A simulated annealing with multiple-search paths and parallel computation for a comprehensive flowshop scheduling problem

Published in 2015 in the International Transactions in Operational Research Vol. 22, 669-691

A Simulated Annealing with Multiple Search Paths and Parallel Computation for a Comprehensive Flowshop Scheduling Problem

Fantahun M. Defersha[†]

School of Engineering, University of Guelph, 50 Stone Road East,
Guelph, Ontario, Canada, N1G 2W1

Abstract

Recent studies have demonstrated that the performance of a simulated annealing algorithm can be improved by following multiple search paths and parallel computation. In this paper, we use these strategies to solve a comprehensive mathematical model for a flexible flowshop lot streaming problem. In flexible flowshop environment, a number of jobs will be processed in several consecutive production stages, and each stage may involve a certain number of parallel machines that may not be identical. Each job is to be split into several unequal sublots by following the concept of lot streaming. The sublots are to be processed in the order of the stages, and sublots of certain products may skip some stages. This complex problem also incorporates sequence-dependent setup times, the anticipatory or non-anticipatory nature of setups, release dates for machines, and machine eligibility. Numerical examples are presented to demonstrate the effectiveness of lot streaming in hybrid flowshops, the performance of the proposed simulated annealing algorithm and the improvements achieved using parallel computation.

Keywords: *Hybrid flexible flowshop; Lot streaming; Simulated Annealing; Multiple Search Paths; Parallel Computation.*

1. Introduction

Simulated annealing (SA) is a general-purpose optimization technique capable of finding optimal or near-optimal solutions in various applications. As pointed out in Kratica *et al.* (2001), SA is able to converge to an optimum value. However, the basic SA is also limited because the cost incurred in solving many complex optimization problems can be very high in terms of computational time. Numerous efforts have therefore been attempted to make the technique practical for solving larger problems by shortening execution time. Lee and Lee (1996) classified these efforts into two major categories: algorithmic optimization (of

[†]For correspondence: fdefersh@uoguelph.ca, Tel: (519) 824-4120 Ext. 56512; Fax: (519) 836-0227

sequential SA) and parallel computing. In the category of optimizing a sequential SA, typical examples are careful perturbation (Deutsch and Wen, 1998) and adaptive cooling rate (Youhua and Weili, 1996; Azizi and Zolfaghari, 2004). However, Lee and Lee (1996) pointed out that these techniques do not significantly improve the quality of the solution or execution time. On the other hand, the advent of parallel computing has made it possible to achieve speed-up by orders of magnitude in various applications (Lee and Lee, 1996). Recent studies have also demonstrated that simulated annealing algorithms with multiple search paths and parallel computation have superior search performances compared to those with single search path and sequential computation (Lee and Lee, 1996; Czech *et al.*, 2009). As shown in Azencott (1992b), the probability of an error that is, the probability of not getting an optimal solution when performing a multiple search path SA (MSPSA) decreases exponentially as the number of search paths increases in return to a linearly increasing computational time. The use of parallel computing, then, is an effective technique to achieve an exponential reduction in the error probability of an MSPSA while at the same time reducing computational time. This can be done by dividing the search paths into groups and allocating them to concurrently available processors. Parallelizing the MSPSA may also improve the quality of the solution without affecting computational time. This can be accomplished by letting multiple copies of the MSPSA with the same number of search paths run on several concurrently available processors.

In this paper we present a multiple-search path parallel simulated annealing for solving a flexible flowshop scheduling problem with lot streaming. Lot streaming is a technique that splits a given job into sublots, each consisting of identical items. This allows the overlapping of successive operations in multi-stage manufacturing systems thereby reducing production makespan. It is used to implement the time-based strategy in today's competitive global manufacturing environment(Chang and Chiu, 2005). In fact, many world-class manufacturing companies such as Dell and Toyota have adopted this strategy to quickly produce goods and deliver them to their customers (Blackburn, 1991; Bockerstette and Shell, 1993). The concept of lot streaming was formally introduced in Reiter (1966) and its practice in solving flowshop scheduling problems is not new. Many previous studies have considered flowshop lot streaming in two stages or special cases of lot streaming in three stages. These include Potts and Baker (1989), Vickson and Alfredsson (1992), Trietsch and Baker (1993), Baker and Jia (1993), Glass *et al.* (1994), Baker (1995), Chen and Steiner (1996), Sen *et al.* (1998), and Sriskandarajah and Wagneur (1999). Lot streaming in flow-

shops with more than three stages has been studied recently by several researchers (see for example Kumar *et al.*, 2000; Hall *et al.*, 2003; Liu, 2003; Chiu *et al.*, 2004; Bukchin and Masin, 2004; Martin, 2009; Biskup and Feldmann, 2006; Tseng and Liao, 2008; Feldmann and Biskup, 2008; Marimuthu *et al.*, 2008).

The studies mentioned above usually assume that a single machine is being used in each stage. However, so-called hybrid flowshops, where there are parallel machines at certain stages, are very common. If one stage poses a bottleneck in the production process, managers may consider investing in additional machines for that stage. Moreover, hybrid flowshops have important applications in flexible manufacturing systems (FMS); in the manufacturing of electronics and furniture; and in the food processing, petrochemical, and pharmaceutical industries (Xiao *et al.*, 2000; Tang *et al.*, 2005, 2006; Zandieh *et al.*, 2006; Jungwattanakit *et al.*, 2008). To this end, the issue of hybrid flowshop scheduling has attracted much interest and many scientific papers have been written dealing with the diverse aspects of the problem (see for example Botta-Genoulaz, 2000; Bertel and Billaut, 2004; Oğuzc *et al.*, 2004; Tang *et al.*, 2005; Zhang *et al.*, 2005; Tang *et al.*, 2006; Ruiz and Maroto, 2006; Zandieh *et al.*, 2006; Ying and Lin, 2006; Jungwattanakit *et al.*, 2008; Janiak *et al.*, 2007; Ruiz *et al.*, 2008). However, the issue of lot streaming in hybrid flowshop scheduling has received much less attention than lot streaming in pure flowshop scheduling. For instance, in a sizable review of 225 papers on hybrid flowshop scheduling done by Ruis *et al.* (2010), lot streaming was addressed in only two papers, namely Zhang *et al.* (2005) and Liu (2008). However, these two cases, Zhang *et al.* (2005) and Liu (2008), apply to a very special situation where there are parallel machines only in the first stage and the number of stages is limited to two. In an attempt to bridge the gap between research efforts in flowshop lot streaming and hybrid flowshop scheduling, Defersha (2011) developed a comprehensive mathematical model for the general hybrid flexible flowshop (HFFS) lot streaming problem. The author demonstrated that lot streaming can be more effective in hybrid flowshops than in pure flowshops, even though there is far more research on the latter than the former.

In order to further bridge this research gap, in this paper we develop an SA algorithm with multiple search paths to efficiently solve the comprehensive model presented in Defersha (2011). The algorithm was implemented in both sequential and parallel computing platforms. The performance of the parallel SA is evaluated against a sequential SA. The results were very encouraging. The rest of this paper is organized as follows. In Section 2,

we present the considered mixed integer linear programming model for HFFS lot streaming. In Section 3 we provide the basics of a single and multiple search path SA and a technique to parallelize the multiple search path SA. Details of the parallel SA method are presented in Section 4. Numerical examples are given in Section 5. Discussion and conclusion are in Section 6.

2. Mathematical Formulation

The main objective of this paper is to present an efficient SA algorithm that will solve a comprehensive mathematical model proposed by Defersha (2011) for an HFFS lot streaming problem. The model incorporates lot streaming and other practical issues, including the following: (1) the existence of parallel machines that may not be identical; (2) the possibility that certain jobs will skip certain stages; (3) sequence-dependent setup times; (4) the anticipatory or non-anticipatory nature of setups; (5) release dates of machines; and (6) machine eligibility. The problem and mathematical model are further described below to improve readers comprehension of this paper.

2.1. Problem Description and Assumptions

Consider a flowshop consisting of several consecutive production stages that process several jobs. Each stage has a known number of parallel machines that may not be identical. Each job is to be split into several unequal consistent sublots. The sublots are to be processed in the order of the stages, and sublots of certain products may skip some stages. At a given stage, a sublot of a job can be assigned to one of the parallel machines eligible to process that particular job. For each job there is a sequence-dependent setup time on each eligible machine, and this setup time may be anticipatory or non-anticipatory at different stages. Each machine can process at most one sublot at a time. Sublots of different products can be interleaved. The problem is to determine the size of each sublot of each job, and the assignment and processing sequence of these sublots on each machine in each stage. The objective function is to minimize the completion time of the last sublot to be processed in the system

2.2. Notations

Below we define the many notations required to mathematically model the problem presented above.

