Jobshop lot streaming with routing flexibility, sequence-dependent setups, machine release dates and lag time

Published in 2012 in the International Journal of Production Research Vol. 50, 2331 -2352

Please cite this article as:

Defersha, F. M., and Chen, M., (2012). Job shop lot streaming with routing flexibility, sequence-dependent setups, machine release dates and lag time. International Journal of Production Research, Vol. 50, 2331-2352.

The online version can be found at the following link:

http://dx.doi.org/10.1080/00207543.2011.574952

Job-Shop Lot Streaming with Routing Flexibility, Sequence Dependent Setups, Machine Release Dates and Lag Time

Fantahun M. Defersha and Mingyuan Chen*

Department of Mechanical and Industrial Engineering, Concordia University
1455 de Maisonneuve W., Montreal, Quebec, Canada, H3G 1M8

Abstract

Lot streaming is a technique of splitting production lots into smaller sublots in a multi-stage manufacturing systems so that operations of a given lot can be overlapped. This technique can reduce manufacturing makespan and is one effective tool for time-based manufacturing strategy. Research on lot streaming models and solution procedures in flexible job-shops has been limited. Flexible job-shop scheduling problem is an extension of the classical job-shop scheduling problem by allowing an operation to be assigned to one of a set of eligible machines during scheduling. In this paper we develop a lot streaming model for a flexible job-shop environment. The model considers several pragmatic issues such as sequence dependent setup time, attached or detached nature of setups, machine release date and lag time. In order to efficiently solve the developed model, an island-model parallel genetic algorithm is proposed. Numerical examples are presented to demonstrate the features of the proposed model and compare computation performance of the parallel genetic algorithm over the sequential one. The results are very encouraging.

Keywords: *Lot Streaming, Flexible Job-shop Scheduling; Sequence Dependent Setup; Machine Release Date; Lag Time; Parallel Genetic Algorithm*

1. Introduction

Lot streaming (LS) is a technique in which a production lot is split into smaller sublots such that each sublot is treated individually and transferred to the next processing stage upon its completion [3, 18]. Different sublots of the same job can thus be processed simultaneously at different stages. As a result of such operation overlapping, production can be significantly accelerated. This technique has been used to implement the time-based strategy in todays global competition [15]. Many world-class manufacturing companies (e.g., Dell and Toyota) have adopted this strategy to quickly produce and deliver goods to their customers [6, 7]. The concept was first formally introduced in literature by Reiter [48] in 1966. Since then a considerable number of articles have been published on LS where the researcher mainly focus on flowshop [see for example 46, 4, 28, 3, 55, 36, 31, 37, 19, 8, 41, 5, 54, 24, 40]. Research on job-shop scheduling problems (JSP) with lot streaming, however, is quite limited. Jacobs and Bragg [32] studied a LS problem in job shops and considered the before-arrival (detached) type of setups. Smunt *et al.* [53] considered LS problems in stochastic flowshops and job-shops. They used simulation and showed that lot splitting substantially improves mean and standard flow time over no

*For correspondence: mychen@encs.concordia.ca, Tel: (514) 848-2424 Ext. 3134; Fax: (514) 848-3175

lot splitting. Dauzere-Peres and Lasserre [21, 22] proposed an iterative procedure for LS in a job-shop which solves a linear programming sub-problem to determine sublot-sizes for a fixed sequence and searches for different sequences using a heuristic algorism. Jeong *et al.* [33] developed a lot splitting heuristic for JSP in a dynamic environment. Some heuristics are applied to determine the split lots and the sublot sizes and then schedule sublots with a modified shifting bottleneck procedure. Low *et al.* [38] demonstrated the benefits of lot splitting in job-shops. A disjunctive graph was first used to describe the addressed scheduling problem, and an integer programming model was then constructed to obtain an optimal solution for small size problems. Chan *et al.* [11] proposed a genetic algorithm to determine the number of equal sized sublots for each lot and the processing sequence of these sublots in a job-shop environment. In their subsequent works, the authors extended their JSP-LS studies by considering unequal sized sublots in Chan *et al.* [12, 13] and assembly operations in Chan *et al.* [14]. Buscher and Shen [9] proposed a three phase algorithm for JSP-LS. The algorithm incorporates the predetermination of sublot sizes, the determination of schedules based on Tabu search and the variation of sublot sizes. Edis and Ornek [23] applied simulation to study LS problems in a stochastic job shop with equal and discrete sublots. The work presented in our this paper extends the job-shop lot streaming literature reviewed above by considering (1) routing flexibility, (2) sequence dependent setup time, (3) attached/detached setups, (4) machine release dates, and (5) lag time.

Routing Flexibility: The presence of alternative routings is typical in many discrete multi-batch production environments. Routing flexibility increases the number of ways in which one can assign operations to machines in order to come up with a better schedule. Flexible job-shop scheduling problem (FJSP) is an extension of the classical JSP by considering routing flexibility during scheduling to achieve a better schedule. Recent studies in FJSP without LS can be found in Chen *et al.* [17], Saidi and Fattahi [52], Gao *et al.* [25], Pezzella *et al.* [45], Xing *et al.* [56], and Gao *et al.* [26]. In this paper, we propose a model and solution procedure for FJSP with lot streaming (FJSP-LS).

Sequence Dependent Setup: In many real-life situations, a setup operation is often required between operations and it strongly depends on the immediate preceding operation on the same machine [20, 2]. Panwalkar *et al.* [44] noticed that significant portion of jobs required sequence dependent setups in job scheduling. Flow shop problems with sequence-dependent setups are extensively covered in the current literature [see for example 30, 35, 43, 47, 49, 51]. However, as pointed in Manikas and Chang [39], research on job-shops scheduling with sequence-dependent setups has been limited. In this paper, sequence dependent setup time is considered in FJSP-LS. As lot streaming increases the number of setup incidences, the optimization of the sequence to minimize the setup time becomes more important.

Attached/Detached Nature of Setup: In addition to sequence dependence, an important feature of setups is their state of being attached or detached. A setup of a particular operation is attached (non-anticipatory) if it is assumed that setting up the machine for this operation can be performed when the job arrives at the machine. When a setup is performed prior to the arrival of the job, the setup is called detached (or anticipatory). In this case the setup time can overlap with the processing time of the preceding operation if these two consecutive operations are not assigned to the same machine. In most papers in the literature, authors assumed attached setup. In the proposed model, setup of

each operation can be treated as either attached or detached depending on the actual manufacturing requirements.

Machine Release Date: The proposed model also considers machine release date. It is the time at which a machine will complete processing products from previous schedule and be available for processing products of the current schedule. This is a common situation in industry as production environments are seldom found empty and one may have to consider ongoing operations from previous schedule [50]. In this research we noticed that when routing flexibility is considered in scheduling, machine release date becomes very important as the selection of an alternative machine can be affected by its release date. This is because, the machine to be released soon may represent a better choice than the one to be released in a latter time.

Lag Time: It is a requirement for delaying the starting time of an operation from the completion time of the previous operation of a sublot. Such time lag may occur when, for example, drying or cooling of products are performed before further operations can take place. In the proposed model, lag time has been incorporated.

Solving classical JSP is known to be NP-hard [27]. The introduction of sequence dependent setup time, routing flexibility and lot splitting complicates the already difficult classical JSP. In order to efficiently solve the FJSP-LS mathematical model proposed in this paper, we developed a parallel genetic algorithm (PGA) that runs on a high performance parallel computing platform. The algorithm is based on the island model parallelization technique of a genetic algorithm (GA). From early days of its development, the GA's potential for parallelization has been noted. Several authors have applied parallel genetic algorithms in diverse domains while research on using PGAs for JSP has not been seen. This paper contributes to the literature by providing a comprehensive model for FJSP-LS and reporting the use of parallel computing in solving this type of difficult problems. The rest of this paper is organized as follows. In Section 2, we present the MILP model for FJSP-LS. The PGA is detailed in Section 3. Numerical examples are give in Section 4. Discussion and conclusions are presented in Section 5.

2. Mathematical Formulation

In this section we present a mixed integer linear programming model for FJSP-LS. The model formalizes the problem studied and can be used to solve small size problems using branch and bound algorithm. Solutions from such small size problems can be used to validate correctness of the developed heuristic solution method.

2.1. Problem Description and Notations

Consider a job-shop consisting of M machines where certain machines are same or have some common functionalities. The system is processing a set of jobs from previous schedules and each machine m has a release date D_m at which time it will be available for current schedule. Consider also a set of J independent jobs to be currently scheduled in the system. The batch size of job j is given by B_j and this

batch is to be split into S_j number of unequal consistent sublots (transfer batches). A decision variable $b_{s,j}$ is used to denote the size of sublot s of job j. Each sublot of job j is to undergo O_j number of operations in a fixed sequence where each operation can be processed by one of several eligible machines. $T_{o,j,m}$ is unit processing time for an operation o of a sublot of job j on machine m. An operation o of a sublot of job j can be started on machine m after lag time $L_{o,j}$ and after the setup is performed. The setup time for an operation o of job type j on machine m depends on the preceding operations and is denoted by $S_{o,j,m,o',j'}$, where operation o' of a sublot of job j' is the preceding operation on machine m. If operation o of sublot s of job j is the first operation to be processed on machine m, the setup time is simply represented as $S^*_{o,j,m}$. The setup time $S_{o,j,m,o',j'}$ (or $S^*_{o,j,m}$) for operation o of a sublot of job j can be overlapped with the processing time of operation $o-1$ of the same sublot if it is a detached setup and machine m is available for setup. The problem is to determine the size of each sublot, to assign the operation of each sublot to one of the eligible machines and to determine the sequence and starting time of the assigned operations on each machine. The objective is to minimize the makespan of the schedule. We next introduce some additional notations and then present a mixed integer linear programming (MILP) formulation for FJSP-LS.

Additional Parameters:

R_m Maximum number of production runs of machine m where production runs are indexed by r or $u = 1, 2, ..., R_m$; Each of these production runs can be assigned to at most one sublot. Thus the assignment of the operations to production runs of a given machine determines the sequence of the sublots on that machine;

$P_{o,j,m}$ A binary data equal to 1 if operation o of a sublot job j can be processed on machine m, 0 otherwise;

$A_{o,j}$ A binary data equal to 1 if setup of operation o of a sublot of job j is attached (non-anticipatory), or 0 if this setup is detached (anticipatory);

Ω Large positive number.

Variables:

Continuous Variables:

c_{max} Makespan of the schedule

$c_{o,s,j,m}$ Completion time of operation o of sublot s of job j on machine m;

$\widehat{c}_{r,m}$ Completion time of the r^{th} run of machine m;

$b_{s,j}$ Size of sublot s of job j

Binary Integer Variables:

$x_{r,m,o,s,j}$ A binary variable which takes the value 1 if the r^{th} run on machine m is for operation o of sublot s of job j, 0 otherwise;

$y_{r,m,o,j}$ A binary variable which takes the value 1 if the r^{th} run on machine m is for operation o of any one of the sublots of job j, 0 otherwise;

$\gamma_{s,j}$ A binary variable that takes the value 1 if sublot s of job j is non-zero ($b_{s,j} \geq 1$), 0 otherwise,

$z_{r,m}$ A binary variable that takes the value 1 if the r^{th} potential run of machine m has been assigned to an operation, 0 otherwise;

2.2. MILP Model for FJSP-LS

Following the problem description and using the notations given above, the MILP mathematical model for the FJSP-LS is presented below.

