MILP Formulation for the Job Scheduling Problem with Setup Times

Akshay Khandelwal, Grace Mary Matson, Omkar Chaudhari

Abstract

This paper proposes an MILP formulation for the Job Scheduling Problem with Setup Times in between orders being processed. The algorithm attempts to solve for minimum processing cost of the entire process. The paper then expands to solve a multiobjective problem minimizing processing cost, processing time and makespan using the eps constraint method.

Keywords: job scheduling, integer linear programming

1 Problem Structure

The problem contains a set of operations I using a set of parallel dissimilar machines J. Each order $i \in I$ can only begin processing after its release date $r(i)$ and has to be completed before its due date $d(i)$. Order i can be solved on any machines, however, it cannot be split across multiple machines, and once an order starts processing, it cannot be interrupted. Each order $i \in I$ on machine $j \in J$ has a processing time $p(i,j)$ and processing cost $c(i,j)$ associated with it. Each ordered pair of orders $(i, ip) \in I \text{x} I$ has a setup time $setup(i, ip)$ associated with it, which constrains the start time of order j ($s(ip)$) to be atleast $setup(i, ip)$ greater than the start time of order i ($s(i)$). The objective of the problem is to find a least cost schedule to process all the orders within their time frames. We have also expanded the problem to minimize total processing time, and makespan, which is the time between the processing of the first order and the finishing of the last order, The problem is defined as,

$$p_{cost} = \min \sum_{i \in I} \sum_{j \in J} c(i,j) m(i,j)$$

$$p_{time} = \min \sum_{i \in I} \sum_{j \in J} p(i,j) m(i,j)$$

$$makespan = \min \left(\max \left(f(i) \right) - \min \left(s(i) \right) \right)$$

subject to:

$$f(i) = s(i) + \sum_{j \in J} p(i,j) m(i,j) \quad \forall i \in I \tag{1}$$

$$\sum_{j \in J} m(i,j) = 1 \quad \forall i \in I \tag{2}$$

$$f(i) \leq d(i) \quad \forall i \in I \tag{3}$$

$$f(i) + adj(i,ip) * setup(i,ip) \leq s_{ip} + U * (1 - B(i,ip)) \quad \forall i, ip \in I \tag{4}$$

$$1 + s(ip) \leq f(i) + adj(i,ip) * setup(i,ip) + U * B(i,ip) \quad \forall i, ip \in I \tag{5}$$

$$\sum_{j \in J} m(i,j) id(j) - \sum_{j \in J} m(ip,j) id(j) \leq U * (1 - A(i,ip)) \quad \forall i, ip \in I \tag{6}$$

$$1 + \sum_{j \in J} m(ip,j) id(j) \leq U * A(i,ip) + \sum_{j \in J} m(i,j) id(j) \quad \forall i, ip \in I \tag{7}$$

$$1 \leq B(ip,i) + B(i,ip) + U * (1 - (A(i,ip) + A(ip,i) - 1)) \quad \forall i, ip \in I, where, i \neq ip \tag{8}$$

$$(B(i,ip) + (A(i,ip) + A(ip,i) - 1)) - 1 \leq U * Z(i,ip) \quad \forall i, ip \in I \tag{9}$$

$$1 \leq (B(i,ip) + (A(i,ip) + A(ip,i) - 1)) - 1 + U * (1 - Z(i,ip)) \quad \forall i, ip \in I \tag{10}$$

$$C(i) = \sum_{ip \in I} Z(i,ip) - Z(i,i) \quad \forall i \in I \tag{11}$$

$$C(i) - C(ip) \leq U * Y(i,ip) \quad \forall i, ip \in I \tag{12}$$

$$1 \leq C(i) - C(ip) + U * (1 - Y(i, ip)) \quad \forall i, ip \in I \tag{13}$$

$$2 \leq U * Yp(i, ip) + C(i) - C(ip) \quad \forall i, ip \in I \tag{14}$$

$$C(i) - C(ip) \leq 1 + U * (1 - Yp(i, ip)) \quad \forall i, ip \in I \tag{15}$$

$$((Y(i, ip) + Yp(i, ip) - 1) + (A(i, ip) + A(ip, i) - 1)) - 1 \leq U * adj(i, ip) \quad \forall i, ip \in I \tag{16}$$

