A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption

Deming Lei, Youlian Zheng & Xiuping Guo

To cite this article: Deming Lei, Youlian Zheng & Xiuping Guo (2017) A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption, International Journal of Production Research, 55:11, 3126-3140, DOI: 10.1080/00207543.2016.1262082

To link to this article: https://doi.org/10.1080/00207543.2016.1262082

Published online: 25 Nov 2016.

International Journal of Production Research, 2017
Vol. 55, No. 11, 3126–3140, http://dx.doi.org/10.1080/00207543.2016.1262082

A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption

Deming Lei[a], Youlian Zheng[b] and Xiuping Guo[c]*

[a]*School of Automation, Wuhan University of Technology, Wuhan, P.R. China;* [b]*Faculty of Computer Science and Information Engineering, Hubei University, Wuhan, P.R. China;* [c]*School of Economic and Management, Southwest Jiaotong University, Chengdu, P.R. China*

(*Received 16 June 2016; accepted 7 November 2016*)

Flexible job shop scheduling problem (FJSP) has been extensively investigated and objectives are often related to time. Energy-related objective should be considered fully in FJSP with the advent of green manufacturing. In this study, FJSP with the minimisation of workload balance and total energy consumption is considered and the conflicting between two objectives is analysed. A shuffled frog-leaping algorithm (SFLA) is proposed based on a three-string coding approach. Population and a non-dominated set are used to construct memeplexes according to tournament selection and the search process of each memeplex is done on its non-dominated member. Extensive experiments are conducted to test the search performance of SFLA and computational results show the conflicting between two objectives of FJSP and the promising advantages of SFLA on the considered FJSP.

Keywords: flexible job shop; scheduling; total energy consumption; workload balance; shuffled frog-leaping algorithm

1. Introduction

Nowadays manufacturers have to face enormous challenges such as the increased diversity in customer demand, globalisation and environmental pressures and develop some new strategies to deal with these challenges. For example, in the traditional production scheduling problems, time, cost and quality are the main objectives. Reducing energy consumption or environment impacts is required to be considered in production scheduling problem to reduce environmental impacts and implement the responsibility for the environmental outcome of manufacturing activities.

The reductions of energy consumption or carbon footprint have attracted some attention in recent years (Gahm et al. 2016). Mouzon, Yildirim, and Twomey (2007) collected the operational statistics of four machines in a workshop, found that the non-bottleneck machines consume a considerable amount of energy when left idle and proposed a turn-on and turn-off scheduling framework. This framework has been extended for single machine scheduling in their subsequent work (Mouzon and Yildirim 2008). They proposed a multiobjective genetic algorithm (GA) with hybridisation with dominance rules and a heuristic to minimise energy consumption and total completion time on a single machine (Yildirim and Mouzon 2012). Shrouf et al. (2014) adopted the same framework by Mouzon, Yildirim, and Twomey (2007) to minimise total energy consumption cost in single machine.

There is a growing interest in flow shop scheduling problem (FSP) with energy-related objective and the literature is mainly about the trade-off between total energy consumption and another objective including total tardiness. Fang et al. (2011) presented a mathematical programming model for FSP with peak power load and carbon footprint and demonstrated the model using a simple case study. Fang et al. (2013) presented two mixed-integer programming models for FSP with constraint on peak power consumption. Dai et al. (2013) reported a genetic-simulated annealing algorithm to make a significant trade-off between makespan and total energy consumption in flexible flow shop scheduling. Luo et al. (2013) provided an ant colony optimisation for hybrid flow shop scheduling with the consideration on electricity consumption cost. Liu et al. (2013) developed a branch-and-bound method for minimising total wasted energy consumption in permutation flow shop scheduling. Tang et al. (2015) proposed an improved particle swarm optimization for energy-efficient dynamic scheduling in flexible flow shop. An integrated model for processing parameter optimisation and flow shop scheduling was established by Lin et al. (2015). They developed a teaching-learning optimization algorithm to minimise makespan and carbon footprint simultaneously. Ding, Song, and Wu (2016) considered a permutation

*Corresponding author. Email: guoxiuping0029@sina.com flow shop scheduling with carbon emission and designed a modified multiobjective iterated greedy algorithm. Mansouri, Aktas, and Besikci (2016) provided a mixed integer linear multiobjective optimization model, lower bound and a heuristic for two-machine flow shop scheduling with makespan and total energy consumption.

Few papers are about reducing energy consumption in job shop scheduling problem (JSP). Lei and Guo (2015a) presented a dynamical neighbourhood search for a dual-resource constrained job shop with interval processing time and two objectives including carbon footprint. May et al. (2015) proposed a multi-objective GA for energy-efficient job shop scheduling. Zhang and Chiong (2016) developed a multi-objective GA to minimise total weighted tardiness and total energy consumption of JSP.

Flexible job shop scheduling problem (FJSP) is an extension of JSP by allowing an operation to be processed on more than one machine. FJSP has great practical relevance in many industries. It has attracted much attention in the past decade and a number of results have been obtained (Li et al. 2011, 2012; Wang, Wang, and Liu 2013, 2014; Chiang and Lin 2013; Yuan and Xu 2015; Ahmadi et al. 2016).

As stated above, the previous works have been made on FSP and JSP with energy consumption and FJSP with time-related objectives. The literature rarely discusses the reduction of carbon foot-print or energy consumption in FJSP and its frequently optimised objectives are related to completion time or workload in most cases. With the intensive concern on environmental problem and the advent of green manufacturing, it is vital to focus on FJSP with energy-relative objective.

After the new objective is added, FJSP is composed of three sub-problems: machine assignment, scheduling and speed selection. The third one is utilised to choose the operation speed of each job on its processing machines. The complexity of FJSP greatly increases with the introduction of the new sub-problem. On the other hand, workload objectives such as workload balance are often adopted in FJSP and the conflicting between workload and other objective including total tardiness has been investigated; however, the conflicting relation between energy consumption and workload objective is not studied, thus, it is necessary to discuss FJSP with total energy consumption and workload.

Shuffled frog-leaping algorithm (SFLA) is a meta-heuristic obtained by observing, imitating and modelling the behaviour of a group of frogs when searching for the location that has the maximum amount of available food. Its main features include fast convergence speed and effective algorithm structure containing local search and global information exchanges. SFLA is originally developed by Eusuff, Lansey, and Pasha (2006) and has been applied to production scheduling problems (Rahimi-Vahed et al. 2009; Pan et al. 2011; Li et al. 2012; Fang and Wang 2013; Lei and Guo 2015b; Lei and Guo (forthcoming); however, SFLA is seldom applied to solve FJSP with multiple objectives and FJSP with energy-related objective (Li et al. 2012).

In this study, FJSP with total energy consumption and workload balance is considered and a SFLA is proposed to solve the problem. A three-string coding method is used to represent three sub-problems independently. Memeplexes are constructed using solutions in population and a non-dominated set according to tournament selection. The search process within each memeplex is executed on its non-dominated member and consists of two kinds of searches: global search and multiple neighbourhood search. A number of experiments are conducted to test the performance of SFLA. The computational results demonstrate the conflicting relation between two objectives and the advantages of SFLA on the considered FJSP.

The remainder of the paper is organised as follows. Problem under study is described in Section 2 and followed by the introduction to SFLA in Section 3. A novel SFLA for the problem is described in Section 4. Numerical test experiments on SFLA are reported in Section 5 and the conclusions are summarised in the final section and some topics of the future research are provided.

2. Problem description and analyses on conflicting between objectives

2.1 *Problem description*

The considered FJSP is composed of n jobs $J_i(i = 1, 2, \ldots, n)$ and m machines $M_k(k = 1, 2, \ldots, m)$. Each job J_i consists of h_i operations. Operation o_{ij} indicates the jth operation of job J_i and can be processed on a set S_{ij} of compatible machines. There is a finite and discrete set of d different processing speeds for each machine, $V = \{v_1, v_2, \ldots, v_d\}$. The speed of a machine cannot be changed during its execution of a job. It is assumed that operation o_{ij} on machine M_k has a basic processing requirement measured by η_{ijk}.

When an operation o_{ij} is processed on a machine $M_k \in S_{ij}$ at speed v_l, the processing time p_{ijkl} is equal to η_{ijk}/v_l and the energy consumption per unit time is denoted by E_{kl}.