Indexes and Input Data:

I Number of stages where stages are indexed by i or $l = 1, 2, ..., I$,

M_i Number of machines in stage i where machines are indexed by m or $k = 1, 2, ..., M_i$,

N Number of jobs (products) where jobs are indexed by n or $p = 1, 2, ..., N$,

J_n Total number of sublots of job n where sublots are indexed by j or $s = 1, 2, ..., J_n$,

E_n A set of pairs of stages (l, i) for job n constrained by precedence relations, i.e, the processing of job n in stage l is followed by its processing in stage i,

$T_{n,m,i}$ Processing time for one unit of job n on machine m in stage i,

Q_n Batch size of job n,

$R_{m,i}$ Maximum number of production runs of machine m in stage i where production runs are indexed by r or $u = 1, 2, ..., R_{m,i}$.

$S_{m,i,n,p}$ Setup time on machine m in stage i for processing job n following the processing of job p on this machine; if $n = p$, the setup may be called minor setup,

$A_{n,i}$ A binary data equal to 1 if setup of job n in stage i is attached (non-anticipatory), or 0 if this setup is detached setup (anticipatory),

$B_{n,i}$ A binary data equal to 1 if job n needs processing in stage i, otherwise 0,

$D_{n,m,i}$ A binary data equal to 1 if job n can be processed on machine m in stage i, otherwise 0; $D_{n,m,i} \leq B_{n,i}$,

$F_{m,i}$ The release date of machine m in stage i,

Ω Large positive number.

Variables:

Continuous Variables:

$c_{j,n,i}$ Completion time of the j^{th} sublot of job n from stage i,

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

$\lambda_{j,n}$ Size of the j^{th} sublot of job n,

c_{max} Makespan of the schedule,

Binary Variables:

$x_{r,m,i,j,n}$ Binary variable which takes the value 1 if the r^{th} run on machine m in stage i is for the j^{th} sublot of job n, 0 otherwise,

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

In the list of notations above, a production run r is the possessing of a sublot by a machine. The maximum number of production runs, $R_{m,i}$, on a particular machine m, in a particular stage i, is theoretically equal to the total number of sublots of all the jobs. However, because parallel machines are present and certain jobs will skip stages, the number of sublots that will actually be processed on a particular machine at an optimal solution is likely lower than the total number of sublots. Hence, a reasonably lower value for $R_{m,i}$ has to be assumed in order to reduce the number of variables in the mathematical model.

2.3. MILP Model

Using the notation given above, the objective function and the constraints of the considered mixed integer linear programming (MILP) model are presented below.

Minimize:

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

Subject to:

$$\widehat{c}_{r,m,i} \geq c_{j,n,i} + \Omega \cdot x_{r,m,i,j,n} - \Omega \; ; \quad \forall (r, m, i, j, n) \tag{2}$$

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

$$\widehat{c}_{1,m,i} - \lambda_{j,n} \cdot T_{n,m,i} - S_{m,i,n,0} - \Omega \cdot x_{1,m,i,j,n} + \Omega \geq F_{m,i} \; ; \forall (m, i, j, n) \tag{4}$$

$$\widehat{c}_{r,m,i} - \lambda_{j,n} \cdot T_{n,m,i} - S_{m,i,n,p} - \Omega \left\{ \left(\sum_{s=1}^{J_p} x_{r-1,m,i,s,p} \right) + x_{r,m,i,j,n} \right\} + 2\Omega \geq \widehat{c}_{r-1,m,i} \ ;$$

$$\forall (r, m, i, j, n, p) | r > 1 \tag{5}$$

$$\widehat{c}_{1,m,i} - \lambda_{j,n} \cdot T_{n,m,i} - S_{m,i,n,0} \cdot A_{n,i} - \Omega \cdot (x_{u,k,l,j,n} + x_{1,m,i,j,n}) + 2\Omega \geq \widehat{c}_{u,k,l} \ ;$$

$$\forall (u, k, m, l, i, j, n) | (l, i) \in E_n \tag{6}$$

$$\widehat{c}_{r,m,i} - \lambda_{j,n} \cdot T_{n,m,i} - S_{m,i,n,p} \cdot A_{n,i} - \Omega \left\{ \left(\sum_{s=1}^{J_p} x_{r-1,m,i,s,p} \right) + x_{u,k,l,j,n} + x_{r,m,i,j,n} \right\} + 3\Omega \geq \widehat{c}_{u,k,l} \ ;$$

$$\forall (u, r, k, m, l, i, j, n, p) | (l, i) \in E_n, \ r > 1 \tag{7}$$

$$\sum_{n=1}^{N} \sum_{j=1}^{J_n} x_{r,m,i,j,n} \leq 1 \ ; \quad \forall (r, m, i) \tag{8}$$

$$\sum_{n=1}^{N} \sum_{j=1}^{J_n} x_{r+1,m,i,j,n} \leq \sum_{n=1}^{N} \sum_{j=1}^{J_n} x_{r,m,i,j,n} \ ; \quad \forall (r, m, i) | r < R_{m,i} \tag{9}$$

$$\sum_{j=1}^{J_n} \lambda_{j,n} = Q_n \ ; \quad \forall n \tag{10}$$

$$\lambda_{j,n} \leq \Omega \cdot \gamma_{j,n} \ ; \quad \forall (j, n) \tag{11}$$

$$\gamma_{j,n} \leq \lambda_{j,n} \ ; \quad \forall (j, n) \tag{12}$$

$$\sum_{m=1}^{M_i} \sum_{r=1}^{R_{m,i}} x_{r,m,i,j,n} = \gamma_{j,n} \cdot B_{n,i} \ ; \quad \forall (i, j, n) \tag{13}$$

$$x_{r,m,i,j,n} \leq D_{n,m,i} \ ; \quad \forall (r, m, i, j, n) \tag{14}$$

$$c_{max} \geq c_{j,n,i} \ ; \quad \forall (j, n, i) \tag{15}$$

$$x_{r,m,i,j,n} \text{ and } \gamma_{j,n} \text{ are binary} \tag{16}$$

The objective function in Eq. (1) is to minimize the makespan of the schedule which is equal to the completion time of the last sublot processed in the system. The constraints in Eqs. (2) and (3) together state that the completion time of the j^{th} sublot of job n in stage i is equal to the completion time of the r^{th} run of machine m in stage i if this production run is assigned to that particular sublot. The starting time of the setup for the first run ($r = 1$) of machine m in stage i is given by $\widehat{c}_{1,m,i} - \lambda_{j,n} \times T_{n,m,i} - S_{m,i,n,0}$ if the j^{th} sublot of job n is assigned to this first run. This starting time cannot be less than the release date of the machine as enforced by the constraint in Eq. (4). The constraint in Eq. (5) 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. (6) states that for any pair of stages $(l, i) \in E_n$, the setup or the actual processing of the first run on machine m in stage i may not be started before the completion time of run u of machine k in stage l, depending on whether the setup of product type n in stage i is non-anticipatory or anticipatory. This constraint is applied if run u of machine k in stage l and that of the first run of machine m in stage i are both assigned to sublot j of job n. The constraint in Eq. (7) is similar to that in Eq. (6) except Eq. (7) is for run $r > 1$ of machine m in stage i. In this case, the sequence dependent setup time has to be considered by taking into account the type of the job that was processed in run $r - 1$ of machine m in stage i. The constraint in Eq. (8) states that a production run r of a particular machine m at a particular stage i can be assigned to at most one sublot. The constraint in Eq. (9) is to enforce the logic that a production run $r + 1$ of a given machine can be assigned to a sublot if and only if run r of that machine is already assigned to another sublot. The constraint in Eq. (10) enforces that the sum of the sizes of the sublots of a given job should be equal to the lot size of that particular job. The constraints in Eqs. (11) and (12) are to set the binary variable $\gamma_{j,n}$ to a value equal to 1 or zero depending on whether the size of j^{th} sublot of job n is positive or zero, respectively. If sublot j of job n is positive (i.e., $\gamma_{j,n} = 1$) and it requires processing in stage i, then it should be assigned to one of the eligible machine in stage i. This is enforced by the constraints in Eqs. (13) and (14). Eq. (15) states that the makespan of the schedule, c_{max}, is greater or equal to the completion time of any sublot on any stage. At optimality, c_{max} takes the value of the completion time of the last sublot to be processed in the system. Eq. (16) is the integrality requirement.

