Beaver Dam Offspring Study (BOSS) Scheduling Problem

Margaret Cruickshanks, Bradley Jamison, Anna Svirsko, Olivia Zheng

December 15, 2010

Abstract

The purpose of this project is to analyze a basic parallel scheduling problem involving eight patients and ten exams. We will provide several methods to find the optimal schedule including scheduling by hand, creating an integer linear program, and several heuristics. We will conclude that due to our limited technological resources, we believe that finding a solution to this parallel scheduling problem is better done by hand.

1 Introduction

Scheduling research is one of the most analyzed topics in optimization history and has many practical applications including production planning, personnel planning, scheduling in parallel, and many others. The basic backbone of scheduling theory is the optimal sequencing of tasks to increase utilization of time and machine processing subject to resource constraints and task requirements. Some resource constraints include processors, money, and manpower; and some task requirements include deadlines, priority jobs, and sequencing conditions. In this paper, we are going to focus on a form of job scheduling on parallel machines.

The basic idea of job scheduling is as follows. There are n jobs and m machines. Each job must be processed uninterrupted through each machine and each machine is restricted to handling only one job at a time. There are distinct processing times associated with each job operation. The objective of this problem is to schedule each job to each machine in a way that will minimize the longest running operation. For this project, each machine can handle every job without any additional instruction. Other scheduling problems have machines that can only process jobs with certain traits, like gender specification, and thus would limit the number of scheduling possibilities. Furthermore, we have a source machine, the machine that each job must get processed through first before all other machines, and a sink machine, the machine that each job must process last. The objective is to find a scheduling solution that will minimize the maximum time job n takes to run through all the machines.

One of the simplest scheduling solutions is to receive and process each order in series. In other words, each job is processed and finished before the next job begins its process. Although this method will minimize the possible number of machine collisions and satisfy the semantics of the scheduling problem, it takes a long and often impractical amount of time to complete. To shorten our run-time, we must consider parallel instruction scheduling which will allow the execution of running multiple jobs simultaneously.

One possible method of finding an optimal solution is to search through the entire solution space, calculate each jobs run-time for each solution set, and find the optimal solution set. This, however, is also extremely expensive. So over the years, researchers have developed a number of ways to approach a parallel scheduling solution including integer programming, queuing, heuristics, approximations, and optimal techniques. To this date, though, there does not exists a complete polynomial bounded algorithms that can be used to find the optimal solution to all job scheduling problems. Scheduling problems are thus, NP-hard, or have nondeterministic polynomial run-time. The heuristics, approximations, and algorithms that have been developed are made to just find a feasible solution, or approach the optimal.

For our project, we will investigate a slightly more complex parallel scheduling example. We will attempt to schedule eight participants, each of whom will have to take ten exams. Our main solution method is integer programming. We will then explain alternate solutions, specifically queuing and the evolutionary algorithm.

2 Beaver Dam Offspring Study

2.1 BOSS Overview

The Beaver Dam Offspring Study, BOSS, is currently in its second stage of exams. The study provides a comparison to the participants parents who had been taking part in the Epidemiology of Hearing Loss Study or Beaver Dam Eye Study. The goal of BOSS is to study the genetic and environmental factors leading to sensory loss or impairment.

2.2 BOSS Goal

BOSS has a goal of testing about 3,462 participants by the end of December 2012 (exams started in August 2010). Only 228 participants were seen by the end of October because of scheduling conflicts for another study conducted by the same research program.

2.3 BOSS Exam

Each exam for BOSS is an average of three and a half hours. It is composed of ten tasks that every participant should complete. All of our duration times are based on the average time it takes to complete each stage of the exam. The components of the exam are as follows:

Exam Component	Average Time of Completion(in minutes)
Informed Consent	15
Questionnaires	25
Olfaction (and Blood Pressure)	15
Cognitive Testing	20
Vision	20
Hearing	40
Ocular Imaging	15
Vascular Anthropometry	40
Phlebotomy	15
Exit(Receive Compensation)	5

There are some limitations on the order of these exams. Informed consent and the exit tasks must be completed first and last, respectively, for all participants. The olfaction and blood pressure testing has to occur before the phlebotomy is completed. The olfaction and cognitive tests must happen before the vision testing, due to the fact that the participants must be able to read and see images during these exams. In addition there must also be at least 20 minutes but not more than 60 minutes between the completion of the vision testing and the beginning of the ocular imaging test in order for the participants eyes to properly dilate. Space must also be taken into consideration when scheduling these exams. There are separate specific rooms for the hearing, vascular, vision, ocular imaging and phlebotomy testing. The study also has two offices that are configured for registration, interviews, cognitive testing and olfaction, although these components can also be conducted in any room not occupied by another participant.

