Adding Emergency Cases to the Hershey OR Sim Model and Simulation Notes

Drew Bennison, Kevin Gardner, Aaron Harter, Olivia Kucenski, Nathan Kurtz, Michelle Soeganto, Praharsh Verma

I. Adding Emergency Cases into the Simulation

Currently the simulation does not account for emergency cases being added to the schedule and bumping other cases from the schedule. Emergency cases are a main source of disruption and delay for other cases, so it would improve the accuracy of the simulations to account for the possibility of emergency cases.

In this section we will briefly outline an approach that might be taken to add emergency cases to the model and simulate them.

A. Sampling Emergency Cases

The first step to adding emergency cases is to sample for the number and type of emergency cases that will happen that day. For sampling the number of emergency cases that will come in that day, we would first filter the data set for the given month and day of the week and filter to keep only cases that were not added to the schedule before the selected cut off time (the default cut off time is 5 p.m. but this is a parameter that can be changed in the code). This would leave us with a data set of cases that were not on the schedule when it was originally planned; emergency cases.

From this data set, we would build a distribution of the number of emergency cases that get seen each day using the same Kernel Density Estimation (KDE) technique used throughout the model. Once this KDE probability density function (PDF) was created, we would randomly sample from it during the simulation run to get the number of emergency cases to be added to the schedule that day.

The second step would be to sample the level of urgency those cases require. Using the same filtered data set described in the first step, we would create a new KDE PDF using the add on hours column. The add on hours column is the difference between the time the case was added on to the schedule and when that case is scheduled to start. Emergency cases will have a small number for add on hours because they are added on to the schedule very close to when they are scheduled to start, while elective cases are planned well in advance and have large values for add on hours. Emergencies are defined as cases where the add on hours < 1, urgent cases have an add on hours < 6, and priority cases have an add on hours < 24. Once the KDE PDF of add on hours is built, we will sample from that PDF n times where n is the number of emergency cases that will be added to the schedule that day per step one. This list of sampled add on hours will give us the urgency of the cases that must be added to the schedule that day.

The third step would be to create a distribution of the time that these emergency cases will "pop up" and when they must be dealt with. A similar KDE technique could be used based on the historical time when emergency cases come in, or a different technique could be used. No matter the approach, this distribution would also have to be sampled n times.

At the end of these steps, we would have a list of n sampled add on hours and n sampled start times for when these emergency cases will pop up.

B. Adding the Emergency Cases to the Schedule

One key difficulty with adding emergency cases to the model is that you have to be able to asses the current state of all OR rooms when the emergency case arrives in order to make the best scheduling decision. In order to asses the current state of all OR rooms, you have to be simulating each OR room in parallel. Since this is not currently possible as OR rooms rely on the previous case in them for vital scheduling information, we simulate the OR by iterating through each OR room and each case. This makes adding emergency cases onto the schedule challenging during the simulation phase of our model.

What we propose is to generate a planned schedule, simulate that schedule using our HersheyORSim.simulateSchedule() function, and then create a new function for adding on emergency cases. This new function will take in the simulated schedule as input and add on emergency cases using the random sampling approach described above.

More specifically,

1) The function addEmergencyCases() takes as input the simulated schedule.
2) The first part of the function will sample the number, urgency, and time emergency cases will come in for that day using the above steps in section A.
3) Starting at the beginning of the day, step through the schedule at intervals of every 5/10/15/30 minutes, whatever time frame seems reasonable.
4) At each time step, check if any emergency cases have "popped up" by checking whether the start time of an emergency case is earlier than the current time you are at the step of and if it has not been dealt with yet.
5) If an emergency case has popped up, add it to the schedule based on its priority and room availability.

Simulate the case duration and turnover time using the KDE technique used for simulating regular elective cases. Bump cases as necessary to add it to the schedule.

6) Continue iterating through the entire day and repeat the adding/bumping as necessary until you have gone through all emergency cases. Return the new simulated schedule at the end.

The additional benefit of having an extra function for adding emergency cases onto the schedule instead of doing it during the simulateSchedule() function is that you can easily test different scheduling/bumping plans. Dealing with emergency cases is where many different types of scheduling decisions can be made, so having a function dedicated to just adding emergency cases onto the schedule allows the user to vary the bumping policies to test how that effects the schedule.

II. Simulation Notes

The simulation relies heavily on Kernel Density Estimation (KDE) and random sampling in order to build the planned schedule and simulate that schedule. KDE PDFs were created for tasks such as determining the number of cases that will be in each OR that day, the length of the planned/actual cases, the turnover time, and more. Each run of the model randomly samples from these distributions and others to add variation to the model and generate a planned/simulated schedule.

If a KDE PDF was not appropriate for the task, such as when we sample what types of cases will be in an OR that day, we created a discrete distribution to sample from. This is how we sampled the types of cases that are in an OR, the start time of the first case of the day, and whether or not a case was going to be cancelled in the future.

When sampling, we check whether the sampled result makes sense in context and re-sample from the distribution if necessary.

Because we utilize random sampling, each run of the model will produce a different result, which makes it useful for running Monte Carlo Simulations and tracking key metrics such as OR utilization, case delays, and more. Running a for loop n times and storing the results would create n different simulations of a given parameter combination and would be the start of running Monte Carlo Simulations.

Once those simulations are run, key metrics can be calculated over the large sample and compared when different scheduling decisions/plans are introduced.

Comments were added throughout the code to specify the different parts of the model and what is being done.