3. Simulated Annealing

3.1. The Basic Algorithm

A simulated annealing is a stochastic search algorithm where a point X in the search space is analogous to a state of some physical system. A function $E(X)$ is defined as the internal energy of that system at state X. The goal of the search is to bring the system from an arbitrary initial state X_0 to a state where the system will be at its minimum possible energy. In doing so, the algorithm visits a sequence of random points X_0, X_1, $\cdots$, X_n of the search space such that $E(X_n)$ is the minimum possible energy of the system as $n \to \infty$. This sequence of visitation is determined by Eq. (17) where X'_n is a neighborhood point generated by slightly moving away from X_n.

$$x_{n+1} = \begin{cases} X'_n & \text{if } E(X'_n) \leq E(X_n) \\ X'_n & \text{if } \exp\left(\frac{E(X_n)-E(X'_n)}{T_n}\right) > \text{rand}() \\ X_n & \text{otherwise} \end{cases} \tag{17}$$

In the above equation, rand() is a random number generated for making a stochastic decision for a new solution. T_n is the *temperature* at the n^{th} iteration. The sequence of T_n, the *cooling schedule*, is generated such that $T_n \geq T_{n+1}$ with $\lim_{n\to\infty} T_n = 0$. From Eq. (17), it can be seen that the choice of X_{n+1} depends only on the current solution x_n, not on the previously visited solutions. Thus, the search path of such simulated annealing algorithms follows a Markov chain. From here onwards we refer to the basic algorithm as a sequential single search path simulated annealing algorithm (SSSP-SA). For a suitable cooling schedule, the probability of not getting an optimal solution after n iterations using an SSSP-SA is characterized by Eq. (18) where U_{min} is a set of optimal solution points (Chiang and Chow, 1988). In this equation, n is large enough and $K > 0$ and $\theta > 0$ are constants specific to a given energy landscape and neighborhood generation. Thus as $n \to \infty$, the solution converges to one of the optimal points in U_{min} with probability of 1.0.

$$P(X_n \notin U_{min}) \sim \left(\frac{K}{n}\right)^{\theta} \tag{18}$$

3.2. Sequential Multiple Search Paths SA

Most SA schemes in the literature follow a single search path. However, from the point of view of performance, following a single search path may not be necessary or advisable (Lee

and Lee, 1996). A sequential multiple search path SA (SMSP-SA) performs S independent versions of simulated annealing algorithms using the same search space, neighborhood generation and cooling schedule. Each one of these independent versions stops after n iterations to provide S independent terminal solutions $\{X_{n,1}, X_{n,2}, \cdots, X_{n,S}\}$. The best and final solution X_n is then selected from these terminal solutions. A general pseudocode of an SMSP-SA is given in Figure 1. The characteristic equation for the probability that this algorithm will miss the optimal solution can be derived from Eq. (18) and is given by Eq. (19). Thus, from Eq. (19), for $0 < K/n < 1$, it can be seen that the probability of error decreases exponentially as S increases while computational time grows only linearly (Azencott, 1992b). This suggests that performing multiple short simulated annealing runs will yield a better solution than than performing a long single annealing run.

$$P(X_N \notin U_{min}) = \prod_{i=1}^{S} P(X_{n,i} \notin U_{min}) \sim \left(\frac{K}{N}\right)^{\theta S} < \left(\frac{K}{N}\right)^{\theta} \tag{19}$$

3.3. Parallelizing the SA

Parallel computing is a promising technique that can either reduce the computational time needed to solve a sequential multiple search path simulated annealing (SMSP-SA), or exponentially decrease its probability of error, depending on the way the SA is parallelized. In order to reduce computational time, let S search paths be partitioned into equal sub-groups as $S_1, S_2, \cdots, S_p$ and distributed to p concurrently available processors. The computational time for the parallel multiple search path SA (PMSP-SA), t_p, will be equal to t_s/p where t_s is the computational time for the SMSP-SA.

On the other hand, we may parallelize an SMSP-SA to improve the quality of the solution without significantly affecting computational time. In this case, let the SMSP-SA have S search paths that require t_s unit times to perform n iterations in each search path. By running multiple copies of the SMSP-SA on several concurrently available computers, the value of S can be increased many times while t_s and n stay the same. The greater the number of search paths, the smaller the probability that the SA will miss the optimal solution. A good introduction on parallel simulated annealing with multiple search paths can be found in Azencott (1992a), Lee and Lee (1996), and Meise (1998), although the authors assumed only a single search path per processor. In our parallel simulated annealing, however, several shorter search paths will be traced on each processor since the exponential reduction in error probability suggests that multiple short SA runs are better than a single

```
SMSP-SA()    //Sequential Multiple Search Path SA
{
    Randomly generate S solution points {X_{0,1}, X_{0,2}, ..., X_{0,S}}
    Set initial temperature T_0
    Set n = 0, r = 0;
    REPEAT
    {
        FOR( q = 1 to Q )    //Q = number of iterations at temperature T_r
        {
            FOR j = 1 to S    //S = number of independent search paths
            {
                Generate X'_{n,j} from X_{n,j}

                IF E(X'_{n,j}) ≤ E(X_{n,j})
                    X_{n+1,j} = X'_{n,j}
                ELSE IF exp((E(X_{n,j}) - E(X'_{n,j}))/T_r) > ρ
                    X_{n+1,j} = X'_{n,j}
                ELSE
                    X_{n+1,j} = X_{n,j}
            }
            n = n + 1
        }
        r = r + 1
        T_r = α × T_{r-1}
    }
    UNTIL a stoping criterion is met
    X_n = X_{n,j*} such that E(X_{n,j*}) ≤ E(X_{n,j}) for j = 1 to S
}
```

Figure 1: Pseudocode for an instance of a SMSP-SA

long run.

4. Components of the Proposed Algorithm

4.1. Solution Representation

A simple permutation of the jobs in an array constitutes the most widely used encoding of a solution for pure flowshop scheduling. Such simple encoding has also been adopted for hybrid flowshop scheduling. The permutation of the jobs is used to guide the assignment and sequencing of the jobs at every stage through some form of heuristics or priority rules. When lot streaming is considered, we need additional components in the solution

representation to encode the number and sizes of the sublots of each job. Moreover, jobs of the same product will appear on several places of the permutation representing the sublots of this product. Figure 2 illustrates the solution encoding of the proposed SA algorithm assuming four jobs where jobs 1 and 4 have two sublots each and jobs 2 and 3 have three sublots each. The left-hand-side segment of the solution encoding, labeled LHS-Segment, encodes the size of the sublots of each job. In this part of the solution representation, the element $\alpha_{j,n}$ takes a random value in the interval [0, 1]. For a solution under consideration the size of the j^{th} sublot of job n is computed (decoded) using Eq. (20). The right-hand-side segment of the solution representation, labeled RHS-Segment, represents a permutation π of S sublots where S is the total number of sublots of all the jobs given by $\sum_n^N J_n$ and $\pi(s)$ is a pair of numeric values at the s^{th} location of the permutation denoting a sublot at that location. In Figure 2 for example, $\pi(1)$ denotes sublot 2 of job 3 whereas $\pi(2)$ denotes sublot 1 of job 4 and so on.

$$\lambda_{j,n} = \frac{\alpha_{j,n}}{\sum_{j=1}^{J_n} \alpha_{j,n}} \times Q_n \tag{20}$$

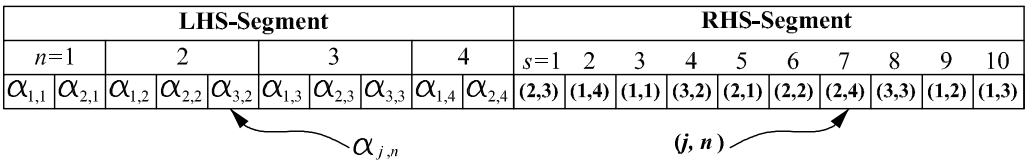


Figure 2: Solution representation

4.2. Energy Function

In the proposed SA, a point X in the search space is a solution encoded as shown in Figure 2 and the value of the corresponding energy function $E(X)$ is equal to the makespan c_{max}. Its value corresponding to a particular solution X can be computed by assigning and sequencing the sublots on each machine in each stage using the information decoded from that solution. In order to perform this evaluation process, the size of each sublot is first decoded from the LHS-Segment of X using Eq. (20). Once the size of the sublots is obtained, the permutation π of the sublots taken from the RHS-Segment of X will be used to guide the assignment and sequencing of the sublots on the parallel machines in each stage. By considering each sublot as a job on its own, the assignment and sequencing of sublots can be performed in a similar way as for hybrid flowshop scheduling without lot

streaming. Indeed, to represent this process, we adopt the assignment rule that appears in Ruiz and Maroto (2006) where the author did not consider lot streaming. This assignment rule requires that we go through all the stages required by each sublot in π and look for the machine that can finish this sublot at the earliest time. The rule has been adjusted to take the following into account: (1) attached and detached setup time; (2) machine release date; (3) skipping of stages by certain jobs; and (4) the possibility that the size of certain sublots may be zero. If a sublot is of zero size, it should not be assigned to any machine in any stage. Considering these adjustments, if sublot j of job n is assigned to an eligible machine m in stage i, its completion time $c_{j,n,i}$ in this stage should be computed based on one of the following four different cases. The complete process of assigning the operations of the various sublots and the determination of the corresponding makespan c_{max} is presented in the flowchart given in Figure 3.

Case 1 a) Sublot j of job n is the first sublot to be assigned on machine m.

b) For job n, stage i is the first stage to be visited.