With respect to relation between p_{ijkl} and E_{kl}, there is an assumption (Ding, Song, and Wu 2016).

It is assumed that if an operation o_{ij} is processed at a higher speed on a machine M_k, the energy consumption increases while its processing time decreases. This implies that

$$\text{For}\,\forall v_l > v_g, l, g \in \{1,2,\ldots,d\}, \quad p_{ijkl} < p_{ijkg}, E_{kl} \times p_{ijkl} > E_{kg} \times p_{ijkg} \tag{1}$$

A machine cannot be turned off completely until all jobs are finished. Each machine M_k will be on a stand-by mode with instantaneous energy consumption of SE_k per unit time during the idle periods.

There are several constraints on jobs and machines, such as:

Each machine can process at most one operation at a time,
No jobs may be processed on more than one machine at a time,
Operations cannot be interrupted,
Set-up times and remove times are included in the processing times et al.

The considered FJSP is made up of three sub-problems: scheduling sub-problem for sequencing the operations on machines, machine assignment sub-problem for assigning an appropriate machine for each operation and speed selection for deciding processing speed of machine used to process operation.

The goal of the problem is to minimise the following objectives simultaneously

$$f_1 = TEC = \int_0^{C_{\max}} \left(\sum_{i=1}^{n} \sum_{j=1}^{h_i} \sum_{k=1}^{m} \sum_{l=1}^{d} E_{kl} y_{ijkl}(t) + \sum_{k=1}^{m} SE_k z_k(t) \right) \mathrm{d}t \tag{2}$$

$$f_2 = WB = \sqrt{\sum\nolimits_{k=1}^{m} \left(W_k - \bar{W}\right)^2} \tag{3}$$

where TEC indicates total energy consumption. $y_{ijkl}(t)$ is a binary variable. If a machine $M_k \in S_{ij}$ is in processing mode at speed v_l at time t, then $y_{ijkl}(t)$ is equal to 1; otherwise $y_{ijkl}(t)$ is 0. $z_k(t)$ is 1 if a machine M_k is in stand-by mode at time t and 0 otherwise. WB denotes workload balance. W_k and $\bar{W}$ are workload of machine M_k and average workload, respectively. The consideration on WB is to diminish the difference of workload on all machines and avoid making some machines overload.

Table 1 shows an example, the entries of which are the processing times.$SE_k = 1$, $E_{kl} = 4 \times v_l^2$, the speed set $V = \{1.0, 1.5, 2.0\}$. In this example, we have total flexibility. In a partial flexibility scenario, an empty entry in the table means that a machine cannot execute the corresponding operation, i.e., it does not belong to the set of the compatible machines for that operation.

2.2 *Conflicting between objectives*

For the problem with f_1, f_2, the optimal result is not a single solution but a set of solutions; moreover, the optimal set cannot be obtained without comparing all solutions produced by the method. When solutions in a set are compared each other, take x and y as an example, if $f_i(x) \le f_i(y)$ for $\forall i \in \{1,2\}$, $f_i(x) < f_i(y)$ for $i = 1$ *or* 2, then x dominates y. If a solution x is dominated by any other solutions in the same set, x is a non-dominated solution regarding the set. If a solution is not dominated by any other solutions in search space, the solution is Pareto optimal. Pareto front consists of the objective vectors of all Pareto optimal solutions.

With respect to TEC and WB, WB is just decided by machine assignment and speed selection and TEC is affected by three sub-problems. Take two solutions (TEC_1, WB_1) and (TEC_2, WB_2) as an example, two cases are discussed:

Table 1. Processing requirement p_{ijk} on machines.

Job	M_1	M_2	Job	M_1	M_2
$J_1 o_{11}$	2	4	$J_3 o_{31}$	6	4
o_{12}	3	5	o_{32}	2	4
o_{13}	4	6	$J_4 o_{41}$	3	5
$J_2 o_{21}$	3	7	o_{42}	6	9
o_{22}	3	2	o_{43}	6	5

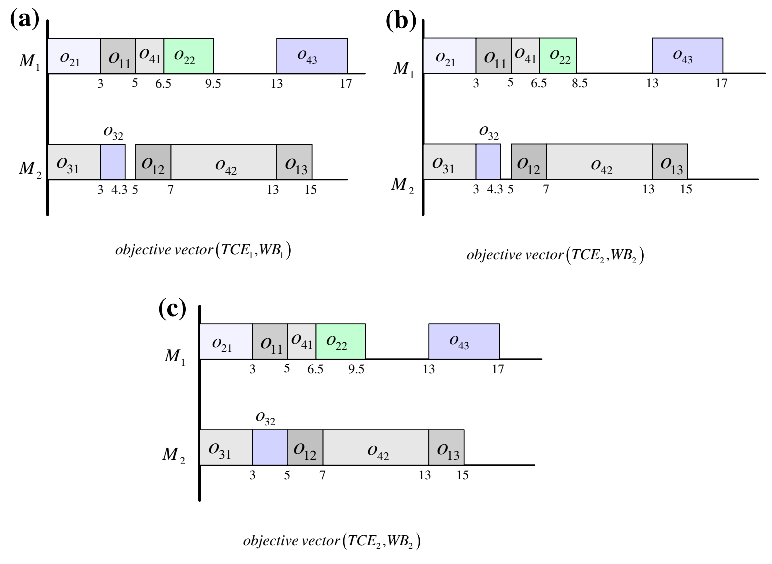

Figure 1. Analyses on conflicting between two objectives.

(1) suppose that they have the same machine assignment and speed selection, that is, $WB_1 > (<) WB_2$; if two objectives don't conflict each other, in this special case, TEC_1 should be equal to TEC_2; however, there exist three possible cases: $TEC_1 > (<) TEC_2$ or $TEC_1 = (<) TEC_2$.

Figure 1 shows three Gantt charts of three schedules. Figure 1(a) is the Gantt chart of a solution of the example in Table 1. When the speed of operation o_{22} increases from v_1 to v_2, Figure 1(b) is obtained. Figure 1(c) is generated from Figure 1(a) by changing the speed of operation o_{32} from v_2 to v_1. The comparison between Figure 1(a) and (b) shows that $(TEC_1$ and $(WB_1 < WB_2)$, (TEC_1, WB_1) dominates (TEC_2, WB_2). It can be found from the comparison between Figure 1(a) and (c) that $TEC_1 > TEC_2$ and $WB_1 < WB_2$. They are non-dominated each other in this case.

The above two cases show that conflict must exist between *TEC* and *WB*.

3. Introduction to shuffled frog-leaping algorithm

In SFLA, a solution is defined as the position of a frog and there is a population of possible solutions defined by a set of virtual frogs. The population is divided into different memeplexes. Each memeplex then performs a search process. After the search process within each memeplex is finished, all evolved memeplexes are shuffled and combined into a new population. The search process and the shuffling process are repeated until the defined convergence criteria are met.

The detailed procedure of SFLA (Eusuff, Lansey, and Pasha 2006) is described below.

(1) Initialize parameters: population size (N), number of memeplexes (s), number of iterations within each memeplex.
(2) Randomly generate initial population P.
(3) Sort the solutions in population in descending order of their fitness.
(4) Divide population into s memeplexes.
(5) Perform search process within each memeplex.
(6) Shuffled the evolved memeplexes.
(7) If the termination condition is met, output the best solution.

The procedure of the dividing of population into s memeplexes is as follows: the first frog goes to the first memeplex $\mathcal{M}_1$, the second one is allocated into the memeplex $\mathcal{M}_2$, the sth frog is assigned into the sth memeplex $\mathcal{M}_s$, the $(s+1)$th frog goes back to the first memeplex $\mathcal{M}_1$, and so on.

The search process in memeplex $\mathcal{M}_i$ is as follows: the best solution x_b of $\mathcal{M}_i$, the worst one x_w of $\mathcal{M}_i$ and the global best solution $x_g \in P$ are first decided, then a new frog position x is produced using x_b and x_w, if this new frog position is better than x_w, then the worst frog is replaced with the new frog; else x_g and x_w are used to produce a new frog position, if the new frog position is better than x_w, then the worst frog is replaced with the new frog; if the above two steps cannot generate new x_w, a new frog position is randomly produced and directly substitutes for the worst frog x_w. The above steps are repeated until the given number of iterations is met.

