Proceedings of the 2004 IEEE
International Symposium on Intelligent Control
Taipei, Taiwan, September 2-4, 2004

Equal Size Lot Streaming to Job-shop Scheduling Problem Using Genetic Algorithms

Felix T. S. Chan*, T. C. Wong and P. L. Y. Chan
Department of Industrial and Manufacturing Systems Engineering,
The University of Hong Kong, Pokfulam Road, Hong Kong
*Email address: ftschan@hkucc.hku.hk

Abstract — **A novel approach to solve Equal Size Lot Streaming (ESLS) in Job-shop Scheduling Problem (JSP) using Genetic Algorithms (GAs) is proposed. LS refers to a situation that a lot can be split into a number of smaller lots (or sub-lots) so that successive operation can be overlapped. By adopting the proposed approach, the sub-lot number for different lots and the processing sequence of all sub-lots can be determined simultaneously using GAs. Applying Just-In-Time (JIT) policy, the results show that the solution can minimize both the overall penalty cost and total setup time with the development of multi-objective function. In this connection, decision makers can then assign various weightings so as to enhance the reliability of the final solution.**

1 INTRODUCTION

In conventional job-shop systems, lots are processed on workstations or machines in different orders. A feasible solution to this problem can be defined by the processing sequence of all lots on all machines such that lots can be finished completely. The total possible solution can be up to $(n!)^m$ if there are 1...m machines and 1...n lots. So it is hard enough to solve this type of problem in practical time limit. In general, it is commonly assumed that a lot cannot be split into sub-lots because of single lot size [1,2]. If LS is considered, it is a must to relax this assumption. By applying LS to JSP, lots can be finished earlier, hence, minimize the overall lot lateness as defined by the deviation beyond due dates. On the other hand, the total setup cost may increase due to the increment in split lots. To solve this problem, it is prevalent to define a multi-objective function considering the overall penalty cost and total setup cost. For practical applications, decision makers are required to assign weights to these 2 objectives. Overall penalty cost, as defined by the sum of earliness cost per hour per early unit and tardiness cost per hour per late unit, is weighted. Total setup cost is regarded as the product of processing cost per hour and the total setup time during fixture changeover between lots or sub-lots. Then the objective becomes the minimization of the sum of these 2 cost types.

2 LITERATURE REVIEW

For most applications, it is quite often to apply LS to Flow-shop Scheduling Problem (FSP) [3-5]. In traditional flow-shop, lots are processed in the same order. Therefore, it is extremely useful to split lots into sub-lots in order to expedite the production process as shown in Fig. 1. Kalir and Sarin [3] presented a new heuristic method called the Bottleneck Minimal Idleness (BMI) heuristic to equally sized LS in FSP. The principle of BMI is to minimize the idle time on the bottleneck machine such that the near-optimal schedule can be obtained. Kumar *et al.* [4] addressed LS in 3 elements, i.e. the number of sub-lots, sub-lot size, and sub-lots processing sequence. By applying i) GAs to determine the number of sub-lots and the processing sequence of sub-lots, and ii) Linear Programming (LP) to determine the sub-lot size, the results are insightful and rewarding. Yoon and Ventura [5] proposed a Hybrid Genetic Algorithm (HGA) which incorporates LP and a Pair-wise Interchange (PI) method for LS in FSP. The objective is to minimize the weighted absolute deviation of lot completion time from due date.

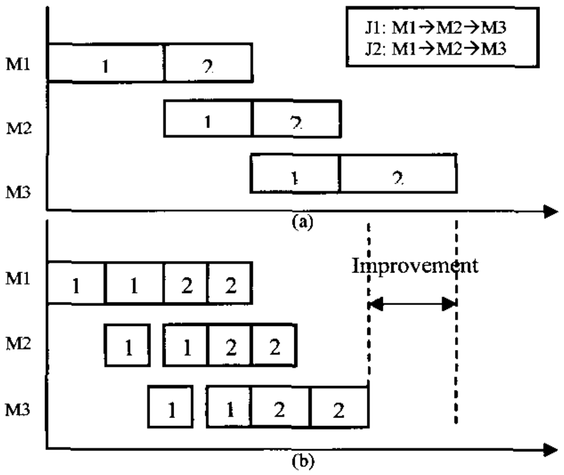

Figure 1. FSP a) without LS; b) with LS

If the objective is to minimize makespan, the optimum solution is to split lots into single unit sub-lots with no setup time. If the objective is to minimize the earliness or tardiness of lots, it is important to split lots in consideration of lot sequencing. However, the problem is less complex as compared to a simple JSP. But since the number of inherent constraints of JSP is bigger than that of FSP, the application of LS to JSP is still useful in modern manufacturing environment. Other than huge capital investment, any improvement to the scheduling issues can implicitly reduce the overall production cost by economically utilizing the fixed assets. Fig. 2 shows the potential of LS to JSP.

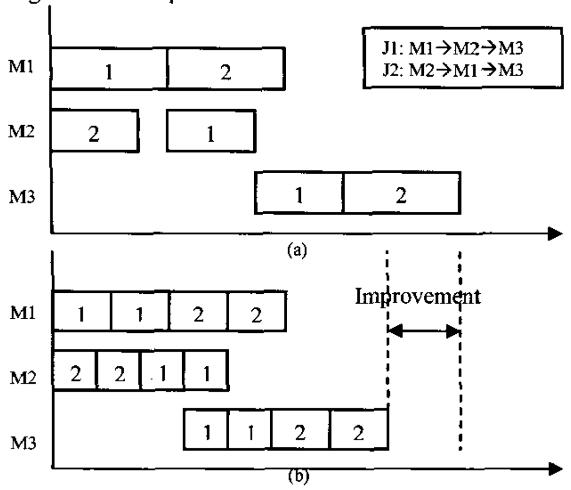

Figure 2. JSP a) without LS; b) with LS

From Fig. 2, the improvement is noted with LS. It is assumed that the processing sequence of sub-lots follows the sequence of its original lot. If the objective is to minimize makespan, the improvement is quite significant. In general, there are 4 types of LS approaches to general scheduling problems including i) Equal size sub-lots without intermittent idling means that lots are split into sub-lots with the same size and machine is required to process all sub-lots in a continuous manner; ii) Equal size sub-lots with intermittent idling allows idle time between sub-lots on the same machine; iii) Varied size sub-lots without intermittent idling means that lots are split into sub-lots of different size and no idle time is allowed between sub-lots on the same machine; and iv) Varied size sub-lots with intermittent idling permits idle time between sub-lots of varying size on the same machine. For detailed description of the above 4 LS approaches, please refer to the work done by Trietsch and Baker [6]. In this paper, the second LS type is adopted and an algorithm will be developed. Moreover, it is assumed that once LS is applied to split lots into sub-lots, the total number of sub-lots and the sub-lot size are fixed throughout the schedule.

Although few studies are dedicated to LS in JSP, some have attempted to address the benefits. Dauzère-pérès and Lasserre [7] presented an iterative procedure to determine the sub-lot size for a given sub-lot sequence and JSP with fixed sub-lot size using LP and GA. The procedure stops until the makespan converges or the maximum number of iterations is achieved. They have shown that results near to lower bound can be obtained by using the proposed procedure. However, no attention is given to the selection of split lots. Jeong *et al.* [2] studied a lot splitting heuristic for JSP in dynamic environment. Some heuristics are applied to determine the split lots and the sub-lot size and then schedule sub-lots with a modified shifting bottle procedure. Also, the proposed heuristics has been applied in different dynamic cases such as machine failure and rush orders. Particularly, the importance of the before-arrival setup time to the schedule performance is highlighted. However, they did not explicitly explain the impact of different dynamic factors on this before-arrival setup time.

Since there are limited applications on LS to JSP, an innovative approach using GA will be studied. This paper will be organized as follows. The proposed approach will be elucidated in the next section. In section 4, computational results will be investigated to examine the performance of the proposed method. An industrial case study will be studied in section 5. Finally, conclusions will be drawn together with future research direction.

3 THE PROPOSED ALGORITHM

3.1 Model Notations

W1	Weightings on overall penalty cost
W2	Weightings on total setup cost
m	Total number of machines
n	Total number of original lots
n'	Total number of sub-lots
J_i	Lot i
J_{ij}	jth lot of J_i
S_i	Number of sub-lots of J_i
F_i	Fixture of J_i
L_i	Original lot size of J_i
Q_{ij}	Quantity of J_{ij}
MS_{ik}	kth machine for J_i
Pt_{ik}	Processing time on kth machine of J_i
St_{ijk}	Start time of J_{ij} on machine k
C_{ij}	Completion date of J_{ij}
D_i	Due date of J_i
su_k	Total setup time on machine k
ec_i	Earliness cost / hour / unit of early J_i
tc_i	Tardiness cost / hour / unit of late J_i
mc_k	Machining cost of machine k per hour

3.2 Model Formulation

Min. (W1 x Overall penalty cost + W2 x Total setup cost)

$$= W1 \times \left(\sum_i \sum_j (\alpha_{ij} \times ec_i + \beta_{ij} \times tc_i) \times Q_{ij} \right) + W2 \times \left(\sum_k su_k \times mc_k \right) \quad (1)$$

where
If $C_{ij} < D_i$, $\alpha_{ij} = D_i - C_{ij}$ and $\beta_{ij} = 0$.
If $C_{ij} > D_i$, $\beta_{ij} = C_{ij} - D_i$ and $\alpha_{ij} = 0$.
If $C_{ij} = D_i$, $\alpha_{ij} = \beta_{ij} = 0$.

$$\sum_j Q_{ij} - L_i = 0 \quad \forall i \tag{2}$$

$$St_{ijMS_{i(k+1)}} \geq Pt_{ik} \times Q_{ij} + St_{ijMS_{ik}} \quad \forall i,j \tag{3}$$

$$Q_{ij} \geq 0 \tag{4}$$

$$St_{ijk} \geq 0 \tag{5}$$

$$1 \leq i \leq n \tag{6}$$

$$1 \leq j \leq S_i \tag{7}$$

$$1 \leq k \leq m \tag{8}$$

The objective function is illustrated by equation (1). Overall penalty cost is the sum of earliness and tardiness costs. Earliness (tardiness) cost is defined as the sum of the total early (late) hour by each unit of all lots/sub-lots times ec_j (tc_j) where j = 1...n'. Total setup cost is regarded as the sum of individual setup time on each machine (su_k) times its associated cost (mc_k) where k = 1...m. Constraint (2) requires the sum of sub-lot size should satisfy the original lot size. Limitation (3) ensures that the processing sequence of sub-lots corresponds to the predetermined order. Constraints (4) and (5) set the non-negativity conditions for all sub-lot sizes and start time of sub-lots. Equations (6)-(8) specify the range of variables i, j, and k.

3.3 Genetic Algorithms

GAs is a kind of stochastic optimization methods as proposed by Holland [8]. The principle of GAs is based on the natural evolution and has been applied to a wide range of combinatorial problems. In terms of GAs, solutions are encoded in a set of strings (or chromosomes) to form a solution pool called *population.* Then strings are evaluated based on the objective function to obtain the *fitness value.* Then, strings with higher fitness value will be combined to produce new strings forming a new pool, this process is called *crossover.* For extensive review on genetic crossover operators, please refer to Cheng *et al.* [9]. These new strings then may be subjected to self-tuning called *mutation* with a probability called *mutation rate (mrate).* Each population then represents a *generation,* thus the procedure will continue to run until the terminating criteria are met such as the maximum number of generation is reached. Finally, the best solution is obtained at the end of the procedure. Recall that the size of solution pool refers to population size (*psize*) and the number of pools defines the maximum number of generations (*gen*).

3.4 Lot Streaming Problem

In this paper, a GA is applied to determine the sub-lot number for all lots called *GA1.* A solution in GA1 is defined as a string of size n. Each bit of the string represents the number of sub-lots for the corresponding lot. For example, a string {1 2 2 5 5} means that lots 2 and 3 are split into 2 sub-lots; and lots 4 and 5 are split into 5 sub-lots while lot 1 remains unsplit. Because integer-sized lots are considered, it is assumed that $S_i \leq L_i$. According to the second LS type, $Q_{ij} = L_i/S_i$. In some cases, the value of L_i/S_i is not an integer, so the last sub-lot of J_i equals to $L_i - \sum Q_{ij}$ for j = 1...S_i-1. For example, if $L_i = 10$ and $S_i = 4$, then $Q_{ij} = 10/4 = 2.5 \approx 2$ (no rounding). Then $Q_{i1} = Q_{i2} = Q_{i3} = 2$, $Q_{i4} = 10-(2+2+2) = 4$. After splitting n lots to n' sub-lots according to the string, the next step is to solve a JSP with n' independent lots. The value of n' can be obtained by equation (9). The scheduling results will then become the objective value (OV) of the string by equation (1). And the OV will be transformed to the fitness value by equation (10). The procedure of GA1 will continue to run until terminating criteria are met. Then good strings will be used to perform crossover operation according to *ranked fitness list,* i.e. only solutions rank top can perform crossover. For GA1, a simple 2-cut-point crossover (2X) operator is implemented (Fig. 3a) and mutation operation (Fig. 3b).

$$n' = \sum_i S_i \tag{9}$$

$$Fitness\ Value(FV) = \frac{MAX - OV + MIN}{AVERAGE} \tag{10}$$

MAX→the maximum OV of the same generation
MIN→the minimum OV of the same generation
AVERAGE→the average OV of the same generation

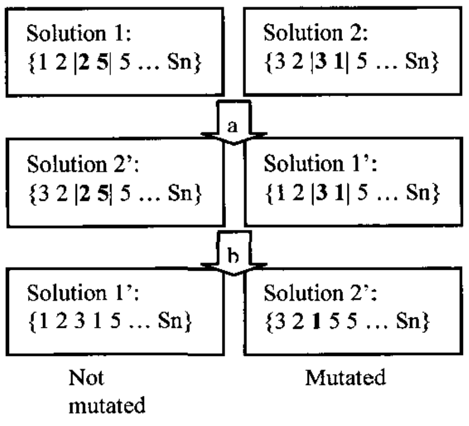

Figure 3. a) 2X operator; b) Mutation operation

3.5 *Job-shop Scheduling Problem*

After LS, a new JSP is formed after splitting n lots into n' sub-lots. To this end, another GA is applied to solve the new JSP called *GA2*. A string for GA2 is defined as the preference list of lot priorities on each machine. For example, if $m = n = 2$, $L_1 = 5$, $L_2 = 8$, $S_1 = S_2 = 2$, then we have $n' = 4$ sub-lots with $Q_{11} = 2$, $Q_{12} = 3$, $Q_{21} = Q_{22} = 4$. Hence, a string can be defined as $\{J_{11}\ J_{12}\ J_{21}\ J_{22} \mid J_{21}\ J_{11}\ J_{22}\ J_{12}\}$. It means that the preference list on machine 1 is $J_{11}>J_{12}>J_{21}>J_{22}$, similarly $J_{21}>J_{11}>J_{22}>J_{12}$ for machine 2. Then Non-Delay (ND) schedule will be generated. ND means that no machine is allowed to idle when there is at least one lot or sub-lot waiting for that machine [10]. This policy is of extremely useful to workshops with only local buffers available on machines because frequent rescheduling is not always applicable. Hence, the preference list of lot processing order is proposed particularly to this type of working condition.

Similar to GA1, the fitness values of strings are obtained by equation (10) and strings of GA2 will perform crossover operation according to *roulette wheel selection scheme*. Job-based Order Crossover (JOX) which has been proven to preserve job order on all machines between different generations well [11], is applied to GA2. The working mechanism of JOX is depicted in Fig. 4. Mutation operation is defined as the interchange of position between 2 lots or sub-lots on the same machine. Tab. 1 shows the comparison between a well-known method Shifting Bottleneck (SB) [12] and GA2 to some standard JSP.

Step 1: A number of random lots or sub-lots is preserved, say J13 and J21.
Solution 1: {... J12 J22 **J13 J21** J11 ...}
Solution 2: {... J22 J11 **J21** J12 **J13** ...}

Step 2: JOX is applied by fixing the position of preserved lots or sub-lots.
Solution 1: {... x x **J13 J21** x ...}
Solution 2: {... x x **J21** x **J13** ...}

Step 3: Then inherit the lot processing order of solution 2 to solution 1 and solution 1 to solution 2.
Solution 1: {... J22 J11 **J13 J21** J12 ...}
Solution 2: {... J12 J22 **J21** J11 **J13** ...}

Figure 4. JOX operator

Table 1. Comparisons between SB and GA2

JSP	OPT	SB	GA2
FT5x20	1165	1178	1178
LA01	666	666	666
LA06	926	926	926
LA16	945	978	998
LA21	1046	1084	1125
LA31	1784	1784	1784

4 COMPUTATIONAL RESULTS

To examine the performance of the proposed algorithm, various JSP will be employed. For each problem, Pt_{ik} from [1~10], L_i ranges from [1~30], F_i from [1~20], ec_i from [1~10], tc_i from [1~10], and mc_k from [1~20]. Precedence processing constraints are maintained such that lots should visit each machine once in a predetermined order. Also, fixed setup time is considered during fixture changeover between lots or sub-lots. Let $m = n = 3$, different weightings will be employed to generate the solution to each of 10 3x3 problems using objective function (1). Average results are shown in Tab. 2. It is noted that the primary goal is the minimization of the overall cost, i.e. the sum of overall penalty cost and total setup cost.

Table 2. Overall cost with different weightings

10	W1/W2				
Problems	1.0/0	0.9/0.1	0.7/0.3	0.5/0.5	0/1.0
Average	7886 [657]	7266.7 [638]	8389.4 [739]	8006 [662]	16944.8 [568]

[ ] – Total setup cost

From this experiment, it is observed the lowest overall cost cannot be obtained by considering either the overall penalty cost (i.e. W1/W2 = 1.0/0) or total setup cost (i.e. W1/W2 = 0/1.0). On average, it seems that W1/W2 = 0.9/0.1 gives the best result on the overall cost in this example. In fact, the ratio W1/W2 should be case-dependent and there should not be any single best ratio to all problems. However, the results show that there is at least one balance point captured between these 2 objectives using the proposed method. And it is shown that the proposed algorithm works quite well to the weightings. From Fig. 5, the convergence of GA1 is examined. GA1 (gen, psize, mrate) = (10, 20, 0.01) is implemented. It is observed that the convergence is quite well with respect to the objective (1) over 200 GA1 solutions. Recall that each GA1 solution defines the sub-lot number for all lots. For GA2, (gen, psize, mrate) = (20, 20, 0.1) is employed. Fig. 6 shows the convergence of GA2 for one particular GA1 solution. Recall that the objective function of GA2 is the minimization of overall penalty cost only and each GA2 solution defines the processing sequence of all lots or sub-lots. From Fig. 6, it is seen that the fluctuation is minimized across generations. To this end, it is observed that the convergence of GA2 is quite well with respect to overall penalty cost. It is noted that the proposed approach explores 200 GA1 solutions. Each GA1 solution is evaluated by 400 different processing sequences. So there are total 80000 individual solutions to every single ESLS to JSP. However, the configuration of GA parameters is also case-dependent. Probably, more emphasis must be placed on the determination of suitable GA configuration to various ESLS to JSP so as to minimize the subjective errors due to human judgment.

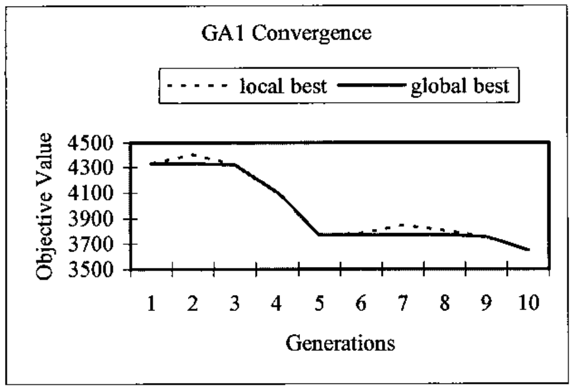

Figure 5. GA1 Convergence

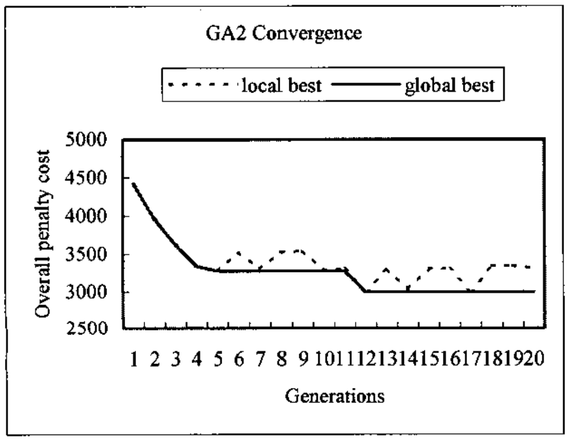

Figure 6. GA2 Convergence

5 CONCLUSION

In this paper, a new approach using GAs is developed to ESLS in JSP. The primary goal of the proposed GA approach is to determine the split lot, the sub-lot number for each lot, and the lot processing sequence simultaneously so as to minimize the value of the multi-objective function. The assignment of weightings also facilitates the decision makers to control the outcomes to different problems in dynamic working environment. Moreover, the proposed GA approach can be easily applied to other applications such as demand splitting in supply chains, allocation and distribution problems in system design, general lot streaming problems in different areas, etc. However, the study of GA configuration is surely encouraged to further strengthen the reliability of the GA mechanism.

6 REFERENCE

[1] Woo S., Kang S. and Park J., A Batch Splitting Heuristic for Dynamic Job Shop Scheduling Problem, Comput. Ind. Eng., vol. 33, 1997, pp 781-784.
[2] Jeong H. I., Park J. and Leachman R. C., A Batch Splitting Method for a Job Shop Scheduling Problem in an MRP Environment, Int. J. Prod. Res., vol. 37, 1999, pp 3583-3598.
[3] Kalir A. A. and Sarin S. C., A Near-optimal Heuristic for the Sequencing Problem in Multiple-batch Flow-shops with Small Equal Sublots, Int. J. Manage. Sci., vol. 29, 2001, pp 577-584.
[4] Kumar S., Bagchi T. P. and Sriskandarajah C., Lot Streaming and Scheduling Heuristics for m-machine No-wait Flowshops, Computers Ind. Eng., vol. 38, 2000, pp 149-172.
[5] Yoon S. H. and Ventura J. A., An Application of Genetic Algorithms to Lot-streaming Flow Shop Scheduling, IIE Trans., vol. 34, 2002, pp 779-787.
[6] Trietsch D. and Baker K. R., Basic Techniques for Lot Streaming, Ops. Res., vol. 6, 1993, pp 1065-1076.
[7] Dauzère-pérès S. and Lasserre J. B., Lot Streaming in Job-shop Scheduling. Ops. Res., vol. 45, 1997, pp 584-595.
[8] Holland J. H., Adaption in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor; 1975.
[9] Cheng R., Gen M. and Tsujimura Y., A Tutorial Survey of Job-shop Scheduling Problems Using Genetic Algorithms, Part II: Hybrid Genetic Search Strategies. Computers Ind. Eng., vol. 36, 1999, pp 343-364.
[10] Croce F. D., Tadei R. and Volta G., A Genetic Algorithm for the Job Shop Problem. Computers Ops. Res., vol. 22, 1995, pp 15-24.
[11] Ono I., Yamamura M. and Kobayashi S., "A Genetic Algorithm for Job-shop Scheduling Problems Using Job-based Order Crossover", in International Conference on Evolutionary Computation, Nayoya, Japan, 1996, pp 547-552.
[12] Adams J., Balas E. and Zawack D., The Shifting Bottleneck Procedure for Job Shop Scheduling. Manage. Sci., vol. 34, 1988, pp 391-401.