Minimize:

$$Objective = c_{max} \tag{1}$$

Subject to:

$$c_{max} \geq c_{o,s,j,m} \ ; \quad \forall(o,s,j,m) \tag{2}$$

$$\widehat{c}_{r,m} \geq c_{o,s,j,m} + \Omega \cdot x_{r,m,o,s,j} - \Omega \ ; \quad \forall(r,m,o,s,j) \tag{3}$$

$$\widehat{c}_{r,m} \leq c_{o,s,j,m} - \Omega \cdot x_{r,m,o,s,j} + \Omega \ ; \quad \forall(r,m,o,s,j) \tag{4}$$

$$\widehat{c}_{1,m} - b_{s,j} \cdot T_{o,j,m} - S^{*}_{o,j,m} - \Omega \cdot x_{1,m,o,s,j} + \Omega \geq D_m \ ; \quad \forall(m,o,s,j) \tag{5}$$

$$\widehat{c}_{r,m} - b_{s,j} \cdot T_{o,j,m} - S_{o,j,m,o',j'} - \Omega \cdot (y_{r-1,m,o',j'} + x_{r,m,o,s,j}) + 2\Omega \geq \widehat{c}_{r-1,m} \ ;$$
$$\forall(r,m,o,s,j,o',j') | (r > 1) \tag{6}$$

$$\widehat{c}_{1,m} - b_{s,j} \cdot T_{o,j,m} - S^{*}_{o,j,m} \cdot A_{o,j} - \Omega \cdot (x_{1,m,o,s,j} + x_{r',m',o-1,s,j}) + 2\Omega \geq \widehat{c}_{r',m'} + L_{o,j} \ ;$$
$$\forall(m,r',m',o,s,j) | \{ \big((1,m) \neq (r',m')\big) \wedge (o > 1) \} \tag{7}$$

$$\widehat{c}_{r,m} - b_{s,j} \cdot T_{o,j,m} - S_{o,j,m,o',j'} \cdot A_{o,j} - \Omega \cdot (y_{r-1,m,o',j'} + x_{r,m,o,s,j} + x_{r',m',o-1,s,j}) + 3\Omega \geq \widehat{c}_{r',m'} + L_{o,j} \ ;$$
$$\forall(r,m,r',m',o,s,j,o',j') | \{ (r > 1) \wedge (o > 1) \wedge (r,m) \neq (r',m') \wedge (o,j) \neq (o',j') \} \tag{8}$$

$$y_{r,m,o,j} \leq P_{o,j,m} \ ; \quad \forall(r,m,o,j) \tag{9}$$

$$y_{r,m,o,j} = \sum_{s=1}^{S_j} x_{r,m,o,s,j} \ ; \quad \forall(r,m,o,j) \tag{10}$$

$$\sum_{m=1}^{M} \sum_{r=1}^{R_m} x_{r,m,o,s,j} = \gamma_{s,j} \ ; \quad \forall(o,s,j) \tag{11}$$

$$b_{s,j} \leq B_j \cdot \gamma_{s,j} \ ; \quad \forall(s,j) \tag{12}$$

$$\gamma_{s,j} \leq b_{s,j} \ ; \quad \forall(s,j) \tag{13}$$

$$\sum_{s=1}^{S_j} b_{s,j} = B_j \ ; \quad \forall(j) \tag{14}$$

$$\sum_{j=1}^{J}\sum_{s=1}^{S_j}\sum_{o=1}^{O_j} x_{r,m,o,s,j} = z_{r,m} \ ; \quad \forall(r,m) \tag{15}$$

$$z_{r+1,m} \leq z_{r,m} \ ; \quad \forall(r,m) \tag{16}$$

$$x_{r',m,o',s,j} \leq 1 - x_{r,m,o,s,j} \ ; \quad \forall(r,r',m,o,o',s,j)|\{(o' > o) \wedge (r' < r)\} \tag{17}$$

$$x_{r',m,o',s,j} \leq 1 - x_{r,m,o,s,j} \ ; \quad \forall(r,r',m,o,o',s,j)|\{(o' < o) \wedge (r' > r)\} \tag{18}$$

$$x_{r,m,o,s,j},\ y_{r,m,o,j},\ \gamma_{s,j} \text{ and } z_{r,m} \text{ are binary} \tag{19}$$

The objective function in Eq. (1) is to minimize the makespan of the schedule. Constraint in Eq. (2), along with the objective function, determines the makespan. The constraints in Eqs. (3) and (4) together state that the completion time of the o^{th} operation of sublot s of job j is equal to the completion time of the r^{th} run of machine m if this production run is assigned to that particular operation. The staring time of the setup for the first run ($r = 1$) of machine m is given by $\widehat{c}_{1,m} - b_{s,j} \cdot T_{o,j,m} - S*_{o,j,m}$ if the o^{th} operation of sublot s of job j is assigned to this first run. This starting time cannot be less than the release date of machine D_m as enforced by the constraint in Eq. (5). The constraint in Eq. (6) is to enforce the requirement that the setup of any production run $r > 1$ of a given machine cannot be started before the completion time of run $r - 1$ of that machine. The constraint in Eq. (7) states that for any pair of machines (m, m'), the setup (if $A_{o,j} = 1$) or the actual processing (if $A_{o,j} = 0$) of the first run on machine m cannot be started before the completion time of run r' of machine m' plus lag time $L_{o,j}$. This constraint is applied if first run of machine m is assigned to operation o of sublot s of job j and run r' of machine m' is assigned to operation $o-1$ of this same sublot. The constraint in Eq. (8) is similar to that in Eq. (7) except that Eq. (8) is for run $r > 1$ of machine m. In this case, the sequence dependent setup time has to be considered by taking into account the operation that was processed in run $r - 1$ of machine m. The constraint in Eq. (9) states that a production run r of machine m can be assigned to operation o of any one of sublots of job j if this operation can be performed on this machine. Constraint 10 depicts the logical relation between the binary variable $y_{r,m,o,j}$ and $x_{r,m,o,s,j}$. If the size of sublot s of job j is positive ($\gamma_{s,j} = 1$), an operation o of this sublot must be assigned to exactly one production run of one machine (Eq. 11). However, if the size of this sublot is zero, it should not be assigned to any production run. The constraint in Eq. (12) forces the binary variable $\gamma_{s,j}$ to take the value 1 if the sublot size $b_{s,j}$ is greater than zero. If the sublot size $b_{s,j} = 0$, the binary variable $\gamma_{s,j}$ is forced to take the value 0 by the constraint in Eq. (13). The constraint in Eq. (14) states that the sum of the sizes of the sublots of job j equals the batch size of this job. Each production run of a given machine can be assigned to at most one operation (Eq. 15), and production run $r + 1$ can be assigned to an operation if and only if run r of that machine is already assigned (Eq. 16). The constraints given in Eqs. (17) and (18) are used speed up the branch and bound procedure in solving small size problems. These constraint sets are not required to model the problem as the relations have been imposed by the constraints in Eqs. (7) and (8). The constraint in Eq. (17) accounts for the fact that if an operation

o of sublot s job j is assigned to a production run r of machine m, any upcoming operation o' of this sublot cannot be assigned to any earlier run r' of machine m. The constraint in Eq. (18) is a mirror image of constraint Eq. (17). It states that if an operation of a sublot of a given job is assigned to a production run of a machine, any earlier operation of that sublot cannot be assigned to any upcoming production run of that machine. Integral requirements on the variable $x_{r,m,o,s,j}$, $y_{r,m,o,j}$, $\gamma_{s,j}$ and $z_{r,m}$ are given Eq. (19).

3. Genetic Algorithm

Solving the classical JSP is known to be NP-hard [27] and so is solving the FJSP-LS model presented in the previous section. In order to efficiently solve this model, we developed an island-model Parallel Genetic Algorithm (PGA) that runs on a high performance parallel computing platform. The various elements of this algorithm are presented in the following subsections.

3.1. Solution Representation

A Genetic algorithm processes population of individuals, each representing a solution of the problem to be solved. In solving FJSP using genetic algorithm, Chen *et al.* [16], Kacem [34], Pezzella *et al.* [45], Gao *et al.* [26] used solution representations encoding both assignment and sequencing of operations on the various machines. Similar representations can be used in solving the proposed FJSP-LS if each sublot is considered as a job and the representations are augmented to encode the size and the number of sublots of each job. In order to illustrate such a solution representation, let us consider a small example problem processing three jobs in a four-machine flexible job shop. The number of operations, maximum number of sublots for each job, and the set of eligible machines for each operation are given in Table 1. By considering each sublot as a job and using the technique proposed in Kacem [34], a typical feasible operation to machine assignment and sequencing is encoded in a chromosome as shown in Figure 1. In this chromosome, each gene is represented by a quadruple (j, s, o, m) denoting the assignment of the o^{th} operation of sublot s of job j to machine m. The sequence of the genes in the chromosome represents the sequences of the operations in the machines. For example, by reading the chromosome from left to right, the assignment and sequencing of operations on machine-1 can be decoded as $(j1, s3, o1) \rightarrow (j3, s2, o3) \rightarrow (j3, s3, o3)$. This information is obtained from the genes at locations 10, 22 and 23 on the chromosome where $m = 1$. The assignment of operations to the other machines and their sequences is given in Table 2 as decoded from the chromosome. In this chromosome, for a given j and s, the gene (j, s, o, m) always lies to the right of all the other genes (j, s, o', m') having $o' < o$.

Table 1: An example small flexible job-shop problem

Job	No. of Operations	Max No. of Sublots	Set of eligible machines for operation		
			$o1$	$o2$	$o3$
$j1$	3	3	$\{m1, m2\}$	$\{m3\}$	$\{m2, m4\}$
$j2$	2	2	$\{m3, m4\}$	$\{m2\}$	
$j3$	3	3	$\{m3\}$	$\{m2, m4\}$	$\{m1, m3\}$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
3, 2, 1, 3	**1, 2, 1, 2**	**2, 1, 1, 3**	**3, 1, 1, 3**	**2, 2, 1, 4**	**3, 2, 2, 4**	**3, 3, 1, 3**	**1, 3, 1, 1**	**3, 1, 2, 2**	**1, 1, 1, 2**	**3, 3, 2, 2**	**1, 2, 2, 3**	**1, 1, 2, 3**	**3, 1, 3, 3**	**1, 3, 2, 3**	**2, 1, 2, 2**	**1, 1, 3, 4**	**2, 2, 2, 2**	**1, 3, 3, 4**	**3, 2, 3, 1**	**3, 3, 3, 1**	**1, 2, 3, 2**	***j, s, o, m***

j **= job index,** ***s*** **= sublot index,** ***o*** **= operations index,** ***m*** **= machine index**

Figure 1: Representation of the assignment of operations to machines and their sequencing

Table 2: Operation assignment and sequencing decoded from Figure 1

	Operation assigned to production run							
Machine	$r1$	$r2$	$r3$	$r4$	$r5$	$r6$	$r7$	$r8$
$m1$	$(j1, s3, o1)$	$(j3, s2, o3)$	$(j3, s3, o3)$					
$m2$	$(j1, s2, o1)$	$(j3, s1, o2)$	$(j1, s1, o1)$	$(j3, s3, o2)$	$(j2, s1, o2)$	$(j2, s2, o2)$	$(j1, s2, o3)$	
$m3$	$(j3, s2, o1)$	$(j2, s1, o1)$	$(j3, s1, o1)$	$(j3, s3, o1)$	$(j1, s2, o2)$	$(j1, s1, o2)$	$(j3, s1, o3)$	$(j1, s3, o2)$
$m4$	$(j2, s2, o1)$	$(j3, s2, o2)$	$(j1, s1, o3)$	$(j1, s3, o3)$				

The chromosome in Figure 1 encodes only the assignment and sequencing of the operations of the sublots. For solving the proposed FJSP-LS model, it is augmented to encode the number of sublots of each job and their sizes. To accomplish this encoding process, a left hand side segment (LHS-Segment) has been added to this chromosome as shown in Figure 2. In this segment, the gene $\alpha_{s,j}$ takes a random value in the interval [0, 1]. For a chromosome under consideration the size of the s^{th} sublot of job j is computed using Eq. (20). From this equation, it can be seen that certain sublots may have a size of zero if their corresponding genes have the value zero. Thus, the maximum and the actual number of sublots for each job and their sizes are encoded in the LHS-Segment. The right hand-side segment (RHS-Segment) is essentially the same as the chromosome shown in Figure 1.

