An Effective Heuristic Method to Minimize Makespan and Flow Time in a Flow Shop Problem

Miguel Fernández¹, Avid Roman-Gonzalez²
Department of Engineering, Pontifical Catholic University of Peru, Lima 32, Peru¹
Image Processing Research Laboratory (INTI-Lab), Universidad de Ciencias y Humanidades, Lima, Perú¹,²

Abstract—In this paper, it is presented a heuristic method for solving the multi-objective flow shop problem. The work carried out considers the simultaneous optimization of the makespan and the flow time; both objectives are essential in measuring the production system’s performance since they aim to reduce the completion time of jobs, increase the efficiency of resources, and reduce waiting time in queue. The proposed method is an adaptation of multi-objective Newton’s method, which is applied to problems with functions of continuous variables. In this adaptation, the method seeks to improve a sequence of jobs through local searches recursively. The computational experiments show the potential of the proposed method to solve medium-sized and large instances compared with other existing literature methods.

Keywords—Flow shop problem; multi-objective optimization; non-dominated solution

I. INTRODUCTION

In a flow shop environment, J jobs must be processed on a set of N machines following the same order. The flow shop problem (FSP) consists of determining the sequence of jobs that optimizes one or more performance measures within the J! possible sequences. The FSP is classified as NP-hard for most of the classic problems, for example [1]: \( F_2 | \sum c_j \), an FSP with two machines and with the aim of minimizing the sum of the completion time of all the jobs (flow time); \( F_2 | L_M \), an FSP with two machines and with the objective of minimizing the maximum delay; \( F_3 | c_M \), an FSP with three machines and with the aim of minimizing the completion time of the jobs (makespan). Given the computational complexity that the FSP presents, various heuristics and metaheuristics methods have been proposed in the literature to solve medium-sized and large instances.

Widmer and Hertz (1989) [2] proposed a heuristic method to solve the problem to minimize the makespan. This method consists of two phases: the first phase considers an initial sequence based on a solution to the traveling salesman problem, and the second phase consists of improving this solution using tabu search techniques. Ho (1995) [3] proposed a heuristic to minimize flow time. In this paper, a simulation study was carried out to test the proposed heuristic effectiveness, comparing it with other methods. Murata et al. (1996) [4] proposed a multi-objective genetic algorithm. In this paper, it is considered a weighted sum of multiple objective functions with variable weights. Ponnambalam et al. (2004) [5] proposed a multi-objective evolutionary search algorithm; the authors solve a traveling salesman problem and employ a genetic algorithm to minimize the makespan, flow time, and downtime. Pasupathy et al. (2006) [6] proposed a multi-objective genetic algorithm, using local search techniques and minimizing makespan and flow time. This algorithm makes use of the principle of non-dominance in conjunction with an agglomeration metric. One can mention other works that adopt the generic algorithm for the FSP [7, 8, 9, 10, 11].

II. PROBLEM FORMULATION

The FSP is a working system of J jobs and N machines in series, where each job must be processed in each of the N machines. All jobs must follow the same processing sequence: first on machine 1, then on machine 2, so on consecutively. The assumptions are as follows:

- Each machine works continuously and without interruptions.
- Each machine can process just one job at a time.
- Each job can be processed by one machine at a time.
- The processing times of the jobs in the machines are deterministic data.
- The setup times of the machines are included within the processing time.

The performance measures or objective functions considered are the makespan (\( c_M \)) and the flow time (\( c_F \)). The makespan optimization seeks to reduce the completion time of the jobs and aims to efficiently use resources, while the optimization of flow time reduces the average number of jobs in the queue [6]. The following notation is used to formulate the FSP:

Sets

\( i \): Job index, \( i = \{1, \ldots, J\} \)
\( k \): Order index, \( k = \{1, \ldots, K\} \)
\( m \): Machine index, \( m = \{1, \ldots, N\} \)

Parameters

\( J, K \): Numbers of Jobs
\( N \): Numbers of machines
\( d_{im} \): Processing time of job \( i \) on the machine \( m \)
The FSP is formulated as follows:

\[ \min c_M = c_{K,N} \]  
\[ \min c_F = \sum_{k=1}^K c_{k,N} \]  
Subject to:

\[ \sum_{k=1}^K R_{ik} = 1 \quad \forall i \]  
\[ \sum_{i=1}^I R_{ik} = 1 \quad \forall k \]  
\[ p_{km} = \sum_{i=1}^I d_{im} R_{ik} \quad \forall k, m \]  
\[ c_{k,m} = \text{Completion time of the job to be executed in the order } k \text{ and on the machine } m \]  

