Chapter 2
Multi-population and Self-adaptive Genetic Algorithm Based on Simulated Annealing for Permutation Flow Shop Scheduling Problem

Huimin Sun, Jingwei Yu and Hailong Wang

Abstract In order to solve the permutation flow shop scheduling problem, a multi-population and self-adaptive genetic algorithm based on simulated annealing is proposed in this paper. For the precocity problem of traditional genetic algorithm, the multi-population coevolution strategy is adopted. We introduce a squared term to improve traditional self-adaptive genetic operators, which can increase the searching efficiency and avoid getting into local optimum. A new cooling strategy is proposed to reinforce the ability of overall searching optimal solution. The algorithm is used to solve a series of typical Benchmark problems. Moreover, the results are compared with SGA, IGA, and GASA. The comparison demonstrates the effectiveness of the algorithm.

Keywords Permutation flow shop scheduling problem · Multi-population · Self-adaptive · Simulated annealing · Genetic algorithm

2.1 Introduction

Flow Shop Scheduling Problem (FSSP) aims at minimizing the completion time/makespan (or other criterion) for cases where n jobs are executed through m machines. It is the simplified model of many modern factory pipeline production scheduling problems and has extensive application both in the integrated manufacturing industry and in the processing industry. The FSSP has been extensively investigated in the last several decades [1], since first proposed by Johnson [2].

H. Sun (✉) · J. Yu · H. Wang
School of Astronautics Institution, Harbin Institute of Technology, Aviation University of Air Force, Changchun, 92 West Dazhi Street, Nan Gang District, Harbin 150001, China
e-mail: sunminggehuimin@163.com

Z. Deng and H. Li (eds.), *Proceedings of the 2015 Chinese Intelligent Automation Conference*, Lecture Notes in Electrical Engineering 338,
DOI 10.1007/978-3-662-46466-3_2

Permutation Flow Shop Scheduling Problem (PFSSP), with all machines processing their jobs following an identical route, is a special case of FSSP. It also belongs to an extremely complex and difficult combinational optimization problem. In the existing literature [3], the PFSSP has proved to be an NP-hard problem and is quite difficult to be solved. Therefore, it has important theoretical and practical significance to develop and study efficient algorithm.

A lot of algorithms have been proposed to solve the PFSSP with some certain optimum criterion (e.g., makespan). The computational intelligence algorithms have become a hot research topic such as ant colony algorithm [4], genetic algorithm (GA) [5], etc. All of them have better performance than traditional algorithms (e.g., dynamic programming, etc.).

In this paper, we propose multi-population and self-adaptive genetic algorithm based on simulated annealing (MSGASA). We adopt multi-population coevolution strategy to solve the precocity problem. We introduce a squared term to improve traditional self-adaptive genetic operators in order to increase the searching efficiency and avoid getting into local optimum. We proposed a new cooling strategy to reinforce the ability of overall searching optimal solution. A series of typical benchmark problems have been solved using the proposed algorithm and the results have been compared with simple GA (SGA), an improved GA (IGA) [6] and GA based on simulated annealing (GASA) [7]. The comparisons show the effectiveness of proposed algorithm.

2.2 Description of Permutation Flow Shop Scheduling Problem

2.2.1 Assumptions

In order to establish the mathematical model for PFFSP, the following assumptions are given at first.

- All jobs are processed by all machines in the same order.
- Each job is processed by each machine once.
- Each machine can process one job and each job can be processed by one machine at the same time.
- The processing time for each job in each machine is known in advance.
- There is no breakdown or interruption in the process.

2.2.2 Mathematical Model

There is a set of n jobs (1, 2, … n) to process in a set of m machines (1, 2, … m) in the same order. Let t_{ij} ($i = 1, 2, 3, \ldots n; j = 1, 2, 3, \ldots m$) donates the processing time of job i on machine j. $T(i, j)$ ($i = 1, 2, 3, \ldots n; j = 1, 2, 3, \ldots m$) is the completion

time of job i on machine j. Therefore, the mathematical model of PFSSP can be written as

$$\begin{aligned} T(1,1) &= t_{11}; \\ T(1,j) &= T(1,j-1) + t_{1j};\ (j = 2,3,\ldots m) \\ T(i,1) &= T(i-1,1) + t_{i1};\ (i = 2,3,\ldots n) \\ T(i,j) &= \max\{T(i-1,j), T(i,j-1) + t_{ij}\};\ (i = 2,3,\ldots n; j = 2,3,\ldots m) \end{aligned} \tag{2.1}$$

The total completion time/makespan is $C_{\max}$

$$C_{\max} = T(n,m) \tag{2.2}$$

In this paper, the optimization goal is to minimize the makespan $C_{\max}$.

