Using Lagrangian Relaxation Approach to Generate a Lower Bound for No-wait Flow Shop Problem

Ahmad Jafarnejad  
Professor, Department of Industrial Management, Tehran University, Tehran, Iran.

Mohammad Reza Mehregan  
Professor, Department of Industrial Management, Tehran University, Tehran, Iran.

Morteza Kazemi  
Department of Industrial Engineering, Shiraz University of Technology, Shiraz, Iran.

Ali Reza Rahmansetayesh*  
Ph.D. Student, Department of Industrial Management, Kish International Campus, Tehran University, Kish Island, Iran.  
*Corresponding author

Abstract  
No wait flow shop scheduling problem (NWFSP) by minimizing make-span is an NP-hard problem with various applications in the industry. In this paper a new lower bound for NWFSP is presented. For this, the MIP model of the problem is relaxed and divided into two separate problems with polynomial solution. Then by applying Lagrangian relaxation method an algorithm is present to obtain a lower-bound. Computational result show the efficiency of proposed algorithm.

Keywords: no-wait flow shop scheduling, Lagrangian relaxation, Allocation problem

INTRODUCTION  
In flow shop scheduling problems, the objective is to find a sequence of jobs to obtain a best objective function value. In these problems it is assumed that the machines are located in series and the operation of the machine is the same for all jobs. Most of flow shops scheduling problems are NP-hard and finding the optimal solution for them is not possible in an acceptable time.

In this paper, the no-wait flow shop scheduling problem (NWFSP) which is widely used in industry [1-2], is considered. In NWFSP once the processing of a job is begin on the first machine; it should not be any delay between the processing of all of its consecutive tasks. The purpose is to find a sequence of jobs in such a way that the completion time of the last job (makespan) is minimized.

This problem can be denoted by $F_m/nw^{t}/C_{max}$ [3] where $F_m$ denotes that it is a flow shop with m-machine, $nw^{t}$ is the stand of no-wait environment, and $C_{max}$ denotes the objective function.

As mention above, the target of scheduling is to minimize the makespan. Gilmore and Gomory [4] solved the $F_2/nw^{t}/C_{max}$ optimally by sub-tour patching technique. Their complexity of algorithm is O(nlogn). Sahni and Cho [7] show that NWFSP is NP-hard. Moon et al. [8] proposed a mixed Integer programing model for it.

Hall and Sriskandarajah [9] declare some application of NWFSP in real world. They mention that main reason for no-wait constraint is usually due to technological restrictions of the production process. Selen and Hott [10] presented a mixed integer goal programming model for solving the multi-objective NWFSP. Van der Veen and Van Dal [11] have proven that some special cases of NWFSPs are solvable in polynomial time. In 2016, Lin and Ying [12] developed a hybrid algorithm. At first, they generate a feasible solution by applying MNEH algorithm. The generated solution is transfer to its equivalent solution in asymmetric traveling salesman problem. By using Lin-Kernighan algorithm, the initial solution is improved, and Branch and Bound technique is employed to find optimal solution.

Bonney and Gundry [13] and King and Spachis [14] presented constructive heuristic algorithms to solve the $F_m/nw^{t}/C_{max}$ problem. In 1976, Bonney and Gundry [13] developed a slope matching (S/M) method which used geometrical relationships between the cumulative process times.

King and Spachis [14] proposed two single-chain heuristics and three multiple-chain heuristics to solve the $F_m/nw^{t}/C_{max}$ problem. Gangadharan and Rajendran [15] and Rajendran [16] presented additional heuristics, named GAN-RAJ and RAJ, for solving the same problem; their heuristics were shown to be superior to the S/M heuristics. Tseng and Lin [17] presented a hybrid genetic algorithm. Bertolissi [18] developed a heuristic algorithm for NWFSP.