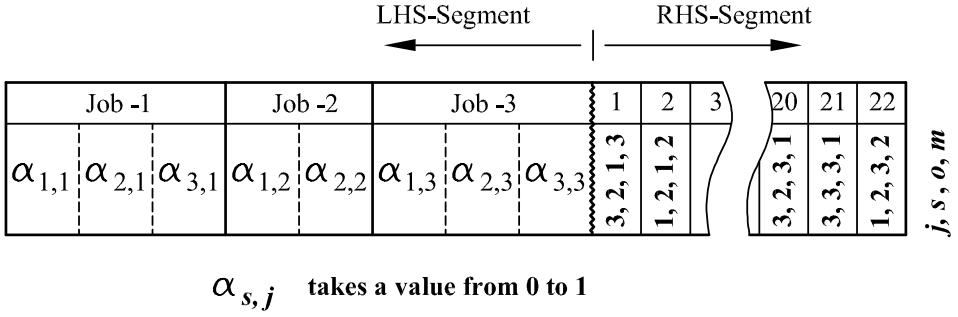


Figure 2: Solution representation used in solving the proposed FJSP-LS using GA

$$b_{s,j} = \begin{cases} \frac{\alpha_{s,j}}{\sum_{s=1}^{S_j} \alpha_{s,j}} \times B_j & ; \text{if } \sum_{s=1}^{S_j} \alpha_{s,j} > 0 \\ B_j / S_j & ; \text{otherwise} \end{cases} \tag{20}$$

3.2. Genetic Operators

Genetic operators make the population evolve by creating promising candidate solutions to replace the less-promising ones. These operators are generally categorized as selection, crossover, and mutation operators.

3.2.1 Selection Operator

A simple way to simulate the natural selection process in a GA is through tournament selection. In the proposed GA, we use a k-way tournament selection operator. In this operator, k individuals are randomly selected and the one presenting the highest fitness (smallest makespan) is declared the winner and a copy of this individual is added to the mating pool to form the next generation. Then, the k individuals in the tournament are placed back to the current population and the process is repeated. This continues until the number of individuals added to the mating pool is equal to the population size.

3.2.2 Crossover Operator

Once the mating pool is generated using the selection operator, the individuals in the pool are randomly paired to form parents for the next generation. Then for each pair, the algorithm arbitrarily selects one of the available crossover operators and applies it with certain probability to create two child individuals by exchanging information contained in the parents. The crossover operators are:

- Single Point Crossover-1 (SPC-1)
- Single Point Crossover-2 (SPC-2)
- Operation-to-Machine Assignment Crossover (OMAC)
- Job Level Operations Sequence Crossover (JLOSC)
- Sublot Level Operations Sequence Crossover (SLOSC)

The above five crossover operators are applied with probabilities equal to ρ_1, ρ_2, $\cdots$, ρ_5, respectively. The crossover operator SPC-1 generates an arbitrary crossover point in the LHS-Segment and swaps the part of this segment at the left of the crossover point. The crossover SPC-2 exchanges the portion of the LHS-Segment of the parents at the right of an arbitrarily chosen crossover point. These two crossover operators are illustrated in Figure 3. The crossover operators OMAC, JLOSC and SLOSC are specific to the RHS-Segments of the parent chromosomes. These operators are adapted from Kacem [34], where the authors did not consider lot streaming. They can be distinguished as assignment or sequence crossover operators. An assignment crossover operator generates two offsprings by exchanging the assignment of a subset of operations between two parents. OMAC is such an operator. From a

given pair of parent chromosomes, this operator creates two child chromosomes where each child chromosome retain the order of the operations supplied by the other parent. The creation of child-1 by this operator, retaining the order of the operations as obtained from parent-1, is illustrated in Figure 4. In step-1, operations from parent-1 are randomly selected. In step-2, all the genetic information of parent-1 without the assignment properties of the chosen operations is copied to child-1. In the last step, step-3, the assignment properties of the chosen operations are copied from parent-2 to complete child-1. Child-2 is created in a similar way where step-1 begins from parent-2.

Sequencing crossover operators only exchange the sequencing property of the operations in the parent chromosomes, i.e., the assignment of operations to machines is reserved in the offspring. JLOSC and SLOSC are such operators. The creation of child-1 by JLOSC, preserving the operation-assignment information of parent-1, is illustrated in Figure 5. In step-1, an operation is arbitrarily chosen from parent-1. In step 2, all the operations of all the sublots of the job which the chosen operation belongs to are copied to child-1. Step 3 is to complete the new individual with the remaining operations, in the same order as they appear in parent-2 while their assignment properties are kept unchanged as they were in parent-1. SLOSC is similar to JLOSC except step-2 of SLOSC is limited only to a single sublot which the arbitrarily chosen operation belongs to.

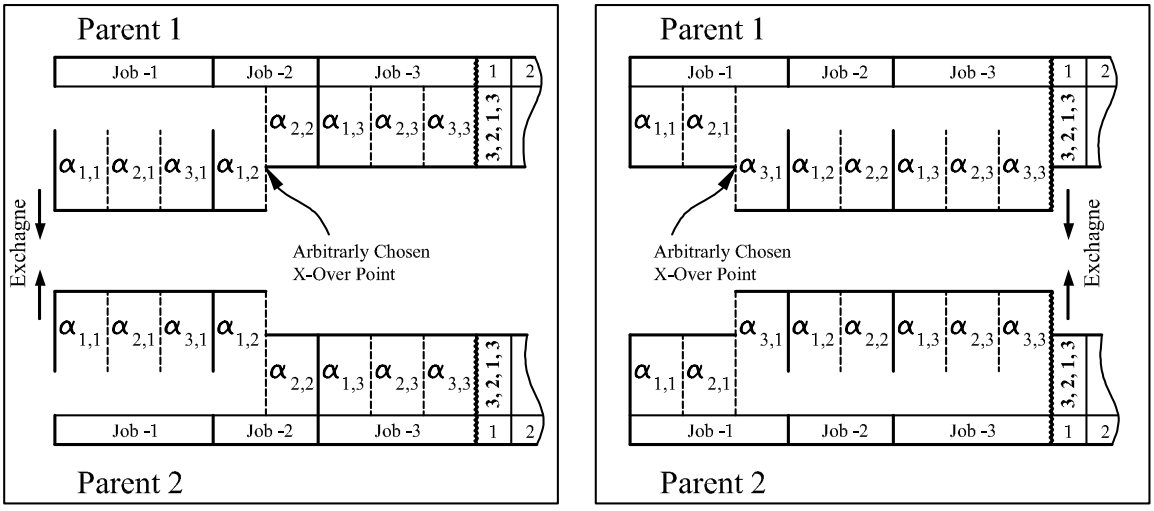


Figure 3: Single Point Corssover operators SPC-1 and SPC-2

3.2.3 Mutation Operator

Crossover operators do not introduce new genetic material into the population pool. This task is performed by the mutation operators acting on a single chromosome at the gene level to alter information contained in the gene. These operators are usually applied on each child chromosome with small probabilities. The six mutation operators used in the proposed PGA are:

- Sublot Step Mutation (SStM)

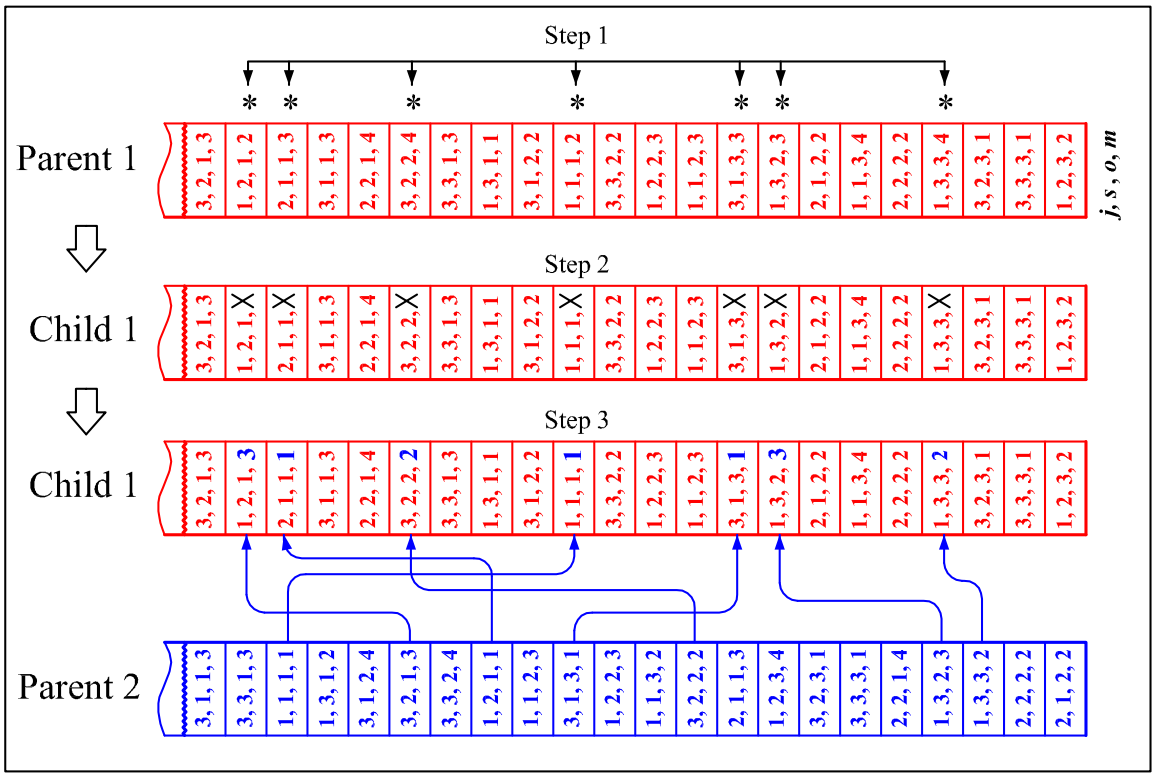


Figure 4: Operation assignment corssover operator.

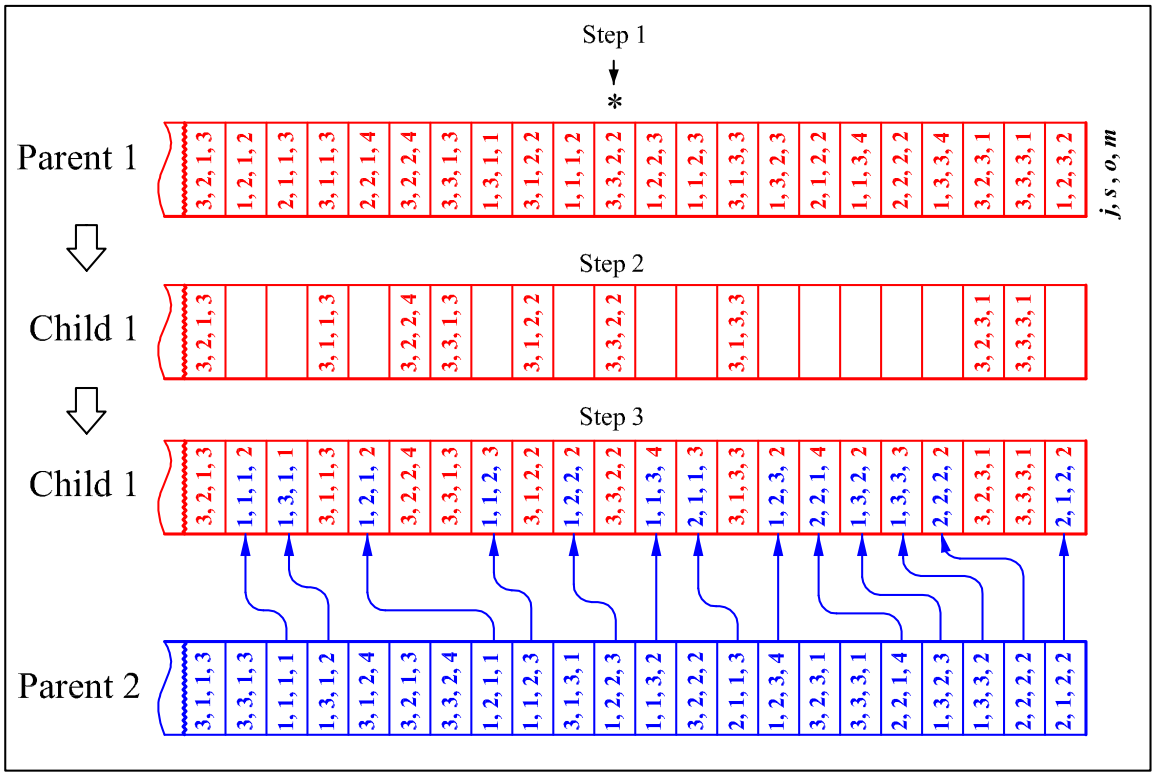


Figure 5: Job level operation sequence corssover operator.

