7th AIMMS-MOPTA competition report, Team RensPolymathIa

Xin Shen[a], Jubiao Yang[a], John E. Mitchell[a,1]

[a]Rensselaer Polytechnic Institute, Troy, NY 12180

1. Mathematical Formulation for problem without bad luck or with deterministic bad luck

Our approach to model the system is based upon the observation that the durations of all projects and the time span are integral, therefore the time span can be divided into time segments with equal lengths (one month in this case). A binary variable X_{jt} is assigned to Project j at the tth time segment, which indicates whether Project j starts at the beginning of the tth month. This results in a mixed integer model, which contains the following parameters:

$$
\begin{aligned}
c_j &= \{\text{the cost of Project j}\} \\
b_j &= \{\text{the benefit of Project j}\} \\
d_j &= \{\text{the duration of Project j}\} \\
c_j^+ &= \{\text{the added cost to Project j if delayed}\} \\
d_j^+ &= \{\text{the added duration to Project j if delayed}\} \\
T &= \{\text{the number of months in the total time span, integer}\}
\end{aligned}
$$

A binary delay flag q_j is introduced to indicate whether Project j is expected to be delayed, hence the cost and duration of each project under what-if scenarios would be:

$$\tilde{c}_j = c_j + q_j c_j^+, \quad \forall j \in J \tag{1}$$

$$\tilde{d}_j = d_j + q_j d_j^+, \quad \forall j \in J \tag{2}$$

The following constraints are added to the model:

- A project can be started at most once throughout the entire time span:

$$\sum_{t=1}^{T} X_{jt} \leq 1, \forall j \in J \tag{3}$$

---

[1]Team Advisor

- A project cannot be started if it is too late, that is, we can only start a project if it can be finished before the deadline:

$$\sum_{t \geq T+1-\tilde{d}_j} X_{jt} = 0, \forall j \in J \tag{4}$$

- For all $i \not\sim j$, i cannot be started within d_j units of time after j started, same for j.

$$\sum_{t-\tilde{d}_j+1 \leq t' \leq t+\tilde{d}_i-1} X_{jt'} + X_{it} \leq 1, \forall i \not\sim j, \forall t \in \{1..T\} \tag{5}$$

- For all $i > j$, j can be started only if i has been finished:

$$\sum_{t' \leq t-\tilde{d}_i} X_{it'} \geq X_{jt}, \forall i > j, \forall t \in \{1..T\} \tag{6}$$

- For all $I \vdash j$, j cannot be started until at least one of the projects in I has been finished:

$$\sum_{i \in I} \sum_{t' \leq t-\tilde{d}_i} X_{it'} \geq X_{jt}, \forall I \vdash j, \forall t \in \{1..T\} \tag{7}$$

Apparently Constraint (6) is a special case of Constraint (7), with only Project i rather than multiple elements in Set I, and can thus be generalized into Constraint (7) for simplicity.

The objective function is defined as:

$$NPV_\gamma(S, T) = - \sum_{\substack{j \in S \\ 0 \leq t_j/12 \leq 3}} \gamma^{t_j} \tilde{c}_j + \sum_{\substack{j \in S \\ 0 \leq (t_j+\tilde{d}_j)/12 \leq 3}} \gamma^{t_j+\tilde{d}_j} \tilde{b}_j \tag{8}$$

In the optimization problem, it is written as:

$$NPV_\gamma(S, T) = - \sum_{i,t} \gamma^t \cdot \tilde{c}_i \cdot X(i,t) + \sum_{i,t} \gamma^{t+\tilde{d}_i} \cdot \tilde{b}_i \cdot X(i,t) \tag{9}$$

It is easy to construct the one-to-one correspondence between (S, T) and X, where:

$$S = \{i | \exists t, \text{ s.t. } X(i,t) = 1\} \tag{10}$$

and for each element t_i in T, which is the starting time of project i in S:

$$t_i = t, \quad \text{where } X(i,t) = 1 \tag{11}$$

Thus to get the optimal schedule without bad luck or with deterministic bad luck, we can formulate the optimization problem as:

$$\begin{aligned} \max_X \quad & NPV_\gamma(S, T) \\ s.t \quad & (3) - (7) \\ & X_{it} \in \{0, 1\} \end{aligned} \tag{12}$$

2

The above model can be efficiently solved by typical MIP solvers such as Cplex.