- In this case, the machine is available after its release date, $F_{m,i}$.
- Setup can commence immediately after $F_{m,i}$ and the processing of the sublot can begin after the setup is completed.
- Thus, $c_{j,n,i} = F_{m,i} + S_{m,i,n,0} + \lambda_{j,n} \cdot T_{n,m,i}$.

Case 2 a) Sublot j of job n is not the first sublot to be assigned on machine m.

b) Sublot x of job p was the last sublot assigned on machine m.

c) For job n, stage i is the first stage to be visited.

- In this case, the machine is available after $c_{x,p,i}$ (the completion time of sublot x of job p).
- Setup for sublot j of job n can commence immediately after $c_{x,p,i}$ and the processing of the sublot can begin after the setup is completed.
- Thus, $c_{j,n,i} = c_{x,p,i} + S_{m,i,n,p} + \lambda_{j,n} \cdot T_{n,m,i}$.

Case 3 a) Sublot j of job n is the first sublot to be assigned on machine m.

b) Job n has to visit stage l immediately before it visits stage i, i.e., $(l, i) \in E_n$

- Similar to Case 1, in this case as well the machine is available after its release date, $F_{m,i}$.
- Setup for sublot j of job n can commence immediately after $F_{m,i}$ if it is detached (i.e., $A_{n,i} = 0$) or at $\max\{F_{m,i};\ c_{j,n,l}\}$ if it is attached (i.e., $A_{n,i} = 1$).
- The processing of the sublot j of job n can begin at $[\max\{F_{m,i};\ c_{j,n,l}\} + S_{m,i,n,0}]$ if its setup is attached or at $\max\{F_{m,i} + S_{m,i,n,0};\ \ c_{j,n,l}\}$if its setup is detached.
- Thus, $c_{j,n,i} = \max\{F_{m,i} + (1 - A_{n,i}) \cdot S_{m,i,n,0};\quad c_{j,n,l}\} + A_{n,i} \cdot S_{m,i,n,0} + \lambda_{j,n} \cdot T_{n,m,i}$.

Case 4 a) Sublot j of job n is not the first sublot to be assigned on machine m.

b) Sublot x of job p was the last sublot assigned on machine m.

c) Job n has to visit stage l immediately before it visits stage i, i.e., $(l, i) \in E_n$

- Similar to Case 2, in this case the machine is available after $c_{x,p,i}$ (the completion time of sublot x of job p).
- Setup for sublot j of job n can commence immediately after $c_{x,p,i}$ if it is detached or at $\max\{c_{x,p,i};\ c_{j,n,l}\}$ if it is attached.
- The processing of the sublot j of job n can begin at $[\max\{c_{x,p,i};\ c_{j,n,l}\} + S_{m,i,n,p}]$ if its setup is attached or at $\max\{c_{x,p,i} + S_{m,i,n,p};\ \ c_{j,n,l}\}$ if its setup is detached.
- Thus, $c_{j,n,i} = \max\{c_{x,p,i} + (1 - A_{n,i}) \cdot S_{m,i,n,p};\quad c_{j,n,l}\} + \lambda_{j,n} \cdot T_{n,m,i} + A_{n,i} \cdot S_{m,i,n,p}$.

4.3. Move Operators

At each iteration, several move operators are applied on each solution with small probabilities. In the proposed SA algorithm, we used four move operators: (i) Sublot Size Step Move (SSStM); (ii) Sublot Size Swap Move (SSSwM); (iii) Sublot Order Shift Move (SOShM); and (iv) Sublot Order Swap Move (SOSwM). The operator SSStM is applied with a small probability σ_1 on each element $\alpha_{j,n}$ in the LHS-Segment to step up or down the value of this element with a step amount θ using the equation $\alpha_{j,n} = \min\{1, \alpha_{j,n} + \theta\}$ or $\max\{0, \alpha_{j,n} - \theta\}$, respectively. The step amount θ is calculated every time this operator is applied to a given $\alpha_{j,n}$ with the equation $\theta = \theta_{\max} \times \text{rand}()$, where a $\theta_{\max} \in [0, 1]$ is a parameter of the algorithm and rand() is a random number generator in (0,1). The operator SSSwM is applied with a small probability σ_2 on each n in the LHS-Segment to swap the values of two arbitrarily selected $\alpha_{j,n}$ and $\alpha_{j',n}$. The perturbation operator SOShM

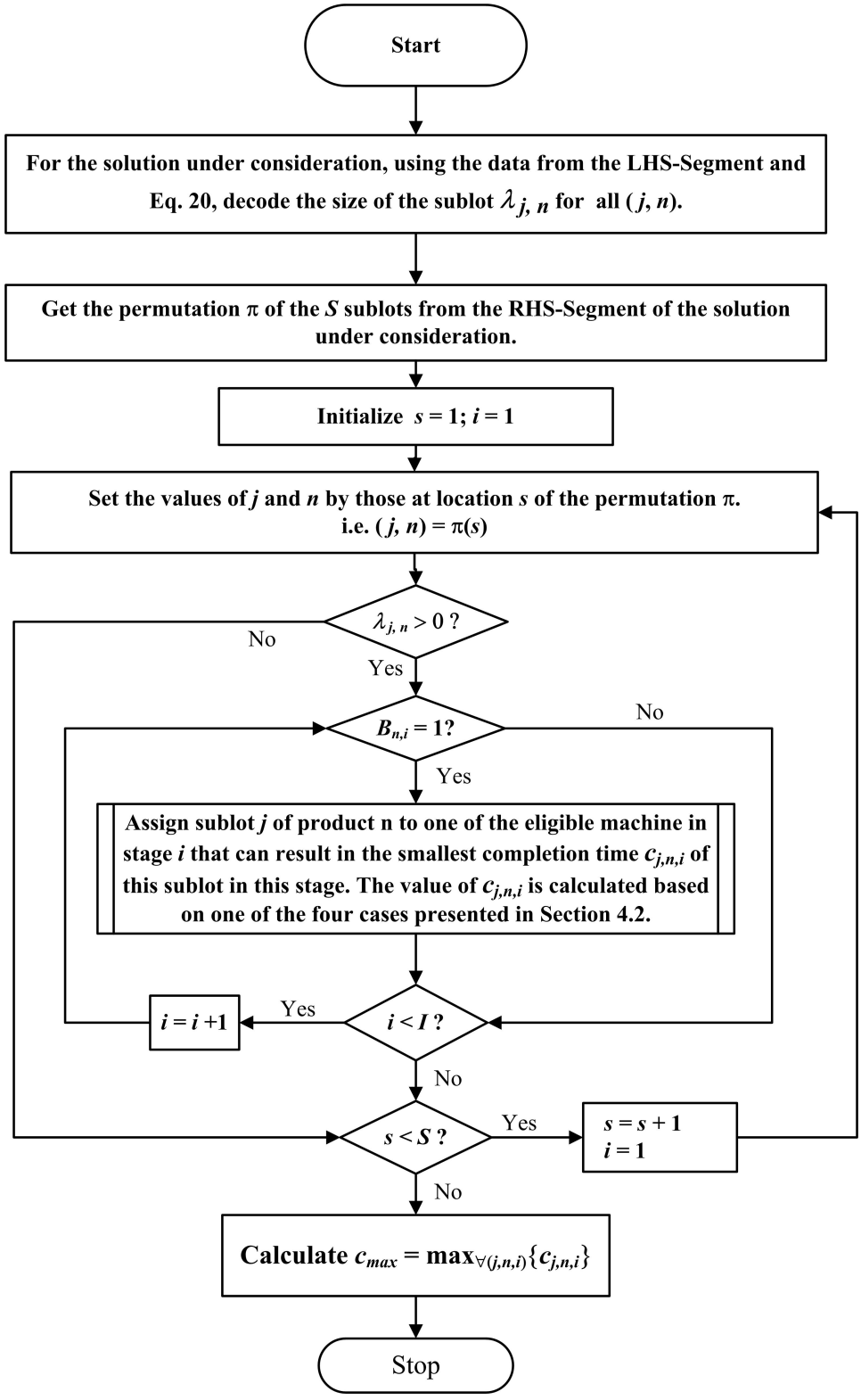


Figure 3: The process of assigning and sequencing the sublots on each machine in each stage and the determination of the makespan corresponding to a particular solution.

arbitrarily selects a sublot in the RHS-Segment and relocates the sublot to another arbitrary location in the RHS-Segment. This perturbation operator has a small probability σ_3. The operator SOSwM arbitrarily selects two sublots in the RHS-Segment and swaps these sublots with a small probability σ_4.