2.4 Staffing

While we did not take into consider staffing issues in our program, other than the fact that we are constrained to a maximum of 4 examiners working at a given time, we were given more information. Currently there are only about 3.8 full time examiners (one is working at 80% capacity until July 1, 2011). One of these examiners is not currently certified to complete 5 minutes of the vision test, any of the ocular imaging, and the ultrasound portion of the vascular exam (20 minutes of the 40 minute exam). This examiner must swap with another examiner during these portions of the exams. She is expected to be fully certified by March 1, 2011. In addition to the examiners there is a receptionist and two phlebotomists. The receptionist works half time, not after five oclock and not on weekends. One of the phlebotomists works 75% of the time and the other works 50% of the time. A phlebotomist must be present during some portion of every exam to draw the participants blood. For necessary simplicity we only considered the four fully trained, full time examiners in our model.

Exams usually take place Monday through Friday but can also include weekends. The current policy for weekend exams is that examiners are given a weekday off to replace the weekend day they worked. A current schedule includes two Saturdays a month in which exams are conducted. Within a current working month there are about six to eight evening exams and the rest are normally scheduled day exams. However, twice a month, there are staff meetings on Tuesday mornings leaving just the afternoon to hold exams. One day in December each year there is a full day of staff meetings. In addition, the study also conducts about two days of quality assurance resulting in lost exam time. These staff meetings and quality assurance procedures eliminate about 28 exam days in the next two years.

In addition to the staff meeting and quality assurance days, time for exams is also lost due to holidays, vacation, sick time and furlough. The examiners are each allowed 22 days of vacation, nine holidays plus four personal days, about one sick day per month. Due to lack of funding, the University of Wisconsin has also mandated that all academic and classified staff have to have eight furlough days each academic year.

2.5 Current Schedule

The researchers planning the Beaver Dam Offspring Study originally created a schedule to both satisfy the minimal time to complete the exam but also to allow their examiners some freedom in conducting the examinations. They decided that eight exams could be completed in a day using two different exam schedules: one for evening examinations and one for the daytime exams. The current schedules were created using a chart and moving pieces of paper of different color and size, corresponding to different exams and participants, until they were satisfied with the result. However, it has been noted that the examiners either simply move to a free room or follow the general order listed in the table above. They also can complete the questionnaire in several sections allowing for more freedom; however, for simplicity we have only considered completing the questionnaire in one time block rather than dividing it into smaller parts.

Currently exams are being scheduled at: 8:00, 8:30, two exams at 9:00, noon, 12:30, and two exams at 1:00. For evening exam days to allow for more participants to be seen after five oclock they follow a different schedule: two exams at 12:30, 1:00, 1:30, 4:30, 5:00, 5:15 and 5:30. This allows the examiners to complete eight exams a day only working 40 hours a week.

They currently follow schedules similar to the following examples, the start times of each exam component is listed in the table. All of the participants will have completed their exams in 3.5 hours.