Li et al. [19] introduced a composite heuristic (CH), based on an objective increment method, which outperformed GAN-RAJ and RAJ and had the lowest CPU time of all the algorithms to which it is compared. Huang et al. [21] considered the two-stage multiprocessor or flexible no-wait flow shop scheduling problem with unit setup time. In a flexible no-wait flowshop, there exists more than one machine in at least one stage. They presented an integer programming...
model and proposed an ant colony optimization heuristic. Laha and Chakraborty [22] presented a constructive heuristic, called the LC heuristic, for solving the Fm/nwt/Cmax problem, based on the principle of job insertion. The empirical results demonstrated that the solutions found using the LC heuristic were significantly better than those using the GAN-RAJ RAJ and two other compared heuristics. Gonzalez et al. [22] developed a hybrid genetic algorithm (GA) for solving the Fm/nwt/Cmax problem; it produced comparable or better solutions to benchmark problems than known heuristic algorithms. Aldowaisan and Allahverdi [24] proposed six meta-heuristics (SA, SA-1, SA-2, GEN, GEN-1, GEN-2) based on simulated annealing (SA) and GA to solve the problem. Their computational results showed the best two of the six algorithms to be SA-2 and GEN-2, which outperformed GAN-RAJ and RAJ, but required significantly more processing time. Schuster and framinan [25] developed the variable neighborhood search (VNS) and hybrid GA and SA methods. Qian et al. [26] proposed a hybrid differential evolution algorithm. Arabameri and Salmasi [27] who presented several heuristics based on tabu search and particle swarm optimization algorithm. Pan et al. [30] conducted a series of studies and proposed a discrete particle swarm optimization (DPSO) algorithm [29], a hybrid discrete particle swarm optimization (HDPSO) algorithm [30] and an improved iterated greedy (IG) algorithm [31]. Samarghandi and ElMekkawy [31] developed a hybrid TS and PSO algorithm (TS-PSO). Davendra et al. [32] proposed a discrete self-organizing migrating algorithm (DSOMA) for solving the problem Fm/nwt/Cmax. Ding et al. [33] proposed a Tabu-mechanism-improved iterated greedy (TMIIG) algorithm for solving the problem. Their empirical results confirmed that the TMIIG algorithm was more effective than all of the other well-performing heuristic algorithms. 

This paper, by using Lagrange relaxation method, has presented the lower bound by using integer modeling and without changing the flow shop problem to the traveling salesman problem (TSP). In next section, the given problem in this article defines and its integer modeling presents. In third section, we present Lagrange relaxation method and determination of Lagrange coefficients. Calculation results of some samples are given in the section 5. Finally, conclusions and suggestions for further research are presented.

DEFINITION OF THE PROBLEM AND MATHEMATICAL MODEL

In no-wait flow shop problem scheduling, after beginning the operation of each job on the first machine, it should not be any delay between the ending time of the job on the mashie and its beginning time on the consecutive machine. In other words, job process completion time on the machine is exactly the same as job processing beginning time on that machine. Therefore, legging of the job on the first machine should be scheduled such that to observe this limitation. In figure 1, an example of feasible scheduling for 3 machine and 4 jobs is shown.

Formally, NPWFS can be described as follows: Suppose there are n jobs and m machines. Each job \( j \in \{1,2,\ldots,n\} \) has a sequence of \( m \) operations \( O_{ij}, O_{i2}, \ldots, O_{im} \). The processing time of \( O_{ij} \) where \( i \in \{1,2,\ldots,m\} \) is denoted by \( p(j,i) \). To satisfy the no-wait restriction, the completion time of \( O_{jk} \) must be equal to the start time of \( O_{jk} \), for \( k \in \{1,2,\ldots,m-1\} \). Suppose that the job permutation \( \pi = (\pi_1, \pi_2, \ldots, \pi_n) \) represents the schedule of jobs to be processed. In [30] it shown that \( C_{\max}(\pi) \) is equal to \( \sum_{j=1}^{n} d(\pi_{j-1}, \pi_j) + \sum_{k=1}^{m} p(\pi_n, k) \) where \( d(\pi_{j-1}, \pi_j) \) represent the minimum delay on the start time of job \( \pi_j \) and \( \pi_{j-1} \) on the first machine. The value of \( d(\pi_{j-1}, \pi_j) \) can be obtain by:

\[
d(\pi_{j-1}, \pi_j) = p(\pi_{j-1}, 1) + \max\{0, \sum_{h=2}^{\max{\pi_{j-1}}m} p(\pi_{j-1}, h) - \sum_{h=1}^{\pi_j} p(\pi_j, h)\}\]