4.4. Cooling Schedule

The cooling schedule can be defined by the rules that determine three factors: (a) how high the starting temperature should be; (b) when the current temperature should be lowered; and (c) by how much the temperature should be lowered.. In this paper we use a popular cooling schedule in which a specified number of iterations are performed at a constant temperature $T_r = \beta \times T_{r-1}$. In this cooling schedule, r is the index of temperature levels, and the constant β, in $[0, 1]$, is the cooling coefficient and generally set as close to 1. Thus, the parameters of the proposed simulated annealing method associated with the cooling schedule are the initial temperature T_0, the cooling exponent β, and the number of iterations at each temperature level.

4.5. Interactions of Search Paths

In the proposed simulated annealing algorithm, each of the P concurrently available processors (CPUs) follows S search paths.. These search paths are allowed to interact to improve performance. We track two levels of interactions: local and global. In the local interaction, after every L number of iterations, a CPU will restart all of its S search paths at the current temperature from the best solution it has found so far (where L is defined as the local interaction frequency). In the global interaction (across CPUs), one of the possessing elements is chosen to coordinate the communication among the CPUs. After every $L \times F$ iterations, this CPU, in addition to executing its S-search path SA, will gather the best solutions found so far by all the CPUs and determine a winner solution as the best of all (where F may be defined as global communication factor). It will then broadcast this winner solution to all the CPUs. In turn, each CPU will restart the S-search paths in its domain from the winner solution at the current level of temperature. If F is chosen to be a large value (say between 20 and 40), global communication happens less frequently. This infrequent communication between processors promotes exploitation of the search space within each processor and exploration of the entire search space across the processors.

4.6. Steps of the Parallel Simulated Annealing

In this section, we present the steps of the parallel simulated annealing algorithm. We also redefine some exiting notations and introduce new notations. These are explained below.

p Processor Index (processor ID), $p = 0,\ 2,\ ...,\ P-1$ where P is the number of concurrently available processors.

s Index of search paths, $s = 1,\ 2,\ ...,\ S$ where S is the number of search paths followed by each processor.

n Iteration counter, $n = 1,\ 2,\ ...,\ N$ where N is the maximum number of iterations in each search path.

$X_{n,s,p}$ The solution at the n^{th} iteration along the s^{th} search path in the p^{th} processor.

α Cooling schedule coefficient.

r Index for the temperature levels in the cooling schedule.

T_r Temperature at the r^{th} level, $T_r = \beta \times T_{r-1} = \beta^r \times T_0$.

Q Number of iterations to be performed in each search path at each temperature level.

L Number of iterations performed in each search path before a processor restarts all of its search paths (at the current level of temperature) from the best solution it has found so far.

F Global Communication frequency factor where product $F \times L$ defined as the number of iterations to be performed by each search path before communication is effected among the processors.

BS_p Best solution found so far in the p^{th} processor.

BS Best solution found so far by all the processors

rand() Random number generator. Each processor uses a different seed for the random number generator.

Using the above notations, the steps of the proposed parallel SA are outlined in the flowchart in Figure 4. These steps were coded in C++ with MPI message-passing library for communication. The code was tested in a parallel computation environment composed of several computing nodes, each containing eight processors (3.2 GHz, 4GB RAM).

5. Numerical Examples

5.1. Pure vs Hybrid flow-shop lot streaming

Most existing models and solution techniques in the literature about flowshop lot streaming apply to pure flowshop. However, lot streaming is more effective in hybrid flowshops than in pure flowshops because of the following two reasons. Firstly, hybrid flowshops can overlap the operation of sublots of a given job not only across stages but also within stages. Secondly, some stages in pure flowshop can easily pose bottlenecks that inhibit further splitting of certain jobs. Defersha (2011) illustrated this point in a paper that addressed a small problem where two jobs were processed in three-stage flowshop. In this section we further illustrate the greater effectiveness of lot streaming in a hybrid flowshop by considering several arbitrarily generated examples composed of five to 25 parts and four to 15 stages. The general features of these sample problems are given in Tables 1 and 2. Each test problem has four categories: a) pure flowshop without lot streaming; b) pure flowshop with lot streaming; c) hybrid flowshop without lot streaming; and d) hybrid flowshop with lot streaming. Table 3 provides the makespans of the schedules obtained for the four categories in every test case by using the proposed algorithm. The table shows that in all the test cases, the reduction of makespan through lot streaming was much greater in hybrid flowshop than in pure flowshop. The average makespan reduction of the ten test cases was 35% in hybrid flowshop, whereas it was only 20% in pure flowshop. Yet, contrary to these findings and their practical potential in several manufacturing settings, lot streaming research in hybrid flowshops is quite limited.

5.2. Computational Performance

In this section, we illustrate the performance of the proposed algorithm. We solved a small test problem by using CPLEX, a state-of-the-art optimization package and the developed algorithm. Figure 5 shows the convergence histories of CPLEX (version 12.1.0) and the SMSP-SA in solving a small test problem (Problem-1d). The figure shows that the makespan of the schedule generated by CPLEX was 1580 minutes after about 46 hours of

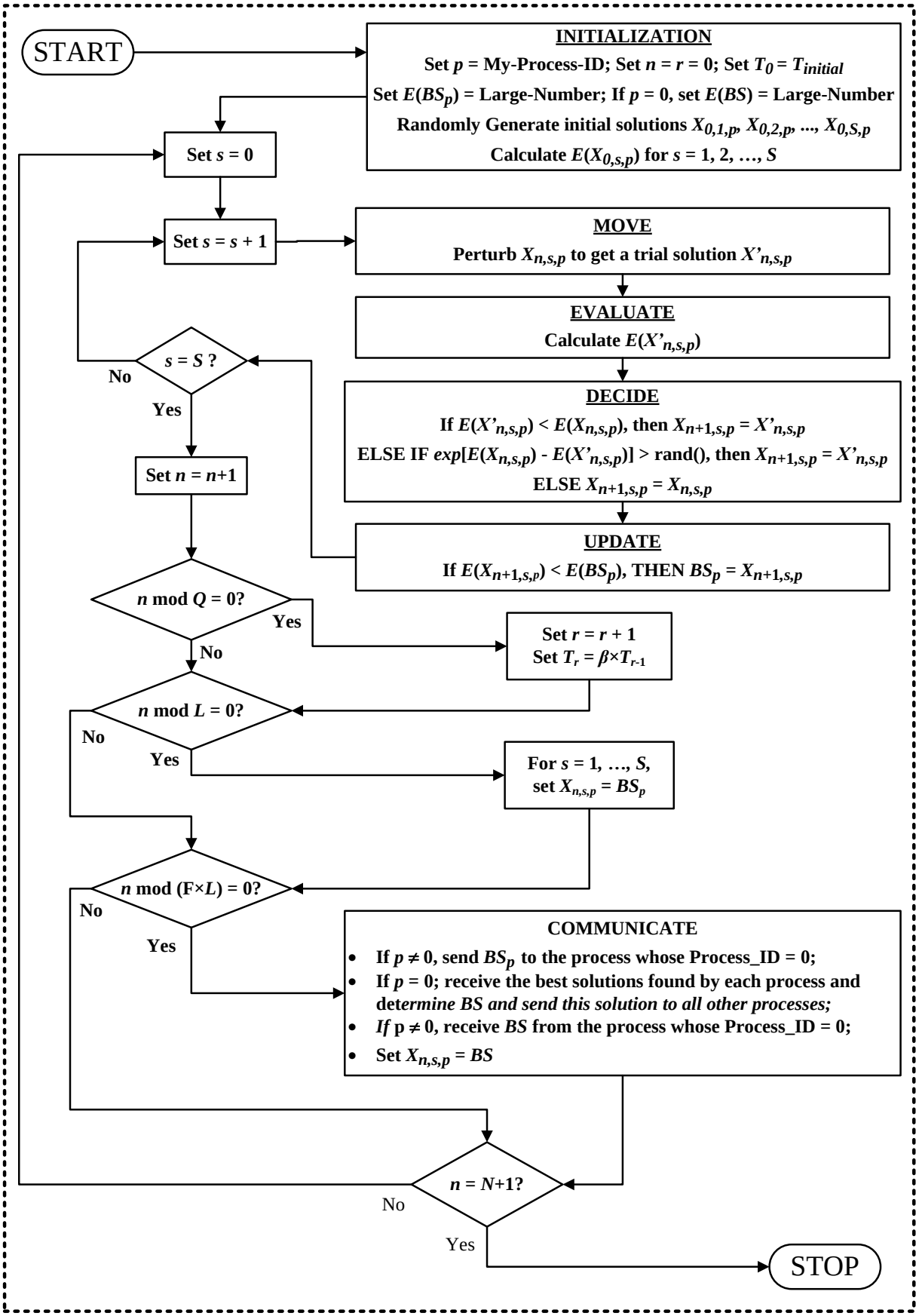


Figure 4: Steps of the algorithm.