- Sublot Swap Mutation (SSwM)
- Sublot Size degenerator (SSD)
- Random Operation Assignment Mutation (ROAM)
- Intelligent Operations Assignment Mutation (IOAM)
- Operations Sequence Shift Mutation (OSSM)

The operator SStM is applied with small probability σ_1 on each gene $\alpha_{s,j}$ in the LHS-Segment to step up or down the value of this gene with a step amount θ using equations $\alpha_{s,j} = \min\{1, \alpha_{s,j} + \theta\}$ and $\alpha_{s,j} = \max\{0, \alpha_{s,j} - \theta\}$, respectively. The step amount θ is calculated every time this operator is applied on a given $\alpha_{s,j}$ with the equation $\theta = \theta_{\max} \times \text{rand}()$ where $\theta_{\max} \in [0, 1]$ is a parameter and rand() is random number generator in [0,1]. The operator SSwM is applied with small probability σ_2 on each j in the LHS-Segment to swap the values of two arbitrarily selected genes $\alpha_{s,j}$ and $\alpha_{s',j}$. The operator SSD is a non-probabilistic mutation operator to set the value of $\alpha_{s,j} = 0$ if $\alpha_{s,j} / \sum_{s=1}^{S_j} \alpha_{s,j}$ is less than a degeneration limit d. The parameter d is chosen to be close to 0. The reason is that very small sublot sizes (less than $d \times 100\%$ of the lot size) may not lead to acceptable solution because of setup requirement for the sublots.

The mutation operators ROAM, IOAM and OSSM are specific to the RHS-Segment of the chromosomes. These operators are adapted from Kacem [34], and similar to the RHS-Segment specific crossover operators discussed in the previous subsection, these mutation operators can also be distinguished as assignment or sequence operators. Assignment mutation operators only change the assignment property of the chromosome undergoing the mutation while the sequencing property is reserved. ROAM is an assignment mutation operator applied with a small probability σ_3 on each gene of the RHS-Segment of each chromosome. Whenever it gets effected on a particular gene, it alters the assignment of the operation represented by the gene to one of its alternative machines. The other assignment mutation operator IOAM selects an operation on the machine with the maximum workload, and assign it to the machine with the minimum workload, if compatible. This operator is effected on each chromosome with a probability σ_4. The shift mutation operator OSSM, applied with a probability σ_5, selects an operation from RHS-Segment of the chromosome and moves it into another position, taking care of the precedence constraints for that operation.

3.3. Initial Population

In generating an initial population, we need to initialize both LHS- and RHS-Segments of each chromosome. In this study, the LHS-Segment of each chromomere is randomly initialized. This provides the sizes and numbers of the sublots of each job of the corresponding initial solution. Once the sizes of the sublots are known, the processing times (sublot size $\times$ unit processing time) of each operation on the various alternative machines can be computed. Using this processing time information and regarding each sublot as a job, the RHS-Segment of the chromosome (i.e., the operation assignment and sequencing) can be initialized using the technique outlined in [45]. This technique takes into account

both the processing times and the workload of the machines, i.e., the sum of the processing times of the operations assigned to each machine. The procedure proceeds in finding, for each operation, the machine with the minimum workload.

3.4. Fitness Evaluation

The makespan of the schedule corresponding to a given chromosome is used as the fitness measurement of this chromosome. In calculating the makespan, we take into account: (1) the dependance of setup time on sequence, (2) the nature of the setup (attached or detached), (3) lag time requirement of certain operations, (4) machine release dates, and (5) the possibility of the sizes of certain sublots of becoming zero. The procedure is outlined below.

Step 1. Using the information obtained from the LHS-Segment of the chromosome and Eq. (20), calculate the sizes of the sublots of the various jobs.

Step 2. Set $l = 1$

Step 3. Set the values of indices j, s, o and m as obtained from the gene at location l of the RHS-Segment of the chromosome.

Step 4. If $b_{s,j}$ is greater than zero, then go to Step 5; otherwise go to Step 6.

Step 5. Calculate the completion time $c_{o,s,j,m}$

- If (1) operation o of sublot s of job j is the first operation assigned to machine m and (2) $o = 1$, then:
  $c_{o,s,j,m} = D_m + S^*_{o,j,m} + b_{s,j} \cdot T_{o,j,m}$.
- If (1) operation o of sublot s of job j is the first operation assigned to machine m, (2) $o > 1$, and (3) operation o-1 is assigned to machine m', then:
  $c_{o,s,j,m} = \max\{D_m + (1 - A_{o,j}) \times S^*_{o,j,m}; \quad c_{o-1,s,j,m'} + L_{o,j}\} + b_{s,j} \times T_{o,j,m} + A_{o,j} \times S^*_{o,j,m}$.
- If (1) operation o' of sublot s' of job j' is the operation to be processed immediately before operation o of job sublot s of j on machine m and (2) $o = 1$, then:
  $c_{o,s,j,m} = c_{o',s',j',m} + S_{o,j,m,o',j'} + b_{s,j} \cdot T_{o,j,m}$.
- If (1) operation o' of sublot s' of job j' is the operation to be processed immediately before operation o of sublot s of job j on machine m, (2) $o > 1$, and (3) operation o-1 is assigned to machine m', then:
  $c_{o,s,j,m} = \max\{c_{o',s',j',m} + (1 - A_{o,j}) \times S_{o,j,m,o',j'}; \quad c_{o-1,s,j,m'} + L_{o,j}\} + b_{s,j} \times T_{o,j,m} + A_{o,j} \times S_{o,j,m,o',j'}$.

Step 6. If l is less than the total number of genes of the RHS-Segment of the chromosome, increase its value by 1 and go to Step 3; otherwise go to Step 7

Step 7. Calculate the makespan of the schedule as $c_{max} = \max\{c_{o,s,j,m}; \quad \forall(o, s, j, m)\}$ and set the fitness of the solution to c_{max}.

The above procedure (in particular Step 5) is based on the property of the chromosomes that, for a given j and s, the gene (j, s, o, m) always lies to the right of all the other genes (j, s, o', m') having $o' < o$. Because of this property of the chromosome, whenever the completion time of operation (j, s, o, m) on machine m is to be calculated, the completion time of operation $(j, s, o-1, m')$ is already calculated and available, regardless to which machine this preceding operation is assigned. Moreover, the completion time of the operation (j', s', o', m) to be processed on machine m immediately before operation (j, s, o, m) is also calculated and available.

3.5. Parallelization of the GA

¿From early days of its development, the GAs potential for parallelization has been noticed with all its attendant benefits of efficiency. GAs can be parallelized in different ways and detailed taxonomy can be found in Nowostawski and Poli [42] and Cantú-Paz [10]. These studies show three major types of PGAs: (1) single-population master-slave PGAs, (2) single-population fine-grained PGAs, and (3) multiple-population island model PGAs. Figure 6 presents schematic representations of these parallelization techniques.

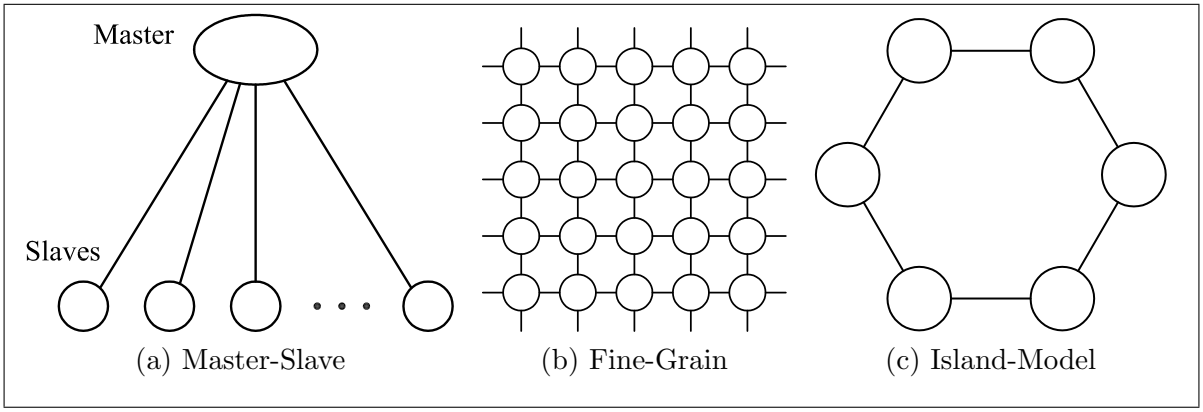
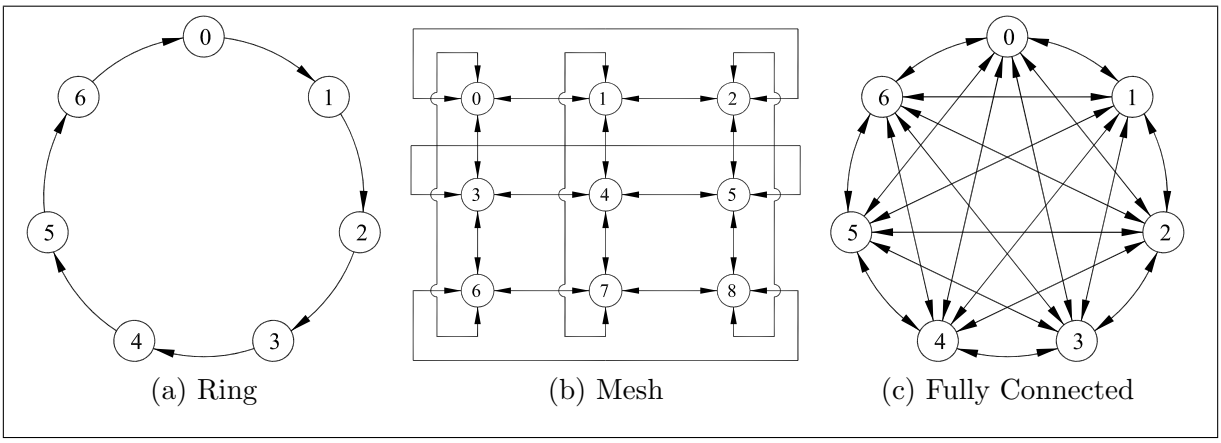


Figure 6: Different parallelization schemes of genetic algorithm.

In this research, we follow the island model PGAs to efficiently solve the proposed FJSP-LS model. This class of PGAs has an appealing trait in that it often reduces the computational effort to solve the same problem as compared to SGAs, even on a single processor computer [29]. This characteristic makes a difference with respect to other search algorithms in that island-model PGAs are not simple parallel versions of sequential algorithms. Thus they represent a new class of algorithms that search the solution space differently [42]. The reason for this can be found in the most significant characteristics of this class of parallelization [1]: (1) their decentralized search, which allows speciation (different subpopulations evolve towards different solutions), (2) the larger diversity levels (many search regions are sought at the same time), and (3) exploitation inside these subpopulations, i.e., refining the better solutions found in each subpopulation. In addition to these interesting characteristics, this class of PGAs can easily be implemented using public domain libraries such as MPI (Message Passing Interface) on a cluster of inexpensive computers connected to a slow network. These characteristics are the main motivating factors to consider island-model PGAs for solving the comprehensive model proposed in this paper.