Day Exams	Person 1	Person 2	Person 3	Person 4	Person 5	Person 6	Person 7	Person 8
Exam 1	8:00	8:30	9:00	9:00	12:00	12:30	1:00	1:00
Exam 2	10:45	8:45	9:15	9:15	2:45	12:45	1:15	1:15
Exam 3	8:55	9:20	9:40	9:40	12:15	1:10	1:40	1:40
Exam 4	9:50	9:25	10:10	9:55	1:50	1:25	2:10	1:55
Exam 5	10:10	9:45	10:30	10:55	2:10	1:45	2:30	2:55
Exam 6	9:10	10:05	10:50	11:30	1:10	2:05	2:50	3:30
Exam 7	11:10	10:45	11:30	12:10	3:10	2:45	3:30	4:10
Exam 8	8:15	11:00	11:45	10:15	12:30	3:00	3:45	2:15
Exam 9	10:30	11:40	9:55	11:15	2:30	3:40	1:55	3:15
Exam 10	11:25	11:55	12:25	12:25	3:25	3:55	4:25	4:25

Evening Exams	Person 1	Person 2	Person 3	Person 4	Person 5	Person 6	Person 7	Person 8
Exam 1	12:30	12:30	1:00	1:30	4:30	5:00	5:15	5:30
Exam 2	12:45	3:30	2:50	4:30	7:30	6:50	8:15	7:20
Exam 3	1:10	1:05	1:55	2:05	5:25	5:55	6:10	5:45
Exam 4	1:25	12:45	2:10	1:45	5:40	6:10	6:25	6:00
Exam 5	1:45	1:20	2:30	3:15	6:00	6:30	7:00	7:45
Exam 6	2:05	2:50	1:15	3:35	4:45	7:30	5:30	6:35
Exam 7	2:45	2:20	3:30	4:15	7:00	7:15	8:00	8:00
Exam 8	3:00	1:40	3:45	2:20	6:20	5:15	7:25	8:40
Exam 9	3:40	2:35	3:15	3:00	7:15	8:10	6:45	6:20
Exam 10	3:55	3:55	4:25	4:55	7:55	8:25	8:40	8:55

It should be noted that, in practice, the examiners do not actually follow any set schedule but rather go to an available room once they finish with an exam.

3 Solving the BOSS Problem

3.1 Purpose

While the method currently used seems to work and allows enough time to complete all exams, it would seem that there might be a better way to approach the problem. We decided to use an integer linear program to determine how to schedule exams for an optimal day. From the optimal day BOSS researchers can shift the schedule five hours for the evening exam type. While our model does not account for all variables it will be able to provide an optimal order of the exams for each participant for a particular number of examinations in a day.

3.2 Assumptions

Due to the complexity of this problem, we had to make several assumptions to help eliminate some variables and constraints. With respect to staffing, we assumed that all four examiners are fully trained and working full time on the project. This helps eliminate extra constraints about the fourth examiners lack of training. Lunches will be fit into half hour breaks during the day, rather than being planned by the linear program. The phlebotomists schedules will be planned in accordance with the daily exam schedule rather than planning their schedules and solving the linear program based off of their work schedules.

In addition to staffing assumptions, we also assumed that participants do not have preferences about when their exams are scheduled. This eliminates the need for two different programs based on the participants desire to be seen at particular times during the day. Also, all participants complete all tests. Furthermore, we assumed that all exam components for each participant are completed in exactly the average amount of time. We also made the assumption that each exam component is completed in one continuous block of time. While in some ways this would eliminate some flexibility in the schedule, it eliminates a large quantity of constraints and variables.

4 BOSS Integer Linear Program

Indices	
i	subjects from 1 to n
j	exams from 1 to 10
h	time from 0 to 102

Variables	
t_{ij}	time subject i begins exam j
k_{ijh}	$\begin{cases} 1 \text{ if subject } i \text{ takes exam } j \text{ during time } t \\ 0 \text{ otherwise} \end{cases}$
d_{ij}	duration of exam j

Tests and Durations			
j	Test	d_j	Value
1	Registration and Consent	d_1	3
2	Questionnaires	d_2	5
3	BP and Olfaction	d_3	3
4	Cognitive Tests	d_4	4
5	Vision	d_5	4
6	Hearing	d_6	8
7	Ocular Images	d_7	3
8	Vascular and anthropometry	d_8	8
9	Phlebotomy	d_9	3
10	Exit	d_{10}	1

Modeling the idea of Basis Parallel Job Scheduling the Integer Linear Program (ILP) we created to solve the BOSS Scheduling Problem works to minimize the maximum time it takes for a subject to complete all 10 exams.

Scheduling ILP

min: $\max_i \{t_{i10} + 1 - t_{i1}\}$ Minimizes the maximum wait time.