Table 1: Problems Features (Problems 1a to 5d)

Features	Problem 1				Problem 2				Problem 3				Problem 4				Problem 5			
	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
NoOfProducts	5	5	5	5	8	8	8	8	10	10	10	10	12	12	12	12	14	14	14	14
MaxNoOfSublots	1	5	1	5	1	5	1	5	1	5	1	5	1	6	1	6	1	5	1	5
NoOfStages	4	4	4	4	5	5	5	5	6	6	6	6	8	8	8	8	8	8	8	8
MaxNoOfParaMach	1	1	3	3	1	1	3	3	1	1	3	3	1	1	4	4	1	1	4	4
MinNoOfSublots	1	5	1	5	1	5	1	5	1	5	1	5	1	6	1	6	1	5	1	5
MinNoOfParaMach	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2
MinimumLotsize	120	120	120	120	120	120	120	120	200	200	200	200	200	200	200	200	150	150	150	150
MaximumLotsize	340	340	340	340	340	340	340	340	400	400	400	400	450	450	450	450	380	380	380	380

Note: MaxNoOfParaMach = Maximum number of parallel machines. The number of machines across the stage varies between MaxNoOfParaMach and MinNoOfParaMach.

Table 2: Problems Features (Problems 6a to 10d)

Features	Problem 6				Problem 7				Problem 8				Problem 9				Problem 10			
	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
NoOfProducts	16	16	16	16	18	18	18	18	20	20	20	20	22	22	22	22	25	25	25	25
MaxNoOfSublots	1	5	1	5	1	6	1	6	1	8	1	8	1	6	1	6	1	5	1	5
NoOfStages	8	8	8	8	10	10	10	10	10	10	10	10	12	12	12	12	15	15	15	15
MaxNoOfParaMach	1	1	3	3	1	1	4	4	1	1	5	5	1	1	6	6	1	1	5	5
MinNoOfSublots	1	5	1	5	1	6	1	6	1	8	1	8	1	6	1	6	1	5	1	5
MinNoOfParaMach	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2
MinimumLotsize	300	300	300	300	400	400	400	400	400	400	400	400	300	300	300	300	100	100	100	100
MaximumLotsize	600	600	600	600	600	600	600	600	800	800	800	800	600	600	600	600	500	500	500	500

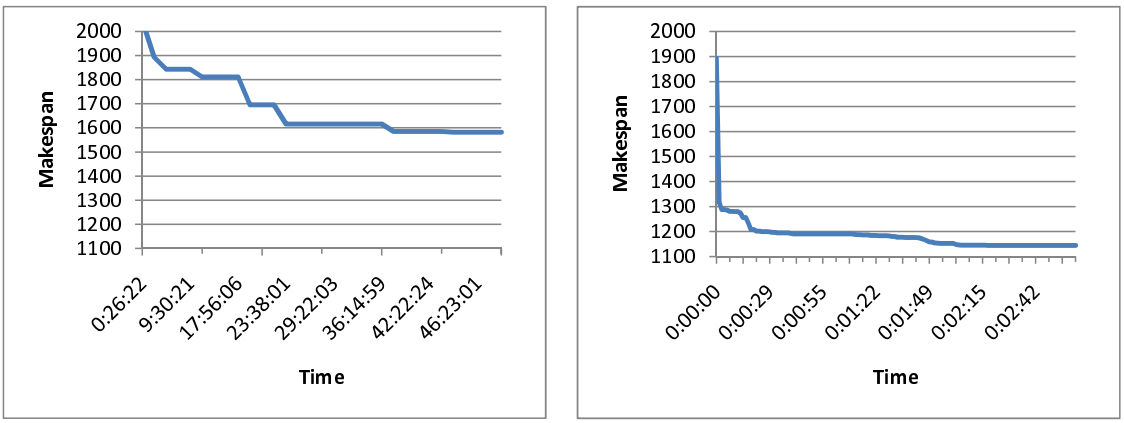
computation. The SMSP-SA, on the other hand, was able to generate a schedule with a makespan of 1143 minutes (27% improvement) in less than three minutes. To validate the improved solution generated by the SMSP-SA, we submitted it to CPLEX as a starting incumbent solution. CPLEX accepted it as a feasible starting solution but was unable to improve it. Regarding the larger problems studied in this paper, CPLEX was unable to start computation because of the large amount of memory required, whereas the SMSP-SA was able to generate solutions in a few minutes and progressively improve them. This illustrates the computational performance of the developed algorithm and its potential in solving larger problems.

5.3. Improvement through Parallelization

In the previous section, we illustrated the effectiveness of the sequential SA by comparing its performance with that of the state-of-the-art optimization package CPLEX. The performance of the sequential SA can be further improved using the multi-processor parallel implementation procedure outlined in Section 4.6. Figure 6 shows the makespan of the schedule for problem 10d using the sequential SA and a 32-processor (four nodes with eight cores each) parallel SA over 16 test runs conducted by varying the algorithm parameters

Table 3: Makespan improvements

	Pure Flowshop			Hybrid Flowshop		
	Makespan		Improvement	Makespan		Improvement
Problem	a	b	%	c	d	%
1	2889	2363	18	1813	1143	37
2	3085	2696	13	2945	2054	30
3	6271	5113	18	5635	3900	31
4	10003	8302	17	5168	3093	40
5	9625	8059	16	5604	3740	33
6	14467	11723	19	11521	7728	33
7	23700	17909	24	13097	8016	39
8	31853	24619	23	16111	10611	34
9	34776	27025	22	21987	13487	39
10	21886	15565	29	13683	8257	40
Average			20			35



CPLEX SMSP-SA

Figure 5: A 46-hour and 3-minute convergence graphs of CPLEX and SMSP-SA, respectively,in solving a small problem instance

as shown in Table 4. The parallel SA outperformed the sequential SA in 14 out of 16 test runs. Using parallel computation, we achieved a maximum improvement of 750 minutes in test run 12, and the average improvement over the 16 test runs was 270 minutes. We observed such performance improvements in several other problems whose general features are given in Table 5. Figure 7 shows the convergence of the sequential and parallel SAs (the latter with eight and 32 processors) in solving Problems 10d, 11, 12, 13 and 14. The convergence graphs in the first three columns show results for 16 test runs that utilized the 16 parameter sets given in Table 4, and the last column shows the average convergence of these test runs. These convergence graphs also show the improvements in the algorithms performance as we moved progressively from the sequential SA to the parallel SAs using eight and then 32 processors. Figure 8 shows the average and standard deviations of

makespans of the final solution of 16 test runs in each problem. Again, we can clearly see that the average makespan improved in each problem as we moved progressively from the sequential to the parallel SAs. More interestingly, the standard deviation of the makespans also decreased as we increased the number of processors. Consequently, an important issue in parallelizing the simulated annealing is not only how to build the algorithm to achieve maximum performance, but also how to make it robust in its ability to offer a consistently high level of performance over different parameter settings. Thus, the parallel SA requires less extensive parameter calibration efforts than the sequential SA.

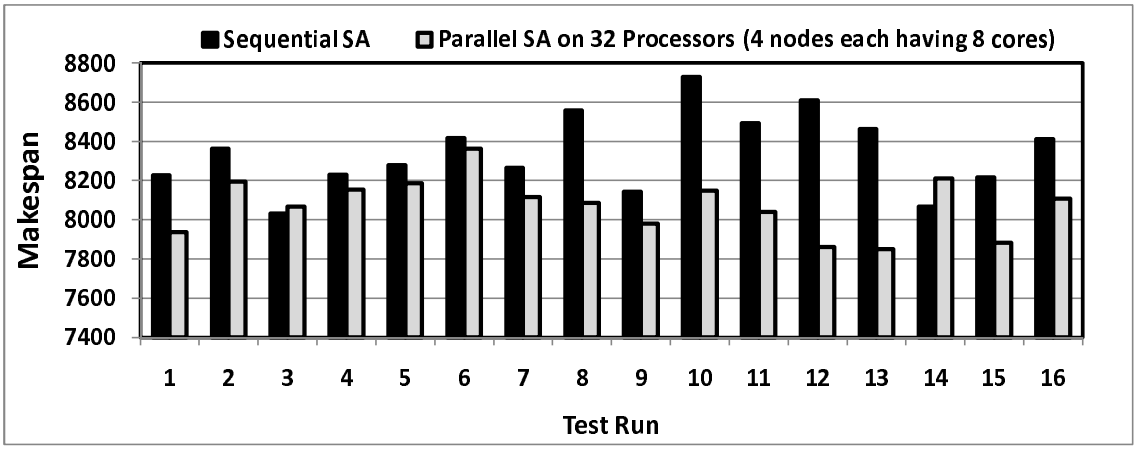


Figure 6: Makespan of problem 10d over 16 test runs

6. Discussion, Conclusion and Future Research

Lot streaming allows different sublots of the same job to be processed simultaneously at different stages. Production can be significantly accelerated as a result of such overlapping in operations. Previous research on flowshop lot streaming has generally featured the utilization of single machines in each stage. However, so-called hybrid flowshops that feature parallel machines at certain stages are very common in some industries. Indeed, hybrid flowshop scheduling has attracted much interest, and many scientific papers dealing with the diverse aspects of the problem have been published. Nevertheless, the issue of lot streaming in hybrid flowshop scheduling has gained less attention than lot streaming in pure flowshop scheduling, even thoughit is more effective in the former case. In an attempt to bridge this gap in research, the author of this paper developed a comprehensive mathematical model for the general hybrid flexible flowshop (HFFS) lot streaming problem, and a simulated annealing algorithm with multiple search paths to efficiently solve that comprehensive model. The proposed algorithm was implemented in both sequential and