2. Scheduling Under Bad Luck

From now on, the problem is to take failures and delays into consideration while determining the project portfolio. A parameter $W \geq 0$ is introduced to represent the budget of bad luck. We define:

$$I_f = \{\text{Project that will be failed by the bad luck}\}$$
$$I_\delta = \{\text{Project that will be delayed by the bad luck but eventually will succeed}\}$$
$$I_{f|\delta} = \{\text{Project that will be delayed by the bad luck and eventually will fail}\}$$

Each project has probabilities to delay, to fail, or to delay and then fail. The probabilities can help with calculation of information entropy, which is used to measure the cost of each occurrence of bad luck. The total cost of bad luck is defined as:

$$w(I_f, I_\delta, I_{f|\delta}) := \sum_{j \in I_f}[-log_2(p_{f,j})] + \sum_{j \in I_\delta}[-log_2((1 - p_{f,j})p_{\delta,j})] + \sum_{j \in I_{f|\delta}}[-log_2((1 - p_{f,j}))]$$

Our purpose is to find a schedule that yields the highest profit under certain budget of bad luck. A general formulation is the following maxmin problem:

$$\max_{S,T} \quad \min_{I_f, I_\delta, I_{f|\delta}} \quad NPV'_\gamma(S, T, I_f, I_\delta, I_{f|\delta})$$
$$subject to \quad i > j, \ \forall (i, j) \in E,$$
$$i \neq j, \ \forall (i, j) \in E', \tag{13}$$
$$I \vdash j, \ \forall (I, j) \in E'',$$

The net present value of the portfolio is computed by summing up the costs and benefits of all projects in the schedule, discounted appropriately at the time they are incurred:

$$NPV_\gamma(S, T) = \sum_{j \in S, 0 \leq t_j/12 < 3} \gamma^{t_j} \tilde{c}_j + \sum_{j \in S, 0 \leq t_j/12 < 3} \gamma^{t_j + d_j} \tilde{b}_j \tag{14}$$

3. Adapative Scheduling of Projects

In practice, the original schedule is subject to update once any bad luck occurs. For example, consider a scenario where an intermediate project delays, which could lead to inability of some other projects to be completed in time. Although we can continue with the subsequent projects, a possible better option could be to remove a proportion of projects while updating the schedule. The purpose is to reduce the potential loss based upon information at the current stage. Our analysis will take the update into consideration. At each time $t^{badluck}$ when a bad luck happens, the update must follow the rules:

- Only projects selected in the original schedule can be chosen in the updated schedule.

- The update will take place when a bad luck occurs, and we assume that bad luck only occurs at the scheduled ending time of a project. The cases where projects delay and then fail can be regarded as special occasions where bad luck occurs twice, in which the first time is at the scheduled ending time of the previous plan, when we know the project is delayed, and the second time is the moment it fails at the end of the delay.

- Only the projects that are scheduled at and after the current bad luck can be rescheduled.

- When updating the schedule, we make an optimal choice without assuming that bad luck will take place after $t^{badluck}$.

The impact of bad luck can be quantified by the loss it incurs. Given an initial schedule (S, T) and a realized schedule (S', T'), the loss is calculated as the difference between the NPV value of them, which can be expressed as:

$$Loss_{(S,T)}(S', T') = NPV(S', T') - NPV(S, T) \tag{15}$$

With the assumptions above, the final realization of schedule can be obtained by solving a series of optimization problems.We begin with an initial schedule (S, T), which can be the solution in part 1, and at each time t_i when bad luck occurs to a single project, the schedule is updated to (S_i, T_i) based upon current progress and the category of bad luck before and at t. Let (S_{i-1}, T_{i-1}) be the schedule after last bad luck happens at time t_{i-1} with corresponding decision variable $X'_{i,t}$, and $(I_{tf}, I_{t\delta})$ be the set of projects that has been delayed or failed before and at time t. The treatment for the case of delayed and fail is a little tricky. If at time t we know a project has been delayed and failed, then we put it into both set I_{tf} and $I_{t\delta}$, otherwise if we only get the information that the project has been delayed, it will only be selected into the set I_{tf}. We can find an updated schedule(S_i, T_i) after bad luck at time t_i by solving a problem which has the following constraints with decision variable $X_{i,t}$:

- As in part 1, a single project can only be started once.

$$\sum_t X_{it} \leq 1, \forall i \tag{16}$$

- The starting time of a project cannot be too late. This constraint is also the same with that in part 1. Note that here since we made the choice without assuming any delay will happen, it is possible that a project will not be completed in time if it is delayed. However, we don't take the delay in the future into account while deciding whether it should start.

$$\sum_{t \geq T+1-d_i} X_{it} = 0, \forall i \tag{17}$$

- All decision variables in (S_i, T_i) prior to the time t, which can be considered as realized, are fixed to the same value with the decision variable in the previous schedule (S_{i-1}, T_{i-1}):

$$X_{i,t} = X'_{i,t}, \forall t < t_i \tag{18}$$

- For any $i > j$, if at time t_i i has failed, or $i \in I_{tf}$, j cannot be started at any time.

$$\sum_t X_{jt} = 0 \tag{19}$$

4

- For any $i > j$, if at time t_i i has been delayed, or $i \in I_{t\delta}$,j cannot be started unless i is finished.

$$\sum_{t' \leq t-d_i-d_i^+} X_{it} \geq X_{jt'}, \forall i > j, t \tag{20}$$

- For any $I \vdash j$ and $i \in I$, the starting time of j will depend on other components in the set I.

$$\sum_{p \in (I/\{I_{tf}\} \cap I_{t\delta}} \sum_{t' \leq t-d_p-d_p^+} X_{pt'} + \sum_{p \in (I/\{I_{tf}\} \cap \bar{I}_{t\delta}} \sum_{t' \leq t-d_p} X_{pt'} \geq X_{jt}, \forall t \tag{21}$$

- Projects that are not selected in the initial schedule will not appear in the updated schedule

$$X_{it} = 0, \forall i \notin S_0 \tag{22}$$

Thus given the schedule (S_{i-1}, T_{i-1}) after bad luck happens at time t_{i-1}, the formulated optimization problem to get the updated schedule after bad luck happens at time t_i is:

$$\begin{aligned} \max_{X} \quad & NPV(S_i, T_i) \\ s.t \quad & (16) - (22) \\ & X_{it} \in \{0, 1\} \end{aligned} \tag{23}$$

The algorithm to get the final realization of the original schedule with the given triple of bad luck $(I_f, I_\delta, I_{f|\delta})$ is as follows:

Data: Initial Schedule (S_0, T_0), Triple of bad luck $(I_f, I_\delta, I_{f|\delta})$

Result: Final Realization (S',T')

(S1,T1) = (S,T);

while *Not every element in $(I_f, I_\delta, I_{f|\delta})$ is selected* **do**

    Choose a bad luck in $(I_f, I_\delta, I_{f|\delta})$, which has not be selected before, with the earliest time of occurence;

    Update the set $(I_{tf}, I_{t\delta})$, which is the set of bad luck that we know has happened;

    Solve the problem (23)

end

Output $(S', T') = (S_n, T_n)$, where n is the last time bad luck occurs.

4. Estimation of Worst Case Scenario

In the previous schedule we developed a way to estimate the loss of a schedule subject to a given set of bad luck in the context of adaptative scheduling. The question arises as to what the worst case would be like given a budget of bad luck. Currently we do not have efficient methods to incorporate adaptative scheduling into a single optimization problem. A brute force approach for assessing the worst case scenario is to try every combination of bad luck, evaluate its total benefit and choose the one with the largest loss. The method is compuationally infeasible especially when the size of scheduling network grows extremely large, since there will be exponetial number of possible combination of bad luck.