After the search process within each memeplex is done, the evolved memeplexes are shuffled: the solutions of these memeplexes are combined together and a new population P is obtained, then the solutions of the population are sorted in the descending order of fitness.

4. SFLA for the considered FJSP

Lei and Guo (2015b, forthcoming) provided some new principles to design SFLA. These principles are listed as follows: the search process is done on x_b of each memeplex and all solutions in the memeplex can be combined together with x_b in the same probability; tournament selection is used to divide population P. In this study, we present some new paths to design SFLA for multi-objective FJSP.

4.1 *Three-string coding*

The considered FJSP is composed of three sub-problems. A three-string coding method is adopted to represent them independently. For FJSP with n jobs, m machines and d speeds, its solution is denoted by applying a scheduling string $((\theta_1, r_1), (\theta_2, r_2), \ldots, (\theta_i, r_i), \ldots, (\theta_h, r_h))$, a machine assignment string $(q_{11}, q_{12}, \ldots, q_{1h_1}, \ldots, q_{nh_n})$ and a speed selection string $(z_{11}, z_{12}, \ldots, z_{1h_1}, \ldots, z_{nh_n})$, $h = \sum_{i=1}^{n} h_i$.

The first string is for scheduling sub-problem, a doublet (θ_i, r_i) indicates an operation$o_{\theta_i r_i}$ and the whole string corresponds to an ordered operation list, $\theta_i \in \{1, 2, \ldots, n\}$, $1 \leq r_i \leq h_{\theta_i}$. Each gene $q_{ij} \in S_{ij}$ denotes a compatible machine for operation o_{ij} and z_{ij} is the speed of the machine $q_{ij} \in S_{ij}$ when it processes operation o_{ij}.

An initial population is randomly produced. To obtain a feasible schedule, the following decoding procedure is adopted for each solution:

(1) Translate the first string into a list of the ordered operations, and assign a machine for each operation and decide the speed of the assigned machine for each operation according to other two strings;
(2) The first operation of the ordered operation list is first arranged, and then the second one is done and so on; each operation is allocated in the best available time on its assigned machine. The procedure is repeated until a schedule is obtained.

For the example shown in Tables 1, a possible solution is shown in Figure 2.

4.2 *Memeplex construction*

Lei and Guo (2015b) applied binary tournament selection to construct$s$ memeplexes by dividing population P, the procedure is as follows: the first tournament selection is done and two solutions are randomly selected from P, the better one of them goes to $\mathcal{M}_1$ and the worse one of them goes back to P; then the second selection is executed and two solutions are chosen from P again, the better one goes to $\mathcal{M}_2$ and the worse one is back to P; the better solution of the sth tournament selection is allocated into $\mathcal{M}_s$; the better one of the $(s+1)$th tournament selection is allocated into $\mathcal{M}_1$, and so on.

In this study, we provide a new path to construct memeplexes using population P and a non-dominated set Ω. The set Ω is used to store the non-dominated solutions generated by SFLA. To obtain s memeplexes, an extended population $\bar{P} = P \cup \Omega$ is obtained, solutions x_1, x_2 are randomly chosen from $\bar{P}$; if $x_1 > x_2$ or $x_2 > x_1$, the dominating solution is directly added into memeplex $\mathcal{M}_1$; if x_1 and x_2 are non-dominated each other, then they can be added into $\mathcal{M}_1$ according to the same probability of 0.5 and one of them is randomly chosen, where $x_1 > x_2$ means that x_1 dominates x_2; the

doublet string	((2,1)	(1,1)	(1,2)	(3,1)	(4,1)	(4,2)	(2,2)	(1,3)	(4,3)	(3,2))
list of ordered operations	o_{21}	o_{11}	o_{12}	o_{31}	o_{41}	o_{42}	o_{22}	o_{13}	o_{43}	o_{32}
	o_{11}	o_{12}	o_{13}	o_{21}	o_{22}	o_{31}	o_{32}	o_{41}	o_{42}	o_{43}
machine assignment string	$(M_1$	M_2	M_2	M_1	M_1	M_2	M_2	M_1	M_2	$M_1)$
speed selection string	$(v_1$	v_2	v_3	v_1	v_1	v_3	v_2	v_3	v_1	$v_2)$

Figure 2. An example of three-string coding.

unselected solution goes back to $\bar{P}$, the above steps are repeated until the size of $\mathcal{M}_1$ reaches. The same procedure is done to construct $\mathcal{M}_2, \ldots \mathcal{M}_s$ sequentially.

Unlike the existing SFLA (Li et al. 2012; Lei and Guo 2015b, forthcoming), both population and the non-dominated set are used to obtain memeplexes, as a result, some solutions can be added into more than one memeplex and some cannot be assigned into any memeplexes; moreover, it is not required to sort all solutions of $\bar{P}$ and the memeplexes are constructed in a different way.

4.3 Search process within each memeplex

In multi-objective case, more than one best solution x_b exists and these solutions are non-dominated, so unlike SFLA proposed by Lei and Guo (2015b), the search process in memeplex $\mathcal{M}_i$ is executed on a randomly chosen non-dominated member $x_b \in \mathcal{M}_i$ and consists of two kinds of searches: global search on x_b and a randomly selected solution $x \in \mathcal{M}_i$, and multiple neighbourhood search of $x_b \in \mathcal{M}_i$.

4.3.1 *Global search*

There are three global search operators for three strings of $x_b \in \mathcal{M}_i$. The first operator is on scheduling string and described as follows (Mattfeld and Bierwirth 2004): for the first string of x_b and x, start with the first position, randomly generate a random number α, if $\alpha < \delta$, the first doublet of x_b is selected; else the first doublet of x is chosen; suppose that the doublet (θ_i, r_i) is chosen, it is appended to the off-spring and then delete (θ_i, r_i) from x_b and x, if the doublet (θ_i, r_i) is the gth doublet of x, then shift the doublets of x between position $g + 1$and last position left once. The above step is repeated until the first string of x_b and x is empty; as a result, a new doublet string is obtained, where δ is a probability.

The second one is done between machine assignments of two solutions, which is as follows: for x_b and x, two positions $g_1, g_2 \in [1, h]$ are stochastically decided and then the machines of x_b between two positions are replaced with those of x in the same position.

The third operator is similar with the second one and described below: for x_b and x, two positions $g_1, g_2 \in [1, h]$ are stochastically decided and then the speed of x_b between two positions are replaced with those of x in the same position. Unlike the works of Lei and Guo (2015b), the execution of three global search operators is decided by two probabilities.

In the following, global search is summarised: if $\alpha < \beta$, the first operator is applied; if $\alpha \in [\beta, \eta]$, then the second one is done; if $\alpha > \eta$, the third operator is executed.

4.3.2 *Multiple neighbourhood search*

Neighbourhood structure *insert* is designed for generating neighbourhood solutions of doublet string. For scheduling string $((\theta_1, r_1), (\theta_2, r_2), \ldots, (\theta_i, r_i), \ldots, (\theta_h, r_h))$, a string $(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_h)$ is first obtained, then an element θ_j and a position $k \neq j$ are decided and the element θ_j is inserted into the position, a new doublet string is constructed.

For example in Figure 2, the scheduling string is ((21), (11), (12), (31), (41), (42), (22), (13), (43), (32)), a string (2, 113, 442, 143) is first generated, $\theta_3 = 1$, $k = 5$, a new string (2, 134, 142, 143) is then produced after insertion. For (2, 134, 142, 143), $\theta_1 = 2$is the first '2' occurring in the string, so $r_1 = 1$, $\theta_7 = 2$ is the second '2' in the string, so $r_7 = 2$. When all r_i are determined in the same way, a new doublet string ((21), (11) (31), (41), (12), (42), (13), (43), (32)) is finally constructed.

Neighbourhood structure *change* is executed on the second string $(q_{11}, q_{12}, \ldots q_{1h_1}, \ldots, q_{n1}, \ldots, q_{nh_n})$, which is shown below: a set $\Theta = \{q_{ij} \| S_{ij}| > 1, i = 1, 2 \ldots, n, j = 1, 2 \ldots, h_i\}$ is first decided, some elements are stochastically selected from the set Θ, suppose that q_{ij} is selected, then q_{ij} is replaced with a chosen machine from S_{ij} in a random way

Neighbourhood structure *speed* is used to select a new speed for some chosen machines. It obtains new solution by randomly choosing several elements from the third string and assigning a new speed to these elements randomly.

