4th International Conference on Industrial Engineering and Industrial Management
XIV Congreso de Ingeniería de Organización
Donostia- San Sebastián , September 8th -10th 2010

Lot-streaming for sequence dependent setup time flowshop problems*

Rubén Ruiz[1]

[1]Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática (ITI), Ciudad Politécnica de la Innovación, Edificio 8G, Acceso B, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022, Valencia, Spain. rruiz@eio.upv.es

Abstract

This paper considers an n-job m-machine lot-streaming flow shop scheduling problem with sequence-dependent setup times under both the idling and no-idling production cases with the objective to minimize the maximum completion time or makespan. We present a novel estimation of distribution algorithm (EDA). An estimation of probabilistic model is constructed to ensure the algorithm searching towards good regions by taking into account both job orders and similar blocks of jobs. A simple but effective local search is employed as well. A comparative evaluation is carried out and the results show that the proposed EDA is very effective in comparison.

Keywords: lot-streaming, flowshop, sequence dependent setup times, scheduling

1. Introduction

The permutation flowshop scheduling problem is one of the most extensively studied combinatorial optimization problems, which has important applications among others, in manufacturing systems or assembly lines. In a flowshop, there are *n* jobs that have to be processed on *m* machines. All jobs visit machines in the same sequence. Each job is assumed to be indivisible, and thus, it cannot be transferred to the downstream machine until the whole operation on the preceding machine is finished. Nevertheless, this is not the case in many practical environments where a job or lot consists of many identical items. For example, in the fastener production process, jobs are batches of thousands of bolts, dowels, or rivets and the whole batch does not need to be finished in order to move on to the next machine. Another example comes from the electronics and semiconductor production environment where a job is comprised of thousands of identical electronic components and it is also not necessary to wait for all items to be completed before transporting to the downstream machine. In order to accelerate production, a job is allowed to overlap its operations between successive machines by splitting it into a number of smaller sub-lots and moving the completed portion of the sub-lots to downstream machines (Yoon and Ventura 2002). Job splitting into sub-lots is usually referred to as lot-streaming and was first studied by Reiter (1966). Generally, there are two different production situations when processing the sub-lots of a job, namely, the idling and no-idling cases. In the no-idling case, the job must be continuously processed without idling production interruption time (i.e., idle time) between any two adjacent sub-lots at the same machine. The idling case is more flexible as there might be idle time in between sub-lots of the same job. We center our study around makespan minimization of the production sequence. It is well known that in mostly all situations, lot-

* This work is co-authored together with Dr. Quan-Ke Pan, College of Computer Science, Liaocheng University, Liaocheng, 252059, PR China.

streaming results in lower makespan values. Furthermore, makespan in the idling case is smaller than that of the no-idling case given the same sub-lot sizes. The potential benefits of lot streaming, as mentioned by Truscott (1986) are: (a) reduction in production lead times (thus, leading to better due-date performance); (b) reduction in work-in-process inventory and associated costs; (c) reductions in interim storage and space requirements; (d) reduction in material handling system capacity requirements. Therefore, in recent years, lot streaming has received extensive attention and has been applied to flowshop scheduling problems by many authors, one example is the work of Tseng and Liao (2008).

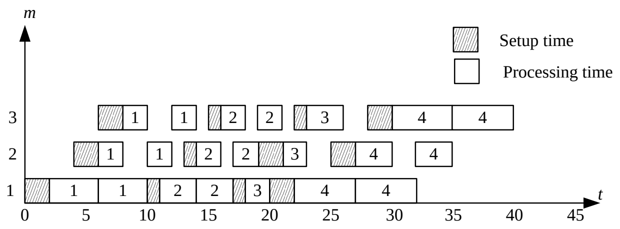
In general, setup times are one of the important factors to be considered in scheduling decisions, which involve operations such as cleaning, obtaining or adjusting tools, fixing or releasing parts to machines, and many others. Although they are not part of the job processing times, these operations have to be done prior to the processing of the jobs. Setup times are often job and machine sequence dependent and have been considered critical in may review papers, for example in the one of Allahverdi and Soroush (2008).