Table 4: Algorithm parameters for various test runs

Parameter Set	S	T_0	α	Q	σ_1	σ_2	σ_3	σ_4	$\theta_{\max}$	L	F
1	240	170000	0.96	400	0.018	0.067	0.100	0.186	0.183	800	20
2	290	130000	0.97	300	0.04	0.140	0.060	0.200	0.172	900	30
3	210	160000	0.97	300	0.066	0.100	0.110	0.270	0.127	1500	10
4	170	100000	0.98	400	0.024	0.100	0.100	0.267	0.150	1200	50
5	280	190000	0.99	300	0.027	0.100	0.100	0.290	0.187	500	40
6	280	100000	0.99	300	0.04	0.062	0.137	0.145	0.160	700	40
7	160	110000	0.97	200	0.019	0.100	0.127	0.108	0.120	1200	30
8	270	150000	0.99	200	0.067	0.068	0.100	0.135	0.150	800	20
9	140	110000	0.97	500	0.031	0.103	0.053	0.299	0.171	1000	30
10	220	90000	0.97	500	0.04	0.100	0.100	0.200	0.150	700	50
11	160	140000	0.96	500	0.045	0.115	0.100	0.240	0.150	600	40
12	220	160000	0.97	400	0.013	0.100	0.086	0.200	0.150	1300	10
13	230	90000	0.97	500	0.013	0.056	0.057	0.274	0.192	500	30
14	230	190000	0.98	200	0.05	0.100	0.102	0.200	0.150	1200	10
15	170	180000	0.96	200	0.03	0.067	0.100	0.200	0.150	1000	20
16	270	170000	0.97	300	0.058	0.132	0.096	0.136	0.197	1100	20

$S-$number of search path followed by each processor; T_0-initial temperature; $\alpha-$ cooling exponent; $Q-$number of iteration at each temperature level along each search path; σ_1, σ_2, σ_3, σ_4, and $\theta_{\max}-$parameters for move operators; $L-$number of iteration before a processor restart its search paths from the best solution it found so far; $F-$communication factor.

Table 5: Features of the Additional Problems

Features	Problem 11	Problem 12	Problem 13	Problem 14
NoOfProducts	25	20	10	20
MaxNoOfSublots	5	5	5	5
NoOfStages	30	40	60	50
MaxNoOfParaMach	6	6	6	6
MinNoOfSublots	5	5	5	5
MinNoOfParaMach	3	3	3	3
MinimumLotsize	200	300	800	800
MaximumLotsize	500	800	1200	1200

Note: MaxNoOfParaMach = Maximum number of parallel machines. The number of machines across the stage varies between MaxNoOfParaMach and MinNoOfParaMach.

parallel computing platforms, and its performance was evaluated against a state-of-the art optimization package and a sequential SA. The results were very encouraging. In our future research, we plan to expand the model presented in this paper to account for capacitated buffer size and capacitated material handling equipment, interdependence of jobs because of assembly requirements and differences in job priority. We also plan to incorporate several objective functions in addition to makespan reduction within the context of lot streaming. These objectives may include due date, work-in-process inventory, flow time, etc. We will also consider advancing the research presented in this article by developing models and solution procedures for lot streaming in non-conventional manufacturing systems, such as

those based on cellular, fractal and distributed layouts, about which we currently have limited research

Acknowledgements: We thank SciNet (http://www.scinet.utoronto.ca/) and SHARCNET (https://www.sharcnet.ca/) for their assistance in providing access to parallel computing facilities.

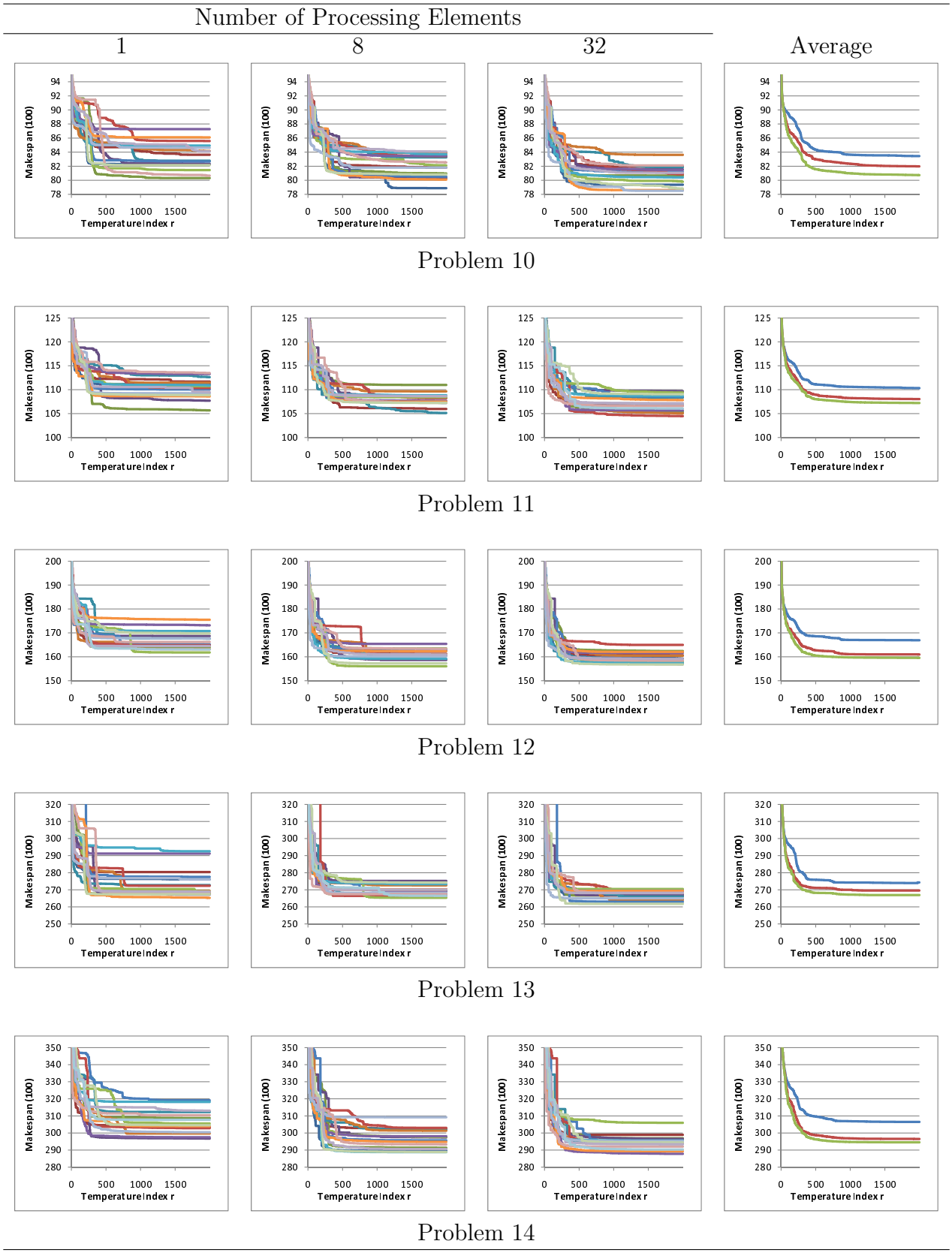


Figure 7: Individual and average convergence graphs of the 16 test runs for problems 10d, 11, 12, 13, and 14 by varying the number of processing elements