Let $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3$ be *insert*, *change* and *speed*. $\mathcal{N}_i(x)$ represents the neighbourhood of x obtained using $\mathcal{N}_i$.

After global search and multiple neighbourhood search are executed on x_b, a new solution z is produced, a replacement principle is used to decide if x_b can be replaced with z: if z is dominated by x_b, then x_b is not changed; otherwise, z substitutes for x_b.

The non-dominated set Ω is updated in the following: after x_b is replaced with a new solution, it is included into the set Ω and all members of Ω are compared according to Pareto dominance. After the dominated members are deleted from the set Ω, if the actual size of Ω exceeds its maximum size H, crowding distance is calculated for each member in terms of the methods proposed by Deb et al. (2002) and some members with the smallest crowding distance are removed from Ω to make the actual size be equal to the maximum size.

4.4 *Algorithm description*

The detailed procedure of SFLA is as follows:

(1) Randomly produce an initial population P, construct an initial set Ω, $\sigma_i \leftarrow 1$ $i = 1, 2, \cdots, N, \cdots, N + H$.
(2) Repeat the following steps (3)–(5) until the termination condition is met
(3) Decide $\bar{P} = P \cup \Omega$ and apply tournament selection to constructs memeplexes using $\bar{P}$.
(4) For each memeplex$\mathcal{M}_i, i = 1, 2, \ldots, s$, $w \leftarrow 1$, if $w \le \mu$, repeat the following steps:
  (4.1) Randomly select a non-dominated solution x_b and a solution from $\mathcal{M}_i$. Suppose that x_b is the gth solution of $\bar{P}$.
  (4.2) Global search between x_b and x is done, a solution z is obtained and compared with x_b; if the replacement condition is met, then x_b is replaced with z, the set Ω is updated, $w \leftarrow w + 1$, go to step (4.1).
  (4.3) A new solution $z \in \mathcal{N}_{\sigma_g}(x_b)$ is obtained and compared with x_b, if the replacement condition is met, then x_b is replaced with z and the set Ω is updated, $w \leftarrow w + 1$and go to (4.1); else $\sigma_g \leftarrow \sigma_g + 1$, if $\sigma_g \ge 4$, then let $\sigma_g \leftarrow 1$.
  (4.4) A new solution $z \in \mathcal{N}_{\sigma_i}(x_b)$ is obtained and compared with x_b, if the replacement condition is met, then x_b is replaced, the set Ω is updated, $w \leftarrow w + 1$ and go to (4.1).
(5) Shuffle the evolved memeplexes.
(6) Output the set Ω.

where μ is an integer.

The flow chart of SFLA is shown in Figure 3. The stopping condition is the predetermined number of objective function evaluations. If it is met in step (4), the search of SFLA directly goes to step (6) and then is stopped. In step (4), a new path is applied to cooperate global search and neighbourhood search. In step (5), all memeplexes are combined together after deleting those solutions selecting from Ω and a new population P is obtained using the combined memeplexes and the remained part of population P. The computational complexity of SFLA is $O(s\mu W)$, where W is the repeated times of steps (3)–(5).

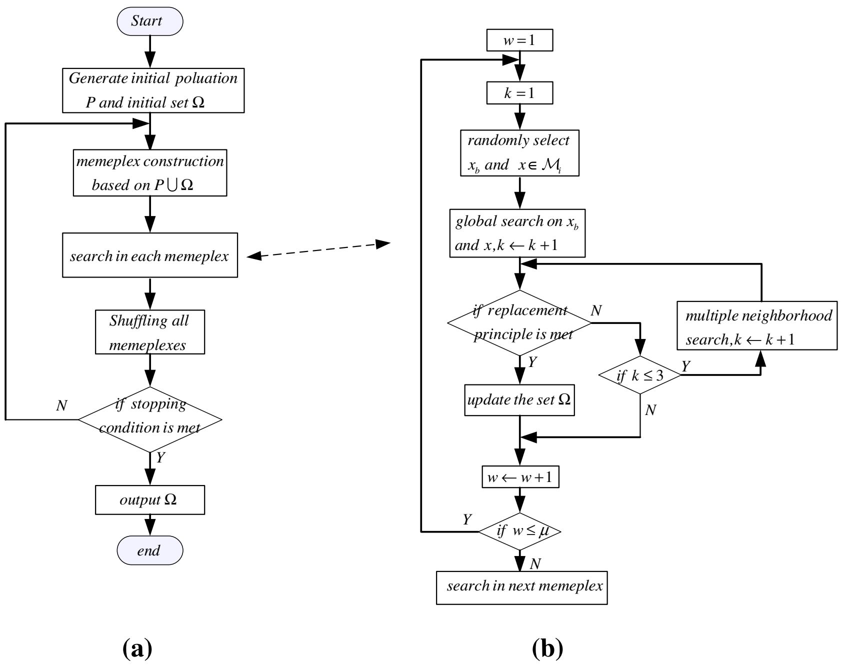
Start
Generate initial poluation P and initial set Ω
memeplex construction based on P∪Ω
search in each memeplex
Shuffling all memeplexes
if stopping condition is met
N
Y
output Ω
end
(a)
w=1
k=1
randomly select x_b and x∈M_i
global search on x_b and x, k←k+1
if replacement principle is met
N
Y
multiple neighborhood search, k←k+1
if k≤3
Y
N
update the set Ω
w←w+1
if w≤μ
Y
N
search in next memeplex
(b)

Figure 3. The flow chart of SFLA.

5. Computational experiments

To test the performance of SFLA for the considered FJSP, extensive experiments are conducted on a set of problems. All experiments are implemented using Microsoft Visual C++ 6.0 and run on 4.0G RAM 1.70 GHz CPU PC.

5.1 *Test instances, performance metrics and the chosen algorithms*

We use 41 instances MK1-13 (Brandimarte 1993), DP1-18 (Dauzère-Pérès and Paulli 1997) and LA21-30 (Hurink et al. 1994). These instances are designed for testing FJSP and JSP. We extend them by adding machine speed information: $V = \{1.00, 1.30, 1.55, 1.80, 2.00\}$, $E_{kl} = 4 \times v_l^2$, $SE_k = 1$.

Many performance metrics have been used to compare the results of multi-objective optimisation on the different algorithms. The following two metrics are chosen in this study.

Distance metric DI_R is applied to measure the performance of the non-dominated solution set Ω_l relative to a reference set Ω^*.

$$DI_R(\Omega_l) = \frac{1}{|\Omega^*|}\sum\nolimits_{y \in \Omega^*} \min\{d_{xy} | x \in \Omega_l\} \tag{4}$$

where d_{xy} is the distance between a solution x and a reference solution y in the normalised objective space, $d_{xy} = \sqrt{\left(f_1^*(x) - f_1^*(y)\right)^2 + \cdots + \left(f_D^*(x) - f_D^*(y)\right)^2}$, D is the number of objectives and f_i^* is the ith normalised objective. The details of normalisation are shown in Ishibuchi et al. (2013). The set Ω^* consists of the non-dominated solutions of $\bigcup_{l=1}^{A} \Omega_l$, A is the total number of algorithms.

The smaller the value of $DI_R(\Omega_l)$ is, the better the solutions of Ω_l are.

Metric ρ_l indicates the ratio of number of the elements in the set $\{x \in \Omega_l | x \in \Omega^*\}$ to $|\Omega^*|$.

As stated above, the conflicting between two objectives of the considered FJSP is not studied, so there are no existing methods for our FJSP. In this study, non-dominated sorting genetic algorithm-II (NSGA-II, Deb et al. 2002) and VNS are chosen.

NSGA-II has very competitive performance on solving multi-objective problem. In NSGA-II, a non-dominated sorting approach is used for each individual to create a Pareto rank, and a crowding distance assignment method is applied to implement density estimation. NSGA-II prefers the point with a lower rank value or the point located in a region with fewer points if both of the points belong to the same front. By combining a fast non-dominated sorting method, an elitism scheme and a parameter- less sharing method with its origin, NSGA-II is claimed to produce a better spread of solutions in some testing problems. The strong search ability on multi-objective problem motivated us to choose NSGA- II as the comparative algorithm.

