Salp Swarm Algorithm Based on Blocks on Critical Path for Reentrant Job Shop Scheduling Problems

Zai-Xing Sun[1], Rong Hu[1](✉), Bin Qian[1], Bo Liu[2], and Guo-Lin Che[1]

[1] School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China
ronghu@vip.163.com

[2] Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China

Abstract. In this paper, salp swarm algorithm based on blocks on critical path (SSA_BCP) is presented to minimize the makespan for reentrant job shop scheduling problem (RJSSP), which is a typical NP-complete combinational optimization problem. Firstly, the mathematical model of RJSSP based on the disjunctive graph is established. Secondly, the extended reentrant-smallest-order-value (RSOV) encoding rule is designed to transform SSA's individuals from real vectors to job permutations so that SSA can be used to perform global search for finding high-quality solutions or regions in the solution space. Thirdly, four kinds of neighborhood structures are defined after defining the insert operation based on block structure, which can be used to avoid search in the invalid regions. Then, a high-efficient local search integrating multiple neighborhoods is proposed to execute a thorough search from the promising regions found by the global search. Simulation results and comparisons show the effectiveness of the proposed algorithm.

Keywords: Reentrant job shop scheduling problem · Makespan
Salp swarm algorithm · Block structure

1 Introduction

Job shop scheduling problem (JSSP) is one of the most typical and most difficult combinatorial optimization problems [1]. Garey et al. [2] has proved that JSSP is a NP-complete problem when the number of machines exceeds two. Its algorithm research has always been an important topic of common concern in academia and engineering. JSSP is limited to machining or manufacturing parts that can only be machined once on a single machine. In the production systems of high-tech industries such as semiconductor manufacturing and computer networks, some parts are required to be machined many times by the same machine. Therefore, the research of the reentrant job shop scheduling problem (RJSSP) has important engineering value and practical significance. Since JSSP has NP-complete property, and JSSP can be reduced to RJSSP, RJSSP is also NP-complete. Therefore, the research of RJSSP for minimizing the makespan has high practical and theoretical value, which can provide feasible guidance and help for relevant production enterprises.

As far as JSSP is concerned, the design of neighborhood structure is very important for the performance of algorithm, and it also has some universality. Therefore, the research is of great significance. Some scholars have studied the neighborhood structure of JSSP with the goal of minimizing makespan. Zhang et al. [3] were respectively proposed the neighborhoods N7 on the basis of predecessors. Zhang and Wu [4] proposed a neighborhood structure to solve the JSSP with the goal of minimizing total weighted tardiness. And using the property of the dual model of the problem, [4] was proved that the proposed neighborhood structure can not improve the target value when satisfying certain constraints, and effectively enhances the local search efficiency of the algorithm. Kuhpfahl and Bierwirth [5] proposed six new neighborhood structures, and the simulation experiments compared the performance and search capability of different structures and combinations. Then Bierwirth and Kuhpfahl [6] derived six further neighborhoods from [5]. By adopting a suitable neighborhood structure, the above-mentioned algorithms have achieved good performance. However, there is no relevant research on the effective neighborhood structure of RJSSP.

For RJSSP, Topaloglu and Kilincli [7] proposed a modified shifting bottleneck heuristic to solve the RJSSP with makespan minimization, which shows that the proposed algorithm is very effective compared with the simulation of other algorithms. Qian et al. [8] proposed a hybrid differential evolution algorithm for solving multi-objective RJSSP with total machine idleness and maximum tardiness criteria. Simulation results and comparisons show the effectiveness of the proposed algorithm. Thus it is very important to develop effective algorithms for this kind of problem.

Mirjalili et al. [9] first proposed a new meta-heuristic swarm intelligence optimization salp swarm algorithm (SSA) based on the swarming behaviour of salps when navigating and foraging in oceans. SSA is novel algorithm that attracts the attention in many applications as engineering design problem [9] and Electrical Engineering problem [10]. In Ref. [9], the authors indicate that the cropped results in comparison to other challenging optimizers demonstrate the value of the SSA in solving optimization problems with hard and unidentified search spaces. On top of that, to the author's knowledge, this is a novel application and first time to use the SSA in solving the scheduling problem, so it is very important to carry out relevant research.

In this paper, a SSA based on blocks on critical path (SSA_BCP) is proposed to minimize makespan of the RJSSP. In SSA_BCP, SSA is used to obtain promising regions over the solution space, and a problem-dependent local search with different neighborhoods is applied to perform exploitation from these regions. Simulation results and comparisons demonstrate the effectiveness of the proposed SSA_BCP. The rest of this paper is arranged as follows.

