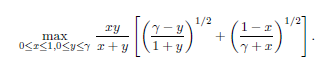
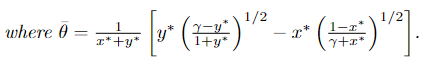
一、The programs in folder “Performance of various due dates --slove w & d\_w ” are used for solving and

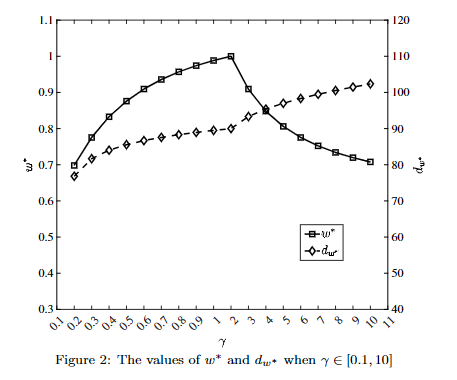
1. “solveg.m ” is used to solve the following problem，and then to solve .





2. Then we can solve and by according to the following two equations.

“ mainprog.m ” is used to implement the calculation process and draws Figure 2.  

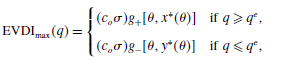
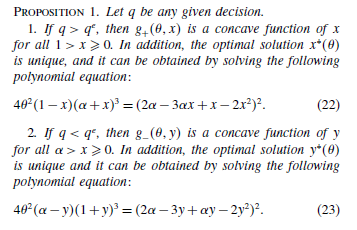



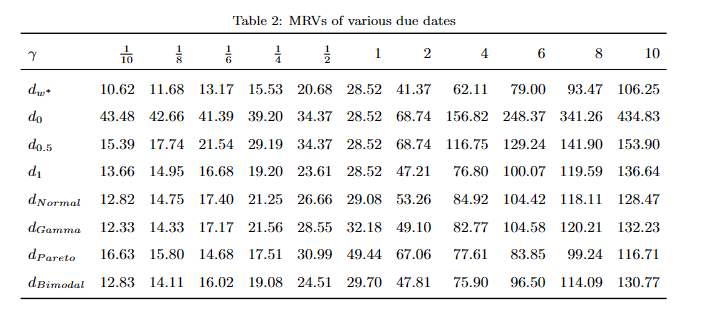
二、The programs in folder “Performance of various due dates -- slove MRV & RV” are used for solving MRV and RV

**1. "mianMRV.m" realizes the calculation process of MRV, and Table 2 is obtained.**

Where "getMRV.m" is called to solve the MRV value for , and the MRV value for when the job completion times of jobs follow the normal distribution, gamma distribution, pateto distribution, and bimodal distribution.

When calling "getMRV.m", it will further invoke "solvex.m" and "solvey.m", which are specifically designed to implement the process functions called during the calculation of MRV. The computation method refers to Proposition 1 from Yue et al. (2006).



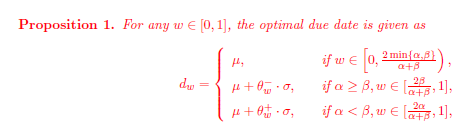


**2. "mianRV" is used to solve for the corresponding RV when the job completion times follow normal, gamma, Pareto, and bimodal distributions, for the cases of 、、、.**

"getxw" is called to solve for ，the formulas are as follows:

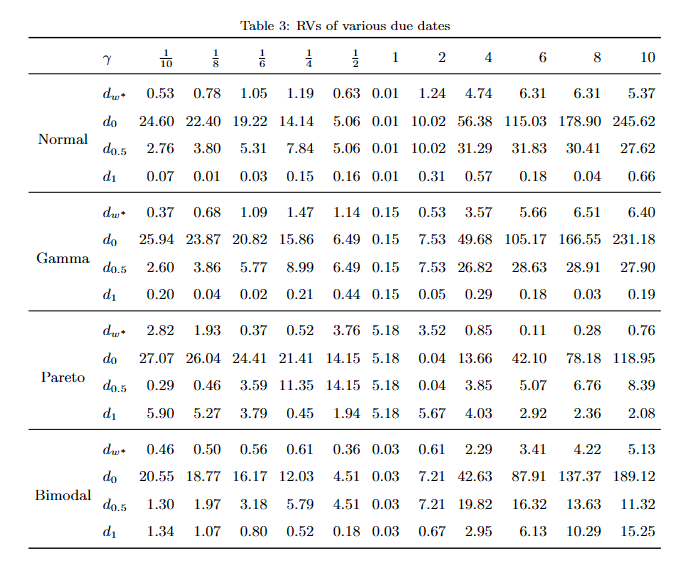


“getxwo” is called to solve for （we need to get 、、）， the formulas are as follows:



"getf" function is called to calculate the expected earliness and tardiness cost of each job, and the formulas are as follows:





3. “generatemusigma.m ” is used to generate data for the mean () and standard deviation () of completion times of 100 jobs，where the mean and standard deviation of the completion time are randomly generated from uniform distributions U(10, 100) and U(0.1, 0.25).