Ahmadi et al. (2016) applied NSGA-II to FJSP with machine breakdown. We extended NSGA-II to energy-efficient FJSP in the following: the representation for machine assignment and scheduling and crossovers and mutation related to this representation are adopted directly, the handling part on machine breakdown is deleted and a speed selection string is added for each individual, two-point crossover and mutation using *speed* are also added for speed selection sub-problem.

As stated above, SFLA has an effective algorithm structure containing local search and global information exchanges. To investigate the local search ability of SFLA, we construct a VNS using $\mathcal{N}_1$, $\mathcal{N}_2, \mathcal{N}_3$, which is as follows.

(1) Produce an initial solution x and an initial set Ω. $k \leftarrow 1$.
(2) Repeat the following steps until the termination criterion is met.
(3) Randomly generate a solution $z \in \mathcal{N}_k(x)$ and compare with x, if the replacement condition is met, the current solution x is replaced with z, the set Ω is updated using the strategy shown in Section 4.3 and $k \leftarrow 1$; otherwise $k \leftarrow k + 1$and if $k = 4$ then $k \leftarrow 1$.

In our SFLA, a new way is applied to construct memeplexes and x_b is the optimisation object of the search in each memeplex. To investigate the impacts of these two strategies on the performance of SFLA, two variants of SFLA named SFLA1 and SFLA2 are obtained. SFLA1 has the same steps as SFLA except that memeplexes are built according to the method of BSFLA. SFLA2 differs from SFLA in that the search process within each memeplex is executed on a dominated member x_w, which has maximum value of γ, where γ denotes the number of solutions dominating a solution in the same set, for example, x_w.

5.2 *Results and analyses*

5.2.1 *Parameter setting*

Five parameters of SFLA including N (population scale), μ, H (maximum size of Ω), s (number of memeplexes) and *max_it* (the maximum number of the generated solutions) are considered, which can influence the performances of SFLA.

The stopping criterion of SFLA is set to be $max\_it = 10^5$ for the instances with $h \le 400$ and 1.5×10^5 for other instances. It can be found that the convergence process of SFLA is nearly finished when the above criterion is met on all instances. We then observed that SFLA can generate better results if H is 20. We finally used three settings of 40, 50, 60 for N, 100, 120, 150 for μ, 5, 8, 10 for s. There are 27 combinations for parameters. Based on the results measured using the above metrics, the best results are achieved when we use the following settings: $N = 40$, $\mu = 100$, $s = 5$.

We set same parameters for SFLA1 and SFLA2.

The parameters of VNS and NSGA-II are listed below. For NSGA-II, population scale of 100, crossover probability of 0.8, mutation probability of 0.1 and maximum generation of $max\_it/100$. For VNS, *max_it* is its unique parameter. The search of each algorithm is stopped when *max_it* solutions are generated. All parameters of VNS and NSGA-II are set based on extensive experiments.

Table 2. The computational results of five algorithms on $DI_R(\Omega)$.

Instance	SFLA	SFLA1	SFLA2	VNS	NSGA-II
MK1	0.0339	44.329	8.3971	38.634	3.6293
MK2	0.2733	36.947	8.1725	33.304	1.5197
MK3	0.0891	42.981	11.827	32.710	1.0476
MK4	0.3307	41.548	9.4125	31.576	3.9671
MK5	1.4335	46.989	9.5512	40.543	3.3719
MK6	1.6638	45.972	14.368	32.326	3.7560
MK7	0.6507	34.458	7.4197	28.152	3.0762
MK8	2.2532	54.875	16.118	37.884	8.7260
MK9	1.2274	41.737	17.876	37.429	5.9239
MK10	4.9566	49.374	23.584	38.342	11.850
MK11	8.3329	101.88	94.336	46.693	21.484
MK12	0.7507	54.826	11.993	37.845	2.4246
MK13	0.5234	46.494	13.523	29.656	5.2576
DP1	0.5459	53.148	14.082	43.192	6.6498
DP2	1.0836	47.489	11.453	39.052	3.6651
DP3	0.9821	44.198	11.757	36.084	6.6217
DP4	1.5764	49.483	9.0961	38.235	5.9962
DP5	0.6931	44.604	9.7627	33.790	0.8829
DP6	0.9405	50.812	15.523	43.890	6.3374
DP7	2.0487	50.517	6.9822	41.949	7.5042
DP8	2.1697	42.715	13.441	45.136	12.562
DP9	1.1164	45.934	14.536	42.510	1.3704
DP10	1.1984	56.353	19.914	35.706	4.5097
DP11	1.7558	48.708	16.781	43.103	2.3679
DP12	0.6334	55.456	19.252	43.184	5.4849
DP13	0.6440	56.650	16.081	51.453	5.0042
DP14	0.5738	53.763	19.022	45.183	7.3328
DP15	0.9672	56.542	13.896	45.883	4.3167
DP16	2.0595	60.997	14.523	45.367	6.1034
DP17	1.7670	61.825	18.231	50.691	5.6331
DP18	0.2000	53.496	14.077	44.249	7.7569
LA21	1.6845	43.470	3.8244	18.923	2.2384
LA22	1.4657	42.864	4.4884	14.556	0.4451
LA23	2.1712	45.136	5.2472	13.821	0.6040
LA24	3.0788	44.263	7.5038	14.835	0.2374
LA25	1.8025	45.737	4.0377	14.846	0.2236
LA26	3.2899	44.283	5.7832	28.047	6.9750
LA27	2.3192	49.308	10.821	30.794	10.548
LA28	3.2751	43.599	7.9128	28.805	16.452
LA29	0.5906	47.261	7.3946	22.410	13.266
LA30	2.2087	48.317	9.3768	21.643	4.5765

When all algorithms are applied to solve LA21-30, the machine assignment part is deleted.

The computational results of five algorithms are reported in Tables 2 and 3, in which Ω^* is a set of the non-dominated solutions in the set $\bigcup_{l=1}^{5}\Omega_l$. Ω_1, Ω_2, Ω_3, Ω_4, Ω_5 are the non-dominated set of SFLA, SFLA1, SFLA2, VNS and NSGA-II. $DI_R(\Omega_l)$ and ρ_l are the corresponding metric values of Ω_l. Table 4 lists the computational times of five algorithms.

To make the results statistically convincing, paired-sample *t*-test is done to compare SFLA with other algorithms. The *p*-value results for the above tests are shown in Table 5. The term '*t*-test (A, B)' means that a paired *t*-test is conducted to judge whether algorithm B gives a better sample mean than algorithm A. We assume a significance level of 0.01, which indicates that A performs better than B in the statistical sense if the corresponding *p*-value is smaller than 0.01.

In order to visualise the solution quality, we depict the discovered front by each algorithm under different number of jobs. We provide charts for four instances of MK series with n = 10, 15, 20, 30 and three instances of DP series. Figures 4 and 5 give the intuitive illustration for results derived from the performance metrics analyses and show the conflicting between two objectives of the problem.

Table 3. The computational results of five algorithms on ρ.

