Algorithm for finding a minimal makespan in FJSSP by means of particle swarm optimization

Asen Toshev*, Vassil Guliashki*

* Institute of Information and Communication Technology at Bulgarian Academy of Sciences,
Department Information Processes and Decision Support Systems Bulgaria,
Acad. G. Bonchev St., Block 25A, 1113 - Sofia, BULGARIA

Email: tochevassen@yahoo.com, vggul@yahoo.com

This article offers algorithm for solving the flexible job shop scheduling problem. It uses a particle swarm optimization. A new problem formulation, including binary variables for operations performance notation, is proposed. The discrete space is presented in a way, allowing it to be used as a continuous space. Some results are presented.

Алгоритъм за намиране на минимално разписание в FJSSP с помощта на оптимизация чрез рояка частици (Асен Тошев, Васил Гуляшки). *Тази статия предлага алгоритъм за решаване на задачата за разписания в гъвкаво производство. В него се прилага оптимизация чрез рояка частици. Предложена е нова формулировка, включваща нотация с двоични променливи за моделиране на операциите. Разглежданата формулировка позволява дискретното пространство да бъде третирано като непрекъснато пространство. Представени са някои резултати.*

Introduction

1.1. The particle swarm optimization (PSO) method was proposed first from Kennedy and Eberhart (1995). It could be associated with flocks of birds, swarms of bees, fishes and others. Every particle has its own velocity *V* and position *X*. The values of *V* and *X* can be calculated by the following formulas:

$$(1)\quad V^{t+1} = \omega.V^t + c_1.rand.(P_{best} - X^t) + \\ +c_2.rand.(G_{best} - X^t)$$

$$(2)\quad X^{t+1} = X^t + V^{t+1}$$

Here ω is a positive inertia weight, c_1 and c_2 are acceleration coefficients and *rand* denotes random numbers with uniformly distribution between 0 and 1.

The PSO method was defined first for continuous space. It could not be used directly for discrete problems. That's why the particle swarm optimization idea was adapted by Kennedy and Eberhart (1997) for binary spaces.

1.2. In case the job shop scheduling problem has more than two jobs, or two machines it is NP-hard. Only the case, when every job has 1 or 2 operations, or the number of machines is 2 and all operations have processing time 1, can be solved for polynomial time.

Well known are tree models of job shop scheduling problem. They are:

- Model for Job shop scheduling problem (JSSP);
- Model for Flexible job shop scheduling problem (FJSSP);

and

- Model for Extended flexible job shop scheduling problem (EFJSSP).

In recent years the researchers devoted great attention to the FJSSP. It is used as a model also in this paper.

This model is an extension of the JSSP model. Unlike the JSSP model, in which every operation can be executed only on one machine, every operation O_{ij} in this model can be processed on machine $\boldsymbol{M_k}$ from a group of machines $\boldsymbol{M_k} \in \boldsymbol{M_{ij}}$, $\boldsymbol{M_{ij}} \subseteq \boldsymbol{M}$. This subgroup is different for each operation. In other words, it is not known in advance which operation on which machine will be assigned.

This model is closer to real-life production situations and can be applied when some or all of the machines are multi-tasking (Multitask machines). They can perform more than one operation (but not at the same time simultaneously) with correspondent processing time. Among the first researchers to offer this model are Brucker and Schlie (1990).

As noted by Chiang and Lin (2013) FJSSP is a task with high computational complexity. Since its solution is important to practice, it has been widely explored over the past two decades. Research on the multi-criteria version of FJSSP began about ten years ago, but most researches are focused on seeking a single optimal solution to a general criterion, including all criteria from the problem formulation. In the last three years, many researchers made attempts to find a set of Pareto optimal solutions.

Although several formulations for multi-criterial FJSSP are proposed, this approach cannot be easily implemented.

The following observations have been made:

(1) Focusing on the optimization of a single criterion can lead to a very poor outcome with respect to the other criteria;

(2) It is not easy to predict the compromise relationship between the criteria.

There are two versions of this model:

If $M_{ij} \subset M$ for at least one processed operation, and for at least one operation $M_{ij} \equiv M$, this is partial flexibility of JSSP (P-FJSSP); while if $M_{ij} \equiv M$ for all processed operations, there is total flexibility of JSSP (T-FJSSP).

The problem for solving FJSSP can be classified in two subproblems:

1) a routing subproblem (assignment), wherein each operation is assigned to the respective machine selected from the set of machines $M_k \in M_{ij}$ suitable for its execution;

2) a subproblem compiling a schedule, in which the initial times of the assigned operations of all machines must be calculated in order to obtain a feasible timetable and to optimize the predetermined criteria (fitness functions).

