Define an algorithm in Matlab for the following scheduling problem.

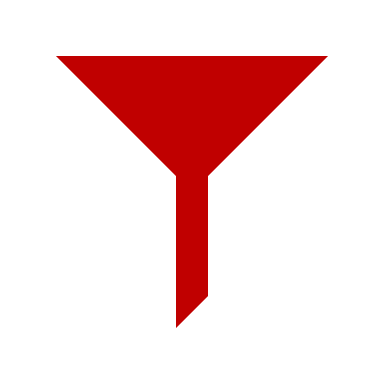
Schedule the following *n* jobs, where *pj* is the processing time on machine i. The goal is to minimize Cmax.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Job | J1 | J2 | J3 | J4 | J5 | J6 | J7 | J8 | J9 | J10 |
| *P1* | 5 | 3 | 6 | 8 | 4 | 12 | 12 | 5 | 3 | 2 |
| *P2* | 12 | 6 | 1 | 5 | 6 | 15 | 3 | 2 | 8 | 8 |
| *P3* | 1 | 20 | 2 | 5 | 7 | 11 | 12 | 2 | 5 | 4 |
| *P4* | 1 | 1 | 1 | 15 | 6 | 2 | 1 | 4 | 4 | 3 |
| *P5* | 2 | 6 | 2 | 11 | 5 | 3 | 2 | 7 | 8 | 3 |

Verify that the solution that has been obtained is optimal or not comparing with the solution obtained in a mathematical programming problem defined in Excel (or other spreadsheet tool with optimization module) or Lingo or Cplex or Matlab.

PROJECT SOLUTION

**PART 1) Problem:** we have five machines (Mn) and ten jobs (Jn) to schedule into the five machines. There is and issue in the system presented because the jobs have 2 possible routes that converge into only one machine (M4) with the risk of a bottleneck.

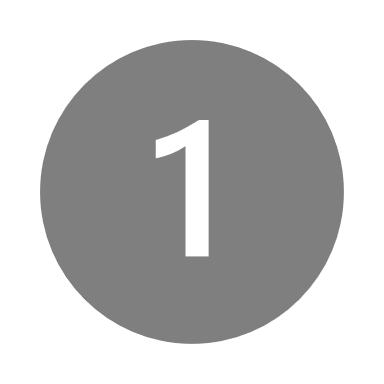


Bottleneck risk

So the idea is to have a smart way to define the order and how to route the jobs to avoid the bottleneck into the machine 4. Our proposal is to consider machine 2 and machine 3 as a unique machine exploiting the parallelism between the two machines.

After deciding how to manage the machines, we sorted the jobs in ascending order by duration of execution of the jobs on machines 1 to 4/5.

At this point we divided them into two groups: **Way2** for the jobs with short time if they pass for the Machine2 compared to the other group, **Way3** with the jobs with short time if they pass for the Machine3.







Way2



Way3

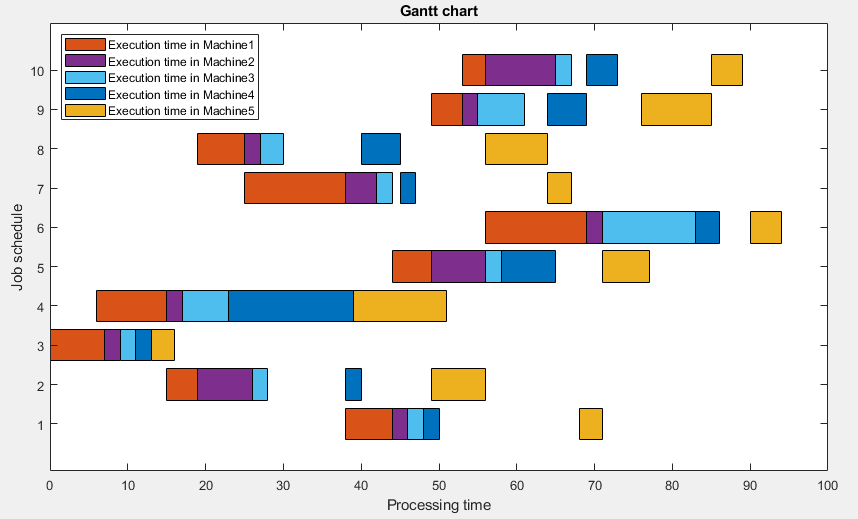


In the following tables all the values saved in **listCompletedJob** for each Job:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Numero del Job | 3 | 4 | 2 | 8 | 7 |
| Direzione macchine | Way2 | Way3 | Way2 | Way3 | Way2 |
| Tempo tot esecuzione | 10 | 39 | 16 | 18 | 18 |
| Tempi esecuzione macchine | 6 / 1 / 2 / 1 / 2 | 8 / 5 / 5 / 15 / 11 | 3 / 6 / 20 / 1 / 6 | 5 / 2 / 2 / 4 / 7 | 12 / 3 / 12 / 1 / 2 |
| Tempi di Inizio Job | 0 / 7 / -1 / 9 / 11 | 7 / -1 / 16 / 22 / 38 | 16 / 20 / -1 / 38 / 50 | 20 /-1 / 26 / 40 / 57 | 26 / 39 / -1 / 45 / 65 |
| Tempi di Fine Job | 6 / 8 / 0 / 10 / 13 | 15 / 0 / 21 / 37 / 49 | 19 / 26 / 0 / 39 / 56 | 25 / 0 / 28 / 44 / 64 | 38 / 42 / 0 / 46 / 67 |
| Tempi di attesa | 0 / 0 / 0 / 0 / 0 | 6 / 0 / 0 / 0 / 0 | 15 / 0 / 0 / 10 / 9 | 19 / 0 / 0 / 10 / 11 | 25 / 0 / 0 / 1 / 17 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Numero del Job | 1 | 5 | 9 | 10 | 6 |
| Direzione macchine | Way3 | Way2 | Way3 | Way2 | Way3 |
| Tempo tot esecuzione | 20 | 21 | 23 | 16 | 32 |
| Tempi esecuzione macchine | 5 / 12 / 1 / 1 / 2 | 4 / 6 / 7 / 6 / 5 | 3 / 8 / 5 / 4 / 8 | 2 / 8 / 4 / 3 / 3 | 12 / 15 / 11 / 2 / 3 |
| Tempi di Inizio Job | 39 / -1 / 45 / 47 / 68 | 45 / 50 / -1 / 57 / 71 | 50 / -1 / 54 / 64 / 77 | 54 / 57 / -1 / 69 / 86 | 57 / -1 / 70 / 82 / 90 |
| Tempi di Fine Job | 44 / 0 / 46 / 48 / 70 | 49 / 56 / 0 / 63 / 76 | 53 / 0 / 59 / 68 / 85 | 56 / 65 / 0 / 72 / 89 | 69 / 0 / 81 / 84 / 93 |
| Tempi di attesa | 38 / 0 / 0 / 0 / 18 | 44 / 0 / 0 / 0 / 6 | 49 / 0 / 0 / 3 / 7 | 53 / 0 / 0 / 2 / 12 | 56 / 0 / 0 / 0 / 4 |

In the following image there is the Gantt chart which arranges and shows with its formatting the execution of jobs on individual machines according to our algorithm.



In the end the Total Execution Time of the Jobs scheduling for our algorithm is **94 seconds**.

PROJECT SOLUTION

**PART 2)** Define the Jobs scheduling of our system using the **Mathematical Programming**