This paper considers lot-streaming flowshop scheduling problems with sequence-dependent setup times and makespan criterion. Both the idle and no-idle variants are studied. Although the literature on lot-streaming flowshop is rich, the addition of sequence dependent setup times has not been studied before, to the best of our knowledge, in the literature. Without loss of generality, this problem is denoted as $F_m, L_n | prmu, ST_{sd} | C_{\max}$. by using the notation of T'Kindt and Billaut (2006), Chang and Chiu (2005), and Allavherdi et al (1999) Allahverdi et al (2008), where ST_{sd} represents the sequence-dependent setup time; F_m and L_n stand for the *n*-job *m*-machine and lot-streaming flow shop configuration, respectively. Clearly, this setting, being a generalization of the traditional flowshop, is an NP-Hard problem.

A fairly recent type of metaheuristic method is also studied along with the problem. We propose an Estimation of Distribution Algorithm (EDA) for this problem. EDA was introduced by Muhlenbein and Paab (1996) and are a class of novel population-based evolutionary algorithms. Unlike the traditional evolutionary algorithms, the EDA samples new solutions from a probabilistic model which characterizes the distribution of promising solutions in the search space at each generation. EDA has recently attracted much attention in the field of evolutionary computation and has been recently applied to solve the flowshop scheduling problem in Jarboui et al, (2009) and Salhi et al, (2007). However, to the best of our knowledge, there is no published work dealing with the lot-streaming version of flowshop scheduling problem with EDA, let alone with the sequence-dependent setup times.

The main motivation behind the selection of the EDA method as a solution technique is purely out of scientific curiosity as it is a type of method that has been seldom studied for flowshop settings, let along for the complex problem studied in this setting.

Figure 1 shows, for clarification, the two cases studied in this work. Also, we show the resulting sequence if no lot-streaming is allowed.



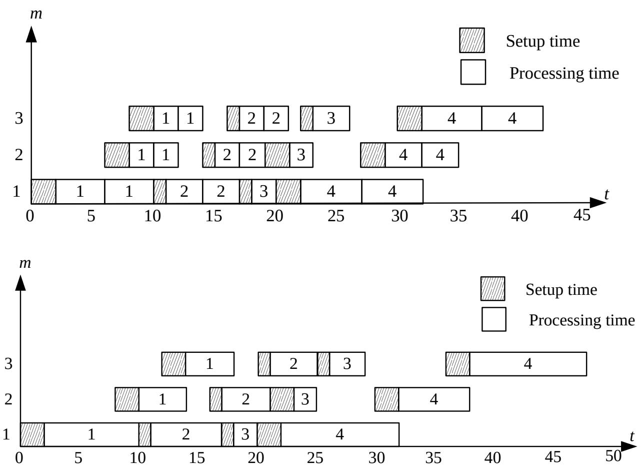


Figure 1. Four job, three machine example under the idle (top) and no-idle (middle) lot-streaming cases. Also processing sequence without lot-streaming (bottom). Note the anticipatory sequence-dependent setup times.

2. An estimation of distribution algorithm for the lot-streaming flowshop problem

EDA is based on populations of solutions that evolve within the search process with a theoretical foundation in probability theory. Instead of using the conventional crossover and mutation operations of regular genetic algorithms, EDA adopts a probabilistic model learned from a population of parent individuals to reproduce offspring in the next generation. Starting from a population of *PS* randomly generated individuals, EDA estimates a probabilistic model from the genetic information of the selected *Q* individuals in the current generation, and represents it by conditional probability distributions for each decision variable. *M* offspring are then sampled in the search space according to the estimated probabilistic model. Finally, the next population is determined by replacing some individuals in the current generation with new generated offspring. The above steps are repeated until some stopping criterion is reached. The pseudo code for the basic EDA is briefly summarized as follows:

```
Begin
    Generate a population of PS individuals;
    Calculate fitness for each individual;
    While termination criterion not met, do
        Select Q individuals and estimate a probabilistic model;
        Sample M offspring from the estimated probabilistic model;
        Evaluate the M generated offspring;
        Generate next population;
    End while;
End.
```

The previous EDA template has to be instantiated for the flowshop problem studied in this paper. We now briefly summarize how this instantiation has been carried since a full explanation is clearly not possible in the allowed limited space.