This class of PGAs can be implemented in different connection topologies. In this paper we considers three commonly used topologies shown in Figure 7. Each circle represents a processing element evolving its own subpopulation.



(a) Ring (b) Mesh (c) Fully Connected

Figure 7: Common connection topologies of island model parallel genetic algorithm.

4. Numerical Example

In this section, we present several example problems and computational results to illustrate the features of the proposed model and the performance improvement achieved using parallellization of the GA.

4.1. Model Illustration

A small instance of lot streaming problem consisting of processing three jobs in a four-machine flexible job-shop is considered to illustrate the features of the developed model. The batch size of each job, and for each operation the nature of setup (attached or detached), lag time, alternative machines and corresponding processing times are given in Table 3. The sequence dependent setup time data are given in Table 4. This small example problem was solved using the proposed algorithm for three different cases. In case-1, it was assumed that there is no lot streaming while in case-2 it was assumed that each job is to split into a maximum of three sublots. Case-3 is same as case-2 except the first machine is not available for the first 800 minutes. The sizes of the various sublots and the Gantt charts of the resulting schedules are given in Figure 8. Numerical values of the starting and the ending times of setups and the operations are given in Table 5.

When lot streaming is not consisted, the makespan of this small problem is 2876 minutes as shown in Figure 8-a. In case-2, the problem was solved again by letting the maximum number of sublots for each job to be 3. A schedule with 2 sublots for each job and a makespan of 2290 minutes was obtained. This is about 20% reduction of the makespan from that in case 1. In the literature, lot streaming has been often touted to enable the overlapping of successive operations of a given job in a multistage manufacturing systems. In this research, we observed that when routing flexibility is considered, lot streaming can also enable the overlapping of identical operations, i.e., operation o of a given job j on one machine with operation o of the same job on another alternative machine. The overlapping of successive

Table 3: Processing Data for Jobs

j	B_j	o	$A_{o,j}$	$L_{o,j}$	Alternative routes, (m, $T_{o,j,m}$)		
					1	2	3
1	540	1	na	na	(1, 3.00)	(3, 2.90)	
		2	1	200	(1, 1.00)	(2, 0.90)	
		3	0	0	(1, 0.60)	(3, 0.70)	(4, 0.60)
2	580	1	na	na	(3, 2.50)	(4, 2.40)	
		2	1	0	(1, 0.60)	(2, 0.70)	(4, 0.60)
3	490	1	na	na	(1, 0.50)	(4, 0.50)	
		2	1	180	(1, 0.50)	(3, 0.40)	(4, 0.40)
		3	1	0	(1, 2.50)	(3, 2.40)	(4, 2.60)

na = not applicable

Table 4: Sequence Dependent Setup Time Data

j	o	m	Setup time ($S^*_{o,j,m}$), $\cdots$,(j', o', $S_{o,j,m,o',j'}$) $\cdots$
1	1	1	(150), (1,1,80), (1,2,160), (1,3,160), (2,2,270), (3,1,270), (3,2,240), (3,3,210)
		3	(200), (1,1,80), (1,3,160), (2,1,210), (3,2,240), (3,3,210)
	2	1	(50), (1,1,120), (1,2,60), (1,3,140), (2,2,150), (3,1,180), (3,2,240), (3,3,300)
		2	(100), (1,2,80), (2,2,180)
	3	1	(150), (1,1,120), (1,2,160), (1,3,60), (2,2,270), (3,1,270), (3,2,270), (3,3,210)
		3	(150), (1,1,140), (1,3,60), (2,1,180), (3,2,210), (3,3,270)
		4	(100), (1,3,60), (2,1,240), (2,2,180), (3,1,270), (3,2,270), (3,3,270)
2	1	3	(200), (1,1,180), (1,3,300), (2,1,70), (3,2,240), (3,3,270)
		4	(100), (1,3,210), (2,1,70), (2,2,160), (3,1,180), (3,2,180), (3,3,180)
	2	1	(100), (1,1,180), (1,2,210), (1,3,210), (2,2,70), (3,1,150), (3,2,300), (3,3,180)
		2	(150), (1,2,180), (2,2,80)
		4	(150), (1,3,180), (2,1,100), (2,2,60), (3,1,270), (3,2,270), (3,3,270)
3	1	1	(100), (1,1,210), (1,2,210), (1,3,210), (2,2,240), (3,1,80), (3,2,180), (3,3,180)
		4	(100), (1,3,210), (2,1,240), (2,2,240), (3,1,60), (3,2,200), (3,3,120)
	2	1	(100), (1,1,180), (1,2,300), (1,3,210), (2,2,270), (3,1,140), (3,2,50), (3,3,140)
		3	(200), (1,1,270), (1,3,270), (2,1,300), (3,2,60), (3,3,180)
		4	(150), (1,3,180), (2,1,270), (2,2,240), (3,1,180), (3,2,80), (3,3,120)
	3	1	(100), (1,1,270), (1,2,180), (1,3,150), (2,2,180), (3,1,160), (3,2,160), (3,3,70)
		3	(100), (1,1,180), (1,3,180), (2,1,270), (3,2,180), (3,3,80)
		4	(50), (1,3,270), (2,1,180), (2,2,150), (3,1,180), (3,2,140), (3,3,70)

operations can be observed in Figure 8-b where the first and the second operations of job 2 (operations (2,2,1) and (2,1,2)) are overlapped when they are processed on machines 4 and 2, respectively. Other such overlapping are also present in the schedule as shown in the Gantt chart. If alternative rouging is not considered, this overlapping cannot happen. This entails that lot streaming can result in a greater makespan reduction in flexible job-shop scheduling than in classical job-shop scheduling.

The schedule shown in Figure 8-c, case-3, is obtained by assuming that machine 1 is not available in the first 800 minutes. This schedule is quite different from the one shown in case-2. For example, in case-2, six operations are assigned to machine 1 whereas in case-3 only two operations are assigned to this machine. This is because, when routing flexibility is considered, the alternative machines of a given operation that will be released soon from previous schedule may represent better choices for this operation. In flexible job-shop, there may be several such operations having alternative machines with different release dates. This demonstrates the importance of considering machine release date in this type of shops.

Other several features of the proposed model are also illustrated in this example. As shown in Table 3, lag times of 200 and 180 minutes were assumed for the second operations of jobs 1 and 3, respectively. Because of such lag time assumption, in case-1 for example, the setup of operation (3,1,2) on machine 1 and that of (1,1,2) on machine 2 started at 525 and 1966 minutes. Without these lag times, they will be started at 345 and 1766 minutes, respectively. Another important consideration in the model was the nature of setup of being attached or detached. The third operation of job 1 has a detached setup allowing overlapping of the setup of this operation with that of the second operation of the same sublot. Such overlapping can also be observed in the Gantt charts in Figure 8. Also shown in this figure, no such overlaps occur for all the operations where attached setups was assumed.

In this example problem one can also see that although lot streaming decreases the overall makespan, it increases the number of setups and the total setup time. In case-1, when lot streaming was not considered, we have 8 setup incidences with a total setup time of 1110 minuets. In case case-2, the total number of setups increased to 16 with a total setup time of 2060 minutes. This is about 86% increase in total setup time. This demonstrates that optimizing the sequences to minimize setup time is more important when LS is considered.

4.2. Computational Performance

In this section, we show the performance of the developed genetic algorithm and the improvements achieved through parallel computation. Figure 9 shows the convergence history of CPLEX (version 11.2.0) and that of the sequential genetic algorithm in solving a small problem instance presented in the previous section (Problem-1, Case-c). In this figure it can be seen that the makespan of the schedule generated by CPLEX after about 6 and half hours of computation is 2692 minutes and does not improve afterwards. Whereas, the genetic algorithm was able to generate a schedule with a makespan of 2530 in less than one minuet of computation. In order to validate this improved solution generated by the genetic algorithm, we submitted it to CPLEX as a starting incumbent solution and it was accepted as a feasible starting solution, though CPLEX was unable to further improve it. For larger problems studied

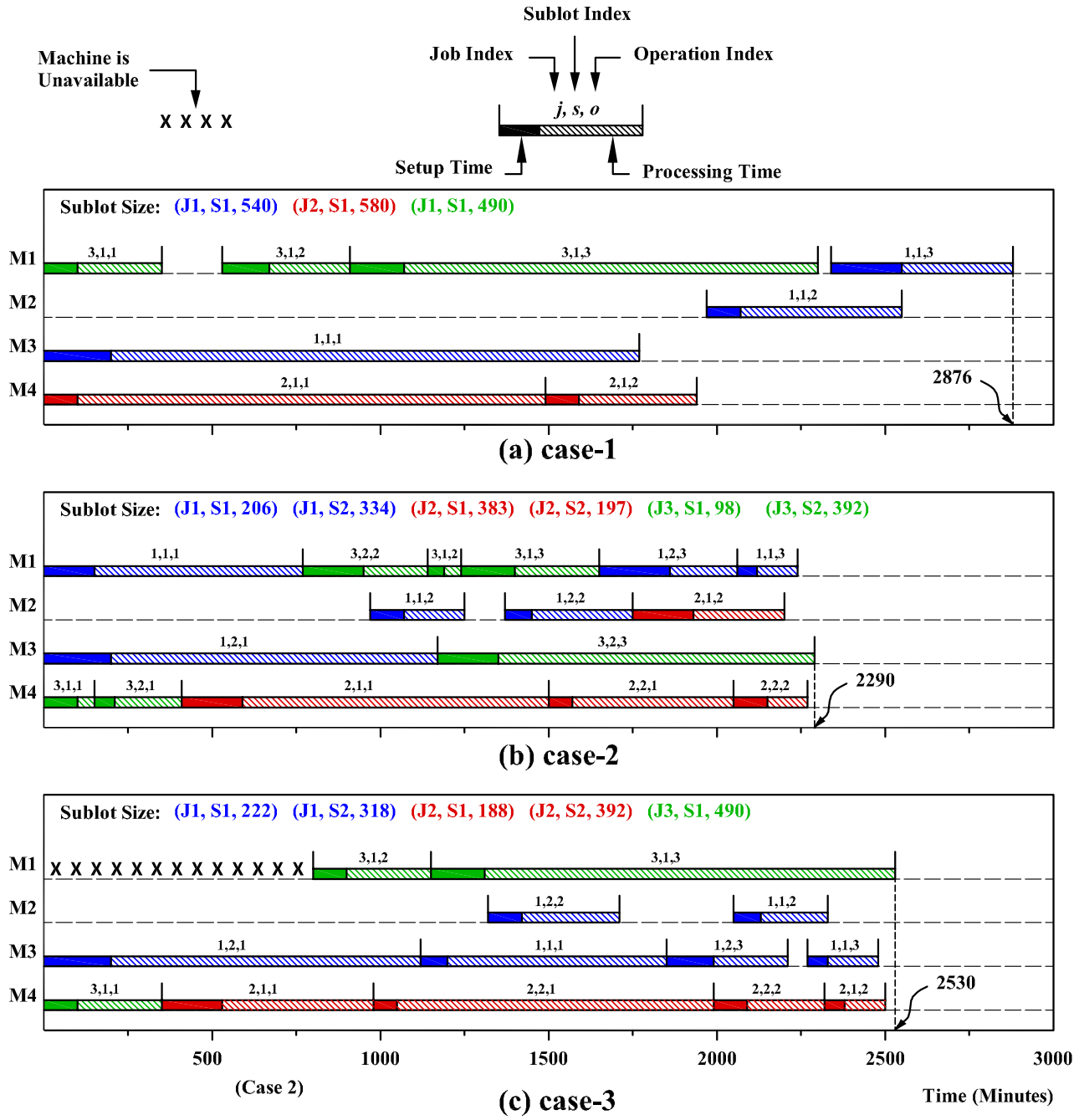


Figure 8: Schedule for problem-1: (a) without lot streaming, (b) with lot streaming, and (c) when machine M1 is not available for the first 800 minutes. Note: The detail numerical values of the starting and the ending times of the setups and the operations are given in Table 5.