The FSP is formulated as follows:

\[
\begin{align*}
\text{Min } c_M &= c_{K,N} \\
\text{Min } c_F &= \sum_{k=1}^K c_{k,N} \\
\text{Subject to:} & \\
\sum_{k=1}^K R_{ik} &= 1 \quad \forall i \\
\sum_{i=1}^I R_{ik} &= 1 \quad \forall k \\
p_{km} &= \sum_{i=1}^I d_{im} R_{ik} \quad \forall k, m \\
c_{k,m} &= \text{Completion time of the job to be executed in the order } k \text{ and on the machine } m 
\end{align*}
\]

Variables

\[ R_{ik} \]: 1, if the job \( i \) is executed in the order \( k \); 0, in other cases.

\[ p_{km} \]: Processing time of the job to be executed in the order \( k \) and on the machine \( m \)

\[ c_{km} \]: Completion time of the job to be executed in the order \( k \) and on the machine \( m \)

A multi-objective optimization problem is defined as follows [12]:

\[ \text{Min } F(x) = \{F_1(x), \ldots, F_r(x)\} \]
\[ s.t \; x \in X \]

Where, \( x \) is a decision variable of dimension \( n \), \( x = \{x_1, \ldots, x_n\} \), and \( X \) is the search space contained in \( \mathbb{R}^n \). Generally, the search space \( X \) is generated by a set of restrictions and ranges of the decision variables. The multi-objective optimization problem consists of finding a solution \( x^* \in X \), so that for \( y \in X \) such that:

\[ F_i(y) \leq F_i(x^*) \quad \text{for all } i = 1, \ldots, r \]
\[ F_j(y) < F_j(x^*) \quad \text{for some } j = 1, \ldots, r \]

IV. NEWTON’S METHOD FOR MULTI-OBJECTIVE OPTIMIZATION

Newton's method for solving multi-objective optimization problems was developed by [13]. The method is based on a multi-start descent algorithm, which consists of generating initial solutions, which will be improved recursively, following a search direction (Newton's direction), with the objective functions.

A. Newton’s Direction

Given a function \( F: U \subset \mathbb{R}^n \rightarrow \mathbb{R}^m \) twice continuously differentiable and a non-stationary point \( x \in X \), Newton's direction in \( x \), denoted by \( s(x) \), is obtained by solving the following problem:

\[
\min \max_{j=1, \ldots, r} \nabla F_j(x)^T s + \frac{1}{2} s^T \nabla^2 F_j(x) s \quad \text{s.t. } s \in \mathbb{R}^n.
\]

The optimal value of the problem, denoted by \( \theta(x) \), and Newton's direction are determined as:

\[
\theta(x) = \inf_{s \in \mathbb{R}^n} \max_{j=1, \ldots, r} \nabla F_j(x)^T s + \frac{1}{2} s^T \nabla^2 F_j(x) s
\]
\[ s(x) = \arg \min_{s \in \mathbb{R}^n} \max_{j=1, \ldots, r} \nabla F_j(x)^T s + \frac{1}{2} s^T \nabla^2 F_j(x) s
\]

This problem is solved recursively, determining in each step \( t \), the values of \( x_t \) and \( \theta(x_t) \), and then doing \( x_{t+1} = x_t + s(x_t) \), until \( \theta(x_t) \approx 0 \) (with a certain level of tolerance), that is, until it is not possible to continue improving the objective functions simultaneously.

V. HEURISTIC METHOD FOR THE FLOW SHOP PROBLEM

In this article, a heuristic method based on Newton's method is proposed for the FSP. The proposed method adapts Newton's method, considered a discrete search space.

A. Principal Structure

The procedure starts from a randomly generated sequence of \( s^* \) jobs (initial solution). This solution is improved recursively by applying local searches in neighborhoods by the insertion method [14] and by the two-job exchange method [2]. If \( J \) is the number of jobs, the insertion method consists of removing a job placed in the \( i \)-th position and inserting it in the \( k \)-th position (see Fig. 1a), the size of the generated neighborhood is \( (J-1)^2 \). The two-job exchange method consists of exchanging the job placed in the \( i \)-th position with the job placed in the \( k \)-th position (see Fig. 1b), the size of the generated neighborhood is \( J(J-1)/2 \).
The pseudo-code of the main structure is presented below:

Main structure (NS)
\[ S \leftarrow \emptyset, ND \leftarrow \emptyset; \]
for \( i = 1, \ldots, NS \) do
\[ \text{Generate a initial solution } s^*; \]
\[ \text{improvement} \leftarrow \text{TRUE}; \]
while (improvement = TRUE) do
\[ \text{Improve the jobs sequence } s' \leftarrow s^*; \]
\[ \text{improvement} \leftarrow \text{FALSE}; \]
end
\[ S \leftarrow S \cup \{s^*\}; \]
end
\[ ND \leftarrow \text{non-dominated solutions in } S; \]
Return (ND)

Here, NS represents the number of solutions generated initially, \( S \) the set of all sequences that have been enhanced, and ND the non-dominated set of \( S \).