Methods for solving FJSSP can be classified into the following two subgroups:

• Hierarchical methods: By them the two subproblems are solved separately and successively. Such methods are developed by Brandimarte (1993), Paulli (1995), Zhang and Gen (2005).

• Integrated procedures: There is no clear distinction between the stage of appointment of machine operations and the stage of searching the optimum schedule. The methods applying the integrated approach are, for example, those developed by Fattahi et al. (2007), Kacem et al. (2002, b).

Algorithm for FJSSP

2.1. Generating initial solutions (schedules) $X_h^0, h = 1, 2, \dots, N_p$, where N_p is the number of particles in the population:

The initial population is calculated as follows: The current schedule is compiled on the basis of generated random numbers $r \in (0, 1)$. The interval $(0, 1)$ is divided into n parts corresponding to each job. Each random number specifies the current job from which an operation is taken to be included in the schedule. When strictly executing the current operation from the selected job, the strict order of operations is strictly observed. The operations are set on machines according to the first free machine rule, and if there are several free machines at the same time, the machine with the smallest index is selected. If the current job is completed (i.e. all operations are included in the schedule), it continues with unfulfilled job that has the smallest index to the end of the jobs.

Let O be the number of operations for the task under consideration. Proceed with the following algorithm:

$$If\ 1 <= O <= 25, then\ N_p = 5;$$
$$If\ 25 < O <= 50, then\ N_p = 10;$$
$$If\ 50 < O <= 75, then\ N_p = 15;$$
$$If\ 75 < O <= 100, then\ N_p = 20;$$
$$If\ 100 < O, then\ N_p = 30.$$

2.2. In the starting moment the velocity of particle h is: $V_{hij}^0 = 0, i = 1, 2, \dots, O;\ j = 1, 2, \dots, M$; ($i$ is the operation number); (j is machine number), where M is the number of machines. X_{hij}^0 and V_{hij}^0 are the elements of X_h^0 and V_h^0 respectively.

2.3. Constructing a binary matrix B_{hij}^0, $B_{hij}^0 \in \{0,1\}$, where 1 is put on positions, in which $X^0{}_{hij}$ has a nonzero value. A specific schedule may have only one 1 in B_{hij}^0 on the i-th row, because only one machine can process a given operation. The other positions have 0-value.

2.4. Finding the velocity V_{hij}^k (h is the particle number in the population, i is the number of the operation, j is the machine number and k - the current iteration) by the formula:

(3) $V_{hij}^{k} = \omega . V_{hij}^{k-1} + r_1 . c_1 . dist\left(P_{hij}^{k-1}, X_{hij}^{k-1}\right) + + r_2 . c_2 . dist(G_{ij}^{k-1}, X_{hij}^{k-1})$

where **ω**, $\boldsymbol{c_1}$, $\boldsymbol{c_2}$ are coefficients determined by the decision maker and control the behavior of the particle swarm optimization (PSO) method.

$$\omega = 1 ,$$
$$c_1 = 1.8 ,$$
$$c_2 = 2.2 .$$

$\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are two random numbers generated with the uniform distribution, $r_1, r_2 \in (0, 1)$.

P_h^k is the best position in the neighborhood of the X_h^k particle, G_h^k is the best position for the X population.

The distance is defined as follows:

(4) $dist\left(P_{hij}^{k}, X_{hij}^{k}\right) = t . [a . \frac{|f(P_{hij}^{k}) - f(X_{hij}^{k})|}{C} + + b . \frac{D - S(P_{hij}^{k}, X_{hij}^{k})}{D}],$

$\boldsymbol{t}$ is a positive integer named accelerating coefficient, $\boldsymbol{a}$ and $\boldsymbol{b}$ are two positive weights that can be obtained experimentally and satisfy the condition that their sum is equal to 1. The values of $\boldsymbol{t}$, $\boldsymbol{a}$ and $\boldsymbol{b}$ are taken from the literary source, Hongwei Ge et al. (2007):

$$t = 1 ,$$
$$a = 0.7,$$

and

$$b = 0.3 .$$

Let the makespan be denoted by $\boldsymbol{C}$. $f(X_{hij}^{k})$ is a fitness function, where $0 \leq f(X_{hij}^{k}) \leq C$.

The constant D is the space dimension, in concrete case $D = O$ – the operations number.