Table 5: The details of the schedules shown in Figure 8

		Case-1				Case-2				Case-3			
Machine	Run	(j,s,o)	SB	SE/PB	PE	(j,s,o)	SB	SE/PB	PE	(j,s,o)	SB	SE/PB	PE
M1	R1	(3,1,1)	0	100	345	(1,1,1)	0	150	768	(3,1,2)	800	900	1145
	R2	(3,1,2)	525	665	910	(3,2,2)	768	948	1144	(3,1,3)	1145	1305	2530
	R3	(3,1,3)	910	1070	2295	(3,1,2)	1144	1194	1243				
	R4	(1,1,3)	2342	2552	2876	(3,1,3)	1243	1403	1647				
	R5					(1,2,3)	1647	1857	2058				
	R6					(1,1,3)	2058	2118	2241				
M2	R1	(1,1,2)	1966	2066	2552	(1,1,2)	968	1068	1253	(1,2,2)	1322	1422	1708
	R2					(1,2,2)	1369	1449	1750	(1,1,2)	2046	2126	2326
	R3					(2,1,2)	1750	1930	2197				
	R4												
	R5												
	R6												
M3	R1	(1,1,1)	0	100	1766	(1,2,1)	0.0	200	1169	(1,2,1)	0	200	1122
	R2					(3,2,3)	1169	1349	2290	(1,1,1)	1122	1202	1846
	R3									(1,2,3)	1846	1986	2209
	R4									(1,1,3)	2266	2326	2481
	R5												
	R6												
M4	R1	(2,1,1)	0	100	1492	(3,1,1)	0	100	149	(3,1,1)	0	100	345
	R2	(2,1,2)	1492	1592	1940	(3,2,1)	149	209	405	(2,1,1)	345	525	976
	R3					(2,1,1)	405	585	1503	(2,2,1)	976	1046	1987
	R4					(2,2,1)	1503	1573	2047	(2,2,2)	1987	2087	2322
	R5					(2,2,2)	2047	2147	2266	(2,1,2)	2322	2382	2495
	R6												

Note: SB, SE, PB, PE stand for setup begins, setup ends, processing begins, and processing ends, respectively.

in this paper, CPLEX was unable to start computation because of large memory requirement whereas the genetic algorithm was able to generate solutions in just few minutes and progressively improve those solutions. This clearly demonstrates the potential of the genetic algorithm in solving larger problem instances.

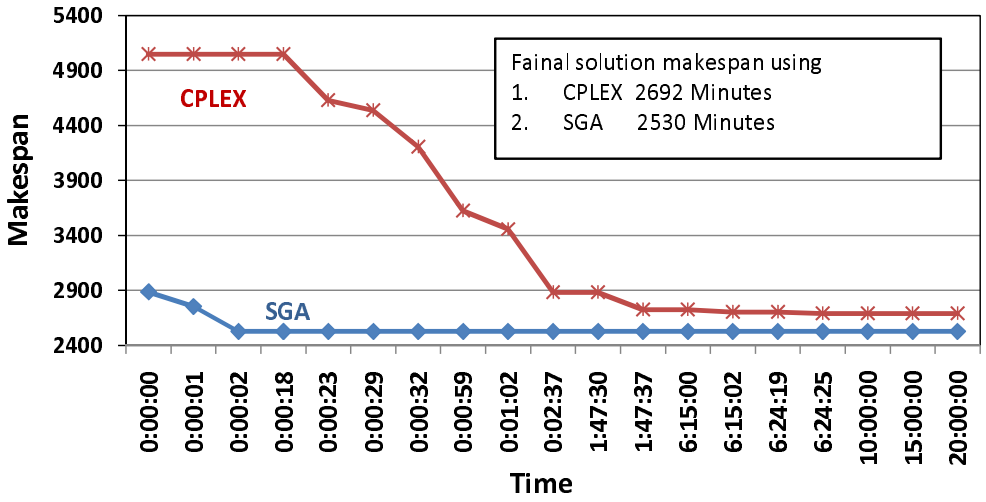


Figure 9: Comparison of CPLEX and the SGA in solving Problem 1

The performance of the genetic algorithm can be further improved by parallel computation using island model PGA presented in Section 3.5. In order to illustrate such performance improvement, we consider problems much lager than the Problem 1 presented in the previous section. The general nature of the considered problems are given in Table 6. Figure 10 illustrates the performance improvement archived in solving Problem 2 through parallel computation as the number of processors increases. Each curve represents an average convergence of the genetic algorithm from 10 test runs with different genetic parameters given in Table 7. In Figure 10 it can be seen that, at its convergence, the sequential genetic algorithm generates a schedule with a makespan of 5227 minutes on the average. The 8-processor parallel genetic algorithm reduces this makespan by about 109 minutes. As we increase the number of processors to 16, 24, 32 and 48, the average makespan of the several test runs is reduced by 125, 165, 170, and 183 minutes, respectively. Such improvements obtained by parallel computation in solving Problem-2 were also observed in solving several other problems considered in this paper. Figure 11 shows the average convergence graphs of the sequential genetics algorithm (SGA) and a 24-subpopulation parallel genetic algorithm in solving problems 2, 3, 4, and 5. In this figure, it can be seen that in all the problems considered there are makespan reductions through parallel computation. The parallel genetic algorithm was codded in C++ programming language using MPI message-passing library for communication. The code was executed in a parallel computation environment composed of more than 250 interconnected workstation each having a 8-core Intel Xeon 2.8GHz processor. The test problems were run using up to 48 cores.

Table 6: The general nature of the problems considered

Problem No.	Number of machines	Number of jobs	Number of sublots for each job	Number of operations for the jobs	Number of alternative routes for the operation
2	8	20	4	3 to 5	1 to 3
3	12	30	4	3 to 6	1 to 3
4	10	25	4	3 to 4	1 to 3
5	12	35	3	2 to 4	1 to 3

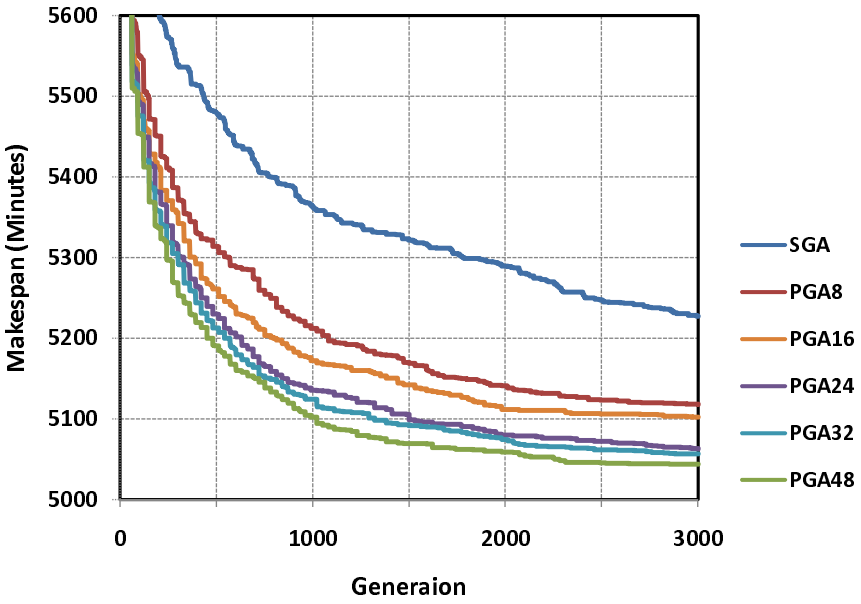
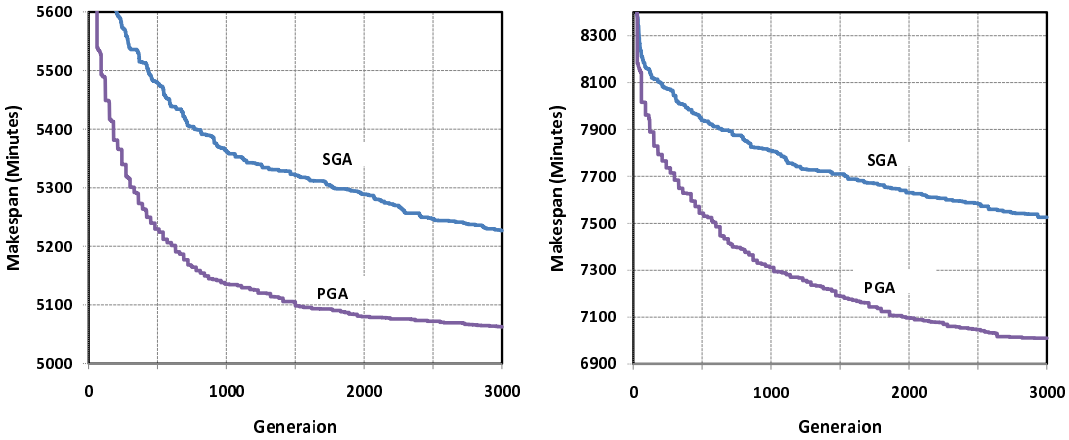
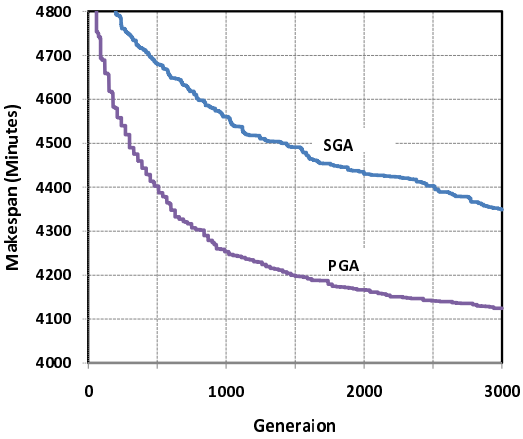


Figure 10: Performance improvement through parallelization of the genetic algorithm as the number of processor is increased from 1 to 8, 16, 24, 32, and to 48.

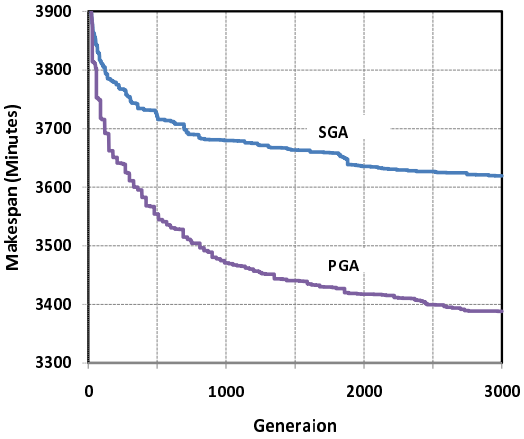


(a) Problem 2

(b) Problem 3



(a) Problem 4



(b) Problem 5

Figure 11: Average convergence of the SGA vs PGA for problems 1, 2, 4, and 5.

4.3. Some Empirical Studies

In this section we present some empirical studies on the impact of the parameters of the proposed algorithm on its performance. In Figure 12 are the plots of the makespan of the schedules obtained after several iterations by SGA and the PGA with 8, 16, 32 and 48 subpopulation under different test runs. The test runs are differentiated by the settings of their genetic parameters as shown in Table 7. In Figure 12, we can see that the patterns by which the final solution quality of the SGA is affected by the genetic parameters are similar to those of the PGA irrespective of number of computing cores (same as the number of subpopulations) used. This indicates that the genetic parameter tuning (such as population size, tournament size, probabilities for the crossover and mutation operators) of PGA can be done by using SGA without committing parallel computing resource. We also obsered that the patterns by which the final solution quality from both SGA and PGA are affected by the change of the genetic parameters are similar for different problem instances as can seen in Figures 12, 13-(a), (b), and (c). This implies that the genetic parameters setting that worked well for one problem instance is likely to work well for other problems.