B. Improve the Jobs Sequence

The procedure starts from an initial sequence of jobs \( s_0 = s^* \), at \( t = 0 \); its objective is to improve at least one objective function in each iteration. In each iteration \( t \), from \( s_t \) a neighborhood \( N(s_t) \) is generated, and evaluating the parameter \( \theta \), the best neighbor (formed solution) of \( N(s_t) \) is chosen. The best neighbor of \( N(s_t) \) will be \( s_{t+1} \), and the value of \( t \) is increased by one. The procedure stops when it is no longer possible to improve a sequence (descent = FALSE). Finally, \( s^* \) is assigned the best sequence found.

VI. COMPUTATIONAL EXPERIMENTS

The computational experiments were carried out in MATLAB and executed on a computer with a 2.4 GHz processor and 2 GB of RAM.

The instances used in the experiments were taken from [15]. Each instance is represented by \( J \times N \), where \( J \) is the number of jobs and \( N \) is the number of machines. In this study, the instances TA31, TA41, TA61, and TA71 are used. The results obtained by the proposed method are compared with the results of the MOGLS [4], ENGA [8], GPWGA [10], and PG-ALS [6]. The proposed method was applied considering 100 initial solutions with ten replicas for each instance. Tables I to IV show the non-dominated solutions of the cited existing methods and the proposed method's non-dominated solutions. Fig. 2 to 5 illustrate the Pareto frontiers that are obtained by different methods.
### TABLE I. COMPUTATIONAL RESULTS OF THE TA31 INSTANCE: 50×5

| Existing Algorithms | Proposed Method |  |
|---------------------|-----------------|---|
| $c_M$               | $c_F$           | $c_M$ | $c_F$ |
| 2724                | 71531           | 2724  | 68516 |
| 2729                | 68036           | 2729  | 68139 |
| 2731                | 67028           | 2733  | 67883 |
| 2752                | 66061           | 2734  | 67826 |
| 2757                | 66052           | 2735  | 66222 |
| 2758                | 66047           | 2743  | 66158 |
| 2763                | 66032           | 2746  | 66024 |
| 2765                | 66024           | 2748  | 65977 |
| 2770                | 65979           | 2752  | 65717 |
| 2799                | 65963           | 2757  | 65531 |

### TABLE II. COMPUTATIONAL RESULTS OF THE TA41 INSTANCE: 50×10

| Existing Algorithms | Proposed Method |  |
|---------------------|-----------------|---|
| $c_M$               | $c_F$           | $c_M$ | $c_F$ |
| 3047                | 93511           | 3133  | 90663 |
| 3052                | 93013           | 3134  | 90641 |
| 3059                | 92666           | 3135  | 90448 |
| 3063                | 92602           | 3137  | 90408 |
| 3070                | 92508           | 3148  | 90364 |
| 3074                | 92493           | 3152  | 90305 |
| 3075                | 92124           | 3156  | 90254 |
| 3076                | 91757           | 3197  | 90207 |
| 3087                | 91688           | 3209  | 90165 |
| 3097                | 91256           | 3237  | 90158 |
| 3099                | 91236           | 3249  | 90099 |
| 3111                | 91149           | 3298  | 90075 |
| 3132                | 90882           |       |       |

### TABLE III. COMPUTATIONAL RESULTS OF THE TA61 INSTANCE: 100×5

| Existing Algorithms | Proposed Method |  |
|---------------------|-----------------|---|
| $c_M$               | $c_F$           | $c_M$ | $c_F$ |
| 5493                | 28784           | 5493  | 26171 |
| 5495                | 262647          | 5495  | 259338 |
| 5498                | 262335          | 5498  | 259088 |
| 5527                | 261411          | 5538  | 258507 |
| 5563                | 261071          | 5539  | 258501 |
| 5564                | 260706          |       |       |