Initialization of the population is carried out with the regular NEH heuristic and one improvement that considers setup times. It is worth noticing that we have developed a set of accelerations in order to keep the computational complexity of NEH to $O(mn^2)$ instead of the original $O(mn^3)$ which results in a very fast and effective EDA population initialization.

The selection of the *Q* individuals for the probabilistic model estimation is simply carried out by selecting the *Q* best solutions from the population. The generation of the *M* offspring from the *Q* best selected individuals is key in the EDA method. This operator is briefly explained below:

Let $\rho_{i,j}$ be the number of times of appearance of job j before or in the position i. Let $\lambda_{j',j}$ represent the number of times that job *j* appears immediately after job j' in the *Q* selected individuals, which indicates the importance of the similar blocks of jobs not only in the same positions but in the different positions as well. Then, the probability of selection of the job j in the *i*-th position of the offspring is given by:

$$\xi_{i,j} = \begin{cases} \left(\dfrac{\rho_{i,j}}{\sum_{l\in\Omega(i)} \rho_{i,l}} + \dfrac{\lambda_{j',j}}{\sum_{l\in\Omega(i)} \lambda_{j',l}} \right) \Big/ 2 & i = 2,3,\ldots,n \\ \dfrac{\rho_{i,j}}{\sum_{l\in\Omega(i)} \rho_{i,l}} & i = 1 \end{cases}$$

An example with four jobs is used to illustrate the presented probabilistic model. Suppose the selected individuals are $\pi(1) = \{1,2,3,4\}$, $\pi(2) = \{2,3,4,1\}$ and $\pi(3) = \{1,4,2,3\}$. $\lambda_{j',j}$ is given below:

$$\left[\lambda_{j',j}\right]_{4\times 4} = \begin{bmatrix} - & 1 & 0 & 1 \\ 0 & - & 3 & 0 \\ 0 & 0 & - & 2 \\ 1 & 1 & 0 & - \end{bmatrix}$$

Then, we calculate the probability of selection of each job in $\Omega(1) = \{1,2,3,4\}$ for the first position as follows: $\xi_{1,1} = 2/(2+1) = 0.67$; $\xi_{1,2} = 1/(2+1) = 0.33$; $\xi_{1,3} = 0/(2+1) = 0$; $\xi_{1,4} = 0/(2+1) = 0$. Suppose job 1 was selected for the first position and $\Omega(2) = \{2,3,4\}$, then we calculate the probability of section of each job in $\Omega(2) = \{2,3,4\}$ as follows: $\xi_{2,2} = (2/(2+1+1) + 1/(1+0+1))/2 = 0.5$;
$\xi_{2,3} = (1/(2+1+1) + 0/(1+0+1))/2 = 0.125$; $\xi_{2,4} = (1/(2+1+1) + 1/(1+0+1))/2 = 0.375$.

With the previous probabilities, the following procedure is applied in order to construct complete solutions:

```
for $i = 1$ to $n$ do
    if $rand() < \varepsilon$ then
        choose the first unscheduled job in the reference sequence.
    else
        select the job $j$ according to its probability $\xi_{i,j}$.
    endif
endfor
```

In the above procedure, ε is a control parameter; $rand()$ is a random function returning a random number sampled from a uniform distribution between 0 and 1. The reference sequence is randomly chosen from the selected individuals for estimating the probabilistic model. When $rand() \ge \varepsilon$, we randomly select θ jobs from the unscheduled job set and the job with the largest $\xi_{i,j}$ is put into the i^{th} position of the new sequence π'. To generate M offspring, the above procedure is repeated M times.

Another aspect considered in the EDA is the population update for the next generation. To maintain the diversity of the population so as to avoid both cycling search and getting trapped in a local optimum, the population is updated in the following way (Ruiz et al 2006):

Step 1: Set $i=1$.