In Sect. 2, the RJSSP description using disjunctive graph representation is briefly introduced. In Sect. 3, a brief review of SSA is provided. In Sect. 4, SSA_BCP is proposed and described in detail. In Sect. 5, simulation results and comparisons are provided. Finally, we end the paper with some conclusions and future work in Sect. 6.

2 Problem Description Using Disjunctive Graph Representation

The RJSSP can be described as follows: given the processing time p_v of operation v, each of n jobs will be processed r times on m machines. At any time, each machine can process at most one job and each job can be processed on at most one machine.

The disjunctive diagram was proposed by Roy and Sussman in 1964 to describe the process sequence constraints and machine uniqueness constraints in the scheduling problem. RJSSP can be represented by the disjunctive graph $G = (N, A, E, Re)$. A simple disjunctive graph for RJSSP is shown in Fig. 1. $N = O \cup \{0\} = \{0, 1, 2, \cdots, TO\}$ is the set of nodes. Node 0 stands for a dummy operation which starts before all the real operations, while the starting time and the processing time of this dummy operation are both zero. $TO = n \times m \times r$ is the total number of operations. A is the set of unidirectional conjunctive arcs, which describes the order constraints of the same job process route. E is the set of disjunctive arcs, where represents the disjunctive arcs related with one machine. The disjunctive arc orientation is not determined before scheduling, for bi-directional. Re is a set of reentrant arcs that is first proposed for reentrant properties which describes the processing constraints of the same job on the same machine and is unidirectional. U is the unidirectional tail arc, describing the last operation of each job pointing to node 0.

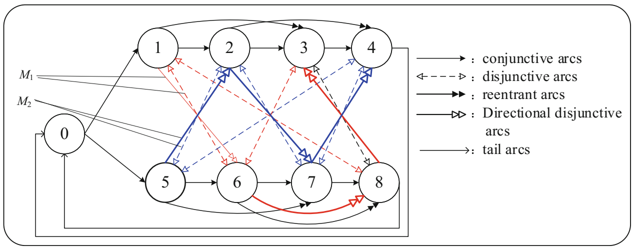

Fig. 1. The disjunctive graph of RJSSP with $2 \times 2 \times 2$

Let s_v be the starting processing time of the operation v, in which $s_0 = 0$. p_v is the processing time of the operation v, in which $p_0 = 0$. μ_j is the last operation of job j. (v, w) is the arc with processing constraints. $\langle v, w \rangle$ is a disjunction arc in an undefined direction. '$\vee$' is a logical OR, describing the operation of the arc in the disjunction. $\mathrm{C}_{\max}$ is the makespan. The disjunctive planning model of RJSSP is described as follows:

$$Min \quad \mathrm{C}_{\max} \tag{1}$$

$$s.t. \quad s_v + p_v \leq s_w \qquad \forall (v, w) \in A \tag{2}$$

$$(s_v + p_v \leq s_w) \vee (s_w + p_w \leq s_v) \quad \forall \langle v, w \rangle \in E \tag{3}$$

$$s_v + p_v \leq s_w \qquad \forall (v, w) \in Re \tag{4}$$

$$\mathrm{C}_{\max} \geq s_{\mu_j} + p_{\mu_j} \qquad \forall \mu_j \in U(O) \tag{5}$$

In the above model, $s_v \geq 0\,(\forall v \in O)$ can be obtained by $s_0 = p_0 = 0$ and formula (2), so this constraint does not need to be given separately.

The important scheduling decision is to determine for each pair of operations that require the same machine, whether one is performed before the other, or the other way around. This decision is often referred to as a selection: for each of the disjunctive constraints (3), either $s_v + p_v \leq s_w$ or $s_w + p_w \leq s_v$ is selected. A selection, denoted by σ, transforms RJSSP into a linear programming problem. Let σ be the collection of ordered pairs of operations (v, w) that require the same machine, and for which it has been decided that v must be performed before w. The linear program is then as follows.

$$Min \quad \mathrm{C}_{\max} \tag{6}$$

$$s.t. \quad s_v + p_v \leq s_w \qquad \forall (v, w) \in A \tag{7}$$

$$s_v + p_v \leq s_w \qquad \forall (v, w) \in \sigma \tag{8}$$

$$s_v + p_v \leq s_w \qquad \forall (v, w) \in Re \tag{9}$$

$$\mathrm{C}_{\max} \geq s_{\mu_j} + p_{\mu_j} \qquad \forall \mu_j \in U(O) \tag{10}$$

3 Brief Review of SSA

SSA is a branch of meta-heuristic proposed by Mirjalili et al. [9] for optimization problems over continuous domains. To mathematically model the salp chains, the population is first divided to two groups: leader and followers. The leader is the salp at the front of the chain, whereas the rest of salps are considered as followers. As the name of these salps implies, the leader guides swarm and the followers follow each other (and leader directly of indirectly). Similarly to other swarm-based techniques, the position of salps is defined in an n-dimensional search space where n is the number of variables of a given problem. Therefore, the position of all salps are stored in a two-dimensional matrix called X. It is also assumed that there is a food source called F in the search space as the swarm's target. The details of SSA can be found in Subsect. 4.4.

4 SSA_BCP for RJSSP

In this section, we will present SSA_BCP for RJSSP after explaining the solution representation and decoding scheme, problem-dependent neighborhood structure, problem-dependent local search.

4.1 Solution Representation and Decoding Scheme

Because of the continuous characters of the individuals in SSA, a canonical SSA cannot be directly applied to the considered problem in this paper. So, we extend the reentrant-smallest-order-value (RSOV) [8] encoding rule to convert SSA's *i*th individual $\boldsymbol{X}_i = \left[x_{i,1}, x_{i,2}, \cdots, x_{i,TO}\right]$ to the job permutation vector $\boldsymbol{\pi}_i = \left[\pi_{i,1}, \pi_{i,2}, \cdots, \pi_{i,TO}\right]$ and disjunctive graph operation permutation vector $\boldsymbol{\varphi}_i = \left[\varphi_{i,1}, \varphi_{i,2}, \cdots, \varphi_{i,TO}\right]$, where $\varphi_{i,j} = \left(\pi_{i,j} - 1\right) \times m \times r + z_{\pi_{i,j}}$, $z_{\pi_{i,j}}$ is the $z_{\pi_{i,j}}$th occurrence of job $\pi_{i,j}$ in $\boldsymbol{\pi}_i$. Table 1 illustrates the representation of vector $\boldsymbol{X}_i$ in SSA for a simple problem $(n = 3, m = 2, r = 2)$.

Table 1. Solution representation

Dimension k	1	2	3	4	5	6	7	8	9	10	11	12
$x_{i,k}$	1.51	2.83	3.25	2.30	1.69	2.57	3.83	0.61	2.40	3.08	0.11	1.03
$y_{i,k}$	4	9	11	6	5	8	12	2	7	10	1	3
$\pi_{i,k}$	1	3	3	2	2	2	3	1	2	3	1	1
$\varphi_{i,k}$	1	9	10	5	6	7	11	2	8	12	3	4

According to the properties of RJSSP, the set of active schedules is a subset of the semi-active ones and an optimal solution must be an active schedule to minimize the makespan. Therefore, when decoding we check the possible idle interval before appending an operation at the last position, and fill the first idle interval before the last operation (called active decoding). In order to adapt to the operation of the block structure, when the scheduling is completed, the process is sorted by the completion time from the smallest to the largest, and the obtained new solution replaces the old solution as the current solution output.

4.2 Problem-Dependent Neighborhood Structure

The design of the neighborhood structure is an important research issue in the field of combinatorial optimization [11]. The effective neighborhood structure can significantly speed up the convergence of the algorithm and improve the search performance of the algorithm.

Definition 1 (Critical path): based on the Machine Predecessor First (MPF) rule [4], the longest path from the last completed operation to the virtual node 0.

Definition 2 (Block structure): a sequence of operations in a critical path is called a block structure if it contains at least two operations and not all belongs to one job.

Definition 3 (Insert operation based on block structure): in disjunctive graph operation permutation vector $\boldsymbol{\varphi}_i$, when the operation v in the block structure is inserted before (after) the operation w, all the operations between v and w that belong to the same job as v are inserted before (after) the operation w.

Neighborhood Structure 1 (NS1): moving the first operation of the block structure into the internal operation within the block structure.

Neighborhood Structure 2 (NS2): moving the last operation of the block structure into the internal operation within the block structure.

Neighborhood Structure 3 (NS3): in disjunctive graph operation permutation vector, the first operation on the block structure is inserted to its previous random operation.

Neighborhood Structure 4 (NS4): in disjunctive graph operation permutation vector, the last operation on the block structure is inserted in its later random operation.

The four neighborhood structures are illustrated in Fig. 2.

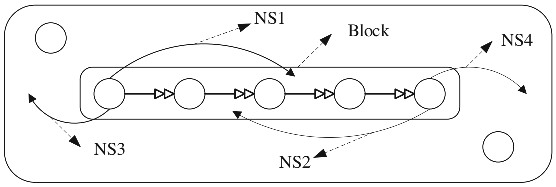

Fig. 2. The four neighborhood structures

4.3 Problem-Dependent Local Search

We design a problem dependent local search based on the neighborhood structure in the above subsection to enrich the search behavior and enhance the search ability. The perturbation phase is based on the following two neighborhoods which are often employed in the literature. (1) remove the job at the vth dimension and insert it in the wth dimension of the solution $\boldsymbol{\varphi}(insert(\boldsymbol{\varphi}, v, w))$, (2) invert the subsequence between the vth dimension and the wth dimension of the solution $\boldsymbol{\varphi}(inverse(\boldsymbol{\varphi}, v, w))$. Let *Block_num* be the numbers of block structure. $Block\_n_i$ is the numbers of operation in ith block structure. $Block_{i,j}$ is ith operation in jth block structure. ***LBest_φ*** is the optimal solution currently searched. The procedure of SSA_BCP's local search is given as follows:

```
Step 1: Convert SSA's individual $X_i$ to disjunctive graph operation permutation vector $\varphi_{i\_0}$ according to the RSOV rule;
Step 2: Set $kk = 0$, $\varphi_{i\_pp} = \varphi_{i\_0}$ and kick-move strength ($ks$) = 10;
  Do
    Randomly select $v$ and $w$, where $v \neq w$;
    If $random < 0.8$ then $\varphi_i$ = insert($\varphi_{i\_pp}$, $v$, $w$), $\varphi_{i\_pp} = \varphi_i$;
    Else $\varphi_i$ = inverse($\varphi_{i\_pp}$, $v$, $w$), $\varphi_{i\_pp} = \varphi_i$;
    $kk = kk + 1$ ;
  While $kk < ks$;
Step 3: Statistics $Block\_num$, $Block\_n$ and $Block$ of $\varphi_i$, let $\varphi_{i\_1} = \varphi_i$, $LBest\_\varphi = \varphi_i$, $x = 1$;
Step 4: the $x$th block structure in $\varphi_{i\_1}$ search using NS1 to get $\varphi_{i\_2}$, if $\varphi_{i\_2} < LBest\_\varphi$, then $LBest\_\varphi = \varphi_{i\_2}$;
let $\varphi_{i\_1} = \varphi_i$, the $x$th block structure in $\varphi_{i\_1}$ search using NS2 to get $\varphi_{i\_2}$, if $\varphi_{i\_2} < LBest\_\varphi$, then $LBest\_\varphi = \varphi_{i\_2}$;
let $\varphi_{i\_1} = \varphi_i$, the $x$th block structure in $\varphi_{i\_1}$ search using NS3 to get $\varphi_{i\_2}$, if $\varphi_{i\_2} < LBest\_\varphi$, then $LBest\_\varphi = \varphi_{i\_2}$;
let $\varphi_{i\_1} = \varphi_i$, the $x$th block structure in $\varphi_{i\_1}$ search using NS4 to get $\varphi_{i\_2}$, if $\varphi_{i\_2} < LBest\_\varphi$, then $LBest\_\varphi = \varphi_{i\_2}$;
Step 5: Set $x = x + 1$. If $x \leq Block\_num$, then go to Step 4;
Step 6: Output $LBest\_\varphi$ and its objective value. Convert $LBest\_\varphi$ back to $X_i$.
```

As can be seen from the above procedure, step 2 is the perturbation phase, which is used to escape local optima and guide the search to a different region, and step 3–5 executes a thorough four neighborhood structures exploitation from the region obtained by step 2.

