Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 23 (2011) 131 – 136

Procedia Engineering

www.elsevier.com/locate/procedia

An Artificial Bee Colony Algorithm Based on Problem Data Properties for Scheduling Job Shops

Rui Zhang*

School of Economics and Management, Nanchang University, Nanchang 330031, PR China

Abstract

To solve the job shop scheduling problem with the objective of minimizing total weighted tardiness, an artificial bee colony algorithm based on problem data analysis is proposed. First, characteristic values are defined to describe the criticality of each job in the process of scheduling and optimization. Then, a fuzzy inference system is employed to evaluate the characteristic values according to practical scheduling knowledge. Finally, a local search mechanism is designed based on the idea that critical jobs should be processed with higher priority. Numerical computations are conducted with an artificial bee colony algorithm which integrates the local search module. The computational results for problems of different sizes show that the proposed algorithm is both effective and efficient.

Keywords: Artificial bee colony algorithm, Job shop scheduling problem, Local search, Problem data

1. Introduction

The job shop scheduling problem (JSSP) has been known as a notoriously hard combinatorial optimization problem since the 1950s. In terms of computational complexity, JSSP is $\mathcal{NP}$-hard in the strong sense. Therefore, even for very small JSSP instances, it is by no means easy to guarantee the optimal solution. In recent years, the meta-heuristics — such as genetic algorithm (GA) [1], tabu search (TS) [2], particle swarm optimization (PSO) [3], and ant colony optimization (ACO) [4] — have clearly become the research focus in practical optimization methods for solving JSSPs.

However, when the problem size grows, meta-heuristic algorithms usually take excessive time to converge. To enhance the efficiency of these algorithms, two types of approaches may be roughly identified in the literature:

(1) Focusing on the optimization algorithm: the conventional operations (or parameters) in the standard version of these algorithms have been modified or redesigned to promote their performance in the neighborhood search.

(2) Focusing on the features of the pending problem: problem-specific or instance-specific information is extracted and utilized in the searching process to accelerate the convergence speed of these algorithms.

The former approach is independent of problem classes. But according to the no free lunch theorem [5], such improvements on the optimization algorithm alone cannot guarantee good performance for all problems. In terms of the latter approach, embedded local search can utilize the characteristic information to improve the solutions in

*Corresponding author
Email address: zhangrui05@gmail.com (Rui Zhang)

1877-7058 © 2011 Published by Elsevier Ltd.
doi:10.1016/j.proeng.2011.11.2478

the optimization process. However, how to effectively extract and describe the characteristic information remains a challenging but rewarding research topic.

In this paper, we devise a fuzzy inference system based on intuitive knowledge to evaluate the criticality value of each job. Then, this information is used in a local search mechanism to promote the optimization efficiency of the artificial bee colony (ABC) algorithm. The paper is organized as follows. The discussed job shop scheduling problem is formulated in Section 2. Sections 3 and 4 presents the detailed algorithms. The computational results and a brief analysis are provided in Section 5. Finally, the conclusions are given in Section 6.

2. Problem formulation

Job shop is one of the most frequently adopted models when dealing with scheduling problems. In the job shop scheduling problem (JSSP), a set of n jobs $\{J_i\}_{i=1}^n$ are to be processed on a set of m machines $\{M_k\}_{k=1}^m$. Each job has a fixed processing route which traverses all the machines in a predetermined order. Besides, a preset due date and a weight are given for each job. JSSP can also be described by a disjunctive graph $G(N, A, E)$, in which $N = \{0, 1, \ldots, *\}$ represents the set of nodes (including two dummy nodes, 0 and $*$); A is the set of conjunctive arcs and $E = \bigcup_{k\in M} E_k$ is the set of disjunctive arcs (E_k represents the disjunctive arcs that correspond to machine M_k). Then the discussed JSSP can be formulated as follows:

$$\begin{cases} \min & TWT = \sum\limits_{j\in C} w_j \left(t_j + p_j - d_j\right)^+, \\ \text{s.t.} & \\ & t_i + p_i \le t_j, \quad (i, j) \in A, \\ & t_i + p_i \le t_j \vee t_j + p_j \le t_i, \quad (i, j) \in E_k, k = 1, \ldots, m, \\ & t_i \ge 0, \quad i \in N. \end{cases}$$

In this formulation, C is the set of the last operations of each job; w_j and d_j are respectively the weight and the due date of the job which operation j belongs to; p_j and t_j (the decision variable) are the processing time and the starting time of operation j, respectively; $(x)^+ = \max\{x, 0\}$. The scheduling objective considered in this paper is to determine the processing sequence of operations on each machine such that the total weighted tardiness is minimized.