Subject to:

$$\sum_j \sum_i k_{ijh} \leq 4 \ \forall h$$ No more than 4 tests at a time

$$\sum_{h=t_{ij}}^{h=t_{ij}+d_j-1} k_{ijh} = dj \ \forall i, j$$ Tests last for a consecutive interval of d_j

$$\sum_h k_{ijh} = d_j \ \forall i, j$$ Tests only last for a period of d_j

$$\sum_j k_{ijh} \leq 1 \ \forall i, h$$ Only one exam is given at a time for subject i during time h

$$\sum_i k_{ijh} \leq 1 \text{ for } j = \{5, 6, 7, 8, 9\}, h = \{1, 102\}$$ The Vision, Hearing, Ocular Images, Vascular and Anthropometry, and Phlebotomy Tests occur at most 1 at a time due to room constraints

$t_{i1} \leq t_{ij}\ \forall i, j$	Registration always occurs first
$t_{i10} \geq t_{ij}\ \forall i, j$	Exit always occurs last
$t_{i3} + d_3 \leq t_{i5}\ \forall i$	BP and Olfaction always occurs before Vision
$t_{i4} + d_4 \leq t_{i5}\ \forall i$	Cognitive Tests always occurs before Vision
$t_{i3} + d_3 \leq t_{i9}\ \forall i$	BP and Olfaction always occurs before Phlebotomy
$t_{i5} + d_5 + 4 \leq t_{i7}\ \forall i$	At least 20 minutes between the completion of the Vision Test and the beginning of the Ocular Images Test
$t_{i5} + d_5 + 12 \geq t_{i7}\ \forall i$	At least 60 minutes between the completion of the Vision Test and the beginning of the Ocular Images Test
$0 \leq t_{ij} \leq 102$	There are a total of 102 time intervals
$1000 * (k_{ijh} - 1) \leq h - t_{ij}\ \forall i, j, h$	For h less than t_{ij} the RHS becomes negative, forcing k_{ijh} to 0 in order for the LHS to be negative.
$1000 * (k_{ijh} - 1) \leq t_{ij} + d_j - 1 - h\ \forall i, j, h$	For h greater than $t_{ij} + d_j - 1$ the right hand side becomes negative, forcing k_{ijh} to 0 in order for the LHS to be negative

The last two constraints force $k_{i,j,h}$ to be 1 for a consecutive d_j time periods.

5 BOSS Scheduling Problem LINGO Code

The following code can be entered into LINGO to create the optimal schedule for BOSS Scheduling Problem.

MODEL:

```
SETS:
    SUBJECTS /1..8/;
    EXAMS /1..9/:DURATION;
    PERIODH /1..101/;
    TIME(SUBJECTS, EXAMS):INTERVAL;
    CONS(SUBJECTS,EXAMS, PERIODH):YESORNO;
ENDSETS

DATA:
    DURATION = 3 5 3 4 4 8 3 8 3;
ENDDATA

[OBJ] MIN = X;
X = @MAX(SUBJECTS(I):@MAX(EXAMS(J) :INTERVAL(I,J)+DURATION(J))-
INTERVAL(I,1));

@FOR(SUBJECTS(I):
    INTERVAL(I,2)-INTERVAL(I,1)>DURATION(1);
    INTERVAL(I,3)-INTERVAL(I,1)>DURATION(1);
    INTERVAL(I,4)-INTERVAL(I,1)>DURATION(1);
    INTERVAL(I,5)-INTERVAL(I,1)>DURATION(1);
    INTERVAL(I,6)-INTERVAL(I,1)>DURATION(1);
    INTERVAL(I,7)-INTERVAL(I,1)>DURATION(1);
    INTERVAL(I,8)-INTERVAL(I,1)>DURATION(1);
    INTERVAL(I,9)-INTERVAL(I,1)>DURATION(1);

    INTERVAL(I,3)+DURATION(3)<INTERVAL(I,5);
    INTERVAL(I,4)+DURATION(4)<INTERVAL(I,5);
    INTERVAL(I,3)+DURATION(3)<INTERVAL(I,9);
    INTERVAL(I,5)+DURATION(5)+4<INTERVAL(I,7);
    INTERVAL(I,5)+DURATION(5)+12>INTERVAL(I,7);
);

@FOR(TIME(I,J):
    @BND(1,INTERVAL(I,J),41);
    @GIN(INTERVAL(I,J));
);

@FOR(SUBJECTS(I):
    @FOR(PERIODH(K):
        1000*(YESORNO(I,J,K)-1)<K-INTERVAL(I,J);
        1000*(YESORNO(I,J,K)-1)<INTERVAL(1,J)-K+DURATION(J)-1;
    );
);

@FOR(SUBJECTS(I):
    @FOR(EXAMS(J):
        @SUM(PERIODH(K):YESORNO(I,J,K))=DURATION(J);
    );
    @FOR(PERIODH(K):
        @SUM(EXMAS(J):YESORNO(I,J,K))<1;
    );
);

@FOR(PERIODH(K):
    @SUM(EXAMS(J):(@SUM(SUBJECTS(I):YESORNO(I,J,K))))<4;

    @SUM(SUBJECTS(I):YESORNO(I,5,K))<1;
    @SUM(SUBJECTS(I):YESORNO(I,6,K))<1;
    @SUM(SUBJECTS(I):YESORNO(I,7,K))<1;
    @SUM(SUBJECTS(I):YESORNO(I,8,K))<1;
    @SUM(SUBJECTS(I):YESORNO(I,9,K))<1;
);

@FOR(CONS(I,J,K):
    @BIN(YESORNO(I,J,K));
);

END
```