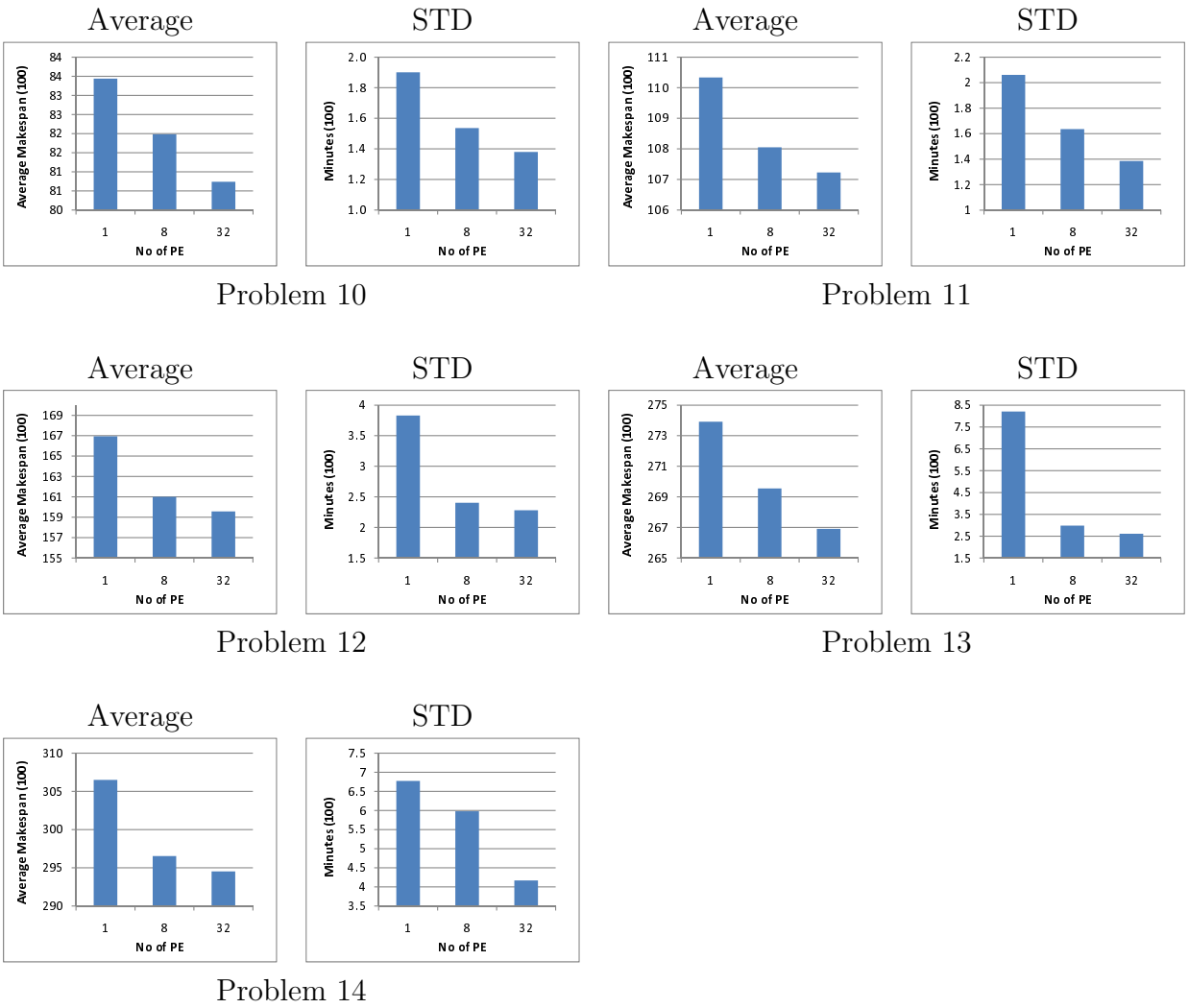


Figure 8: Average and standard deviations (STD) of makespans of 16 test runs for problems 10d, 11, 12, 13, and 14 by varying the number of processing elements

References

Azencott, R., 1992a. Parallel simulated annealing: an overview of basic techniques. In Robert Azencott, editor, Simulated Annealing: Parallelization Techniques. John Wiley and Sons, New York, pp. 25–35.

Azencott, R., 1992b. Sequential simulated annealing: speed of convergence and acceleration techniques. In Robert Azencott, editor, Simulated Annealing: Parallelization Techniques. John Wiley and Sons, New York, pp. 1–9.

Azizi, N. and Zolfaghari, S., 2004. Adaptive temperature control for simulated annealing: a comparative study. *Computers & Operations Research*, **31**, 2439–2445.

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

Baker, K. R. and Jia, D., 1993. A comparative study of lot streaming procedures. *OMEGA International Journal of Management Sciences*, **21**, 561–566.

Bertel, S. and Billaut, J. C., 2004. A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. *European Journal of Operational Research*, **159**, 651–662.

Biskup, D. and Feldmann, M., 2006. Lot streaming with variable sublots: an integer programming formulation. *Journal of Operational Research Society*, **57**, 296–303.

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

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

Botta-Genoulaz, V., 2000. Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *International Journal of Production Economics*, **64**, 101–111.

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

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

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

Chiang, T. S. and Chow, Y., 1988. On the convergence rate of annealing processes. *SIAM journal on control and optimization*, **26**, 1455–1470.

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.

Czech, A., Mikanik, W., and Skinderowicz, R., 2009. Implementing a parallel simulated annealing algorithm. in the porceeding of Parallel Processing and Applied Mathematics. Wroclaw, Poland, September 13-16, pp. 146–155.

Defersha, F. M., 2011. A comprehensive mathematical model for hybrid flexible flowshop lot streaming problem. *International Journal of Industrial Engineering Computations*, **2**, 283–294.

Deutsch, C. V. and Wen, X. H., 1998. An improved perturbation mechanism for simulated annealing simulation. *Mathematical Geology*, **30**, 801–816.

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.

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

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.

Janiak, A., Kozan, E., Lichtenstein, M., and Oğuzc, C., 2007. Metaheuristic approaches to the hybrid flow shop scheduling problem with a cost-related criterion. *International Journal of Production Economics*, **105**, 407–424.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., and Werner, F., 2008. Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *International Journal of Advanced Manufacturing Technology*, **37**, 354–370.

Kratica, J., Tosiæ, D., Filipoviæ, V., and Ljubiæ, I., 2001. Solving the simple plant location problem by genetic algorithms. *RAIRO–Operations Research*, **35**, 127–142.

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.

Lee, S.-Y. and Lee, K. G., 1996. Synchronous and asynchronous parallel simulated annealing with multiple markov chains. *IEEE Transactions on Parallel and Distributed Systems*, **7**, 903–1007.

Liu, J. Y., 2008. Single-job lot streaming in m-1 two-stage hybrid flowshops. *European Journal of Operational Research*, **187**, 1171–1183.

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.

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.

Martin, C. H., 2009. A hybrid genetic algorithm/mathematical programming approach to the multi-family flowshop scheduling problem with lot streaming. *OMEGA International Journal of Management Sciences*, **37**, 126–137.

Meise, C., 1998. On the convergence of parallel simulated annealing. *Stochastic Processes and their Applications*, **76**, 99–115.

Oğuzc, C., Zinder, Y., Do, V. H., Janiak, A., and Lichtenstein, M., 2004. Hybrid flow-shop scheduling problems with multiprocessor task systems. *European Journal of Operational Research*, **152**, 115–131.

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

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

Ruis, R., Antonio, J., and Rodríguez, V., 2010. The hybrid flow shop scheduling problem. *European Journal of Operational Research*, **205**, 1–18.

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

Ruiz, R. and Maroto, C., 2006. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, **169**, 781–800.

Sen, A., Topaloglu, E., and Benli, O. S., 1998. Optimal streaming of a single job in a two-stage flow shop. *European Journal of Operational Research*, **110**, 42–62.

Sriskandarajah, C. and Wagneur, E., 1999. Lot streaming and scheduling multiple products in two-machine no-wait flow shop. *IIE Transactions*, **31**, 695–707.

Tang, L., Liu, W., and Liu, J., 2005. A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment. *Journal of Intelligent Manufacturing*, **16**, 361–370.

Tang, L., Xuan, H., and Liu, J., 2006. A new lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time. *Computers & Operations Research*, **33**, 3344–3359.

Trietsch, D. and Baker, K., 1993. Basic techniques for lot streaming. *Operations Research*, **41**, 1065–1076.

Tseng, C. T. and Liao, C. J., 2008. A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. *European Journal of Operational Research*, **191**, 360–373.

Vickson, R. G. and Alfredsson, B. E., 1992. Two and three machines flow shop scheduling problems with equal sized transfer batches. *International Journal of Production Research*, **30**, 1551–1574.

Xiao, W., Hao, P., Zhang, S., and Xu, X., 2000. Hybrid flow shop scheduling using genetic algorithms. Proceedings of the 3rd World Congress on Intelligent Control and Automation. Hefei, P.R. China, pp. 537–541.

Ying, K.-C. and Lin, S.-W., 2006. Multiprocessor task scheduling in multistage hybrid flowshops: an ant colony system approach. *International Journal of Production Research*, **44**, 3161–3177.

Youhua, W. and Weili, Y., 1996. Adaptive simulated annealing for the optimal design of electromagnetic devices. *IEEE Transactions on Magnetics*, **32**, 1214–1217.

Zandieh, M., Fatemi Ghomi, S. M. T., and Moattar Husseini, S. M., 2006. An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. *Applied Mathematics and Computation*, **180**, 111–127.

Zhang, W., Yin, C., Liu, J., and Linn, R. J., 2005. Multi-job lot streaming to minimize the mean completion time in m-1 hybrid flowshops. *International Journal of Production Economics*, **96**, 189–200.