4.4 SSA_BCP

Based on the above subsections, the procedure of SSA_BCP is proposed as follows:

```
Step 1: Let $G$ denotes a generation, $P$ a population of size $PS$, and $x_i(t)$ the $i$th individual of dimension $TO$ in population $P$ in generation $t$, $x_{i,k}(t)$ the $k$th dimension of individual $x_i(t)$, $random(0,1)$ the random value in the interval [0,1], and $t\_max$ the maximum number of iteration;
Step 2: Input, $m$ $n$, $r$, $PS$. Initial bounds: $lb_k = -2$, $ub_k = 2$, $k = 1, 2, \cdots TO$;
Step 3: Set $t = 1$. Initialize the salp population $x_i$ considering low and upper. Calculate the fitness of each search agent (salp). $F$ = the best search agent;
Step 4: Update $c1$ by Eq.(11). Set $i = 1$;
Step 5: Set $c3$ = $random(0,1)$. If $c3 < 0.5$ or $i = 1$, then update the position of the leading salp by Eq.(12); else update the position of the follower salp by Eq.(13).
```

$$c1 = 2e^{-(4t/t\_max)^2} \tag{11}$$

$$x_{i,k} = \begin{cases} F_k + c1((ub_k - lb_k)c2 + lb_k) & c3 < 0.5 \\ F_k - c1((ub_k - lb_k)c2 + lb_k) & c3 \geq 0.5 \end{cases} \tag{12}$$

$$x_{i,k} = \frac{1}{2}(x_{i-1,k} + F_k) \tag{13}$$

```
Where $c2$ and $c3$ are random numbers;
Step 6: Amend the salps based on the upper and lower bounds of variables;
Step 7: Set $i = i + 1$. If $i \leq PS$, then go to Step 5;
Step 8: Calculate the fitness of each search agent (salp). $F$ = the best search agent.
Step 9: Apply the problem-dependent local search to $F$;
Step 10: Set $t = t + 1$. If $t \leq t\_max$, then go to Step 4;
Step 11: Output $F$.
```

The steps of the algorithm show that SSA used and found high quality solutions for global search in the solution space of regional problems, at the same time into local search for different neighborhood structures for high quality solutions for regional implementation further detailed search, the SSA_BCP in the global and local search to achieve a reasonable balance. SSA_BCP is expected to be an effective algorithm for solving this problem. The fifth section will verify the effectiveness of the proposed algorithm by simulation experiments and algorithm comparisons.

5 Simulation Results and Comparisons

In this section, we first perform experimental analysis based on the Design of Experiment (DOE) for the key parameters of the SSA_BCP to determine the optimal combination of the algorithm parameters. Then by comparing SSA_BCP with other effective algorithms, SSA_BCP is an effective algorithm for solving RJSSP. A set of instances under different scales is randomly generated. The $n \times m \times r$ combinations include $10 \times 10 \times 3$, $20 \times 5 \times 3$, $10 \times 20 \times 3$, $20 \times 10 \times 3$, $20 \times 15 \times 3$, $50 \times 6 \times 3$, $20 \times 20 \times 3$, $40 \times 10 \times 3$, $50 \times 10 \times 3$, $100 \times 5 \times 3$. The processing time is generated from a uniform distribution [1,99]. All algorithms are coded in Delphi10.2 and are executed on Intel Core I7-7700 k 4.20 GHz PC with 8 GB memory.

For the purpose of evaluating the effectiveness of SSA_BCP, we carry out simulations to compare our SSA_BCP with NIMGA [12]. NIMGA is an effective algorithm for JSSP. The performance of NIMGA is superior to a variety of effective algorithms.

For each instance, each algorithm is run 20 times independently, running at the same time each time. SSA_BCP's parameters are set as follows: the population size $PS = 10$, and the maximum number of iteration $t\_max = 200$. Other parameters of SSA or NIMGA set reference [9, 12]. *Min*, *Max*, *Avg* and *Std* are the best makespan, worst makespan, average makespan, and standard derivation of the 20 times. The optimal results for each problem are shown in bold, and the test results are shown in Table 2.

Table 2. Comparison of SSA_BCP and NIMGA

$n \times m \times r$	SSA_BCP				NIMGA			
	Min	*Max*	*Avg*	*Std*	*Min*	*Max*	*Avg*	*Std*
$10 \times 10 \times 3$	**2275**	2420	**2347.25**	41.22	2299	**2406**	2348.2	**23.57**
$20 \times 5 \times 3$	**3250**	3314	3262.75	18.53	**3250**	**3270**	**3253.1**	**6.24**
$10 \times 20 \times 3$	**3996**	4267	**4096.1**	69.55	4090	**4184**	4135.75	**28.02**
$20 \times 10 \times 3$	**3543**	**3757**	**3646.75**	54.46	3671	3845	3768.1	**44.91**
$20 \times 15 \times 3$	**4529**	**4760**	**4643.75**	57.08	4759	4975	4836.05	**51.8**
$50 \times 6 \times 3$	**8085**	**8085**	**8085**	**0**	**8085**	8094	8085.55	1.96
$20 \times 20 \times 3$	**5086**	**5400**	**5230.55**	80.89	5353	5595	5477.35	**58.98**
$40 \times 10 \times 3$	**6767**	**6887**	**6842.25**	**32.73**	6912	7115	7023.85	53.78
$50 \times 10 \times 3$	**8175**	**8323**	**8237.59**	41.26	8340	8453	8394.35	**26.70**
$100 \times 5 \times 3$	**15592**	**15592**	**15592**	**0**	**15592**	**15592**	**15592**	**0**