6 BOSS Integer Liner Program Solution

Due to the number of subjects and exams for BOSS, the ILP became extremely large for the set time periods that were given to us. The problem had approximately 7500 integer variables that we needed to find. We tried using the software LINGO 5, which only handles 8000 variables but only 800 integer variables. Due to these constraints, we reduced the number of time periods from 102 to 41, and the number of subjects from eight to two. We also eliminated exam 10 and all of its respective constraints, deciding that we would simply add the final exam to the end of each testing sequence. While the code had no errors, the program was unable to run completely. After running for four hours, the program seemed to be stuck at a certain constraint. We tried running the program again but had a similar result. After a bit of research we discovered that the software may have reached a suboptimal solution. Because the scheduling problem is NP complete the solutions are distributed across several different schedules, causing the optimization programs to freeze. Due to the way LINGO is set up, we cannot stop the solver and view the suboptimal solution that it reached. Instead, when the solver is stopped LINGO shows all the variables to be zero. In conclusion, we have found a model that would ideally produce a model that schedules eight people to ten exams in a minimal amount of time. However, due to the software constraints we are incapable of solving this problem and smaller version of it.

7 Excluded Realistic Factors in BOSS Scheduling Problem

Although we were able to determine an optimal solution for our system using a manual technique, we had to simplify the inputs so that they were easier to work with. We did not take into account vacation days since for all intensive purposes they are random. When examiners take vacation this would deplete the amount of people who could be tested at a single time by however many examiners were off. We also did not take into account staff meetings. Every other Tuesday morning there is a staff meeting and once every December there is a full day staff meeting. This takes away from times that could have been used to test participants and therefore changes the optimal daily scheduling. We also assumed that every participant shows up to his or her assigned appointment, however in reality there is a 25 to 30 percent probability that the participant will cancel or not show up. This is a significant amount and though these instances do not have an affect on the other appointments, they do leave more time in which the examiners are not being utilized to their full potential.

8 Alternative Method 1: Queueing Systems

One way to approach a scheduling problem is to formulate it as a queuing system. A queuing system consists of the following factors: how customers arrive, the actual servicing system and the way in which customers exit the system (satisfaction measurement).

Arrival is determined both by the customer population and the arrival rate. Customers can arrive from either a finite or infinite population. Finite populations can sometimes lead to downtime in service whereas an infinite one can constantly fill the void once there is room in the system. The arrival rate is the amount of customers who arrive during a given period. If the arrival rate is variable, it can be modeled by an exponential distribution, meaning the probability of arrival is based on time, or a Poisson distribution, meaning the exact arrival probabilities are desired at a given time.