Let $X_i(x_{i1}, x_{i2}, \dots, x_{io})$ and $X_j(x_{j1}, x_{j2}, \dots, x_{jo})$ be two particles in the space of operations. The following function is inserted:

(5) $s(m) = \begin{cases} 1, \mathrm{x}_{im} = \mathrm{x}_{jm} \\ 0, \mathrm{x}_{im} \neq \mathrm{x}_{jm} \end{cases}$

Let the function $S(X_i, X_j)$, which is called function of similarity, be entered.

(6) $S\left(X_i, X_j\right) = \sum_{m=1}^{o} s(m)$

On elements of V is applied the transformation by sigmoid function. Only the V_{hij}^{k} elements having value 1, take part in the calculation of nonzero elements of X. If the sigmoid function converts one V-element to 0 this reflects in the corresponding X_{hij}^{k} element. In case $M = 2$ the matrix B is constructed as follows: 0 is written in the position, corresponding to the V-element having value 1, and the alternative position obtains value 1; In case $M > 2$, the value 1 is put in the position according one of the following options (the decision maker makes the choice):

1. in the position of the least processing time for this *i*-th row;
2. in the position of the longest processing time for this *i*-th row;
3. in the position of the closest to the middle processing time for this *i*-th row.

In matrix X the nonzero elements corresponds to the positions, where the elements of matrix B have value 1. The nonzero elements of X obtain values corresponding to the processing time, applying the above 3 options.

Let

(7) $\sigma\left(V_{hij}^{k}\right) = \frac{1}{1 + e^{-V_{hij}^{k}}}$

is the sigmoid function for V_{hij}^{k}.

To prevent the excessive approximation of $\sigma(V_{hij}^{k})$ to 0 or 1, the $Vmax$ constant is often used to limit the V_{hij}^{k} magnitude. Typically, $Vmax = 4$;

$V_{hij}^{k} \in [-Vmax, Vmax]$. After the transformation (7) the value of $\sigma(V_{hij}^{k})$ is in the range of 0 to 1, i.e. $\sigma(V_{hij}^{k}) \in (0,1)$.

Find the result of the changed position by the following rule using the probability function:

(8) $B_{hij}^{k} = \begin{cases} 1, & if\ rand < \sigma\left(V_{hij}^{k}\right) \\ 0, & otherwise \end{cases}$

2.5. Constructing the matrix B_{hij}^{k}, $h = 1,2, \dots, N_p$; $i = 1, 2, \dots, O$; $j = 1,2, \dots, M$ according to (8). $B_{hij}^{k} \in \{0,1\}$.

2.6. Find a new X_h^k solution in the way described in 2.4, using the B_{hij}^{k} matrix, so that 0-elements in X correspond to 0-elements of B, and the elements equal to processing time in X correspond to the elements of B, having value 1.

If $f(X_h^k) < f(P_h^{k-1})$, then $P_h^k = X_h^k$, otherwise $P_h^k = P_h^k$, $h = 1,2, \dots, N_p$.

If $f(P^k_{hij})<f(G^{k-1})$, then $G^k = P^{\,k}_{h}$, otherwise $G^k = G^{k-1}$, $h = 1,2,\dots,N_p$.

Let the iteration limit in the algorithm be denoted by *maxiter*.

In case one of the stop criteria is met, i.e.:

1) the limit *maxiter* is reached;

2) there is no improvement of the makespan in *z* consecutive iterations;

3) the best makespan in the population is equal to the sum of the optimal times for each job;

or

4) a value of the fitness function, satisfying the decision maker is found,

then END,

otherwise go to step 2.4.

Example

The example M2J2O4 of FJSSP from Fattahi (2007) is used as an illustrative example.

Job	**Operation**	**Machine M1**	**Machine M2**
J1	**O11**	25	37
	O12	32	24
J2	**O21**	45	65
	O22	21	65

Below is presented the search process of the second particle in the swarm with cardinality five from Iteration 0 to Iteration 3. As can be seen in this case, the particle reaches the optimal makespan on the second iteration, which is saved by the algorithm. The search procedure has a limit of 100 iterations.