From Table 2, it can be seen that SSA_BCP performs much better than NIMGA with respecting to solution quality. The values of *Min* and *Avg* obtained by SSA_BCP are obviously better than that obtained by NIMGA for all instances except $20 \times 5 \times 3$. For *Max*, SSA_BCP is better than NIMGA on large-scale problems. For *Std*, although SSA_BCP is larger than NIMGA on several issues, its *Avg* is superior, so SSA_BCP is effective. For $50 \times 6 \times 3$ and $100 \times 5 \times 3$, their *Std* is 0, because the block structure of the current solution is only one and only on one machine, so the solution is the optimal solution. So, it can be concluded that SSA is an effective algorithm for the RJSSP. Moreover, the test results also manifest neighborhood structure based on blocks on critical path is more suitable for guiding the search to the promising regions in the solution space of RJSSP.

6 Conclusions and Future Research

This paper presented a salp swarm algorithm based on blocks on critical path (SSA_BCP) for the reentrant job shop scheduling problem (RJSSP). To the best of the current authors' knowledge, this is the first report on the application of salp swarm algorithm (SSA) approach to solve the problem considered. Firstly, considering that the problem has reentrant features, a reentry arc is added to the disjunctive graph model of Job Shop Scheduling Problem (JSSP), and the RJSSP mathematics planning model based on disassembly graph is established. Secondly, the extended reentrant-smallest-order-value (RSOV) encoding rule is designed so that SSA can be used to perform global search for finding high-quality solutions or regions in the solution space. Then, a high-efficient local search integrating multiple neighborhoods is proposed to execute a thorough search from the promising regions found by the global search. Simulation results and comparisons demonstrated the effectiveness of SSA_BCP. Future work includes developing of multi-objective SSA for the uncertain multi-objective RJSSP.

Acknowledgements. This research is partially supported by the National Science Foundation of China (51665025), and the Applied Basic Research Foundation of Yunnan Province (2015FB136).

References

1. Quanke, P., Ling, W., Liang, G., et al.: Differential evolution algorithm based on blocks on critical path for job shop scheduling problems. J. Mech. Eng. **46**(22), 182–188 (2010)
2. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. INFORMS (1976)
3. Zhang, C.Y., Li, P.G., Guan, Z.L., et al.: A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. Comput. Oper. Res. **34**(11), 3229–3242 (2007)
4. Zhang, R., Wu, C.: A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardiness objective. Comput. Oper. Res. **38**(5), 854–867 (2011)

5. Kuhpfahl, J., Bierwirth, C.: A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective. Elsevier Science Ltd. (2016)
6. Bierwirth, C., Kuhpfahl, J.: Extended GRASP for the job shop scheduling problem with total weighted tardiness objective. Eur. J. Oper. Res. **261**, 835–848 (2017)
7. Topaloglu, S., Kilincli, G.: A modified shifting bottleneck heuristic for the reentrant job shop scheduling problem with makespan minimization. Int. J. Adv. Manuf. Technol. **44**(7–8), 781–794 (2009)
8. Qian, B., Li, Z.H., Hu, R., et al.: A hybrid differential evolution algorithm for the multi-objective reentrant job-shop scheduling problem, pp. 485–489 (2013)
9. Mirjalili, S., Gandomi, A.H., et al.: Salp Swarm Algorithm: a bio-inspired optimizer for engineering design problems. Adv. Eng. Softw. **114**, 163–191 (2017)
10. Elfergany, A.A., Kalogirou, S.A., Christodoulides, P.: Extracting optimal parameters of PEM fuel cells using Salp Swarm Optimizer. Renew. Energy **119**, 641–648 (2018)
11. Abdel-Basset, M., Gunasekaran, M., El-Shahat, D., et al.: A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. Future Gener. Comput. Syst. **85**, 129–145 (2018)
12. Kurdi, M.: An effective new island model genetic algorithm for job shop scheduling problem. Comput. Oper. Res. **67**, 132–142 (2016)