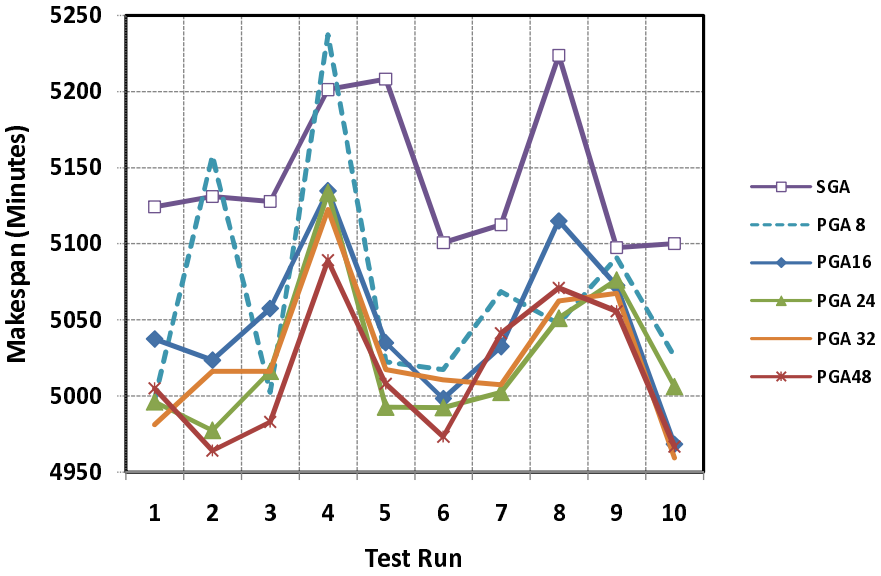
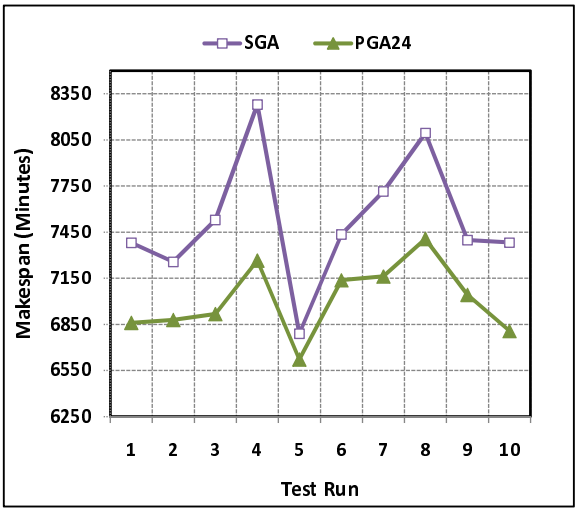
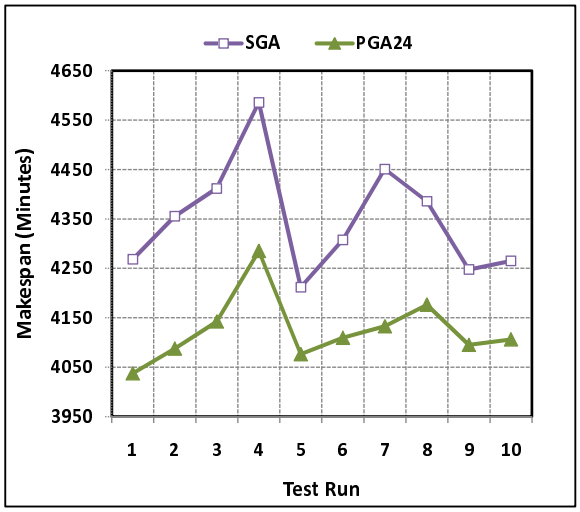


Figure 12: The effect of changing genetic parameters on the final solution quality obtained by the SGA and PGA with different number of subpopulation in solving problem 2

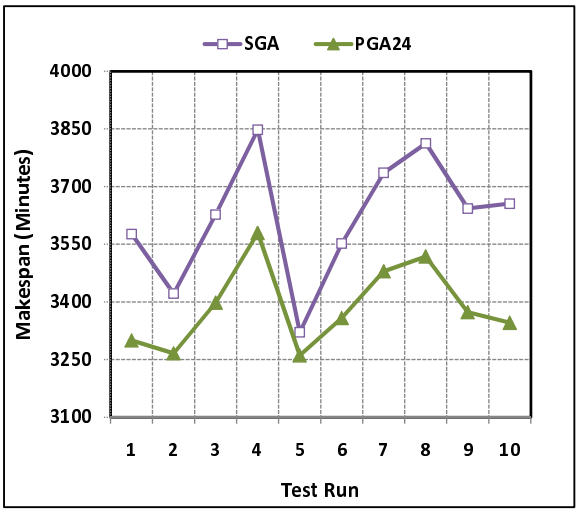
We also performed empirical studies on the convergence behavior of the proposed PGA affected by parameters specific to the island model parallelization. One such parameter is the connection topology. Figure 14 shows the convergence graphs of the 24-subpopulation PGA with different connection topologies as illustrated in Figure 7. The curves show the average convergence of 10 test runs as described previously. From this figure it can be seen that the fully connected topology outperforms other two topologies. Another PGA specific parameter is the migration policy. It determines the individuals to migrate from the source subpopulation and those to be replaced by the migrants in the destination subpopulation. In this empirical study we considered three different migration policies: (1) random-replace-random (RR), (2) best-replace-random (BR), and (3) best-replace-worst (BW). Figure 15 shows



(a) Problem 3



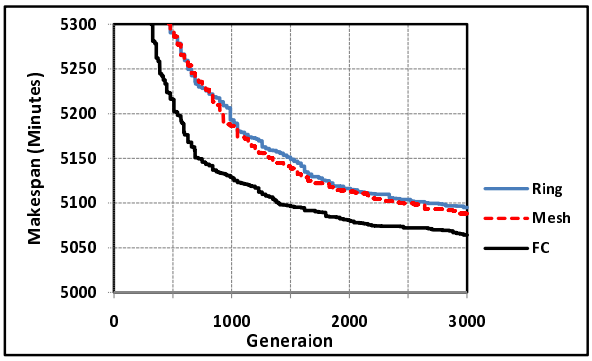
(b) Problem 4



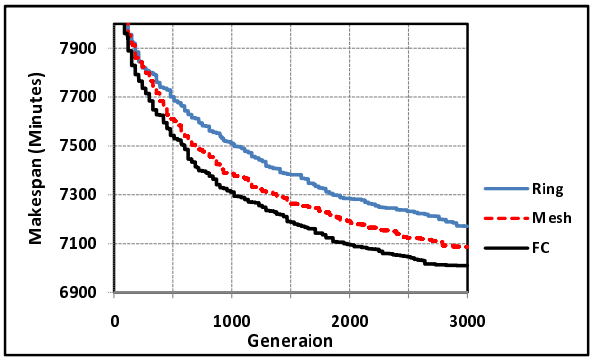
(c) Problem 5

Figure 13: The effect of changing genetic parameters on the final solution quality obtained by the SGA and 24 subpopulation PGA in solving problems 3, 4, and 5

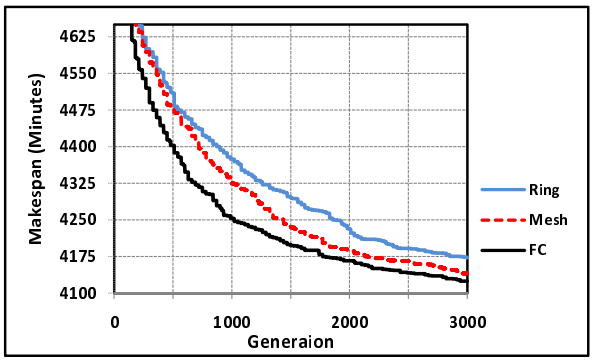
the convergence graphs of the PGA in solving problems 2 and 3. The curves represent the average convergence of the 10 test runs. It can be seen that the PGA is less sensitive to the change of migration policy while the best-replace-random migration policy slightly outperforms the other two policies.



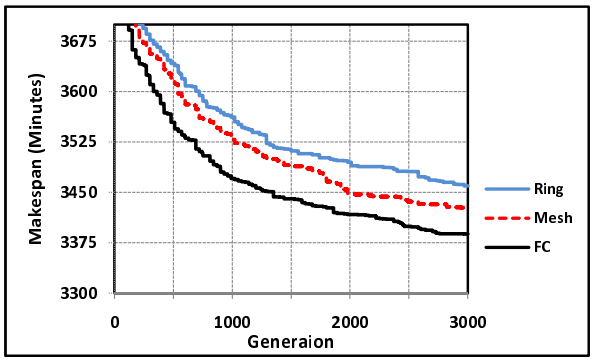
(a) Problem 2



(b) Problem 3



(a) Problem 4

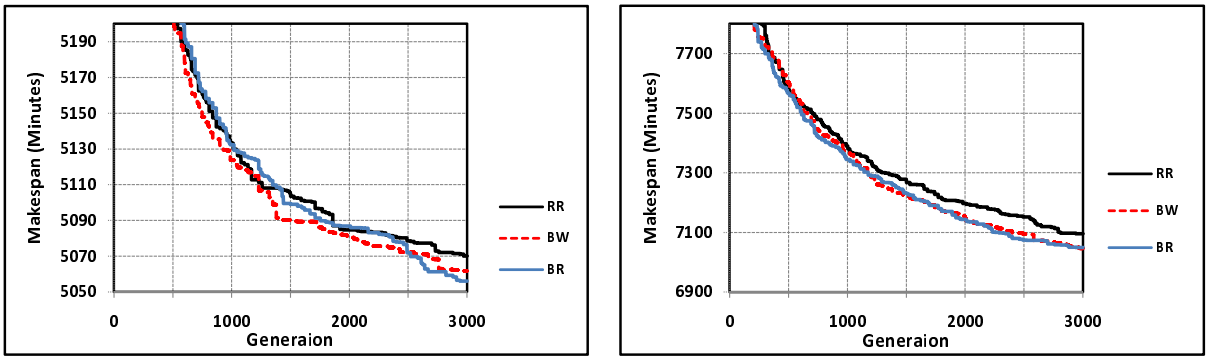


(b) Problem 5

Figure 14: Average convergence of the PGA under different connection topologies for problems 1, 2, 4, and 5.

5. Discussion and Conclusions

Lot streaming is a technique in which a production lot is split into smaller sublots such that each sublot is treated individually and transferred to the next processing stage upon its completion. Thus, different sublots of the same job can be processed simultaneously at different stages, thereby reducing production makespan. In this research we noted that the study of lot streaming in job-shop is very limited. In this paper, we attempted to extend this limited literature by considering (1) routing flexibility, (2) sequence dependent setup time, (3) attache/detached nature of setup, (4) machine release dates, (5) lag time, and (6) high performance parallel computation. Because of the consideration of routing flexibility, the problem studied can be termed as flexible-job shop scheduling with lot streaming. A mixed integer linear programming model was developed to formalize the problem. The developed model is NP-hard and difficult to solve even for small size problems using off the shelf optimization software. To this end, we developed a parallel genetic algorithm that runs on a high-performance parallel computing environment. A numerical example showed the importance of considering several pragmatic issues addressed in this paper in an integrated manner. Several other numerical example problems were



(a) Problem 2 (b) Problem 3

Figure 15: Average convergence of the parallel genetic algorithm under different migration policies for problems 1 and 2.

solved by the developed PGA and the examples show the superior computational performance of the PGA against those of a sequential GA and an off-shelf optimization package. In our future research, we plan to extend the model and the solution procedure to solving lot streaming problems considering multiple objectives such as workload balancing, due dates and mean flow-time requirements, and others factors such as capacitated buffer and transportation time.