Where "getSigma.m" is called to randomly generate the mean and standard deviation of the completion time for each job based on the number of jobs, and the upper and lower bounds of the mean completion time.

4. "generateCompleteTime / generateCompleteTime.m" is used to generate completion times for 100 jobs, with each job’s completion time producing 10^4 examples. The process is repeated 4 times, following a normal distribution, gamma distribution, pareto distribution, and a bimodal distribution. The other functions called within have the same functionality as mentioned above, specifically for finding an appropriate bimodal distribution.

5. Data Description：

"MRV\_matrix.mat" stores the values of MRV for 100 jobs when ranges from 0.1 to 10 (0.1, 0.2, ..., 1, 2, ..., 10) and for the corresponding w values of and 1. It also includes the MRV values corresponding to when the completion times of the jobs follow normal, gamma, pareto and bimodal distributions. Table 2 is derived by taking the average of these data for the 100 jobs.

"saveData/N\_RV.mat", "saveData/G\_RV.mat", "saveData/P\_RV.mat", and "saveData/S\_RV.mat" separately store the RV for 100 jobs when the completion times follow normal, gamma, pareto, and bimodal distributions, with ranging from 0.1 to 10 (0.1, 0.2, ..., 1, 2, ..., 10) and d being ,、**、、**. Table 3 is derived by taking the average of these data for the 100 jobs.

**"mat2excel"** writes the saved .mat data into Excel, allowing direct access to Table 2 (corresponding to MRVresult.xlsx) and Table 3 (corresponding to RVresult.xlsx).

三、“Algorithm efficacy“

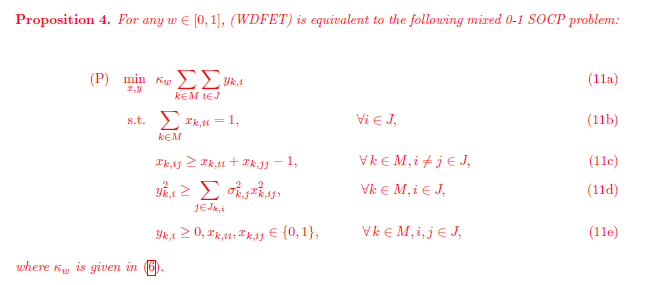
The programs in the "small-scale instances 2 3 5" folder are used to generate Table 4, with the number of machines m = 2, 3, 5 and the number of jobs n = 4m, 5m, 6m. The programs in the "large-scale instances 10 15 20" folder are used to generate Table 5, with the number of machines m = 10, 15, 20 and the number of jobs n = 4m, 5m, 6m.

1. Under the “SOCP” folder

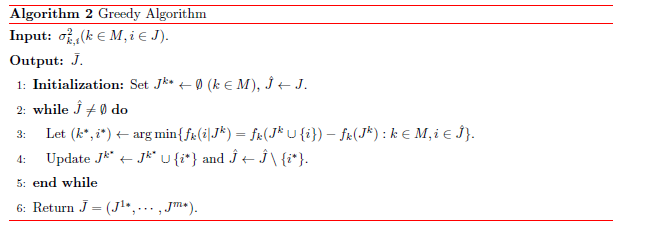
**（1）****The “SOCP/getSOCPresult” is uesed to call the Gurobi solver to solve problem (P).**

The "SOCP/solveByMixedIntegerSOCP.m" is called to implement the modeling of problem (P). The "SOCP/solveByMixedIntegerSOCPnew.m" also implements the modeling of problem (P), but the difference is that after modeling, it assigns an initial solution obtained by a greedy algorithm to Gurobi.

“prepareForSOCP” is called to extract some of the constraints and the objective function from the SOCP to save time on repetitive modeling.



（2）"SOCP/generateSigma.m" calls "SOCP/getSigma" to randomly generate the mean and standard deviation of the completion time for each job based on the number of machines, the number of jobs, and the upper and lower bounds of the mean completion time. The generated datasets are saved in the "SOCP/jobdata/mu10\_100" folder.