2.3 Multi-population and Self-adaptive Genetic Algorithm Based on Simulated Annealing

In this section, we will introduce the multi-population coevolution strategy, improved self-adaptive genetic operators, and simulated annealing cooling strategy.

2.3.1 Multi-population Coevolution Algorithm

The traditional GA has a single population searching through the whole search space. Research shows that multi-population GA has excellent performance with solving the precocity problem of the GA [8]. In this paper, we adopt the multi-population coevolution algorithm to take the place of the single population. The process of multi-population coevolution strategy is shown in Fig. 2.1.

Suppose there are P independent subpopulations and each subpopulation evolves Q generations independently. Then populations migrate once with the effect of immigration operator, namely the worst individuals in a population will be replaced by the best individual in another population. The order of migration can be expressed as: $1 \rightarrow 2$, $2 \rightarrow 3$, …, $P \rightarrow 1$. After migration, we select the best individual from each subpopulation artificially as the quintessence population.

2.3.2 Self-adaptive Crossover and Mutation Operators

In this paper, the linear order crossover method and multi-point exchange mutation are adopted respectively. The population crossover and mutate according to their

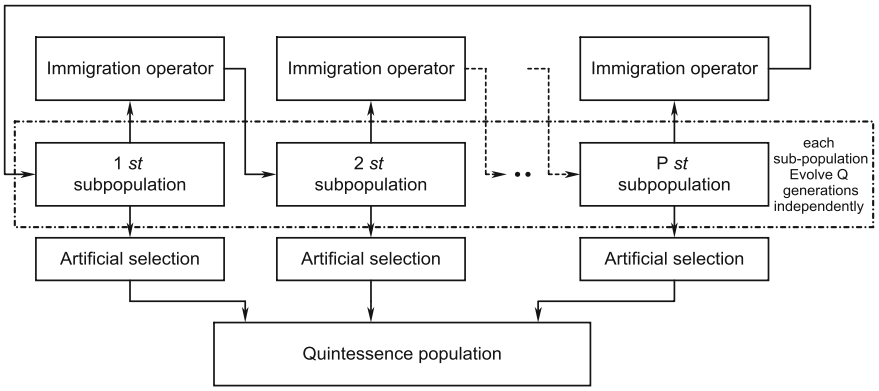

Fig. 2.1 Process of multi-population coevolution strategy

corresponding probability P_c and P_m. The selected values of crossover and mutation probabilities will greatly affect the quality of solution of the algorithm and the convergence rate. In order to improve the searching efficiency and avoid getting into local optimum, the cosine self-adaptive GA algorithm [9] adopts the following crossover probability P_c and mutation probability P_m.

$$P_c = \begin{cases} \frac{P_{c\max}+P_{c\min}}{2} + \left(\frac{P_{c\max}-P_{c\min}}{2}\right)\cos\left(\left(\frac{\overline{f}-f_{\text{avg}}}{f_{\max}-f_{\text{avg}}}\right)\pi\right) & \overline{f} \geq f_{\text{avg}} \\ P_{c\max} & \overline{f} < f_{\text{avg}} \end{cases} \tag{2.3}$$

$$P_m = \begin{cases} \frac{P_{m\max}+P_{m\min}}{2} + \left(\frac{P_{m\max}-P_{m\min}}{2}\right)\cos\left(\left(\frac{f-f_{\text{avg}}}{f_{\max}-f_{\text{avg}}}\right)\pi\right) & f \geq f_{\text{avg}} \\ P_{m\max} & f < f_{\text{avg}} \end{cases} \tag{2.4}$$

Inspired by the idea of Shi Shan, we introduce a squared term to improve self-adaptive genetic operators in genetic algorithms as shown in Eqs. (2.5) and (2.6).

