Our Algorithm :

The heuristic algorithm we designed can roughly be separated to 2 sections, they are the main section, the tardy section , respectively. During the main section, we designate the jobs-to-machine one phase by one phase, until all the jobs are considered either done in due time or be labeled as tardy. In other words, we can tell the definite amount of tardy jobs we obtained through this algorithm in main section, and after the tardy section, we can obtain the makespan.

This is how our main section goes : as long as there are jobs in jobs list that are not ‘finished in due time’ or be ‘labeled as tardy’ and be put in tardy list (which we will use in the tardy section) we will do the following two steps : do the normal allocation and then do 0 to several times of emergency allocation until there are no any emergency jobs. Start from normal allocation, we sort the machines by ‘the least accumulation time first’ (if equal, choose the smaller index first); our criteria to sort the jobs is by the least ‘due time – processing time first’ (if equal, choose the smaller index first), we deem this kind of jobs as the most emergency, and has the highest priority to do. Every time we arrange some job-to-machine, we keep update the job phase and change their judging criteria. For the jobs that aren’t allocated, the due time – processing time is ‘due – processing time 1 – processing time 2’, as for the jobs finished the first part, it is ‘due – processing time 2’. If the job criteria is negative, we consider it tardy. After the sorting, we look at the sorted machine from the beginning one, and find the first one which can do the job. \*If the chosen job is not allocated, we check if the chosen machine can do the job in its due time, we also consider it tardy if it can’t. Note that if the finished job doesn’t have stage2, we remove it from jobs list and say it’s done ; if it has, we put it into another list called ‘stage 2’ , which can be used by the emergency step. It is important that whenever the job’s second phase is finished sometime, we need to delete it from corresponding emergency list(stage 2), vise versa.

Later, to deal with emergency step, we first sort machines, and the jobs in stage 2 list(this time is by ‘due – processing time 2’( if equal, choose the one with larger processing time 2), we set this criteria because we want the jobs in stage 2 can all be finished, or we will waste the time dealing its first stage. And the one with larger processing time 2 maybe harder to put into machine. To reach the goal, we also set up a floating number ‘a’, it fluctuates when the ratio between total machine number and jobs in stage 2 list changes. The larger ‘a’ means the easier the emergency condition may be alerted. Every time we start from looking at the first job in sorted stage 2, if any machine alerts the emergency, we need to deal with this job by put it into the first sorted machine which can do the job. If no machine can handle it, it’s a tardy job and be put into tardy list. However, if emergency being alerted, we need to either put the job into machine and say it’s done or put it in tardy list, and do this emergency step again.

Last, when the main section is finished, the remaining jobs are all tardy and they have all being put in a tardy list. We want to obtain the smallest makespan by put the longer job in the smaller cumulative machine. We run two times of this list, the first time we deal with the jobs which haven’t done phase 1, we sort the machines, and sort tardy list by ‘longest processing time 1’, and deal the jobs one by one. And afterwards, we have all jobs in tardy list with stage 2 not done. This time we sort the machines, and sort tardy list by ‘longest processing time 2’, and deal the jobs one by one. This is the end of algorithm, we can get an feasible&nice solution with few tardy jobs and small makespan.