Instance	SFLA	SFLA1	SFLA2	VNS	NSGA-II
MK1	0.9375	0.0000	0.0000	0.0000	0.0625
MK2	0.7500	0.0000	0.0000	0.0000	0.2500
MK3	0.7500	0.0000	0.0000	0.0000	0.2500
MK4	0.8461	0.0000	0.0000	0.0000	0.1539
MK5	0.2000	0.0000	0.0000	0.0000	0.8000
MK6	0.5161	0.0000	0.0000	0.0000	0.4839
MK7	0.5625	0.0000	0.0000	0.0000	0.4375
MK8	0.8571	0.0000	0.0000	0.0000	0.1429
MK9	0.6875	0.0000	0.0000	0.0000	0.3125
MK10	0.7391	0.0000	0.0000	0.0000	0.2609
MK11	0.8500	0.0000	0.0000	0.0000	0.1500
MK12	0.7692	0.0000	0.0000	0.0000	0.2308
MK13	0.7096	0.0000	0.0000	0.0000	0.2904
DP1	0.8571	0.0000	0.0000	0.0000	0.1429
DP2	0.2727	0.0000	0.0000	0.0000	0.7273
DP3	0.8947	0.0000	0.0000	0.0000	0.1053
DP4	0.8181	0.0000	0.0000	0.0000	0.1819
DP5	0.6785	0.0000	0.0000	0.0000	0.3215
DP6	0.9230	0.0000	0.0000	0.0000	0.0670
DP7	0.8518	0.0000	0.0000	0.0000	0.1482
DP8	0.7692	0.0000	0.0000	0.0000	0.2308
DP9	0.2500	0.0000	0.0000	0.0000	0.7500
DP10	0.8333	0.0000	0.0000	0.0000	0.1667
DP11	0.7058	0.0000	0.0000	0.0000	0.2942
DP12	0.7418	0.0000	0.0000	0.0000	0.2582
DP13	0.8889	0.0000	0.0000	0.0000	0.1111
DP14	0.7241	0.0000	0.0000	0.0000	0.2759
DP15	0.7750	0.0000	0.0000	0.0000	0.2250
DP16	0.6129	0.0000	0.0000	0.0000	0.3871
DP17	0.6486	0.0000	0.0000	0.0000	0.3514
DP18	0.7600	0.0000	0.0000	0.0000	0.2400
LA21	0.4145	0.3411	0.0000	0.0000	0.2513
LA22	0.1004	0.0350	0.0000	0.0000	0.8612
LA23	0.0400	0.2400	0.0000	0.0000	0.7200
LA24	0.0350	0.0000	0.0000	0.0000	0.9650
LA25	0.1165	0.0900	0.0000	0.0000	0.7965
LA26	0.8214	0.0000	0.0000	0.0000	0.1786
LA27	0.8235	0.0000	0.0000	0.0000	0.1765
LA28	0.8885	0.0000	0.0000	0.0000	0.1115
LA29	0.9705	0.0000	0.0000	0.0000	0.0295
LA30	0.8108	0.0000	0.0000	0.0000	0.1891

Table 4. Computational time of five algorithms.

Instance	SFLA	SFLA1	SFLA2	VNS	NSGA-II
MK1	3.009	3.058	2.836	3.254	4.629
MK2	3.179	3.276	2.930	3.358	4.668
MK3	9.139	8.488	7.477	10.26	9.419
MK4	4.564	4.625	4.095	5.292	5.924
MK5	6.359	6.325	5.937	7.609	7.924
MK6	8.091	8.003	7.167	9.700	9.044
MK7	5.601	5.297	5.083	7.035	6.978
MK8	16.61	16.43	14.48	17.72	16.36
MK9	16.43	15.91	14.45	20.80	16.50
MK10	16.16	15.32	14.19	17.40	16.25
MK11	16.17	13.99	12.36	13.84	17.40
MK12	15.88	16.32	13.75	16.15	15.95
MK13	15.88	16.41	13.71	16.95	16.18
DP1	12.19	12.52	11.48	12.58	13.44
DP2	12.37	12.74	11.57	13.15	13.98
DP3	12.45	12.08	11.54	13.52	13.60
DP4	12.35	12.18	11.16	12.81	13.44
DP5	12.35	12.43	11.42	12.79	13.80
DP6	12.40	12.32	11.76	13.21	13.60
DP7	22.71	21.12	20.28	23.12	21.83
DP8	22.86	21.29	20.10	23.86	22.37
DP9	22.03	21.12	19.41	25.23	22.23
DP10	22.58	20.88	19.72	23.01	22.72
DP11	22.10	21.14	19.49	22.97	22.66
DP12	22.76	20.80	19.37	23.02	22.76
DP13	53.10	50.79	46.49	60.21	56.33
DP14	51.56	48.26	45.27	60.36	55.66
DP15	52.39	48.77	46.75	61.80	56.31
DP16	50.97	50.79	48.50	62.65	57.15
DP17	52.88	49.24	45.92	63.71	53.65
DP18	50.76	49.20	46.77	62.11	53.59
LA21	8.846	8.795	8.675	9.354	9.937
LA22	8.829	8.787	8.741	9.463	9.812
LA23	8.851	8.766	8.739	9.412	9.910
LA24	8.858	8.780	8.722	9.409	9.932
LA25	8.844	8.786	8.734	9.369	9.895
LA26	12.65	12.34	12.25	13.15	13.62
LA27	12.67	12.30	12.21	13.17	13.58
LA28	12.52	12.28	12.15	13.20	13.46
LA29	12.53	12.40	12.23	13.19	13.78
LA30	12.81	12.41	12.24	13.18	13.98

5.2.2 *Effect of memeplex construction on the performance of SFLA*

The comparison between SFLA and SFLA1 is to test the impact of memeplex construction on the performance. As shown in Tables 2 and 3, SFLA notably performs better than SFLA1 on all instances. SFLA1 cannot provide any members for the set Ω^* and the solutions of SFLA1 locate far away from those one in Ω^*. In SFLA1, memeplexes are obtained by dividing solutions of P. The solutions of $\bar{P}$are used to construct memeplexes, as a result, members of Ω have more chances to be assigned into memeplex and guide the search of SFLA. This feature of SFLA make it converge better than SFLA1, thus, memeplex construction is a vital step of SFLA.

5.2.3 *Effect of optimisation object*

It can be found from Tables 2 and 3 that SFLA2 is inferior to SFLA. SFLA2 still cannot provide any members of the set Ω^* and $DI_R(\Omega_3)$ is greater than $DI_R(\Omega_1)$ by at least 10 percentage for most of instances. The search process of SFLA2 is always done on a dominated solution x_w and this feature leads to the low convergence rate because the evolution rate of population is always done in a low level; on the contrary, in SFLA, a non-dominated member x_b of $\mathcal{M}_i$ is

Table 5. Paired-sample *t*-test for five algorithms.

t-test experiment	*p*-value ($DI_R(\Omega)$)	*p*-value (ρ)
t-test (SFLA, SFLA1)	0.0000	0.0000
t-test (SFLA, SFLA2)	0.0000	0.0000
t-test (SFLA, VNS)	0.0000	0.0000
t-test (SFLA, NSGA-II)	0.0000	0.0000