Acknowledgements: This research is supported by Discovery Grant from NSERC and by Faculty Research Support Fund from the Faculty of Engineering and Computer Science, Concordia University, Montreal, Canada. We thank RQCHP (Réseau québécois en calcul de haute performance http://www.rqchp.qc.ca/) for its assistance in providing access to parallel computing facilities.

Table 7: Genetic parameters used for the test runs

		Test Run									
Parameter		1	2	3	4	5	6	7	8	9	10
Population Size		5500	2500	4500	3000	4000	3500	2500	2000	2800	3800
Tournament size factor		0.1	0.2	0.05	0.03	0.2	0.15	0.12	0.25	0.2	0.15
Crossover probability for:											
SPC1	ρ_1	0.85	0.8	0.95	0.8	0.75	0.8	0.75	0.9	0.7	0.8
SPC2	ρ_2	0.85	0.8	0.95	0.8	0.9	0.8	0.75	0.8	0.85	0.8
OMAC	ρ_3	0.95	0.85	0.8	0.75	0.75	0.9	0.85	0.75	0.85	0.9
JLOSC	ρ_4	0.8	0.9	0.85	0.95	0.8	0.75	0.8	0.7	0.8	0.85
SLOSC	ρ_5	0.9	0.8	0.85	0.9	0.95	0.9	0.75	0.7	0.9	0.83
Mutation probability for:											
SStM	σ_1	0.15	0.1	0.05	0.1	0.02	0.1	0.15	0.1	0.02	0.1
SSwM	σ_2	0.1	0.08	0.1	0.12	0.1	0.1	0.2	0.15	0.1	0.05
SSD	d	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ROAM	σ_3	0.1	0.05	0.1	0.1	0.03	0.1	0.1	0.15	0.1	0.1
IOAM	σ_4	0.1	0.12	0.2	0.15	0.05	0.2	0.1	0.25	0.2	0.1
OSSM	σ_5	0.2	0.1	0.25	0.1	0.05	0.1	0.05	0.15	0.06	0.15

References

[1] Alba, E. and Troya, J. M., 2000. Influence of the migration policy in parallel distributed gas with structured and panmictic populations. *Applied Intelligence*, **12**, 163–181.

[2] Baker, K., 1974. Introduction to Sequence and Scheduling. Wiley, NY,

[3] Baker, K., 1995. Lot streaming in the two-machine flow shop with setup times. *Annals of Operations Research*, **57**, 1–11.

[4] Baker, P. D., K.R, 1990. Solution procedures for the lot streaming problem. *Decision Sciences*, **21**, 475–491.

[5] Biskup, D. and Feldmann, M., 2006. Lot streamin with varaibal sublots: an integer programming formulation. *Journal of Operational Research Society*, **57**, 296–303.

[6] Blackburn, J., 1991. Time-Based Competition. Business One Irwin, Burr Ridge, IL,

[7] Bockerstette, J. and Shell, R., 1993. Time Based Manufacturing. McGraw-Hill, New York,

[8] Bukchin, J. and Masin, M., 2004. Multi-objective lot splitting for a single product m-machine flowshop line. *IIE Transactions*, **36**, 191–202.

[9] Buscher, U. and Shen, L., 2008. An integrated tabu search algorithm for the lot streaming problem in job shops. *European Journal of Operational Research*, **In Press (DOI: 10.1016/j.ejor.2008.11.046)**.

[10] Cantú-Paz, E., 2000. Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Norwell, MA,

[11] Chan, F., Wong, T., and Chan, P., 2004. Equal size lot streaming to job-shop scheduling problem using genetic algorithm. Proceedings of the 2004 IEEE International Symposium on Intelligent Control. Taipei, Taiwan, September 2-4, 2004, pp. 472–476.

[12] Chan, F., Wong, T., and Chan, P., 2005. Lot streaming technique in job-shop environment. Proceedings of the 13th Mediterranean Conference on Control and Automation. Limassol, Cyprus, June 27-29, 2005, pp. 364–369.

[13] Chan, F., Wong, T., and Chan, P., 2008. The application of genetic algorithms to lot streaming in a job-shop scheduling problem. *International Journal of Production Research*, **In press (DOI: 10.1080/00207540701577369)**.

[14] Chan, F., Wong, T., and Chan, P., 2008. Lot streaming for product assembly in job shop environment. *Robotics and Computer-Integrated Manufacturing*, **24**, 321–331.

[15] Chang, J. H. and Chiu, H. N., 2005. A comprehensive review of lot streaming. *International Journal of Production Research,*, **43**, 1515–1536.

[16] Chen, H., Ihlow, J., and Lehmann, C., 1999. A genetic algorithm for flexible job-shop scheduling. In the proceedings of the 1999 IEEE International Conference on Robotics & Automation. May 1999, Detroit, Michigan, pp. 1120–1125.

[17] Chen, J., Chen, K., Wu, J., and Chen, C., 2007. A study of the flexible job shop scheduling problem with parallel machines and reentrant process. *International Journal of Advanced Manufacturing Technology*, In Press, DOI 10.1007/s00170-007-1227-1.

[18] Chen, J. and Steiner, G., 1996. Lot streaming with detached setups in three-machine flow shops. *European Journal of Operational Research*, **96**, 591–611.

[19] Chiu, H. N., Chang, J. H., and Lee, C. H., 2004. Lot streaming models with a limited number of capacitated transporters in multistage batch production systems. *Computers & Operations Research*, **31**, 2003–2020.

[20] Conway, R. and Maxwell, W., 1967. Theory of Scheduling. Addison-Wesley, MA,

[21] Dauzere-Peres, S. and Lasserre, J., 1993. An iterative procedure for lot streaming in job-shop scheduling. *Computers and Industrial Engineering*, **25**, 231–234.

[22] Dauzere-Peres, S. and Lasserre, J., 1997. Lot streaming in job-shop scheduling. *Operations Research*, **45**, 584–595.

[23] Edis, R. and Ornek, A., 2009. Simulation analysis of lot streaming in job shops with transportation queue disciplines. *Simulation Modelling Practice and Theory*, **17**, 442–453.

[24] Feldmann, M. and Biskup, D., 2008. Lot streaming in a multiple product permutation flow shop with intermingling. *International Journal of Production Research*, **46**, 197–216.

[25] Gao, J., Gen, M., Sun, L., and Zhao, X., 2007. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. *Computers & Industrial Engineering*, **53**, 149–162.

[26] Gao, J., Sun, L., and Gen, M., 2008. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, **35**, 2892–2907.

[27] Garey, M. R., Johnson, D. S., and Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, **1**, 117–129.

[28] Glass, C., JND, G., and Potts, C., 1994. Lot streaming in three-stage process. *European Journal of Operations Research*, **75**, 378–394.

[29] Gordon, V. and Whitley, D., 1993. Serial and parallel genetic algorithms as function optimizers. Proceedings of the Fifth International Conference on Genetic Algorithms, edited by S. Forrest. Morgan Kaufmann, San Mateo, CA., pp. 177–183.

[30] Gupta, J., 1986. Flowshop schedules with sequence dependent setup times. *Journal of Operations Research Society of Japan*, **29**, 206–219.

[31] Hall, N. G., Laporte, G., Selvarajah, E., and Srikandarajah, C., 2003. Scheduling and lot streaming in flow shops with no-wait in process. *Journal of Scheduling*, **6**, 339–354.

[32] Jacobs, F. and Bragg, D., 1988. Repetitive lots: flow time reductions through sequencing and dynamic batch sizing. *Decision Science*, **19**, 281–294.

[33] Jeong, H., Park, J., and Leachman, R., 1999. A batch splitting method for a job shop scheduling problem in an mrp environment. *International Journal of Production Research*, **37**, 3583–3598.

[34] Kacem, I., 2003. Genetic algorithm for the flexible jobshop scheduling problem. In the Proceeding of the IEEE International Conference on Systems, Man, and Cybernetics. Washington, DC, pp. 3464–6469.

[35] Kochhar, S. and Morris, R., 1987. Heuristic methods for flexible flow line scheduling. *Journal of Manufacturing Systems*, **6**, 299–314.

[36] Kumar, S., Bagchi, T., and Sriskandarajah, C., 2000. Lot streaming and scheduling heuristics for m-machine no-wait flow shop. *Computers and Industrial Engineering*, **38**, 149–172.

[37] Liu, S. C., 2003. A heuristic method for discrete lot streaming with variable sublots in a flow shop. *International Journal of Advanced Manufacturing Technology*, **22**, 662–668.

[38] Low, C., Hsu, C., and Huang, K., 2004. Benefits of lot splitting in job-shop scheduling. *International Journal of Advanced Manufacturing Technology*, **24**, 773–780.

[39] Manikas, A. and Chang, Y., 2008. Multi-criteria sequence-dependent job shop scheduling using genetic algorithms. *Computers & Industrial Engineering*, , In press.

[40] Marimuthu, S., Ponnambalam, S. G., and Jawahar, A. N., 2008. Evolutionary algorithms for scheduling m-machine flow shop with lot streaming. *Robotics and Computer-Integrated Manufacturing*, **24**, 125–139.

[41] Martin, C. H., 2006. A hybrid genetic algorithm/mathematical programming approach to the multi-family flowshop scheduling problem with lot streaming. *OMEGA International Journal of Management Sciences*, doi: 10.1016/j.omega.2006.11.002.

[42] Nowostawski, M. and Poli, R., 1999. Parallel genetic algorithm taxonomy. Proceedings of the Third International Conference on Knowledge-Based Intelligent Information Engineering System. Adelaide, Australia, pp. 88–92.

[43] Osman, I. and Potts, C., 1989. Simulated annealing for permutation flow-shop scheduling. *Omega*, **17**, 551–557.

[44] Panwalkar, S. S., Dudek, R. A., and Smith, M. L., 1973. Sequencing research and the industrial scheduling problem. In S. E. Elmaghraby (Ed.), Symposium on the theory of scheduling and its applications. Springer-Verlag, p. 29.

[45] Pezzella, F., Morganti, G., and Ciaschetti, G., 2008. A genetic algorithm for the flexible job-shop scheduling problem. *Computers and Operations Research*, **35**, 3202–3212.

[46] Potts, C. and Baker, K., 1989. Flow shop scheduling with lot streaming. *Operations Research Letter*, **8**, 297–303.

[47] Reeves, C. R., 1995. A genetic algorithm for flowshop sequencing. *Computers and Operations Research*, **22**, 5–13.

[48] Reiter, S., 1966. A system for managing job shop production. *Journal of Business*, **34**, 371–393.

[49] Rios-Mercado, R. and Bard, J., 1999. A branch-and-bound algorithm for permutation flow shops with sequence-dependent setup times. *IIE Transactions*, **31**, 721–731.

[50] Ruiz, R., Şerifoglub, F. S., and Urlings, T., 2008. Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, **35**, 1151–1175.

[51] Ruiz, R., Maroto, C., and Alcaraz, J., 2005. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research*, **165**, 34–54.

[52] Saidi, M. and Fattahi, P., 2007. Flexible job shop scheduling with tabu search algorithm. *International Journal of Advanced Manufacturing Technology*, **35**, 563–570.

[53] Smunt, T., Buss, A., and Kropp, D., 1996. Lot splitting in stochastic flow shop and job shop environments. *Decision Science*, **27**, 215–238.

[54] Tseng, C. T. and Liao, C. J., 2007. A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. *European Journal of Operational Research*, doi: 10.1016/j.ejor.2007.08.030.

[55] Vickson, R., 1995. Optimal lot streaming for multiple products in a two-machine flow shop. *European Journal of Operations Research*, **85**, 556–575.

[56] Xing, L., Chen, Y., and Yang, K., 2008. Multi-objective flexible job shop schedule: Design and evaluation by simulation modeling. *Applied Soft Computing*, **In Press (doi:10.1016/j.asoc.2008.04.013)**.