The servicing system, or queuing system, consists mainly of the waiting line and the available servers, the people serving the customers. The waiting line can be infinite or limited. It is infinite only if there is nothing restricting the number of customers. For example, if there is a finite amount of space to hold the customers in line, then the waiting line must be constrained and is therefore limited. The amount of lines also plays a part depending on how many servers there are. For example if there are two lines and two servers and they are evenly distributed, then the system is treated as a single queuing system. Whereas if there is one line and two servers, it is a multiple queuing system. The final factor of the servicing system is the queue discipline: this is the order in which customers are served. Different ways to serve customers vary from the common first-come-first-served to priority first. Priority first customers are served regardless of their arrival time. The distribution of the service time is known as the service rate, which is the number of customers served per period per server. This can either be constant or approximated using an exponential distribution. The amount of services received by a customer from the servicing system as well as the number of servers available help determine what type of line structure exists. If there is a single server then it is a single channel line, if there is more than one server it is a multichannel line. If a single service is being received then it is a single phase line, where as if multiple services are being received it is a multiphase line. For example, if you have one person performing one task, the line structure is considered single channel, single phase.

The way in which customers exit the queuing system can be classified in two ways depending on if the customer returns. If the customer is likely to come back, they would simply be added back to the population of potential customers. If the customer is unlikely to return for service then they would not be readmitted into the potential customer population.

Since we are trying to optimize the wait time for a scheduling problem, using a queuing system analysis could potentially help heed results.
Arrival for the problem at hand can be modeled with a constant arrival rate and a finite population. The arrival rate can be considered constant since the participants are given appointments and there is roughly one appointment every half-hour. Although the population is relatively large in comparison to the daily participants served, we cannot consider it to be an infinite population since towards the end of the study the pool will have dwindled to a point where it is no longer significantly large.

The servicing system for this problem can be classified as having a limited waiting line, with a single line, and a reservations first queuing discipline. The service rate is 0.29 per hour. The line structure of this system is multichannel, multiphase. The system has a limited waiting line since the line must be kept within the waiting area of the research facility, it would not make sense to have participants waiting outside the facility. It can be classified as single line since there only exists one waiting room where all the participants are held and their place in line is the order they came. The queuing discipline is identified as reservations first since the participants are given a scheduled time to come. If all the participants arrive at their scheduled time it could also be considered a first-come-first-served discipline. The arrival rate is 2 per hour since the time between arrivals is half an hour. Since testing should take 3.5 hours per person then the service rate is 0.29 per hour (1/3.5). The line structure of the system is multichannel since up to four tests by four people can be taken at a time.

Recall that tests 1 through 4 and 9 can be taken the same test at one time but not the others, though a total of four or more tests can be taken at all times. It is multiphase since each participant has to take 9 tests.

Once a participant has gone through the system they are finished with the current part of the study therefore their probability for return of service is zero since they will not reenter the participant pool.

We can use the information from the queuing system in formulas to determine other factors of the study such as the average number of participants in the system and especially the total time in system. More importantly from here we have the option of initializing queue scheduling algorithms to optimize the latency, turnaround and response times. In our situation we would use a first-come-first-served scheduling algorithm or a fixed priority pre-emptive scheduling algorithm, Since one algorithm is not significantly better in throughput time which one we used would not have an impact on our systems optimization. However, using this method would require special scheduling software which we do not have access to and would not likely always give an optimal solution since these programs tend to optimize for turnaround time as opposed to throughput time, which means it allows for participants to be in the system for more than 3.5 hours, which is the desired maximum time in the system.

9 Alternative Method 2: Evolutionary Algorithm

The evolutionary algorithm was developed specifically for the field of genetics, but we can easily apply the basic ideas to this scheduling problem. We will use Mesghouni, Hammandi, and Bornes summary of this heuristic in their article, Evolutionary Algorithm for Job-Shop Scheduling, and apply it to our scheduling problem.