$$P_c = \begin{cases} \frac{P_{c\max}+P_{c\min}}{2} - (P_{c\max}-P_{c\min})\left(\frac{\text{Max\_gen}}{\text{Count\_opt}+k1}\right)^2 \cos\left(\left(\frac{\overline{f}-f_{\text{avg}}}{f_{\max}-f_{\text{avg}}}\right)\pi\right) & \overline{f} \geq f_{\text{avg}} \\ P_{c\max} - (P_{c\max}-P_{c\min}) \times \left(\frac{\text{Count\_opt}+k1}{\text{Max\_gen}}\right)^2 & \overline{f} > f_{\text{avg}} \end{cases} \tag{2.5}$$

$$P_m = \begin{cases} \frac{P_{m\max}+P_{m\min}}{2} - (P_{m\max}-P_{m\min})\left(\frac{\text{Max\_gen}}{\text{Count\_opt}+k2}\right)^2 \cos\left(\left(\frac{f-f_{\text{avg}}}{f_{\max}-f_{\text{avg}}}\right)\pi\right) & f \geq f_{\text{avg}} \\ P_{m\max} - (P_{m\max}-P_{m\min}) \times \left(\frac{\text{Count\_opt}+k2}{\text{Max\_gen}}\right)^2 & f < f_{\text{avg}} \end{cases} \tag{2.6}$$

where $P_{c\,\min}$ and $P_{c\,\max}$ are the lower and upper limits of crossover probability respectively; $\overline{f}$ is the larger fitness of two crossover individuals; $\overline{f}$ donates the fitness of mutating individuals. f_{avg} is the average fitness of population; $f_{\max}$ is the maximum fitness of population; $P_{m\,\min}$ and $P_{m\,\max}$ are the lower and upper limits of mutation probability; Max_gen donates the generation number of optimal value keeping on;

Count_opt is the optimal value counter; $k1$ and $k2$ are constant which are less than Max_gen, which is used to adjust the P_c and P_m.

In the initial phase of algorithm, the value of Count_opt is very small. In this condition: Max_gen > Count_opt + $k1(k2)$. It can be obtained that:

$$\begin{cases} H_1 = \left(\frac{\text{Max\_gen}}{\text{Count\_opt}+k1(k2)}\right)^2 > 1 \\ H_2 = \left(\frac{\text{Count\_opt}+k1(k2)}{\text{Max\_gen}}\right)^2 < 1 \end{cases} \tag{2.7}$$

The H_1 is strengthening factor and H_2 is weakening factor. For those individuals ($\overline{f}, f < f_{\text{av}}$), the value of P_c and P_m will be larger, and more poor individuals will involve in crossover and mutation. That will increase the diversity of population.

In the late period of algorithm, the value of Count_opt will increase gradually. In this case: Max_gen < Count_opt + $k1(k2)$. We can obtain that:

$$\begin{cases} H_1 = \left(\frac{\text{Max\_gen}}{\text{Count\_opt}+k1(k2)}\right)^2 < 1 \\ H_2 = \left(\frac{\text{Count\_opt}+k1(k2)}{\text{Max\_gen}}\right)^2 > 1 \end{cases} \tag{2.8}$$

The H_1 is weakening factor and H_2 is strengthening factor. For those individuals ($\overline{f}, f > f_{\text{av}}$), the P_c and P_m will be larger, more good individuals will involve in crossover and mutation. That will increase the probability of the best individuals.

2.3.3 Simulated Annealing Cooling Strategy

Simulated Annealing (SA) was proposed by Metropolis et al. [10] and Scott Kirkpatrick et al. in 1983 [11]. Combined with simulated annealing mechanism, the genetic algorithm will reinforce the ability of overall searching optimal solution. In order to improve the ability of the sudden jump, we adopt formula (2.9) as cooling strategy.

$$T_{\text{start}} = \left(\frac{\text{Count\_opt}+p}{\text{Max\_gen}}\right)^q \times T_{\text{start}} \tag{2.9}$$

Selecting the appropriate p value, we can get that: when (Count_opt + p < Max_gen), it is a cooling process. When (Count_opt + p<Max_gen), it is heating process. Formula (2.9) has the ability to dynamically adjust the temperature changing. To some extent, it improves traditional monotony of sudden jump probability.