Here we adopted a greedy heuristic to assign the bad luck in order to get a relatively bad scenario. We begin with the empty triple $(I_f, I_\delta, I_{f|\delta})$. In every iteration, for each possible assignment of bad luck on a single project, we add it

into the triple selected temporarily, calculate the loss incured. At the end of the iteration we pick the assignment that yields the most loss and add it to the triple permanently. The realization of the schedule is updated taking the effect of chosen bad luck into consideration. The iteration terminates when no damage that incur loss can be added to the triple. Although there's no theoretical guarantee for the 'optimal' chocie of bad luck, the algorithm provided a way to get a comparatively bad case and help identifiy key components in the job scheduling network in a reasonable amount of time. The algorithm is as follows:

Data: Initial Schedule (S,T), budget of bad luck W

Result: Triple of bad luck in the worst case $(I_f, I_\delta, I_{f|\delta})$

while *1* **do**

    for *i in S, p in {Failure, delay, delay and failure} with* $W_{ip} \leq W$ **do**

        Find the updated schedule (S',T') by solving the optimization problem if with the triple of bad luck contains previously selected bad luck and type p bad luck assigned to project i;

        Compute the loss, which is the total benefit of the plan subject only to the previouly selected bad luck minus the result in the last step;

    end

    if *There exist some project i with type bad luck that incurs positive loss* **then**

        pick the one that with the most loss, or the highest ratio of loss over W_{ip};

        W=W-W_{ip};

    else

        Jump out of the loop;

    end

end

5. Robustification of Schedule

Our way to find a robust schedule is based upon a 'solve-robustify' approach. We solve the problem in part 1 and use the solution as our initial schedule. Then we apply the greedy heuristic in the last section to identify projects in the worst case scenario, which can be regarded as vulnerable to bad luck and plays an important role in the scheduling network. Given the set of these 'weak' but 'important' components in the scheduling network, we take several strategies to robustify the networks, including:

- If project i fails in the worst case scenario.

  - i is the single project chosen in set I with $I \vdash j$ and j is also in the original schedule. A possible choice is to add another project in I to the schedule, this can be done by replacing the relationship $I \vdash j$ and $k \in I$ by $k \succ j$ and $I/\{k\} \vdash j$ and rerun the optimization problem in part 1.

  - i is the prequisite of some project j in the original schedule. It is obvious that j and subsequent projects that rely on the completion on j cannot be started. A possible way to is to solve an optimization problem with the decision variable X_{it} fixed to be 0 for any t, this might result in:

    * i and its subsequenct projects which depend on i will not be started

    * Same with above. However, it is possible that some other projects, which were not selected in the previous schedule are included in the update schedule. Another possible outcome is that the starting time of some projects, which have conflicts with deleted projects, are moved forward.

- If project i is delayed in the worst case scenario,

  - The approach would be checking whether the starting time of the project can be moved forward. This can be achieved by adding the time of delay to the duration of project i and solve the optimization problem in part 1 again.

  - If some project cannot be completed in time because of the delay of i, delete i by restricting decision variables corresponding to i to be 0 and rerun the optimization problem.

- if project i is delayed and then failed in the worst case scenario. The treatment can be a combination of that in the case of delay and case of failure.

There are also many other strategies which are not covered in our program, for example, under the condition of adaptive scheduling, the starting time of certain projects, which is prerequisite to many other projects but also highly susceptible to failure, can be moved forward. If they fail then other projects will get the information and won't start. Note that the above strategies does not necessarily yield a solution whose worst scenario is better than before. So each time we made a single or a set of changes to the original plan, we need to check whether the worst case of updated one is better than the previous one.

we also need to point out that the estimation of final realization for a given triple of bad luck will still rely on the original relation between projects. Thus the scheme for updating needs slight change, for example, some relation between projects will be recovered which is incorporated into our program.

6. Computational Results

6.1. Smaller Dataset

In the second problem, we begin with the solution, where we assumed that no bad luck happens. The greedy heuristic (2) can be applied here. For purpose of illustration we calculate the loss for every possible bad luck happened to a single luck no matter whether the bad luck required is more than the budget. Table 1 gives the loss that incurred by each single occurrence of bad luck when the constant γ is equal to 0.99.

project	Failure	Delay and Failure	Delay
Job1	-12.2333	-12.3193	-12.3193
Job2	-12.2333	-12.3193	-12.3193
Job4	-12.2333	-12.3135	-12.3135
Job6	-12.2333	-12.3211	-0.08775
Job8	-6.11092	-6.30892	-0.198
Job9	-7.77821	-7.95372	-0.1755
Job10	-13.789	-13.9445	-0.15556

Table 1: Loss incurred by each single bad luck occurrence, $\gamma = 0.99$

We study several combinations of γ and W.