For this algorithm we make the following assumptions:

- There are eight independent participants or jobs
- Each participant has an ordered sequence G_i where each G_i contains x_i operations or ten examsthe order of which each participant will take his or her exam
- O_{ij} is a vector (i, j, t_{ijk}) where t_{ijk} is the start time of when person i takes exam j
- There are seven rooms or machines
- Each exam requires a room
- For each i, j, k, there is a processing time $P_{i,j,k}$

- $1 \leq i \leq N = 8$ (participants)
- $1 \leq j \leq x_i = 10$ (jobs)
- $1 \leq k \leq M = 7$ (rooms)

- Each exam must run uninterrupted
- Each room must finish the assigned exam before hosting another exam
- C_{max} is the makespan or how long it takes for a participant to take all ten exams

Our objective is the same as our integer program problem. For each set of operations $\{C_{ijk}\}$ with i, j, k bounded appropriately, we want to minimize C_{max}.

M_1	(i, j, t_{ijk})
M_2	(i', j, t'_{ijk})
...	...
M_M	...

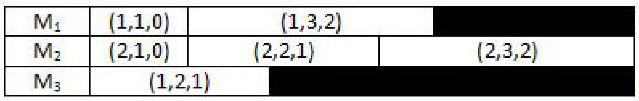
Note: t_{ijk} is the starting time of participant i taking exam j

In this scheme, there are $|i| * |j| * |t_{ijk}|$ or $8 * 10 * 102 = 8160$ possible vectors that can be distributed among the M rooms assuming we include collisions. Each M_k may be assigned as many as 8160 vectors or as little as zero vectors. In the above figure, M_2 cannot have (i, j, t_{ijk}) assigned to it because (i, j, t_{ijk}) is already assigned to M_1. For simplification, if we just have two participants, three rooms, and three exams, a possible scheduling scheme is as follows:

M_1	(1,1,0)	(1,3,2)	
M_2	(2,1,0)	(2,2,1)	(2,3,2)
M_3	(1,2,1)		

Note: Participant 1 takes exams 1 and 3 in room 1 and exam 2 in room 3
Participant 2 takes all 3 exams in room 2

In order to account for processing time, we can graphically represent the time duration or the weight of each exam by the size of each block. If, for example, exam 1 takes the shortest amount of time and exam 3 takes the longest our new chart will change as follows:



M₁	(1,1,0)	(1,3,2)	■
M₂	(2,1,0)	(2,2,1)	(2,3,2)
M₃	(1,2,1)	■	

Crossover

1. Create two schedules called parent schedules by randomly assigning vectors into rooms following all conditions

2. Create one or two children schedules by interchanging assignments between the two parents into the new children schedules, leaving the time assignment unknown

A more structured algorithm for determining the childrens schedule is to randomly choose a room, k, and copy all the elements assigned to that room of parent one to the first childs kth room. We do a similar task using parent two, child two, and the same kth room. The remaining entries in both children can be filled by taking the opposite parents elements keeping the rooms the same. We also need to be careful that operations are not repeated in either childs schedule. If a child is missing a particular exam vector O_{ij}, we apply the appropriate case:

- If $k = 1$, then put the missing exam at the beginning of the kth room assignment
- If $k = x_j$, then put the missing exam at the end of the kth room assignment
- Else, find the column which contains $O_{i-1,j}$ and $O_{i+1,j}$ and put O_{ij} between those columns in the kth machine row

Throughout this whole process, the start times of the childrens exams remain unassigned. To determine the value of t_{ijk} we must first define two new variables and then follow the algorithm created by Mesghouni, Hammandi, and Borne:

- T_F: Set of deadlines of the last exams on each participant. $|T_F| = 8$
- D_k: Set of deadlines of the last exam for each machine. $|D_k| = M = 7$

```
$h = 1$
    $while(h < k)do$
        $if(T_F(i) < D_k(h))$
            $t_{ijk} = D_k(h)$
        $else$
            $t_{ijk} = T_F(i)$
    $T_F(j) = t_{ijk} + P_{ijk}$
    $D_k(j) = t_{ijk} + P_{ijk}$
$end$
```

This model will always produce a new schedule. Its biggest flaw is its inability to account for precedence. Note the following example:

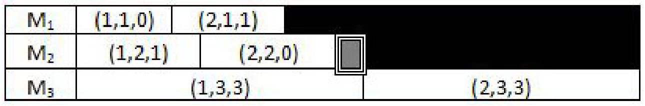
M_1	(2,2,?)	(1,1,?)
M_2	(1,2,?)	(2,1,?)