The problem is to find a permutation of jobs like \( \pi^* \) in the set of all permutations \( \pi \) such that \( C_{\max}(\pi^*) \leq C_{\max}(\pi) \)

To model the problem as a MIP, The following notation is used [12]:

Parameters:

\( N \) : number of jobs to be scheduled
\( M \) : number of machines in the no-wait flowshop
\( P_{ij} \) : processing time of job \( j \) on machine \( i \), \( j \in \{1,2,\ldots,N\}, i \in \{1,2,\ldots,M\} \)

Decision variables:

\( x_{j,k} \) : Equal 1 if job \( j \) occupies position \( k \), otherwise 0. \( j, k \in \{1,2,\ldots,N\} \)
\( C_{k,i} \) : completion time of job in position \( k \) on machine \( i \), where \( k \in \{1,2,\ldots,N\}, i \in \{1,2,\ldots,M\} \)
By this notation, the objective function can be expressed as:

$$\min C_{N,M}$$

$$C_{k,i-1} = \sum_{j=1}^{N} P_{j,i} x_{j,k} = C_{k,i}, \quad i = 2, ..., M, \quad k = 1, ..., N$$

(1)

$$C_{k-1,i} = \sum_{j=1}^{N} P_{j,i} x_{j,k} \leq C_{k,i}, \quad i = 1, ..., M, \quad k = 2, ..., N$$

(2)

$$C_{1,1} = \sum_{j=1}^{N} P_{j,1} x_{j,1}$$

(3)

$$\sum_{j=1}^{N} x_{k,j} = 1, \quad k = 1, ..., N$$

(4)

$$\sum_{k=1}^{N} x_{k,j} = 1, \quad j = 1, ..., N$$

(5)

$$C_{kl} \geq 0, \quad k = 1, 2, ..., N, \quad i = 1, 2, ..., M$$

(6)

$$x_{j,k} \in \{0,1\}; \quad j, k = 1, 2, ..., N$$

(7)

Constraint (1) causes that the operation of each job continues without any delay from the first to last machine. In constraint (2), the completion of kth job on the jth machine is at least equal to the processing time of this job plus job processing time of k-1th place on the same machine. Constraint (3) states that job completion time on the first machine is at least equal to the processing time of first job time on the first machine. Because objective function is minimization, in practice, this inequality changes to equality. Constraints (4) and (5) causes that each work places only in one place of sequence and one job allocates to each sequence. Constraint (6) is sing constraint.

By using constraint (1), we can replace constraint (2) with two following constraints:

$$C_{k,i-1} \geq C_{k-1,i}, \quad i = 2, ..., M, \quad k = 2, ..., N$$

(2-1)

$$C_{k,1} \geq C_{k-1,1} + \sum_{j=1}^{N} P_{j,1} x_{j,k}, \quad k = 2, ..., N$$

(2-2)

LAGRANGE RELAXATION

Lagrange relaxation is used to produce lower bound for the minimization problems [34]. Constraints (4) and (5) in the presented model are constraints of popular problem known as Assignment problem. $C_{k,i}$ are continues variables. By these fact, if the Lagrange coefficient corresponds to constraints (1), (2-2) and (3) is set as $\alpha_{k,i}, \cdot \beta_{k}$ and $\gamma$, respectively, the Lagrange relaxation problem can be write as below:

$$\min C_{M,N} + \sum_{k=1}^{N} \sum_{i=2}^{M} \alpha_{k,i} [C_{k,i-1} + \sum_{j=1}^{N} P_{j,i} x_{j,k} - C_{k,i}]$$

$$+ \sum_{k=2}^{N} \beta_{k}(-C_{k,1} + \sum_{j=1}^{N} P_{j,1} x_{j,k} + C_{k-1,1})$$

$$+ \gamma \sum_{j=1}^{N} P_{j,1} x_{j,1} - C_{11})$$

s. to:

$$C_{k,i-1} \geq C_{k-1,i}, \quad i = 2, ..., M, \quad k = 2, ..., N$$

$$C_{k,1} \geq \sum_{i=2}^{M} \alpha_{k,i}, \quad k = 2, ..., N$$

By definition of coefficient $\alpha_{k,i}$ is free in sign but $\beta_{k}$ and $\gamma$ are positive. This model can be divided into two separate problems that both of them can be solved in polynomial time:

1. Assignment problem

$$\min \sum_{k=1}^{N} \sum_{i=2}^{M} \alpha_{k,i} \sum_{j=1}^{N} P_{j,i} x_{j,k} + \sum_{k=2}^{N} \beta_{k} \sum_{j=1}^{N} P_{j,1} x_{j,k} + \gamma$$

$$\sum_{j=1}^{N} x_{k,j} = 1, \quad k = 1, 2, ..., N$$

$$\sum_{k=1}^{N} x_{k,j} = 1, \quad j = 1, 2, ..., N$$

$$x_{j,k} \in \{0,1\}; \quad j, k = 1, 2, ..., N$$

2. Linear programing problem

$$\min \sum_{k=1}^{N} \sum_{i=2}^{M} \alpha_{k,i} \sum_{j=1}^{N} P_{j,i} x_{j,k} + \sum_{k=2}^{N} \beta_{k} \sum_{j=1}^{N} P_{j,1} x_{j,k} + \gamma \sum_{j=1}^{N} P_{j,1} x_{j,1} - C_{11})$$

s.t: $C_{k,i-1} \geq C_{k-1,i}, \quad i = 2, ..., M, \quad k = 2, ..., N$$

$$C_{k,1} \geq \sum_{i=2}^{M} \alpha_{k,i}, \quad k = 2, ..., N$$

By solving both problems, a lower bound of the problem is obtained. As $\alpha_{k,i}$ is a free in sign, if its value set to negative, the optimal solution of second problem can be unbounded. In order to prevent this situation, following constraints are added to the second problem:

$$C_{k,1} \geq \sum_{i=2}^{M} \alpha_{k,i}, \quad k = 1, ..., N$$

$$C_{k,m} \geq C_{k-1,m}, \quad k = 2, ..., N$$
It is clear that new constraints hold in the main problem and adding it to the problem, does not eliminate any feasible solution.

**DETERMINING LAGRANGE COEFFICIENTS**

For each value of Lagrange coefficients a lower bound of the problem is obtained. To obtain the best estimate of them, we apply a popular method which is called sub-gradient method [34]. Sub-gradient is a recursive method that begins from initial values and update them in a systematic manner.

Supposed $\theta_i^t$ is the lagrangian multipliers of the constraint $\sum_i a_{ij} x_i \leq b_i \forall i$ at step $t$. The sub-gradient method is summarized as follow:

1) Consider UB equal to the upper bound that can be achieved through a heuristic approach.
2) Set $t=0$ and parameter $\lambda$ in $[0, 2]$ and $\theta_i^0 = 0$.
3) Solve the lagrangian problem for current $\theta_i^t$. Suppose LB is the value of the model.
4) Compute $G_i = \sum_j a_{ij} x_j - b_i$ for each constrain.
5) Compute the scalar value $\gamma_i$ as: $\gamma_i = \frac{UB-LB}{\sum_i \theta_i}$
6) Update the multipliers for each constrain by using $\theta_i^{t+1} = \max(\{0, \theta_i + \gamma_i G_i\})$
7) If any of the termination criteria is met, stop the algorithm otherwise go to 2.

**COMPUTATIONAL RESULTS**

In order to evaluate the lower bound obtained by this method, 21Test problems introduced by Reeves [34] are used. Its range varies from 20 jobs and 5 machines to 25 jobs and 20 machines and for each combination, 3 problems are exist. Results of calculations are presented in table (1). In our algorithm the stop criteria is set to 100 repetitions or achieving $\gamma$ value 0.001. The start values of all multipliers are set to 2. In each iteration, if no improvement is occurred in the value of objective function, multipliers reduce to half.