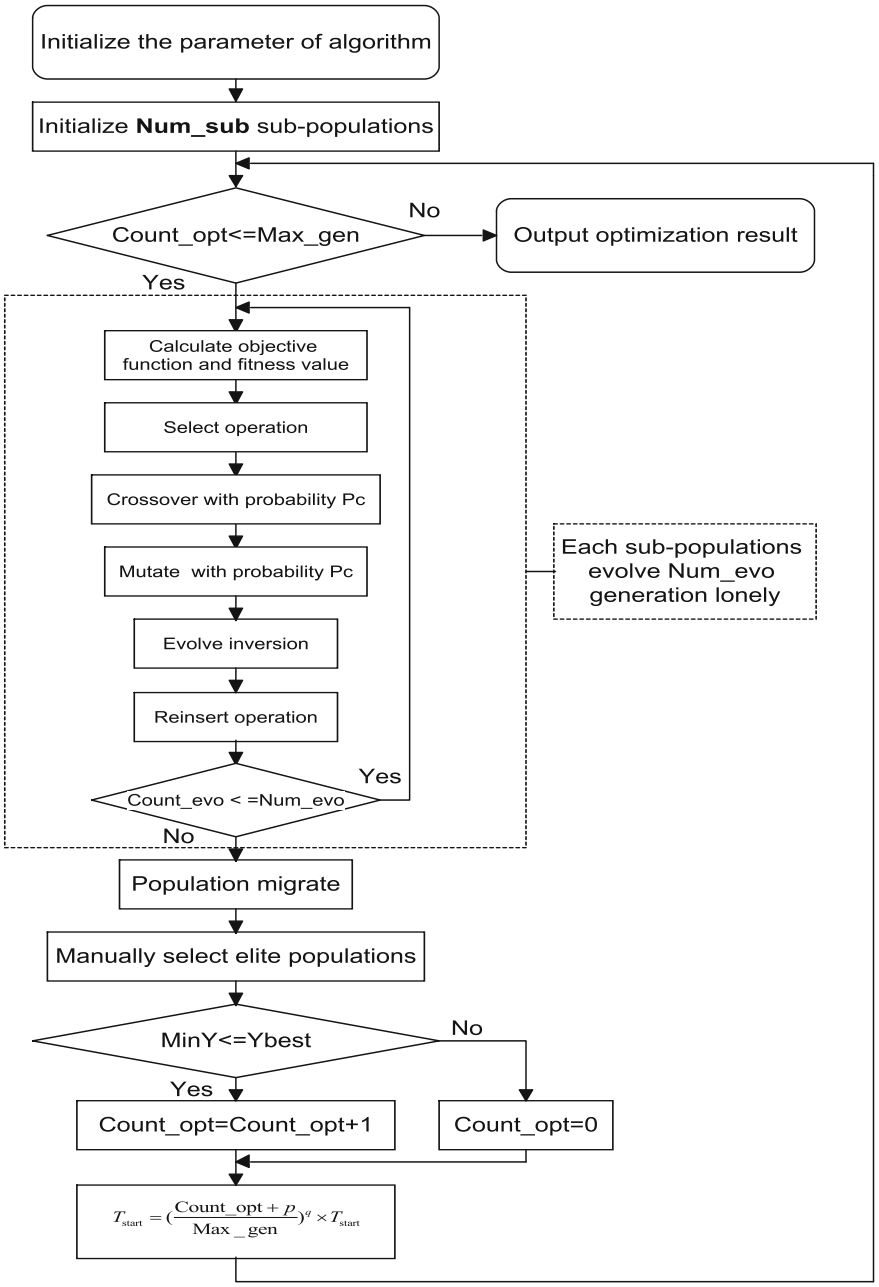

Fig. 2.2 Flowchart of algorithm

2.3.4 Realization of Algorithm

The procedure of the MSGASA for PFFSP is as follows:

Step 1. Initialize the parameter of algorithm.
Step 2. Generate subpopulations; initialize the generation number of optimal values keeping on; set the optimal value counter.
Step 3. Set the independent evolution generation counter.
Step 4. For each subpopulations, make the following operation (1–7), until generate Num_sub new population.

(1) Calculate objective function value;
(2) Select L individuals according to Roulette rule;
(3) For L individuals selected in (2.2), make crossover with self-adaptive crossover probability P_c (calculated in formula (2.5));
(4) Make mutate with self-adaptive mutation probability P_m.
(5) Evolve inversion: keep the high fitness value individuals;
(6) Reinsert: choose the optimum individuals from offspring and parent populations;
(7) If Count_evo < Num_evo, turn to Step 4; if not, turn to Step 5;

Step 5. Populations migrate.
Step 6. Artificially select the quintessence population.
Step 7. Change the annealing temperature in accordance with the formula (2.9).
Step 8. If Count_opt <=Max_gen, return to Step 3; if not, output the optimization result.

The flowchart of algorithm is shown in Fig. 2.2.

2.4 Experimental Results and Comparisons

To test the performance of MAGASA, some benchmark problems [12] have been solved. The parameters of MAGASA are set as in Table 2.1.