Step 2: If offspring i is better than the worst individual of the population and there is no other identical individual existing, it replaces the worst individual and become a member of the population; otherwise, it is discarded.

Step 3: Set $i=i+1$, if $i \le M$, go to step 1; otherwise stop the procedure

We employ a local search based on the job insertion operator, which is very suitable for performing a fine local search and commonly used to produce a neighboring solution in the flow shop literature (Ruiz and Stutzle 2007, Vallada and Ruiz 2009). In the local search, a job is extracted from its original position in the sequence and reinserted in all the other $n-1$ possible positions. If a better makespan value is found, the solution is replaced. We repeat the procedure until no improvements are found. According to the extraction order of jobs in the first step, the local search can be classified as referenced local search (Pan et al 2008) and local search without order (Ruiz and Stutzle 2007). Let $\pi^b = \{\pi^b{}_1, \pi^b{}_2, \ldots, \pi^b{}_n\}$ denote the best job sequence found so far, and $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ be a sequence that undergoes local search. Then the referenced local search is described as follows:

Step 1: Set $i=1$ and counter $Cnt=0$.

Step 2: Find job $\pi^b{}_i$ in permutation π and record the position.

Step 3: Take out $\pi^b{}_i$ from its original position in π. Then insert it in another different position of π, and adjust the permutation accordingly by not changing the relative positions of the other jobs. Consider all the possible insertion positions and denote the best one among the obtained sequences π^*.

Step 4: If π^* is better than π, then set $\pi = \pi^*$ and $Cnt=0$; otherwise set $Cnt = Cnt+1$.

Step 5: If $Cnt < n$, let $i = \begin{cases} i+1 & i<n \\ 1 & i=n \end{cases}$, and go to step 2; otherwise output the current permutation π and stop.

For the local search without order, its procedure is given as follows:

Step 1: Set counter $Cnt=0$.

Step 2: Remove a job at random from its original position in π without repetition. Then insert it in another different position of π, and adjust the permutation accordingly by not changing the relative positions of the other jobs. Consider all the possible insertion positions and denote the best one among the obtained sequences π^*.

Step 3: If π^* is better than π, then let $\pi = \pi^*$.

Step 4: Let $Cnt = Cnt+1$. If $Cnt < n$, go to step 2.

Step 5: If the permutation π was improved in the above Steps 1 through 4, then go to Step 1; otherwise output the current permutation π and stop.

We test both the referenced local search and local search without order in our study. The local search is applied to the generated offspring with a probability P_{ls}, that is, the individual will undergo a local search if a random number uniformly generated in the range of [0,1] is less than P_{ls}. In addition, the local search is also applied to the best individual after the initialization of the population.

We also use a diversity control operator with a partial reinitialization of the population. In the restart mechanism, the 20% best individuals are kept from the current population and the remaining 80% are generated randomly. At the same time, to reduce the computation effort spent on computing the diversity value, the diversity value is calculated at least 100 generations after the algorithm restarts.

Outside this short paper is the detailed statistical calibration of the proposed EDA method. We now jump directly to the computational results.

3. Computational results

We compare the proposed EDA method against 9 state-of-the-art algorithms proposed in the literature for the lot-streaming flowshop, with adaptations to consider setup times. These algorithms include several variants of Tabu Search, Simulated Annealing, Hybrid Genetic Algorithms, Ant Colony Optimization and Particle Swarm Optimization. The tested methods are the seven presented methods of : Marimuthu et al. (2007, 2008, 2009), including a tabu search (TS), simulated annealing with insertion neighborhood (SAi), simulated annealing with swap neighborhood (SAs), hybrid genetic algorithm (HGA), ant colony optimization (ACO), threshold accepting with insertion neighborhood (TAi), and threshold accepting with swap neighborhood (TAs). Recently, Tseng and Liao (2008) developed a discrete particle swarm optimization (DPSO) for a problem similar to the one studied in this paper. We finally consider the most recent and high performing proposed method, the EDA algorithm of Jarboui et al. (2009), referred to as EDA_{J}.