- We begin with a simple case $W = 1$ and $\gamma = 0.99$. It is obvious that none of the bad luck will happen, which is shown in Figure 1.
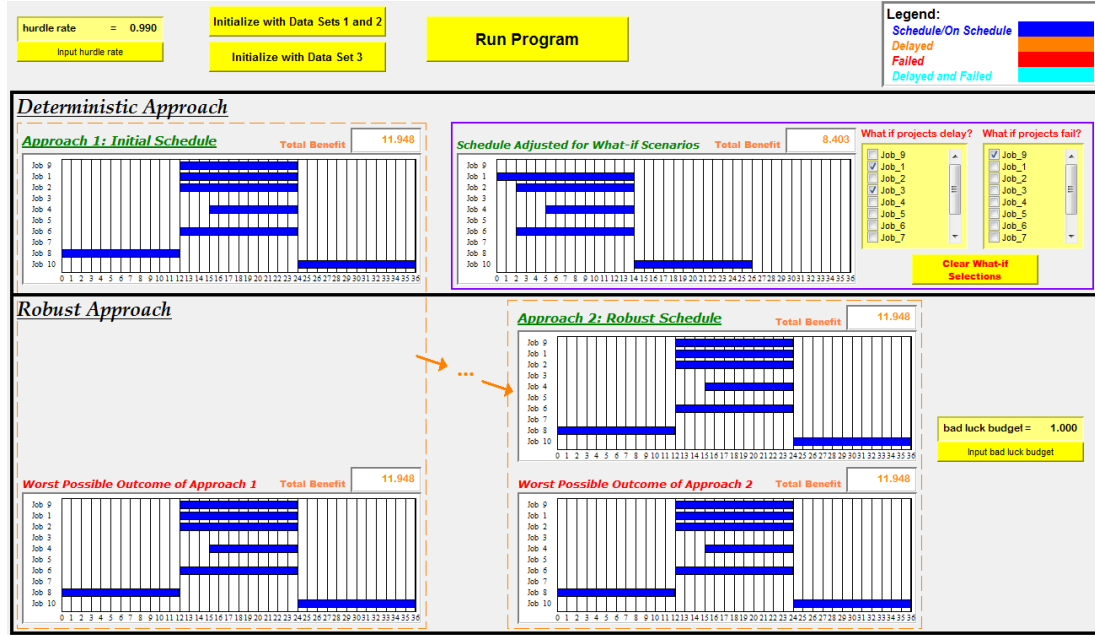
7

Figure 1: GUI with result, $W = 1.0$, $\gamma = 0.99$

- When the budget $W = 1.4$ and $\gamma = 0.99$. The above table gives the loss of each potential choice of bad luck. When W equals 1.4 it reaches the threshold for Project 4 to fail. In the worst case scenario, Project 4 fails and Project 1 cannot be started. Two alternatives are considered. In the first approach, all decision variable for Project 4 are fixed to be 0, while in the second approach, we'll add Project 3 since $\{3, 4, 5\} \vdash 10$, as the failure of Project 4 leads to failure of Project 1. The worst case scenario of the results of the two alternatives are examined, and worst case in the result of second approach is better than both the previous result and Approach 1. Therefore we add 3 into the plan and update the chart of loss to Table 2 listed below.

project	Failure	Delay and Failure	Delay
Job1	-12.2333	-12.3193	-12.3193
Job2	-12.2333	-12.3193	-12.3193
Job4	-3.55E-15	-0.08016	-0.08016
Job6	-12.2333	-12.3211	-0.08775
Job8	-6.11092	-6.30892	-0.198
Job9	-7.77821	-7.95372	-0.1755
Job10	-13.789	-13.9445	-0.15556
Job3	-12.2333	-6.39589	-6.39589

