Metaheuristic approaches for job scheduling in hybrid flowshop with multiple objectives

Fabricio Niebles-Atencio[1], Elyn L. Solano-Charris[2], and Jairo R. Montoya-Torres[2]

[1] Servicio Nacional de Aprendizaje (SENA)
Carrera 43 No. 42-40, Barranquilla, Colombia
fabricioniebles@sena.edu.co

[2] Escuela Internacional de Ciencias Económicas y Administrativas
Universidad de La Sabana
Campus del Puente del Común, Km. 7 autopista norte de Bogotá, D.C.,
Chía, Cundinamarca, Colombia
{erlyn.solano,jairo.montoya}@unisabana.edu.co

1 Introduction

The flow-shop scheduling problem is one of the most studied combinatorial optimization problems, due to its large number of practical application in manufacturing, transportation problems and service systems such as electronics, paper and textile industries, manufacturing of photographic films, internet service architectures, and container handling systems [1]. The general idea is to schedule a set of tasks on a line production system with more than one stage, considering one or multiple objectives to be optimized. This work focuses on a more general case of the flowshop scheduling problem in which each stage has several machines in parallel. This configuration adds more flexibility at each processing station by increasing the overall capacity, and avoiding bottlenecks if some operations are too long [2]. The problem is known as the Flexible or Hybrid Flowshop Scheduling (HFS) and objectives to be minimized are the makespan, the total tardiness and the number of tardy jobs. The HFS problem is known to be NP-hard even for the case of a system with only two processing stages when one stage contains two machines and the other stage contains a single machine [3].

Formally, the problem can be described as follows. A set of n jobs are to be processed on a set of s stages in series, each one containing a set of m_s machines in parallel. This problem denoted as $FHs,\ PM^{(l)}{}_{s=1}^{l}|\ |(C_{max}, \sum T_j, \sum U_j)$, which has $M^{(l)}$ homogeneous or identical machines (i.e., machines with equal capacity and processing speed) at each stage. Each job j ($j = 1, ..., n$) has to be processed on only one machine at each stage. A machine can only execute one job at a given time. The processing route of all jobs is identical (i.e., job j is first processed on one machine at stage 1, then on one machine at stage 2, and so on). The processing time of job j on any machine of stage s is denoted as p_{js}. All jobs are available at the beginning of the time horizon (e.g., all jobs have the same release date) and once the processing of a job is started, it cannot be interrupted (e.g., preemption is not allowed). Besides that, job j has to be finished before a given due date, denoted as d_j. The makespan is denoted as $C_{max} = \max\ \{C_j\}$, where C_j is the completion time of job j; the total tardiness is denoted as$\sum T_j$, where $T_j = \max\ \{0, C_j - d_j\}$ is the tardiness of job j; and the number of tardy jobs denoted as $\sum U_j$, where U_j is a binary variable with value equal to 1 if job j is tardy (that is, if $C_j > d_j$); and 0 otherwise.

The literature has witnessed the proposition of considerable amount of algorithms to solve mono-objective versions of the problem [1]. The multi-objective case has been solved mainly using meta-heuristic algorithms such as simulated-annealing [4], genetic algorithms[5], ant colony system algorithm [6, 7]. Objective functions include the makespan and the total flow time, or the makespan and the total tardiness. The three objective functions under study in the current paper have not been previously taken into account.

2 Solution approach and experiments

In order to solve the multi-objective HFS problem presented previously, two meta-heuristics are considered: Ant Colony Optimization (ACO) and Non-dominated Sorting Genetic Algorithm-II

(NSGA-II). The former has been shown in the literature to be a very efficient and effective meta-heuristic to solve mono-objective HFS problem [8], while the latter has shown a flexible structure and a successful application to a wide range of multi-objective combinatorial optimization problems [9]. The problem is modeled using a disjunctive graph and the basic tenets of such procedures are considered in the first instance.

Concerning the NSGA-II, the population is randomly generated. At each iteration a sorting procedure is applied to rank the solutions according its dominance over the total population. Then a solution is said to be better if it is not dominated by any other solution in the population. Selections of the best solutions found so far are used for applying the Partially Matched Crossover (PMX) and produce new solutions. Mutation with a fixed probability is also considered. The algorithm stops after a predefined number of iterations.

The computational experiments were carried out on a PC Intel Core i7, 2.9 GHz with 8GB of RAM. The proposed meta-heuristics were coded using Visual Basic 6.0. Datasets employed in our experiments were taken from the OR-Library and can be downloaded from the webpage: http://people.brunel.ac.uk/ mastjjb/jeb/orlib/multiflowinfo.htm. Instances with 20 and 100 jobs and shops with 2, 5 and 8 stages were considered. For each meta-heuristic, non-dominated solutions (i.e., the set of Pareto-optimal solutions), were registered. Up to now, the preliminary results obtained include the set of non-dominated solutions for each meta-heuristic for selected instances. An overview of these preliminary results is presented in Table 1. For example, regarding the performance of ACO, results have shown that the quality of the solution is not affected when the numbers of jobs to be scheduled are increased; while this is not the case when the number of stages increases. Further analysis is however required. This includes the evaluation of distance measures between two fronts, the coverage of the solutions of both meta-heuristics, the deviation with respect to a single objective, the deviation with respect to the best initial solution, and the computational time.

	sets of nondominated solutions $(C_{max}, \sum T_j, \sum U_j)$			
	NSGA-II: SOL1	NSGA-II: SOL1	ACO: SOL1	ACO: SOL2
P20S2T01	390, 2617, 10	389, 2601, 11	408, 2729, 12	424, 2721, 12
P20S5T01	1450, 12716, 13	1535, 12600, 14	1612, 13271, 16	1664, 13127, 16
P20S8T01	2001, 22345, 16	2090, 22350, 13	2185, 21845, 16	2105, 23811, 16
PH1S2T01	1885, 71670, 70	1879, 70601, 72	1867, 75017, 79	1898, 74824, 79
PH1S5T01	9230, 45672, 78	9321, 44567, 81	9343, 466996, 83	-
PH1S8T01	9242, 45684, 80	9331, 44577, 83	9287, 426315, 82	9355, 424216, 82

Table 1. Examples of non-dominated solutions

References

1. Ruiz, R., Vázquez-Rodríguez, J.: The hybrid flow shop scheduling problem. European Journal of Operational Research **205** (2010) 1–18.
2. Khalouli, S., Ghedjati, F., Hamzaoui, A.: An ant colony system algorithm for the hybrid flow-shop scheduling problem. International Journal of Applied Metaheuristic Computing **2** (2011) 29–43.
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: Two stage hybrid flow shop scheduling problem. IEEE Transactions on Evolutionary Computation **38** (1988) 359–364.
4. Varadharajan, T., Rajendran, C.: A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. Journal of Intelligent Manufacturing **167** (2005) 772–795.
5. Montoya-Torres, J., Vargas-Nieto, F.: Solving a bi-criteria hybrid flowshop scheduling problem occurring in apparel manufacturing. International Journal of Information Systems and Supply Chain Management **4** (2011) 42–60.
6. B. Yagmahan, M.Y.: A multi-objective ant colony system algorithm for flow shop scheduling problem. Expert Systems with Applications **37** (2010) 1361–1368.
7. Solano-Charris, E., Montoya-Torres, J., Paternina-Arboleda, C.: Ant colony optimization algorithm for a bi-criteria 2-stage hybrid flowshop scheduling problem. Journal of Intelligent Manufacturing **22** (2011) 815–822.
8. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press (2004)
9. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197.