3. Evaluation of the criticality values

In this section, we provide a detailed description of the fuzzy inference system which is designed to calculate the criticality value (CV) of each job.

Based on a given schedule (including the completion time of each job), we can define:

- The relative distance between job i's completion time and its due date: $g_i = (\hat{F}_i - d_i)/\sum_{j\in J_i} p_j$, where $\hat{F}_i$ denotes job i's completion time under the current schedule; $\sum_{j\in J_i} p_j$ equals the total processing time of job i.
- The relative slack time of job i: $h_i = (d_i - c_i - \sum_{j\in J_i'} p_j)/\sum_{j\in J_i} p_j$, where c_i refers to the completion time of the currently considered operation of job i and also the release time of J_i' which is the set of its succeeding operations in job i. Note that h_i corresponds to specific operations of job i and thus reflects the features of different processing stages of the job.
- The normalized weight of each job: $v_i = \overline{w}_i = (w_i - w_{\min})/(w_{\max} - w_{\min})$, where $w_{\max} = \max_{i=1}^n w_i$, $w_{\min} = \min_{i=1}^n w_i$.

Based on these variables, a fuzzy controller is designed to calculate the CV. The fuzzy controller takes g_i, h_i and v_i as input variables, and outputs the CV for each job. In the fuzzy inference system, the four input/output linguistic variables are respectively denoted by G, H, V and B, and are divided into three fuzzy sets as follows.

- $G, H = \{\text{NL}, \text{Z}, \text{PL}\}$, i.e. {Negative, Around Zero, Positive}.
- $V = \{\text{S}, \text{M}, \text{L}\}$, i.e. {Small, Medium, Large}.
- $B = \{\text{N}, \text{PC}, \text{C}\}$, i.e. {Not a critical job, Possibly a critical job, A critical job}.

In this fuzzy inference system, all the relevant membership functions are chosen to have symmetrical triangular shapes. For example, the membership functions of the three fuzzy sets related to G are illustrated in Figure 1. We could

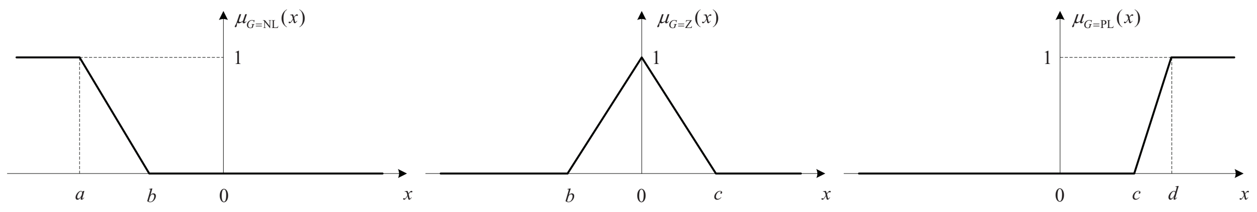

Figure 1: The membership functions adopted by the fuzzy controller

use the turning point values to characterize the membership functions. Thus, the linguistic variable G is accordingly denoted as $\{(-\infty, a, b); (b, 0, c); (c, d, +\infty)\}$.

Critical jobs are those jobs that need to be considered with higher priority in the optimization process. In practical job shop scheduling, there exists some human experience which indicates which jobs should have higher priority under different circumstances. After further abstraction, this kind of knowledge can be expressed in terms of fuzzy rules which have the form of "If..., then...". For example, "If G = PL, H = NL and V = L, then B = C" means that if under the current schedule, a certain job has relatively large tardiness and relatively small slack time, and its weight is large, then this job should be regarded as a critical job.

According to such priori knowledge, we obtain the fuzzy rule table shown in Table 1 by enumerating all possible and feasible combinations of the input variables. These rules try to reflect the basic properties of critical jobs from different perspectives.

Table 1: The fuzzy rule table

G, H	V		
	S	M	L
NL, PL	N	N	N
Z, Z	PC	C	C
Z, PL	N	N	PC
PL, NL	C	C	C
PL, Z	PC	C	C
PL, PL	N	PC	PC

The fuzzy inference system based on the 18 rules in Table 1 adopts the Mamdani model, in which the T-norm is implemented by the "min" operator. Since the output criticality value should be a quantifiable number, here we use the smallest-of-maximum Z_{SOM} [6] as the defuzzification method. By applying the above fuzzy inference process to the current solution, we obtain the criticality values $\{CV_i\}_{i=1}^{n}$ for each job.