$$1 \leq ((Y(i, ip) + Yp(i, ip) - 1) + (A(i, ip) + A(ip, i) - 1)) - 1 + U * (1 - adj(i, ip)) \quad \forall i, ip \in I \tag{17}$$

$$\sum_{i \in I} Xf(i) = 1 \tag{18}$$

$$\sum_{i \in I} Xs(i) = 1 \tag{19}$$

$$lf \geq f(i) \quad \forall i \in I \tag{20}$$

$$es \leq s(i) \quad \forall i \in I \tag{21}$$

$$lf \leq U * (1 - Xf(i)) + f(i) \quad \forall i \in I \tag{22}$$

$$es + U * (1 - Xs(i)) \geq s(i) \quad \forall i \in I \tag{23}$$

Table 1: Variable Table

Variable	Type	Significance
m(i,j)	Binary	1 if order i is processed on machine j
A(i, ip)	Binary	1 iff order i occurs on a machine with number less than or equal to that of ip
B(i, ip)	Binary	1 iff order i finishes before or at start time of ip with setup time required if i is processed before ip in between
Z(i , ip)	Binary	1 iff order i and ip on same machine and i finishes at or before ip
Y(i, ip)	Binary	1 iff no of orders after i on m(i) - no of orders after ip on m(ip) ≥ 1
Yp(i, ip)	Binary	1 iff no of orders after i on m(i) - no of orders after ip on m(ip) ≤ 1
adj(i, ip)	Binary	1 iff order ip is the order next after i on the same machine
Xf(i)	Binary	1 if i is the Latest Order
Xs(i)	Binary	1 if i is the Earliest Order
s(i)	Integer	start time of order i
f(i)	Integer	finish time of order i
C(i)	Integer	number of orders scheduled after order i on the same machine
lf	Integer	largest finish time
es	Integer	earliest start time
U	Scalar	Very Large Constant

- Constraint 2 makes sure only one machine is allotted to any order by summing each row and equating to 1.
- Constraint 3 ensures every order should finish before its due date d(i).
- A is an order NxN matrix which denotes if machine number of order i is less than or equal to order ip . The purpose of this matrix is to use A + A' - 1 to find which orders have been allotted the same machine.
- Constraints 6 and 7 fix matrix A.
- If two orders i and ip are on the same machine, then i must finish before ip or ip must finish before i, this condition is satisfied by constraint 8.
- Constraints 9 and 10 fix matrix Z.
- Z is further used in calculating vector C which stores the number of orders scheduled after i on the same machine in constraint 11.
- Y and Yp are matrices such that Y + Yp - 1 will give whether the number of orders scheduled after ip on its allotted machine is exactly 1 less than the orders scheduled after order i on its machine. This is used in setting matrix adj
- Constraints 12, 13 and 14, 15 are used to fix Y and Yp respectively.
- Matrix adj checks if orders i and ip are ordered on the same machine such that ip is scheduled to be next after i, and is constrained by 16, 17.
- Matrix B uses adj as a mask to calculate whether orders adjacent to each other have required setup time in between.
- Constraints 18, 19 set Xf and Xs respectively
- Constraints 20, 21 are used to confirm ls, es and constraints 22, 23 are used to constrain lf, es.

1.1 Multi-Objective Optimization

A multi objective optimization problem was solved using the eps constraint method. The codes are all available at https://github.com/akshaykhandelwal0710/JSSP-Modification-Project. The process of the eps constraint method is described as follows:

1. Generate the payoff table by minimizing each objective function individually.
2. Find the minimum and maximum value for each objective function from the payoff table.
3. Divide the range of objective values for each objective function into a set number of points, called gridpoints, (in our case 10), these serve as the epsilon constraints below.
4. For each objective function obj_i solve the following problem:
   (a) let objm(i) be the set of all objective functions other than obj_i
   (b) for each possible combination of gridpoints for the objective functions in objm(i):
      i. set gridrhs values for each objective function according to the selected gridpoints
      ii. add additional constraints to the minimization problem, $obj_j \leq$ gridrhs(obj_j) for all obj_j belongs to objm(i)
      iii. solve the problem minimizing obj_i

Table 2: Eps Constraint Variables

Variable	Significance
objm(i)	set of objective functions other than obj_i
payoff(obj_i, obj_j)	found value of obj_j while minimizing obj_i
gridrhs(obj_i)	epsilon constraint value for objective function obj_i