Table 2: Loss incurred by each single bad luck occurrence, $\gamma = 0.99$

The final schedule is highly resilient to bad luck at the level of W, as is shown in Figure 2.
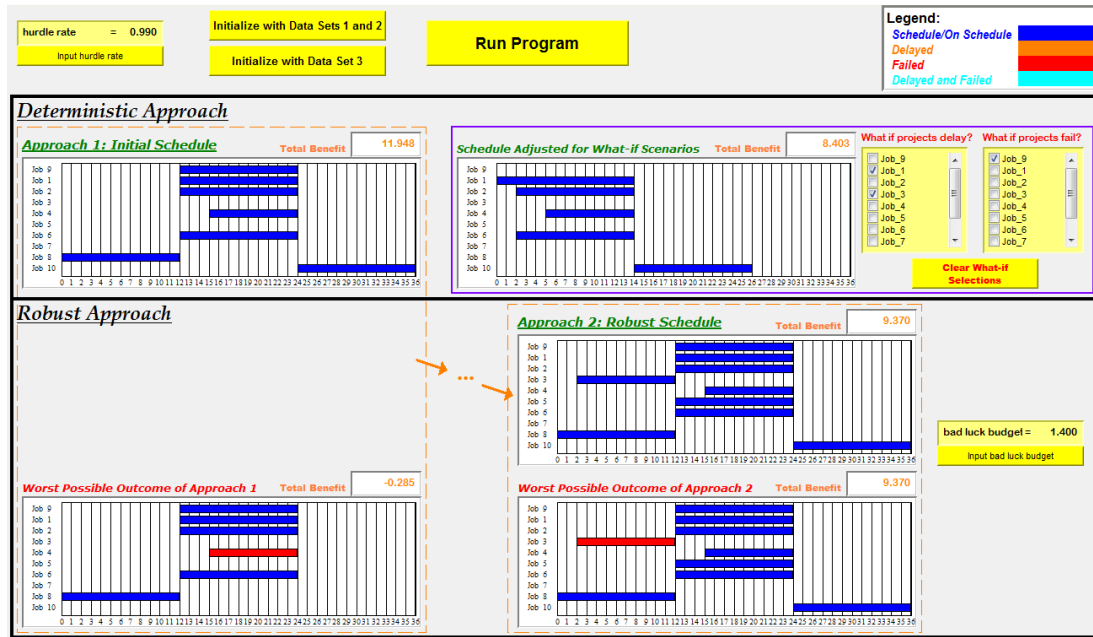
8

Figure 2: GUI with result, $W = 1.4$, $\gamma = 0.99$

- When the budget $W = 2.5$ and $\gamma = 0.99$. Under this condition the failure of Project 1 will lead to the most loss. The only possible method for improvement is to restrict decision variable corresponding to Project 1 to be 0. In the resulting scheduling net work we only have the Project 8 and Project 9.
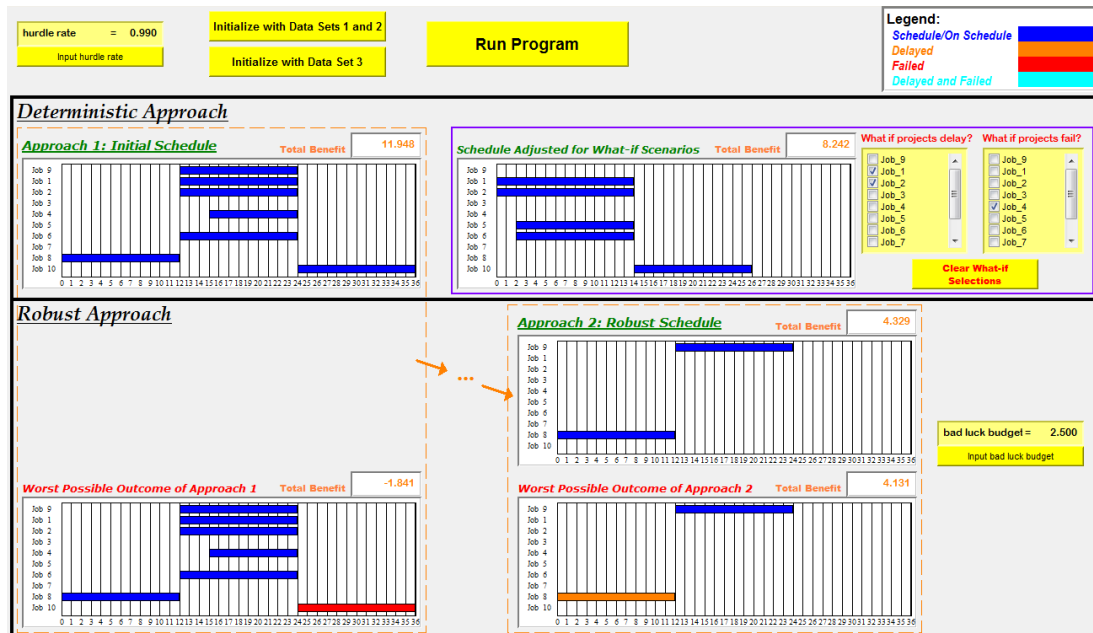

Figure 3: GUI with result, $W = 2.5$, $\gamma = 0.99$

9

As *W* goes larger both Projects 9 and 10 would fail and the best choice would be doing nothing.

- When the budget *W* = 2.5 and *γ* = 0.95, we can observe a dramatic change in the intial condition from the case *γ* = 0.99. Only Project 8 and Project 9 are selected in the initial schedule. The schedule is highly vulnerable to delay, since the bad luck that will result in the most loss is delay for Project 8. Since Project 8 cannot be moved forward, the only way that may lead to a better solution is to delete Project 8. The resulting schedule contains no projects and of course will no longer be influenced by bad luck.
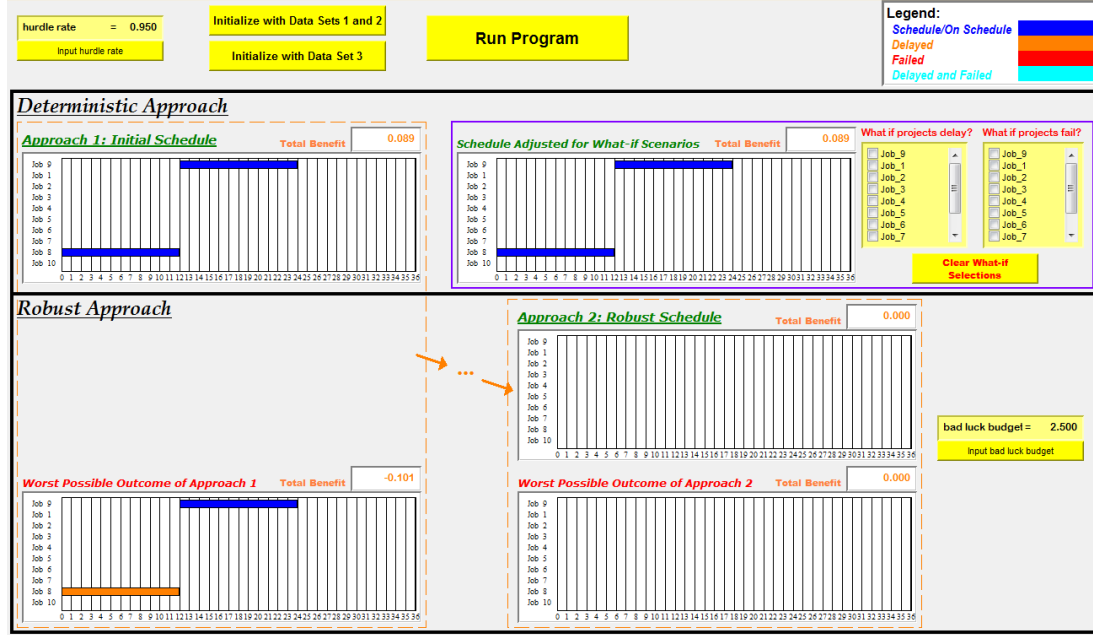

Figure 4: GUI with result, *W* = 2.5, *γ* = 0.95

6.2. Larger Dataset

Similar to the case of the smaller dataset, for the large data set our analysis will begin with the result with Approach 1, which does not take any risk of failure or delay into consideration. Greedy algorithm still plays a central role in acquiring the worst case scenario. However, in the case of the larger dataset, which contains 76 projects, it is time consuming to perform analysis on the loss incurred by every single occurrence of bad luck since the formulated optimization problem has thousands of variables. Two strategies are taken based upon the observation of the initial result from Approach 1:

- Most projects starts before the 18th month, which implies that even delay takes place, it is unlikely that there will be any project that cannot be completed in time. And currently we only limit our discussion to the case where the hurdle rate is large,i.e, greater than 0.99, thus the discount effect, together with additional cost of delay is trivial and the resulting loss of delay is rather negligible when compared with that of failure. Another observation is that in many projects, the budget of failure is comparable to that of delay. As a result, in our analysis for the large data set we'll assume that bad luck only happens in the form of failure.

