CODE:

.mod file:

// Define the number of jobs, operations, and machines

int numJ = ...; // Number of jobs

int numO = ...; // Number of operations

int numM = ...; // Number of machines

range rangeJ = 1..numJ;

range rangeO = 1..numO;

range rangeM = 1..numM;

// Define a large constant for the big-M method

int bigM = 10000000;

// Define tuples for jobs and operations

tuple JO {

int job;

int operation;

}

setof(JO) JOSet = ...; // Set of all job-operation pairs

tuple JJO {

int jobprv;

int jobaft;

int operation;

}

setof(JJO) JJOSet = {<i, j, o> | <i, o> in JOSet, <j, o> in JOSet};

// Define input parameters

float A[JOSet][rangeM] = ...; // The machine assignment for operations

float P[JOSet] = ...; // Processing time of operations of jobs

// Define decision variables

dvar float+ S[JOSet]; // Start time of operations of jobs

dvar float+ C[JOSet]; // Completion time of operations of jobs

dvar boolean X[JJOSet][rangeM]; // =1 if job i is processed before job j on machine m at operation o

dvar boolean Y[JJOSet][rangeM]; // =1 if job i and job j are assigned on machine m at operation o

dvar float+ Co[rangeJ]; // Completion time of jobs

dvar float+ Cmax; // Makespan

// Objective: Minimize the makespan

minimize Cmax;

subject to {

// Constraint 1: Calculate completion time of each operation

forall(<j, o> in JOSet) {

C[<j, o>] == S[<j, o>] + P[<j, o>];

}

// Constraint 2: Calculate completion time of each job and makespan

forall(j in rangeJ) {

Co[j] >= C[<j, numO>]; // Assuming numO is the last operation of each job

Cmax >= Co[j];

}

// Constraint 3: Enforce operation order within each job

forall(j in rangeJ, o in 1..numO - 1) {

C[<j, o>] <= S[<j, o + 1>];

}

// Constraint 4: Prevent overlapping on the same machine

forall(i in rangeJ, j in rangeJ: i != j, o in rangeO, m in rangeM) {

// Ensure that operations are assigned to the same machine

A[<i, o>][m] >= Y[<i, j, o>][m];

A[<j, o>][m] >= Y[<i, j, o>][m];

A[<i, o>][m] + A[<j, o>][m] - 1 <= Y[<i, j, o>][m];

// Ensure that operations do not overlap

Y[<i, j, o>][m] <= X[<i, j, o>][m] + X[<j, i, o>][m];

X[<i, j, o>][m] + X[<j, i, o>][m] <= 1;

S[<j, o>] >= C[<i, o>] - bigM \* (1 - X[<i, j, o>][m]);

S[<i, o>] >= C[<j, o>] - bigM \* (1 - X[<j, i, o>][m]);

}

}

.dat file:

SheetConnection Data("Data.xlsx");

numJ from SheetRead(Data,"indices!B2");

numO from SheetRead(Data,"indices!B3");

numM from SheetRead(Data,"indices!B4");

JOSet from SheetRead(Data,"P!A2:B25");

A from SheetRead(Data,"M!C2:Q25");

P from SheetRead(Data,"P!C2:C25");

File Data: