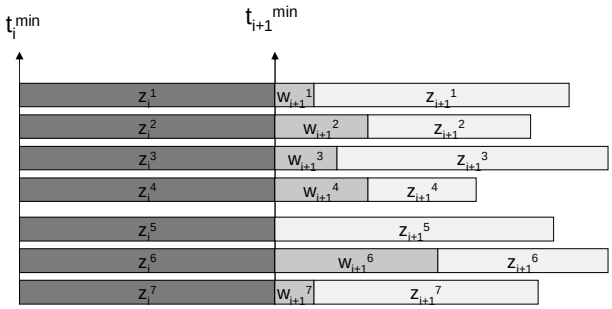

Figure 11: Maximum waiting time

11 Appendix B: Proof of Lemma 3 from Section 4.

The optimal scheduled starting times for a set of n jobs with the cost vector c_i^s=nK, c_i^w=1 $\forall i \in \{1,..,n\}$. It is given by: $t_i^* = \min_{k\in K}\{t_{i-1}^* + w_{i-1}^k + z_{i-1}^k\}$

Proof. We will prove this lemma by contradiction. Case 1: Assume that ∃ i such that $t_i^* < \min_{k\in K}\{t_{i-1}^* + w_{i-1}^k + z_{i-1}^k\}$ in the optimal schedule. This means that $w_i^k > 0\ \forall k \in K$ but this is a contradiction with the fact that this is an optimal solution because we can increase t_i^* until some w_i^k becomes zero without affecting the waiting time of the other jobs. This implies that we were in a suboptimal solution. Case 2:Assume that ∃ i such that $t_i^* > \min_{k\in K}\{t_{i-1}^* + w_{i-1}^k + z_{i-1}^k\}$ in the optimal schedule and z^* is the optimal objective function value. This means that we have idle time for some scenario in job i. If we change the starting time by this rule $t_j = t_j^*\ \forall j < i$, $t_j = t_j^* + \delta$ $\forall j \geq i$. If we recompute the objective function we will get:

$$z_\delta = z^* + \frac{K_s\delta}{K}nK - \frac{(K-K_s)\delta}{K} = z^* + \frac{(nKK_s - K + K_s)\delta}{K} \tag{39}$$

Where, K_s represents the number of scenarios that will incur in idle time. But this is a contradiction with the assumption that z^* is optimal. □

12 References

[1] HFMA, "Achieving operating room efficiency through process integration," *Health Care Financial Management Association Report*, 2005.

[2] E. Weiss, "Models for determining the estimated start times and case orderings," *IIE Transactions*, vol. 22 (2), pp. 143–150, 1990.

[3] B. Denton and D. Gupta, "Sequential bounding approach for optimal appointment scheduling." *IIE Transactions*, vol. 35 (11), pp. 1003–1016., 2003.

[4] L. Robinson and R. R. Chen., "Scheduling doctors appointments: Optimal and empirically-based heuristic policies." *IIE Transactions*, vol. 35, pp. 295–307, 2003.

[5] B. Denton, J. Viapiano, and A. Vogl, "Optimization of surgery sequencing and scheduling decisions under uncertainty," *Health Care Management Science*, vol. 10, pp. 13–24, 2007.

[6] D. Gupta, "Surgical suites operations management," *Production and Operations Management*, vol. 16 (6), pp. 689–700, 2007.

[7] P. P. Wang, "Optimally scheduling n customer arrival times for a single-server system," *Computers & Operations Research*, vol. 24, no. 8, pp. 703 – 716, 1997.

[8] G. C. Kaandorp and G. Koole, "Optimal outpatient appointment scheduling," *Health Care Manage Science*, vol. 10, pp. 217–229, 2007.

[9] P. M. Vanden Bosch and D. C. Dietz, "Scheduling and sequencing arrivals to an appointment system," *Journal of Service Research*, vol. 4 (1), p. 1525, 2001.

[10] ——, "Minimizing expected waiting in a medical appointment system," *IIE Transactions*, vol. 32, pp. 841–848, 2000.

[11] R. Schultz, "Stochastic programming with integer variables," *Mathematical Programming*, vol. 97, pp. 285–309, 2003.

[12] G. Laporte and F. Louveaux, "The integer l-shaped method for stochastic integer programs with complete recourse." *Operations research letters*, vol. 13, pp. 133–142, 1993.

[13] C. C. Caroe and R. Schultz, "Dual decomposition in stochastic integer programming," *Operations Research Letters*, vol. 24, pp. 37–45, 1999.

[14] A. Kleywegt, A. Shapiro, and T. H. de Mello, "The sample average approximation method for stochastic discrete optimization," *SIAM Journal on Optimization*, vol. 12, pp. 479–502, 2002.

[15] V. Norkin, Y. Ermoliev, and A. Ruszczynski, "On optimal allocation of indivisibles under uncertainty," *Operations Research*, vol. 46, no. 3, pp. 381–395, 1998.

[16] M. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. of Operations Research*, vol. 1, pp. 117–129, 1976.

[17] G. Cote and M. A. Laughton, "Large-scale mixed integer programming: Benders-type heuristics," *European Journal of Operational Research*, vol. 16, no. 3, pp. 327–333, 1984.

[18] K. Aardal and T. Larsson, "A benders decomposition based heuristic for the hierarchical production planning problem," *European Journal of Operational Research*, vol. 45, no. 1, pp. 4 – 14, 1990.

[19] J. T. Linderoth, A. Shapiro, and S. J. Wright, "The empirical behavior of sampling methods for stochastic programming," *Annals of Operations Research*, vol. 142, pp. 219–245, 2006.

[20] R. H. Storer, D. S. Wu, and R. Vaccari, "New search spaces for sequencing problems with application to job shop scheduling," *Management Science*, vol. 38, no. 10, pp. 1495–1509, 1992.