In order to evaluate the performance of the performance of the MAGASA, the best relative error (BRE) and average relative error (ARE) are adopted. They are calculated using the following formulae:

$$\begin{cases} \mathrm{BRE}(\%) = \frac{C_{\mathrm{best}} - C^*}{C^*} \times 100\,\% \\ \mathrm{ARE}(\%) = \frac{C_{\mathrm{avg}} - C^*}{C^*} \times 100\,\% \end{cases} \tag{2.10}$$

where C^* = Lower bound makespan; C_{best} = Makespan obtained using algorithm; C_{avg} = Average makespan obtained (20 times experiments)

The comparisons of SGA, IGA, GASA, and MAGASA are listed in Table 2.2.

Table 2.1 Parameters of MAGASA

Parameter	Value	Parameter	Value
$P_{c\ \max}$	0.9	Num_ind	100
$P_{c\ \min}$	0.6	$k1$	10
$P_{m\ \max}$	0.2	$k2$	10
$P_{m\ \min}$	0.05	p	12
Num_sub	4	T_{start}	1000
Max_gen	30	q	0.8
Num_evo	20		

Table 2.2 Comparisons of SGA, IGA, GASA, and MAGASA

Problem $_{n*m}$		SGA		IGA		GASA		MAGASA	
		BRE	ARE	BRE	ARE	BRE	ARE	BRE	ARE
Carl	11*5	0	0	0	0	0	0	0	0
Car2	13*4	0	0	0	0	0	0	0	0
Car3	12*5	0	0	0	0	0	0	0	0
Car4	14*4	0	0	0	0	0	0	0	0
Car5	10*6	0	0	0	0	0	0	0	0
Car6	8*9	0	0	0	0	0	0	0	0
Car7	7*7	0	0	0	0	0	0	0	0
Car8	8*8	0	0	0	0	0	0	0	0
Rec3	20*5	0	0.01	0	0	0	0.1	0	0
Rec7	20*10	0	0.4	0	0.34	0	0.27	0	0.06
Rec13	20*15	0.52	1.39	0.41	0.87	0.31	0.67	0	0.16
Rec19	30*10	0.91	1.7	0.67	1.09	0.38	1.07	0.14	0.34
Rec25	30*15	1.91	3.45	1.03	2.28	0.84	1.28	0.4	0.57

It can be observed that: with the problem of Car class (Car1–Car8), all the four algorithms have excellent performance in global searching and stability. With the problem of Rec class (Rec01–Rec25), the BRE and ARE obtained by MAGASA are less than those obtained by SGA, IGA, and GASA.

2.5 Conclusions

In this work, for the permutation flow shop scheduling problem, we proposed a multi-population and self-adaptive genetic algorithm based on simulated annealing. We improve traditional self-adaptive genetic operators by introducing the squared term. We propose a new cooling strategy to reinforce the ability of overall searching optimal solution. Experimental results show that the proposed algorithm has better performance compared with other existing heuristics.

References

1. Ribas I, Leisten R, Framiñan JM (2010) Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. Comput Oper Res 37(8):1439–1454
2. Johnson SM (1954) Optimal two-and three-stage production schedules with set up times included. Naval Res Logist Q 1:61–68
3. Brucker P (2007) Scheduling algorithm, 5th edn. Springer, Berlin
4. Fardin A (2012) A new ant colony algorithm for makespan minimization in permutation flow shops. Comput Ind Eng 63(2):355–361
5. Zhang Y, Liao XP, Wang Q (2009) Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. Eur J Oper Res 196(3):869–876
6. Rajkumar R, Shahabudeen P (2009) An improved genetic algorithm for the flowshop scheduling problem. Int J Prod Res 47(1):233–249
7. Wang L (2003) Intelligent optimization algorithm and application. Tsinghua University Press, Beijing
8. Huang M, Liu P, Xu L (2010) An improved multi-population genetic algorithm for job shop scheduling problem. In: Proceedings of the 2010 IEEE international conference on progress in informatics and computing, PIC 2010, vol 1, pp 272–275
9. Shi S, Li Q, Wang X (2002) Design optimization of brushless direct current motor based on adaptive genetic algorithm. J Xi'an Jiaotong Univ 36(12):1215–1218
10. Metropolis N, Rosenbluth A et al (1953) Equation of state calculations by fast computing machines. J Chem Phys 56(21):1087–1092
11. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(11):650–671
12. Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64(N.2): 278–285

http://www.springer.com/978-3-662-46465-6

Proceedings of the 2015 Chinese Intelligent Automation Conference
Intelligent Technology and Systems
Deng, Z.; Li, H. (Eds.)
2015, X, 599 p. 318 illus., Hardcover
ISBN: 978-3-662-46465-6