- Instead of giving a precise estimation of the loss of each occurrence of bad luck, we adopted the following algorithm to find a naive estimation of the failure of a single project.

10

Data: Current Realization of Schedule (S',T')
Result: Loss(S,'failure'), each entry represents the naive estimate of loss incured by only failing the
corresponding entry in S under current realization
$S'' = \{\}$;
while *S' is not empty* **do**
    Pick $i \in S'$ with the latest starting time.;
    Loss(i)=$b_i * \gamma^{t_i+d_i}$; if it is not failed, 0 if failed. **for** $j \in S''$ **do**
        if $j > i$ **then**
            Loss(i)=Loss(i)+Loss(j)-$c_j * \gamma^{t_i}$;
        end
        if $j \in S''$ *and* $I > i$ **then**
            Loss(i)=Loss(i)+(Loss(j)-$c_j * \gamma^{t_i}$)/ Cardinality($I \cap (S' \cup S'')$);
   **end**
    end
    Move i from S' to S''.
end

The general idea of the algorithm is to estimate the loss of the failure of a given project based on loss of other projects that completely or partially depend on the completion of the current project. The algorithm is time efficient at the expense of accuracy. An obvious drawback for this algorithm is that if fails to grasp the interdependence of projects. However, it provided us with insight into components that has high potential to be crucial to the structure of the network.Note that the algorithm can also be extended to evaluate a naive estimation for the loss of delay.

The graph below shows the trend for how the total benefit of worst case scenario changes against W. the blue line is for the worst case for the initial plan, and yellow line for the worst case of the robust plan.
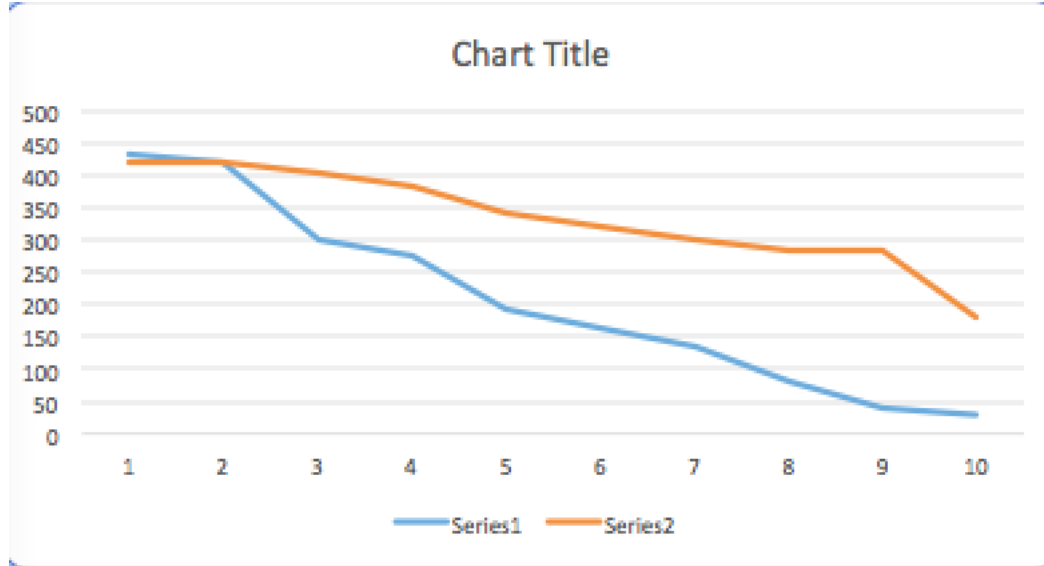
In the robustification of the large data set, we take a conservative approach since our resource of computing is limited. We only consider the case where project i,j fails in the worst scenario, and i is in some set $I \vdash j$. The approach we take is to add other projects in set I that were not previously selected into the plan. Another reason for not deleting failed projects is that the cost for failure does not differ much from project to project, and the deletion will make the

remaining projects in the schedule more susceptible to the attack of bad luck. Here we plot how the value in the worst case changes after the robustification for a given budget of W. From the figure we can see that the robust schedule performs much better than the initial schedule in the worst scenario.