Room 1 must host person 2 taking exam 2, however, person 2 cannot start exam 2 without first taking exam 1. Similarly, person 1 cannot take exam 2 unless he or she finishes exam 1. However, the way the exams are ordered in the room, it is impossible to satisfy the precedence condition.

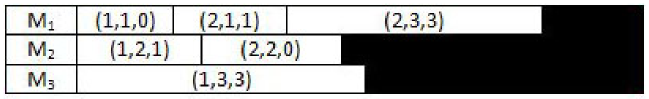
Operator Mutation

1. Create a random schedule that satisfies all constraints and arbitrarily pick an exam from a room with a high load
2. Move this exam to another room that is not heavily filled to reduce the makespan

This is a very intuitive algorithm. The following example will introduce this idea.



If we look at the above schedule, it is optimal for us to move (2,3,3) to room 1 because person 2 will finish his exam in room two but will have to wait the amount of time shaded in grey to start exam 3 in room 3. However, if we move his or her exam to room 1, person 2 will start his exam 3 immediately and thus, reducing our makespan time.



This method is very similar to our initial scheduling solution we produced by hand except that this algorithm has a visual representation that may facilitate solving the problem.

Both the evolutionary algorithms do not produce the optimal solution; rather, it finds a feasible solution and check to see if the makespan time is shorter. It continues to search the reduced solution space until the smallest makespan time is found.

10 Conclusion

The Beaver Dam Offspring Study examines the genetic and environmental factors leading to sensory loss or impairment. Each participant must be examined in Beaver Dam, and must do through several tests. Our task is to come up with a schedule that will minimize the maximum makespan time, or the amount of time any given patient must spend at the site. To tackle this issue, we first created a chart and guessed at a solution. We checked over the solution to make sure the studys conditions were satisfied. This method took us about two hours.

Our next solution involved an integer linear program, involving about 7500 integer variables. Due to our limited technological resources, the only software we had access to was LINGO 5.0 which can only handle 800 integer variables. So we decided to reduce the number of constraints by reducing the number of people involved in the study to make our program work. However, after allowing the program run for over four hours, the program froze at a suboptimal solution and crashed.

While in theory, the integer linear program we created should ideally find our optimal scheduling solution, running the actual program is more expensive than producing a schedule by hand. A problem like BOSS is NP complete and as a result, finding software that can handle an unlimited number of variables and run our program in a reasonable amount of time is extremely difficult. In fact we conclude that producing this optimal schedule by hand is more advantageous. The schedule we created initially not only satisfied all of our conditions and met our objective, but it took a fraction amount of the time we spent on the integer linear program. Unless provided with better software, creating a schedule by hand is our optimal method.

Works Cited

Gere, W.S. "Heuristics in Job Scheduling" *Management Science* 13 (1966): 167-190.

Haq A.N., Balasubramanian K., Sashidharan B., and Karthic R.B. "Parallel Line Job Shop Scheduling Using Genetic Algorithm." *International Journal of Advanced Manufacturing Technology* 35 (2008): 1047-1052.

Heffernan M.E. "Data-Dependency Graph Transformations for Instruction Scheduling." Dissertation, Massachusetts Institute of Technology, 2007.

Jacobs, F. Robert., Richard B. Chase, Nicholas J. Aquilano, and Richard B.Chase. *Operations and Supply Management* Boston: McGraw-Hill, 2009.

Mesghouni K., Hammadi S., and Borne P. "Evolutionary Algorithms for Job-Shop Scheduling." International Journal of Applied Mathematics and Computational Science 14 (2004): 91-103.

Potts C.N. "Analysis of a Linear Programming Heuristic for Scheduling Unrelated Parallel Machines." Discrete Applied Mathematics 10 (1985): 155-164.

Switalski P. and Seredynski F. "Multiprocessor Scheduling by Generalized Extremal Optimization." Journal of Scheduling 13 (2009): 531-543.