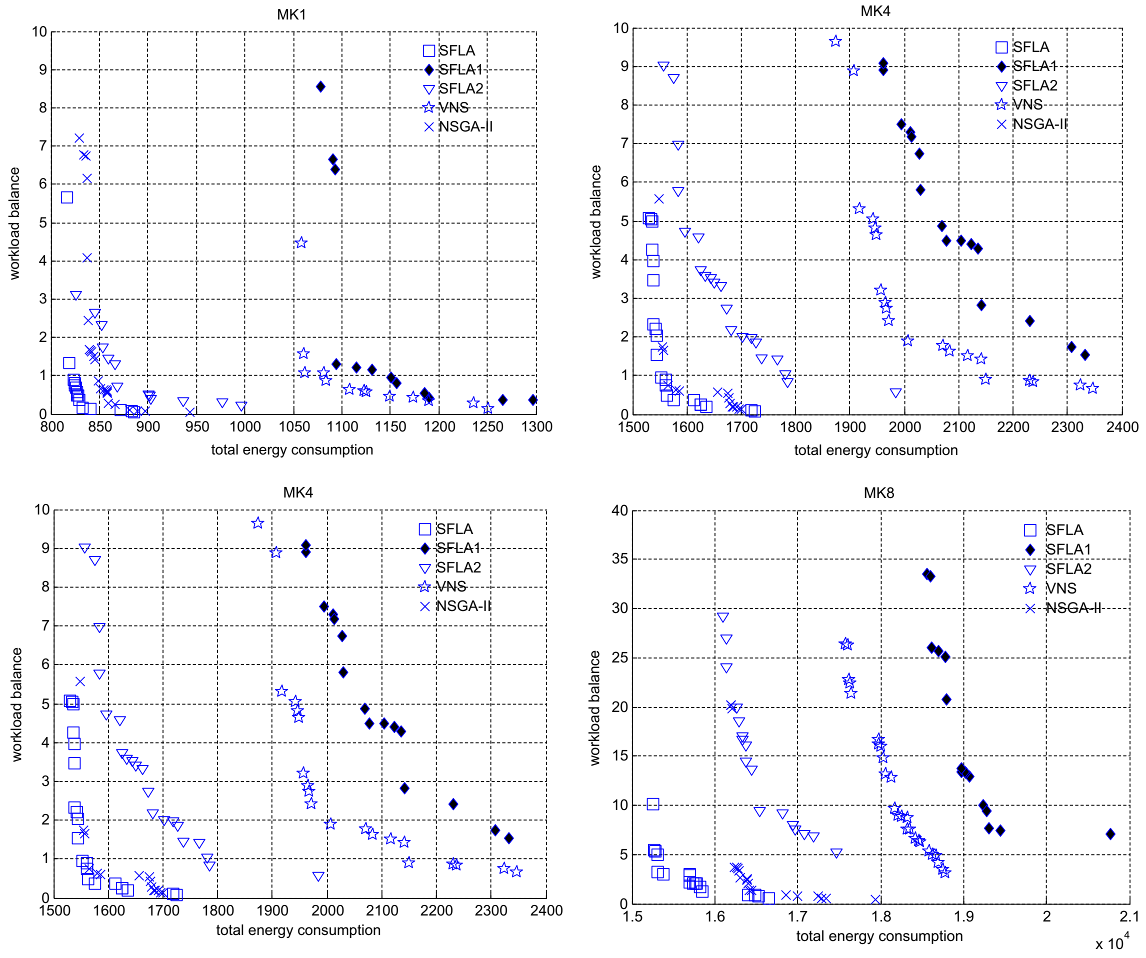

Figure 4. Distribution of the obtained non-dominated solutions of five algorithms for four MK instances.

the object of search process and there is more than one x_b, these features result in the good exploration and high diversity of SFLA, thus, the selection of optimisation object is an importance part of SFLA.

5.2.4 *Comparisons among three algorithms*

As shown in Tables 2–4, SFLA generates better results than NSGA-II and VNS using less time. VNS cannot converge to any members of the set Ω^* and its solutions are always far away from the solutions in Ω^*. SFLA generates most members of Ω^* on 32 instances, NSGA-II obtain most members of Ω^* for only 7 instances; moreover, $DI_R(\Omega_5)$ is always bigger than $DI_R(\Omega_1)$ by at least 3% on 25 instances. Figures 4 and 5 also show the differences on the location of solutions produced by SFLA, VNS and NSGA-II.

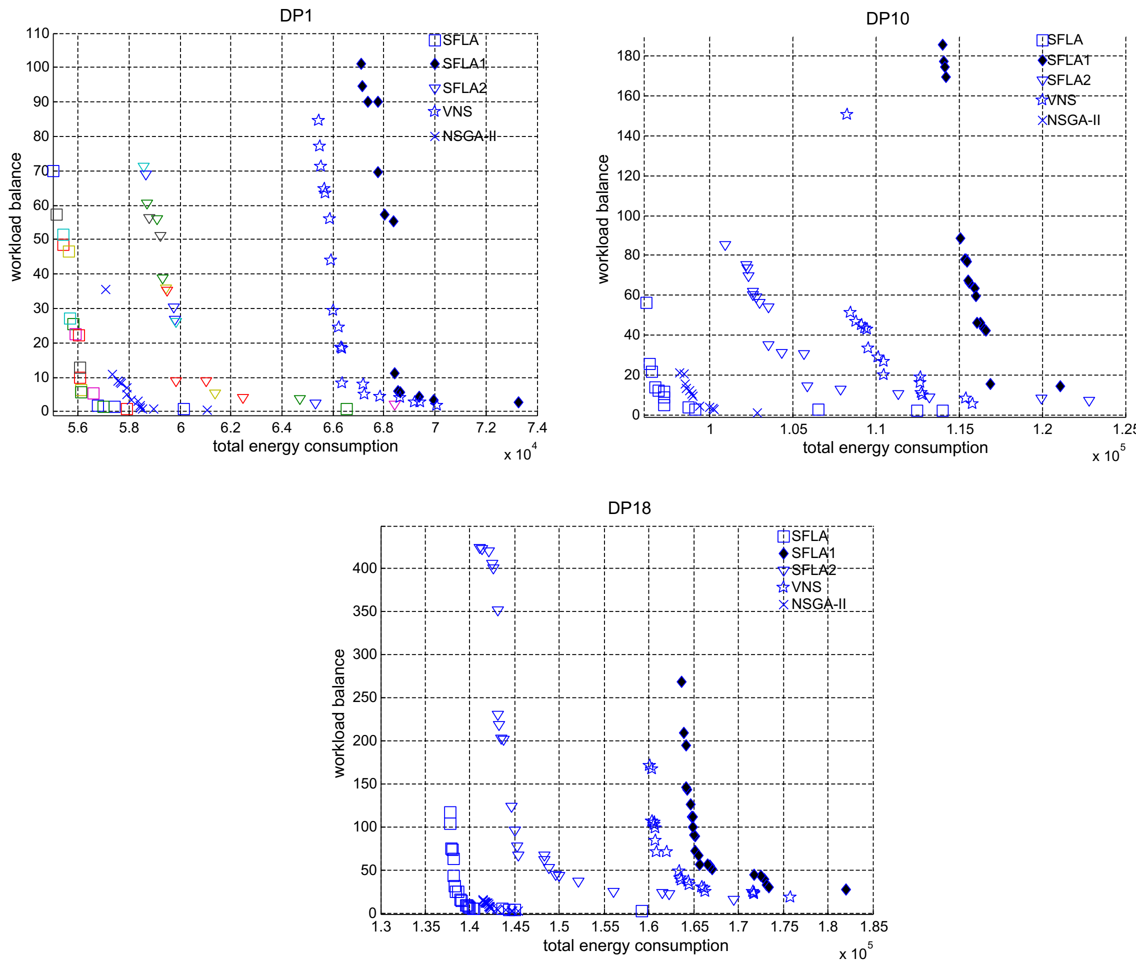

Figure 5. Distribution of the obtained non-dominated solutions of five algorithms for three DP instances.

The statistical results in Table 5 also reveal that SFLA outperforms VNS and NSGA-II in the two metrics. This conclusion is consistent with the above analyses on the performance comparison of three algorithms, thus, further justifies the advantages of SFLA.

In Figures 4 and 5, the distribution of the generated solutions shows that the conflicting really exists between two objectives. Take MK8 as an example, when SFLA obtains a solution with the biggest workload balance, the solution has the smallest energy consumption; moreover, it can be found that workload balance decreases with the increase in energy consumption when we scan solutions of three algorithms from left to right.

As stated in Sections 5.2.2 and 5.2.3, the good performance of SFLA mainly results from its memeplex construction and its selection on the optimisation object in each memeplex. These strategies make it has good exploration, high diversity and good balance between exploration and exploitation. The above results and analyses demonstrate that SFLA has promising advantages on the considered FJSP.

6. Conclusions

FJSP has been extensively considered in the past decade; however, these results are mainly about the optimisation of time-related indices such as total tardiness. Energy-related objective is seldom coped with in flexible job shop. In this study FJSP with total energy consumption and workload balance is investigated, in which the conflicting between two objectives is analysed and shown. A novel SFLA is proposed based on a three-string coding method. Population and the non-dominated set are used to construct memeplexes according to tournament selection. The search process within each memeplex is done on its non-dominated member. Extensive experiments are conducted and the results show the promising advantages of SFLA on the considered FJSP.

We will continue to focus on FJSP in the near future, for example, we pay attention to FJSP with four objectives such as total tardiness, makespan, total workload and total energy consumption and find a good way to deal with these objectives, which lead to very high optimisation difficulties. The applications of SFLA to scheduling problems such as distributed scheduling is also our future main topics.

Acknowledgement

This paper is supported by the National Natural Science Foundation of China (61573264, 71471151).

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Ahmadi, E., M. Zandieh, M. Farrokh, and S. M. Emami. 2016. “A Multi Objective Optimization Approach for Flexible Job Shop Scheduling Problem under Random Machine Breakdown by Evolutionary Algorithms.” *Computers and Operations Research* 73 (1): 56–66.

Brandimarte, P. 1993. “Routing and Scheduling in a Flexible Job Shop by Tabu Search.” *Annals of Operations Research* 41 (3): 157–183.

Chiang, T.-C., and H.-J. Lin. 2013. “A Simple and Effective Evolutionary Algorithm for Multiobjective Flexible Job Shop Scheduling.” *International Journal of Production Economics* 141 (1): 87–98.

Dai, M., D. B. Tang, A. Giret, M. A. Salido, and W. D. Li. 2013. *Energy-Efficient Scheduling for a Flexible Flow Shop Using an Improved Genetic-Simulated Annealing Algorithm Robotics and Computer-Integrated Manufacturing* 29 (2): 418–429.

Dauzère-Pérès, S., and J. Paulli. 1997. “An Integrated Approach for Modeling and Solving the General Multiprocessor Job-shop Scheduling Problem Using Tabu Search.” *Annals of Operations Research* 70: 281–306.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. 2002. “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II.” *IEEE Transactions on Evolutionary Computation* 6 (2): 182–197.

Ding, J. Y., S. J. Song, and C. Wu. 2016. “Carbon-efficient Scheduling of Flow Shops by Multi-objective Optimization.” *European Journal of Operational Research* 248 (3): 758–771.

Eusuff, M. M., K. E. Lansey, and F. Pasha. 2006. “Shuffled Frog-leaping Algorithm: A Memetic Meta-heuristic for Discrete Optimization.” *Engineering Optimization* 38 (2): 129–154.

Fang, C., and L. Wang. 2013. “An Effective Shuffled Frog-leaping Algorithm for Resource Constrained Project Scheduling Problem.” *Computers and Operations Research* 39 (5): 890–901.

Fang, K., N. Uhan, F. Zhao, and J. W. Sutherland. 2011. “A New Approach to Scheduling in Manufacturing for Power Consumption and Carbon Footprint Reduction.” *Journal of Manufacturing Systems* 30 (4): 234–240.

Fang, K., N. A. Uhan, F. Zhao, and J. W. Sutherland. 2013. “Flow Shop Scheduling with Peak Power Consumption Constraints.” *Annals of Operations Research* 206 (1): 115–145.

Gahm, C., F. Denz, M. Dirr, and A. Tuma. 2016. “Energy-efficient Scheduling in Manufacturing Companies: A Review and Research Framework.” *European Journal of Operational Research* 248 (3): 744–757.

Hurink, J., B. Jurisch, and M. Thole. 1994. "Tabu Search for the Job-Shop Scheduling Problem with Multi-Purpose Machines." *Operational Research Spectrum* 15 (4), 205–215.

Ishibuchi, H., T. Yoshida, and T. Murata. 2003. “Balance between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling.” *IEEE Transactions on Evolutionary Computation* 7: 204–223.

Lei, D. M., and X. P. Guo. 2015a. “An Effective Neighborhood Search for Scheduling in Dual-resource Constrained Interval Job Shop with Environmental Objective.” *International Journal of Production Economics* 159: 296–303.

Lei, D. M., and X. P. Guo. 2015b. “A Shuffled Frog-leaping Algorithm for Hybrid Flow Shop Scheduling with Two Agents.” *Expert Systems with Applications* 42 (23): 9333–9339.

Lei, D. M., and X. P. Guo. Forthcoming. “A Shuffled Frog Leaping Algorithm for Job Shop Scheduling with Outsourcing Options.” International Journal of Production Research.

Li, J. Q., Q. K. Pan, P. N. Suganthan, and T. J. Chua. 2011. “A Hybrid Tabu Search Algorithm with an Efficient Neighborhood Structure for the Flexible Job Shop Scheduling Problem.” *The International Journal of Advanced Manufacturing Technology* 52 (5–8): 683–697.

Li, J. Q., Q. K. Pan, and S. X. Xie. 2012. “An Effective Shuffled Frog-leaping Algorithm for Multi-objective Flexible Job Shop Scheduling Problems.” *Applied Mathematics and Computation* 218 (18): 9395–9371.

Li, J. Q., Q. K. Pan, and M. F. Tasgetiren. 2014. “A Discrete Artificial Bee Colony Algorithm for the Multi-objective Flexible Job-shop Scheduling Problem with Maintenance Activities.” *Applied Mathematical Modelling* 38 (3): 1111–1132.

Lin, W. W., D. Y. Yu, C. Y. Zhang, X. Liu, S. Q. Zhang, Y. H. Tian, S. Q. Liu, and Z. P. Xie. 2015. “A Multi-objective Teaching−Learning-based Optimization Algorithm to Scheduling in Turning Processes for Minimizing Makespan and Carbon Footprint.” *Journal of Cleaner Production* 101: 337–347.

Liu, G. S., B. X. Zhang, D. D. Yang, X. Chen, and G. Q. Huang. 2013. “A Branch-and-bound Algorithm for Minimizing the Energy Consumption in the PFS Problem.” *Mathematical Problems in Engineering*.

Luo, H., B. Du, G. Q. Huang, H. P. Chen, and X. L. Li. 2013. “Hybrid Flow Shop Scheduling Considering Machine Electricity Consumption Cost.” *International Journal of Production Economics* 146 (2): 423–439.

Mansouri, S. A., E. Aktas, and U. Besikci. 2016. “Green Scheduling of a Two-machine Flowshop: Trade-off between Makespan and Energy Consumption.” *European Journal of Operational Research* 248 (3): 772–788.

Mattfeld, D. C., and C. Bierwirth. 2004. “An Efficient Genetic Algorithm for Job Shop Scheduling with Tardiness Objectives.” *European Journal of Operational Research* 155 (3): 616–630.

May, G., B. Stahl, M. Taisch, and V. Prabhu. 2015. “Multi-objective Genetic Algorithm for Energy- efficient Job Shop Scheduling.” *International Journal of Production Research* 53 (23): 7071–7089.

Mouzon, G., and M. B. Yildirim. 2008. “A Framework to Minimise Total Energy Consumption and Total Tardiness on a Single Machine.” *International Journal of Sustainable Engineering* 1 (2): 105–116.

Mouzon, G., M. B. Yildirim, and J. Twomey. 2007. “Operational Methods for Minimization of Energy Consumption of Manufacturing Equipment.” *International Journal of Production Research* 45 (18–19): 4247–4271.

Pan, Q. K., L. Wang, L. Gao, and J. Q. Li. 2011. “An Effective Shuffled Frog-leaping Algorithm for Lot-streaming Flow Shop Scheduling Problem.” *The International Journal of Advanced Manufacturing Technology* 52 (5-8): 699–713.

Rahimi-Vahed, A., M. Dangchi, H. Rafiei, and E. Salimi. 2009. “A Novel Hybrid Multi-objective Shuffled Frog-leaping Algorithm for a Bi-criteria Permutation Flow Shop Scheduling Problem.” *The International Journal of Advanced Manufacturing Technology* 41 (11–12): 1227–1239.

Shrouf, F., J. Ordieres-Meré, A. García-Sánchez, and M. Ortega-Mier. 2014. “Optimizing the Production Scheduling of a Single Machine to Minimize Total Energy Consumption Costs.” *Journal of Cleaner Production* 67: 197–207.

Tang, D. B., M. Dai, M. A. Salido, and A. Giret. 2015. “Energy-efficient Dynamic Scheduling for a Flexible Flow Shop Using an Improved Particle Swarm Optimization.” *Computers in Industry* 30 (1): 223–232.

Wang, L., S. Y. Wang, and M. Liu. 2013. “A Pareto-based Estimation of Distribution Algorithm for the Multi-objective Flexible Job-shop Scheduling Problem.” *International Journal of Production Research* 51 (12): 3574–3592.

Yildirim, M. B., and G. Mouzon. 2012. “Single-machine Sustainable Production Planning to Minimize Total Energy Consumption and Total Completion Time Using a Multiple Objective Genetic Algorithm.” *IEEE Transactions on Engineering Management* 59 (4): 585–597.

Yuan, Y., and H. Xu. 2015. “Multiobjective Flexible Job Shop Scheduling Using Memetic Algorithms.” *IEEE Transactions on Automation Science and Engineering* 12 (1): 336–353.

Zhang, R., and R. Chiong. 2016. “Solving the Energy-efficient Job Shop Scheduling Problem: A Multi-objective Genetic Algorithm with Enhanced Local Search for Minimizing the Total Weighted Tardiness and Total Energy Consumption.” *Journal of Cleaner Production* 112: 3361–3375.