### TABLE IV. COMPUTATIONAL RESULTS OF THE TA71 INSTANCE: 100×10

| Existing Algorithms | Proposed Method |  |
|---------------------|-----------------|---|
| $c_M$               | $c_F$           | $c_M$ | $c_F$ |
| 5801                | 325462          | 5858  | 314749 |
| 5803                | 324725          | 5877  | 312785 |
| 5804                | 318924          | 5881  | 312632 |
| 5806                | 318299          | 5892  | 312534 |
| 5816                | 318055          | 5897  | 312349 |
| 5827                | 316972          | 5904  | 312207 |
| 5832                | 316642          | 5915  | 310887 |
| 5836                | 316542          | 5920  | 310515 |
| 5837                | 316292          | 5928  | 310359 |
| 5838                | 316161          | 5934  | 310297 |
| 5840                | 315753          | 5959  | 310227 |
| 5851                | 315184          | 6001  | 310040 |
| 5856                | 314879          | 6009  | 310005 |

**Fig. 2.** Approximation of the Pareto Frontier for Instance TA31: 50 × 5.

**Fig. 3.** Approximation of the Pareto Frontier for Instance TA41: 50 × 10.
The results found show that the proposed method has obtained a good approximation of the Pareto frontier and even surpassing the solutions found by the existing methods. Experiments indicate that generating 100 initial solutions is sufficient to obtain good results in cases with ten or fewer machines.

VII. CONCLUSIONS

In this paper, a heuristic method is proposed to solve the flow shop problem, considering the simultaneous optimization of the makespan and the flow time. This method is inspired by multi-objective Newton's method.

In Section 6, the proposed method demonstrated its ability to obtain a set of satisfactory solutions in medium-sized and large instances, generating 100 initial solutions. However, for more extensive cases (concerning the number of jobs or machines), the initial solutions should be increased.

Lastly, unlike other methods, the proposed method has an advantage because it is not necessary to calibrate several parameters by carrying out previous experiments, as happens, for example, with the genetic algorithm.

REFERENCES

[1] PINEDO, M. Scheduling theory algorithms and system. 3ª Edição. New York: Prentice Hall, 2008.
[2] WIDMER, M.; HERTZ, A. A new heuristic method for the flow shop sequencing problem. European Journal of Operational Research, v. 41, p. 186-193, 1989.
[3] HO, J. Flowshop sequencing with mean flowtime objective. European Journal of Operational Research, v. 81, p. 571-578, 1995.
[4] MURATA, T.; ISHIBUCHI, H.; TANAKA, H. Multi-objective genetic algorithm and its applications to flowshop scheduling. Computers Ind. Eng., v. 30, n. 4, p. 957-968, 1996.
[5] PONNAMBALAM, S. G.; JAGANNATHAN, H.; KATARIA, M.; GADICHERLA, A. A TSP-GA multi-objective algorithm for flow-shop scheduling. The International Journal of Advanced Manufacturing Technology, v. 23, p. 909-915, 2004.
[6] PASUPATHY, T.; RAJENDRAN, C.; SURESH, R. K. A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs. The International Journal of Advanced Manufacturing Technology, v. 27, p. 804-815, 2006.
[7] REEVES, C. R. A genetic algorithm for flowshop sequencing. Computers & Operations Research, v. 22, p. 5-13, 1995.
[8] BAGCHI, T. P. Multiobjective scheduling by genetic algorithms. Boston: Kluwer, 1999.
[9] BUZZO, R. W.; MOCCELLIN, J. V. Programação da produção em sistemas flow shop utilizando um método heurístico híbrido algoritmo genético-simulated annealing. Gestão & Produção, v. 7, p. 364-377, 2000.
[10] CHANG, P. C.; HSIEH, J. C.; LIN, S. G. The development of gradual priority weighting approach for the multi-objective flowshop scheduling problem. International Journal of Production Economics, v. 79, p. 171-183, 2002.
[11] KAMIRI, N.; ZANDIEH, M.; KARAMOOZ, H.R. Bi-objective group scheduling in hybrid flexible flowshop: A multi-phase approach. Expert Systems with Applications, v. 37, p. 4024-4032, 2010.
[12] KONAK, A.; COIT, D.; SMITH, A. Multi-objective optimization using genetic algorithms: a tutorial. Reliability Engineering & System Safety, v. 91, p. 992-1007, 2006.
[13] FLIEGE, J.; DRUMMOND, L. M. G.; SVAITER, B. Newton’s method for multiobjective optimization. Optimization Online, 2008.
[14] NAWAZ, M.; ENSCORE, JR., E.; HAM, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. The International Journal of Management Science, v. 11, n. 1, p. 91-95, 1983.
[15] TAILLARD, E. Benchmarks for basic scheduling problems. European Journal of Operational Research, v. 64, p. 278-285, 1993.