To test all methods (12 in total, since we include three versions of our proposed EDA, the all-included EDA, the one without speed-ups EDA_{nS} and the one without local search, EDA_{nL}) we employ a benchmark of 360 instances with various parameters. Of particular relevance is the parameter *l*(*j*), which indicates in how many sub-lots a given job is allowed to split into. The maximum tested size is 200 jobs and 20 machines. To make a fair comparison, all the compared algorithms adopt the same maximum CPU time limit stopping criterion of $t = n \times (m/2) \times \rho$ milliseconds, where $\rho = 100$, 200, and 300, respectively. There are a total of five replicates for each algorithm and instance. Therefore, the total number of results is 360·5·12·3=64,800.

All the algorithms are coded in Visual C++ and run on a Pentium PIV 3.0 GHz PC with 512 MB of memory. As a result, we can safely say that the computational evaluation setting is fair and comparable. We measure the average relative percentage deviation from the best known solution as a grand total of the 360 instances both for the idling and no-idling case (two separated experiments). Results are given in Tables 1 and 2.

Table 1. Average Percentage Deviation from best known solution for all tested methods under the no-idling case. Our proposed methods in bold. Three different stopping times

	EDA	EDA_{nS}	EDA_{nL}	EDA_J	HGA	ACO	TS	SAi	SAs	TAi	TAs	DPSO
$\rho = 100$	0.2794	1.1832	3.3391	4.3434	3.1241	2.1431	1.5151	3.1760	4.2812	4.3119	5.4874	2.7790
$\rho = 200$	0.2792	1.1224	3.3634	3.9596	3.0988	2.1132	1.3942	3.2879	4.3943	4.4250	5.6016	2.3587
$\rho = 300$	0.2622	1.0604	3.3398	3.7430	3.0661	2.0807	1.2556	3.3199	4.4267	4.4574	5.6343	2.1537

Table 2. Average Percentage Deviation from best known solution for all tested methods under the idling case. Our proposed methods in bold. Three different stopping times

	EDA	EDA_{nS}	EDA_{nL}	EDA_J	HGA	ACO	TS	SAi	SAs	TAi	TAs	DPSO
$\rho = 100$	0.2791	1.5248	3.5485	4.9252	3.5023	2.3994	2.1309	3.1692	4.1075	4.3926	5.2916	4.0322
$\rho = 200$	0.2683	1.4056	3.5548	4.5313	3.3031	2.2995	1.7476	3.2812	4.2206	4.5061	5.4059	3.2562
$\rho = 300$	0.2668	1.3358	3.5691	4.2794	3.2532	2.2539	1.5795	3.3397	4.2795	4.5652	5.4655	2.9193

As can be seen, our three proposed methods outperform all others in most situations, both in the idling, as well as in the no-idling case, and for all three termination criteria. More specifically, if we focus on $\rho = 300$, the proposed EDA method, which contains the local search and the speed up, is just 0.2622% away from the best known optimum solution, on average for the 5 replicates and 360 instances and in the no-idling case. In comparison, the closes competitor is the TS which is 1.2556. This is 4.79 times less deviation. For the idling case, the differences widen, as EDA is 0.2668 and the closest competitor is again TS with 1.5795, i.e., 5.92 times better.

Our hypothesis is that the effective control of the population, with clone avoidance, convergence avoidance and the finely tuned local search is the key aspect to the performance of the three proposed approaches.

4. Conclusions

As we can see from the computational results, the proposed EDA algorithm yields significantly better results than all other tested methods. On average, EDA performs several times better than the closest competitor from the literature both in the idling and no-idling cases. This work shows an interesting evolutionary method tailored for a realistic lot-streaming flowshop problem with the inclusion of sequence-dependent setup times. The finely tuned local search as well as the avoidance of clones and premature convergence in the population are key aspects to performance. As a conclusion, we can safely conclude that the proposed EDA is a new state-of-the-art algorithm for the lot-streaming flow shop scheduling problem with setup times and makespan criterion.