4. The hybrid artificial bee colony (ABC) algorithm

4.1. The principles of ABC

In the ABC algorithm, the artificial bees are classified into three groups: the employed bees, the onlookers and the scouts. A bee that is exploiting a food source is classified as employed. The employed bees share information with the onlooker bees, which are waiting in the hive and watching the dances of the employed bees. The onlooker bees will then choose a food source with probability proportional to the quality of that food source. Therefore, good food sources attract more bees than the bad ones. Scout bees search for new food sources randomly in the vicinity of the hive. When a scout or onlooker bee finds a food source, it becomes employed. When a food source has been fully exploited, all the employed bees associated with it will abandon the position, and may become scouts again. Therefore, scout bees perform the job of "exploration", whereas employed and onlooker bees perform the job of "exploitation". In the algorithm, a food source corresponds to a possible solution to the optimization problem, and the nectar amount of a food source corresponds to the fitness of the associated solution.

In ABC, the first half of the colony consist of employed bees and the other half are onlookers. The number of employed bees is equal to the number of food sources (SN) because it is assumed there is only one employed bee for each food source. Thus, the number of onlooker bees is also equal to the number of solutions under consideration. The ABC algorithm starts with a group of randomly generated food sources. The main procedure of ABC can be described as follows. The detailed description for each step can be found in [7].
Step 1: Initialize the food sources.
Step 2: Each employed bee starts to work on a food source.
Step 3: Each onlooker bee selects a food source according to the nectar information shared by the employed.
Step 4: Determine the scout bees, which will search for food sources in a random manner.
Step 5: Test whether the termination condition is met. If not, go back to Step 2.

4.2. Encoding and decoding

Although ABC was originally intended for continuous function optimization, some efforts have recently been made to adapt it to combinatorial optimization problems. The encoding scheme used here is based on the random key representation and the smallest position value (SPV) rule [8]. Each solution (food source) is described by $n \times m$ continuous values, and in the decoding process, this set of values will be transformed to a permutation of operations by the SPV rule. Formally, let $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, \dots, x_{i,n\times m}]$ denote the i-th solution, where $x_{i,d}$ is the position value of the i-th solution with respect to the d-th dimension ($d = 1, \dots, n \times m$). To decode a solution, the SPV rule is applied to sort the position values within a solution and then determine the relative positions of the corresponding operations.

4.3. The local search module

In each iteration of the proposed ABC, the local search is carried out for the best $e\%$ of solutions in the current population immediately after the onlooker bee phase. The procedure is described as follows.
Step 1: Choose a solution from the current population and then randomly select a machine.
Step 2: Make a copy of the n operations to be processed on the selected machine from the operation list Π obtained after decoding this solution, yielding a temporary operation list L of length n. Then randomly choose an operation L_k from this list.
Step 3: Calculate $\{CV_i\}_{i=1}^{n}$ for all the operations L_i ($i = 1, \dots, n$) in list L.
Step 4: Evaluate $i_1 = \arg\max_{i \in L^{\mathrm{p}}(k)} \{(CV_k - CV_i)^+\}$, where $L^{\mathrm{p}}(k)$ denotes the set of operations that are before L_k in L.
Step 5: Evaluate $i_2 = \arg\max_{i \in L^{\mathrm{s}}(k)} \{(CV_i - CV_k)^+\}$, where $L^{\mathrm{s}}(k)$ denotes the set of operations that are after L_k in L.
Step 6: Let $i^* = \arg\max_{i \in \{i_1, i_2\}} \{|CV_k - CV_i|\}$.
Step 7: Swap operations L_k and L_{i^*} in the original decoded operation list Π.

Finally, the position values of the solution should be repaired after the local search process, so as not to violate the SPV rule. This procedure is illustrated in Table 2, where the position values $x^t_{i,2}$ and $x^t_{i,4}$ should also be swapped when the positions of operation 2 and 4 are exchanged by the local search.