（3） "SOCP/getInitialSolutionByGreedyAlgorithm.m" implements the greedy algorithm to obtain a satisfactory solution, which is then used as an initial solution for Gurobi when calling "solveByMixedIntegerSOCPnew.m". The process of the greedy algorithm is as follows:

（4）Data Description：

"SOCP/result/SOCP\_m2n8tol5.mat", ..., "SOCP/result/SOCP\_m5n30tol5.mat" store the objective values, lower bounds, modeling times, and solution times obtained by calling the Gurobi solver for different scenarios where the number of machines is m = 2, 3, 5, and the number of jobs is n = 4m, 5m, 6m. A total of 10 groups of examples are included, and Table 4 presents the average values of these 10 groups of examples.

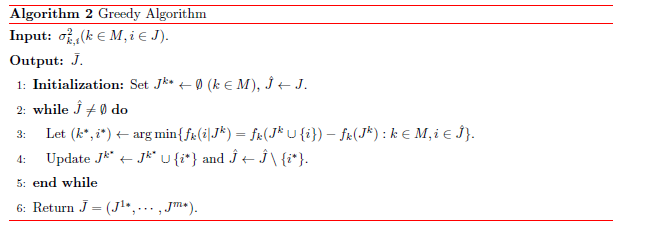
2. Under the "TPA" folder

（1）"TPA/ifupdateX" implements the logic to determine whether an update is necessary based on whether the objective value has decreased.

Within this, "getf" is called to calculate the target value for two machines under the current decision.



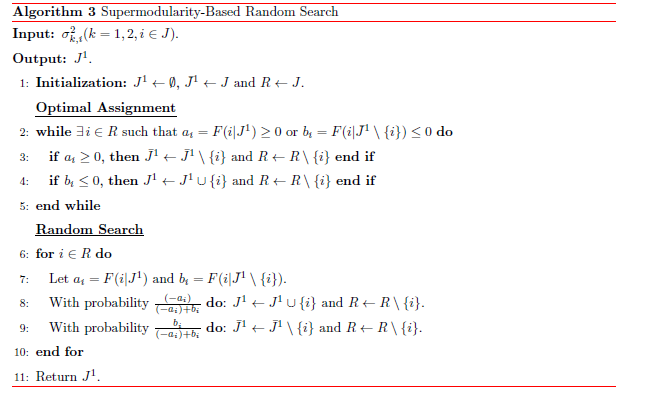
（2）“TPA / getInitialSolutionByGreedyAlgorithm.m“ implements the greedy algorithm to obtain a satisfactory solution. The process of the greedy algorithm is as follows:



（3）"TPA / InsertOperation.m" implements the functionality of inserting the i-th job from the k-th machine into the p-th position of another machine, which essentially means moving a job from one machine to another.

"TPA / ImproveSolutionByInsert.m" will invoke "InsertOperation.m" once to implement the algorithm process of improving a solution through insertion search.

"TPA / ImproveSolutionByExchange.m" will invoke "InsertOperation.m" twice to implement the algorithm process of improving a solution through exchange search. Specifically, it inserts the i-th job from machine k into position p1-th of machine s, and simultaneously inserts the j-th job from mmachine s into position p2-th of machine k.

（4）"TPA/solveByRHA\_2.m" implements the random search algorithm based on supermodularity for two machines, while "TPA/solveByRHAWithMultiM.m" repeatedly calls "solveByRHA\_2.m" to implement the same random search algorithm based on supermodularity for multiple machines.

**（5）TPA / mainprog" implements the TPA algorithm to solve the model. It calls "getInitialSolutionByGreedyAlgorithm.m" once to obtain an initial solution using the greedy algorithm, and repeatedly calls "ImproveSolutionByInsert.m", "ImproveSolutionByExchange.m", and "solveByRHAWithMultiM.m" to improve the solution.**

The solution obtained by the greedy algorithm, the last insertion search, the last exchange search, and the last solution from the random search algorithm based on supermodularity are all saved, including the objective value, number of iterations, iteration time, and total time.

The data are saved in the folders "TPA/GAresult", "TPA/InsertExchange", and "RHAresult" respectively.

3. Under "small-scale instances 2 3 5" folder and "large-scale instances 10 15 20" folder, there is a file named“mat2excel”, which converts the solutions obtained by the aforementioned algorithms into Excel files, named "table4result.xlsx" and "table5result.xlsx" respectively. The results of "table4result.xlsx" are shown in Table 4 and the results of "table5result.xlsx" are shown in Table 5.