**Table 1:** The results with samples Reeves

<table>
<thead>
<tr>
<th>name</th>
<th>n*m</th>
<th>opt.</th>
<th>lower bound</th>
<th>Rat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rec1</td>
<td>20*5</td>
<td>1526</td>
<td>1087</td>
<td>0.71</td>
</tr>
<tr>
<td>Rec3</td>
<td>20*5</td>
<td>1361</td>
<td>920</td>
<td>0.67</td>
</tr>
<tr>
<td>Rec5</td>
<td>20*5</td>
<td>1511</td>
<td>1022</td>
<td>0.67</td>
</tr>
<tr>
<td>Rec7</td>
<td>20*10</td>
<td>2042</td>
<td>1027</td>
<td>0.50</td>
</tr>
<tr>
<td>Rec9</td>
<td>20*10</td>
<td>2042</td>
<td>973</td>
<td>0.47</td>
</tr>
<tr>
<td>Rec11</td>
<td>20*10</td>
<td>1881</td>
<td>945</td>
<td>0.50</td>
</tr>
<tr>
<td>Rec13</td>
<td>20*15</td>
<td>2545</td>
<td>968</td>
<td>0.38</td>
</tr>
<tr>
<td>Rec15</td>
<td>20*15</td>
<td>2529</td>
<td>946</td>
<td>0.37</td>
</tr>
<tr>
<td>Rec17</td>
<td>20*15</td>
<td>2587</td>
<td>937</td>
<td>0.36</td>
</tr>
<tr>
<td>Rec19</td>
<td>30*10</td>
<td>2850</td>
<td>1555</td>
<td>0.54</td>
</tr>
<tr>
<td>Rec21</td>
<td>30*10</td>
<td>2821</td>
<td>1499</td>
<td>0.53</td>
</tr>
<tr>
<td>Rec23</td>
<td>30*10</td>
<td>2700</td>
<td>1479</td>
<td>0.54</td>
</tr>
<tr>
<td>Rec25</td>
<td>30*15</td>
<td>3593</td>
<td>1487</td>
<td>0.41</td>
</tr>
<tr>
<td>Rec27</td>
<td>30*15</td>
<td>3431</td>
<td>1402</td>
<td>0.40</td>
</tr>
<tr>
<td>Rec29</td>
<td>30*15</td>
<td>3291</td>
<td>1364</td>
<td>0.41</td>
</tr>
<tr>
<td>Rec31</td>
<td>50*10</td>
<td>4307</td>
<td>2511</td>
<td>0.58</td>
</tr>
<tr>
<td>Rec33</td>
<td>50*10</td>
<td>4424</td>
<td>2582</td>
<td>0.58</td>
</tr>
<tr>
<td>Rec35</td>
<td>50*10</td>
<td>4397</td>
<td>2585</td>
<td>0.58</td>
</tr>
<tr>
<td>Rec37</td>
<td>75*20</td>
<td>8008</td>
<td>3702</td>
<td>0.46</td>
</tr>
<tr>
<td>Rec39</td>
<td>75*20</td>
<td>8419</td>
<td>3741</td>
<td>0.44</td>
</tr>
<tr>
<td>Rec41</td>
<td>75*20</td>
<td>8437</td>
<td>3714</td>
<td>0.44</td>
</tr>
</tbody>
</table>

This experiment shows that lower bound value calculated from Lagrange relaxation is between 40-70 percent of optimal solution for the problem. As it can be seen, if the number of jobs is increase the bound is tighter. In other words, distance of lower bound reduces with optimal solution.

**CONCLUSION AND SUGGESTIONS**

This paper introduced new lower bound for no wait flow shop problem by using Lagrangian relaxation method. For this purpose, it was shown that relaxed integer model can be divided into assignment problem and a linear programming model which can be solved in polynomial time. Having a lower bound for this problem can be used in the branch and bound method for calculating optimal solution. Lower bound of the problem can be used for closing tree branches and increases the efficiency of problem solving. It is suggested that this method is used in the optimization methods for flow shop problems.

**REFERENCES**