Iteration 0:

V^0_2		M_1	M_2
J_1	O_{11}	0	0
	O_{12}	0	0
J_2	O_{21}	0	0
	O_{22}	0	0

B^0_{2ij}		M_1	M_2
J_1	O_{11}	1	0
	O_{12}	0	1
J_2	O_{21}	1	0
	O_{22}	1	0

X^0_2		M_1	M_2
J_1	O_{11}	25	0
	O_{12}	0	24
J_2	O_{21}	45	0
	O_{22}	21	0

Iteration 1:

$\mathbf{V^1_2}$		$\mathbf{M_1}$	$\mathbf{M_2}$
J_1	O_{11}	0	0.29
	O_{12}	0	0.39
J_2	O_{21}	0.39	0
	O_{22}	0.54	0

B^1_{2ij}		M_1	M_2
J_1	O_{11}	0	1
	O_{12}	0	1
J_2	O_{21}	1	0
	O_{22}	1	0

X^1_2		M_1	M_2
J_1	O_{11}	0	37
	O_{12}	0	24
J_2	O_{21}	45	0
	O_{22}	21	0

Iteration 2:

V^2_2		M_1	M_2
J_1	O_{11}	0	0.48
	O_{12}	0.67	0
J_2	O_{21}	0.71	0
	O_{22}	0	0.88

B^2_{2ij}		M_1	M_2
J_1	O_{11}	0	1
	O_{12}	1	0
J_2	O_{21}	1	0
	O_{22}	0	1

X^2_2		M_1	M_2
J_1	O_{11}	0	37
	O_{12}	32	0
J_2	O_{21}	45	0
	O_{22}	0	65

Iteration 3:

V^3_2		M_1	M_2
J_1	O_{11}	0	0.68
	O_{12}	1.10	0

J_2	O_{21}	1.67	0
	O_{22}	0	1.93

B^3_{2ij}		M_1	M_2
J_1	O_{11}	0	1
	O_{12}	1	0
J_2	O_{21}	1	0
	O_{22}	0	1

X^3_2		M_1	M_2
J_1	O_{11}	0	37
	O_{12}	32	0
J_2	O_{21}	45	0
	O_{22}	0	65

Iteration 4: … (and so on)

Test results

The obtained test results are summarized in the following table:

Number	Problem	Iteration	Makespan
1.	M2J2O4	5	66
2.	M3J2O5	5	63
3.	M3J2O5	5	14
4.	M3J3O8	5	11
5.	M3J3O9	5	52
6.	M3J4O13	5	40
7.	M3J5O20	5	44
8.	M4J3O8	5	19
9.	M2J2O4	5	6
10.	M5J3O7	5	13
11.	M5J3O8	5	15
12.	M5J3O58	5	221

Conclusion

This article presents a new algorithm using binary space model in particle swarm optimization for solving flexible job shop scheduling problems.

The described algorithm is coded on MATLAB. It is implemented on system Windows XP Professional, Intel, Celeron, CPU 430, 1.80 GHz, 1 GB of RAM.

The proposed algorithm was used to solve twelve benchmark examples from the literature. The experiments show that the binary particle swarm optimization can be used to solve the FJSSP effectively.

The following directions for future research can be formulated:

1) Developing new diversification search strategies for eventual improvement of best found solution (schedule);

2) Making comparisons of proposed algorithm to other (meta) heuristics;

3) Performing test experiments on larger sample of (benchmark) examples with higher dimension.

REFERENCES

[1] Brandimarte, P. (1993) Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157-183, 1993.

[2] Brucker, P., R. Schlie (1990) Job shop with multi-purpose machine, *Computing*, vol. 45, 1990, pp. 369-375.

[3] Chiang, Tsung-Che, Lin, Hsiao-Jou (2013) A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling, *Int. J. Production Economics*, 141, 87-98.

[4] Eberhart, R.C. A discrete binary version of the particle swarm algorithm. In: Proceedings of 1997 conference systems man cybernetics, NJ: Piscataway; 1997. p. 4104–8.

[5] Fattahi, P., Saidi Mehrabad, M., Jolai, F., (2007), Mathematical Modeling and Heuristic Approaches to Flexible Job Shop Scheduling Problems, *Journal of Intelligent Manufacturing*, 18:331–342.

[6] Ge H., W. Du, F. Qian, A Hybrid Algorithm Based on Particle Swarm Optimization and Simulated Annealing for Job Shop Scheduling, http://www.paper.edu.cn.

[7] Kacem I, Hammadi S, Borne P (2002b) Pareto-Optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Math. Comput. Simul.* 60:245-276 doi:10.1016/S0378-4754(02)00019-8

[8] Kennedy J., R.C. Eberhart, Particle swarm optimization, Proceedings of IEEE international conference on neural networks, vol. IV, Piscataway, NJ (1995), pp. 1942-1948.

[9] Paulli, J. (1995) "A Hierarchical approach for the FMS scheduling problem", *Europ. J. of Operational Research*, vol. 86, pp. 32-42.

[10] Zhang, H., Gen, M. (2005) “Multistage-based genetic algorithm for flexible job-shop scheduling problem”, *International Journal of Complexity*, vol. 11, pp. 223-232.