Exercise 1

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* | 13 | 10 | 1 | 15 | 6 | 12 | 11 | 4 | 4 | 13 |
| *P5* | 2 | 6 | 2 | 1 | 5 | 13 | 2 | 7 | 18 | 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.

Dopo una prima analisi del problema abbiamo capito che ogni job attraversa sempre tre machine, per cui la sequenza di attraversamento può essere divisa in tre fasi. La prima fase comprende l’attraversamento della macchina 1 oppure della macchina 2, La seconda fase comprende l’attraversamento della macchina 3 oppure della macchina 4, mentre la terza fase comprende l’attraversamento della macchina 5, che è il collo di bottiglia del Sistema, ovvero tutti I job devono essere processati su di essa.

Abbiamo quindi definito un’euristica in Matlab per risolvere il problema senza analizzare tutti i possibili casi.

* In questo algoritmo scegliamo sempre di schedulare i job sulla **prima macchina disponibile**. Se entrambe le macchine sono disponibili **si sceglie quella su cui il job impiega minor tempo**. Utilizziamo un counter per ogni macchina, che rappresenta il tempo totale di processamento dei job schedulati su quella macchina.

Abbiamo poi cercato di formulare il problema in Lingo come un problema di programmazione matematica. L’obiettivo è quello di minimizzare il makespan, cioè il tempo di completamento del job che finisce per ultimo. Abbiamo fatto questo definendo il makespan come maggiore o uguale di tutti i completion time, che unito al fatto che la funzione obiettivo minimizza il makespan, lo renderà uguale al tempo di completamento del job che finisce per ultimo. Abbiamo definito delle variabili opportune, ad esempio x(k, i, j), che vale 1 se il job i è eseguito prima del job j sulla macchina k, e y(k, i), che vale 1 se il job i è eseguito sulla macchina k. La variabile y ci indica quali macchine un job attraversa durante il suo processamento. Perciò avremo dei vincoli su questa variabile, cioè che ogni job è eseguito o sulla macchina 1 o sulla macchina 2, che ogni job è eseguito o sulla macchina 3 o sulla macchina 4 e che sicuramente tutti i job sono eseguiti sulla quinta macchina. Altri tipi di vincoli messi sono quelli di precedenza che indicano l’ordine in cui possono essere eseguiti i job sulle macchine. Poi si deve vincolare i job eseguiti dopo altri su una macchina ad avere uno starting time maggiore del completion time di chi li ha preceduti.