Acknowledgements

Rubén Ruiz is partially funded by the Spanish Ministry of Science and Innovation, under the project “SMPA - Advanced Parallel Multiobjective Sequencing: Practical and Theoretical Advances with reference DPI2008-03511/DPI. The authors should also thank the IMPIVA for the project OSC with reference IMIDIC/2009/198 and the UPV for the project “PPAR - Production Programming in Highly Constrained Environments: New Algorithms and

Computational Advances”with reference 3147. Quan-Ke Pan is partially supported by National Science Foundation of China (60874075, 70871065), and Open Research Foundation from State Key Laboratory of Digital Manufacturing Equipment and Technology (Huazhong University of Science and Technology).

References

Allahverdi, A;, Gupta, J.N.D.; Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. OMEGA, Vol 27, pp. 219-239.

Allahverdi, A.; Ng, C.T.; Cheng, T.C.E.; Kovalyov, M.Y. (2008). A survey of scheduling problems with setup times or costs. European Journal of Operational Research, Vol 187, pp. 985-1032

Allahverdi, A.; Soroush, H.M. (2008). The significance of reducing setup times/setup costs. European Journal of Operational Research, Vol 187, pp. 978-984.

Chang, J.H.; Chiu, H.N. (2005). A comprehensive review of lot streaming. International Journal of Production Research, Vol 43, pp 1515-1536.

Jarboui, B.; Eddaly, M.; Siarry, P. (2009). An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems, Computers and Operations Research, Vol 36, pp. 2638-2646.

Marimuthu, S.; Ponnambalam, S.G.; Jawahar, N. (2007). Tabu search and simulated annealing algorithms for scheduling in flow shops with lot streaming. Proceedings of the Institution of Mechanical Engineers, Part B, Journal of Engineering Manufacture, Vol 221, pp 317-331.

Marimuthu, S.; Ponnambalam, S.G.; Jawahar, N. (2008). Evolutionary algorithms for scheduling m-machine flow shop with lot streaming. Robotics and Computer-Integrated Manufacturing Vol 24, pp 125-139.

Marimuthu, S.; Ponnambalam, S.G.; Jawahar, N. (2009). Threshold accepting and ant-colony optimization algorithm for scheduling m-machine flow shop with lot streaming. Journal of Material Processing Technology, Vol 209, pp 1026-1041.

Muhlenbein, H.; Paab, G. (1996). From recombination of genes to the estimation of distribution. Binary parameters. In: Lecture notes in computer science 1411: Parallel Problem Solving from Nature, PPSN, Vol IV, pp. 178-187.

Pan, Q.-K.; Tasgetiren, M.F.; Liang, Y.C. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. Computers and Industrial Engineering, Vol 55 pp 795-816.

Reiter, S. (1996). A system for managing job-shop production. The Journal of Business, Vol 34, pp. 371-393.

Ruiz, R.; Maroto, C.; Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. Omega-International Journal of Management Science, Vol 34, pp 461-476.

Ruiz, R.; Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flow shop scheduling problem. European Journal of Operational Research, Vol 177, pp 2033-2049.

Salhi, A.; Vázquez-Rodríguez, J.A.; Zhang, Q. (2007). An estimation of distribution algorithm with guided mutation for a complex flow shop scheduling problem. GECCO'7, July 7-11, 2007, London, England, UK, pp. 570-576.

T'kindt V.; Billaut J.C. (2006). Multicriteria scheduling: theory, models and algorithms. 2nd ed., Berlin. Springer.

Truscott, W. (1986). Production scheduling with capacity constrained transportation activities. Journal of Operational Management, Vol 6, pp. 333-348.

Tseng, C.T.; Liao, C.J. (2008). A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. European Journal of Operational Research, Vol 191, pp 360-373.

Vallada, E.; Ruiz, R. (2010). Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem, Omega-International Journal of Management Science, Vol 38, pp 57-67.

Yoon S.H.; Ventura, J.A. (2002). An application of genetic algorithms to lot-streaming flow shop scheduling. IIE Transactions, Vol 34, pp. 779-787.