Table 2: Illustration of position value repairing after the local search

Dimension k	1	2	3	4	5	6	7	8	9
$x^t_{i,k}$	1.80	−0.99	3.01	0.72	−0.45	−2.25	5.30	4.80	1.90
$\pi^t_{i,k}$	6	$\underline{2}$	5	$\underline{4}$	1	9	3	8	7
$x^t_{i,k}$	1.80	**0.72**	3.01	**−0.99**	−0.45	−2.25	5.30	4.80	1.90
$\pi^t_{i,k}$	6	**4**	5	**2**	1	9	3	8	7

5. Computational results

To test the effectiveness of the proposed ABC algorithm, we randomly generate different-scale JSSP instances in which the route of each job is a random permutation of m machines and the processing time of each operation follows a uniform distribution $\mathcal{U}(1, 100)$. The due date of each job is obtained by a series of simulation runs which apply different dispatching rules to the machines and finally we take the average completion time of each job among these simulations as its due date. The weight of each job is generated from a uniform distribution $\mathcal{U}(1, 10)$.

The membership functions of the fuzzy inference system for calculating CV are: $\{(0.1, 0.3); (0.3, 0.7); (0.7, 1)\}$ for variable V; $\{(-1, -0.2); (-0.2, 0.2); (0.2, 1, +\infty)\}$ for variable G; $\{(-\infty, -1, -0.2); (-0.2, 0.2); (0.2, 1)\}$ for variable H and $\{(0, 0.1); (0.1, 0.6); (0.6, 1)\}$ for output B. The parameters for ABC are: $SN = 30$, $limit = 40$, $e\% = 25\%$ and the maximum generation number is 200.

We compare ABC with a hybrid particle swarm optimization (PSO) algorithm whose parameters are set according to [9]. The results for 10 test problems are shown in Table 3, where $f_{\min}$, $f_{\max}$ and f_{avg} respectively denote the minimum, maximum and average objective value in 20 consecutive runs of the corresponding algorithm.

Table 3: Computational results and comparisons

No.	Size ($n \times m$)	ABC			PSO		
		$f_{\min}$	$f_{\max}$	f_{avg}	$f_{\min}$	$f_{\max}$	f_{avg}
1	10×10	9	16	14.6	10	16	14.8
2	20×5	143	162	146.2	142	169	155.7
3	10×20	50	69	58.5	53	79	60.2
4	20×10	184	264	238.3	211	290	258.4
5	20×15	132	176	163.0	148	212	172.1
6	50×6	123	189	164.4	134	197	172.7
7	20×20	400	515	467.8	418	548	483.4
8	40×10	1435	1552	1512.9	1502	1864	1710.3
9	50×10	1187	1534	1387.4	1225	1595	1477.8
10	100×5	7829	8612	8462.3	8150	9727	8812.1

By comparing $f_{\min}$, we see that ABC obtains better result than PSO for 9 instances, which proves the effectiveness of the proposed approach. By comparing f_{avg}, our algorithm outperforms PSO for all the instances in terms of average performances, which suggests the proposed ABC is more robust in different runs.

6. Conclusion

In this paper, an artificial bee colony algorithm based on criticality information for solving job shop scheduling problems is proposed. The defined criticality values reflect the properties of both the objective function and the most crucial jobs at different stages of the optimization process. The criticality information is extracted and used as a local search optimizer to increase the convergence speed of the optimization process. Computational results show that the proposed algorithm is effective.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China under Grant No. 61104176.

References

[1] H. Zhou, W. Cheung, L. C. Leung, Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm, European Journal of Operational Research 194 (3) (2009) 637–649.

[2] J. Q. Li, Q. K. Pan, P. N. Suganthan, T. J. Chua, 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) (2011) 683–697.

[3] G. Moslehi, M. Mahnam, A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search, International Journal of Production Economics 129 (1) (2011) 14–22.

[4] M. Seo, D. Kim, Ant colony optimisation with parameterised search space for the job shop scheduling problem, International Journal of Production Research 48 (4) (2010) 1143–1154.

[5] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.

[6] J. S. R. Jang, C. T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing, Prentice-Hall, Englewood Cliffs, NJ, 1997.
[7] J. Li, Q. Pan, S. Xie, S. Wang, A hybrid artificial bee colony algorithm for flexible job shop scheduling problems, International Journal of Computers, Communications & Control 6 (2) (2011) 286–296.
[8] M. F. Tasgetiren, Y. C. Liang, M. Sevkli, G. Gencyilmaz, Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem, International Journal of Production Research 44 (22) (2006) 4737–4754.
[9] W. Xia, Z. Wu, A hybrid particle swarm optimization approach for the job-shop scheduling problem, The International Journal of Advanced Manufacturing Technology 29 (3-4) (2006) 360–366.