SOLVING THE JOB SHOP SCHEDULING PROBLEM BY AN IMMUNE ALGORITHM

XING-QUAN ZUO, YU-SHUN FAN

Automation Department, Tsinghua University, Beijing 100084, China
E-MAIL: zuoxq@tsinghua.edu.cn

Abstract:

An immune algorithm is presented for solving the job shop scheduling problem. In the algorithm, the niche technology is used to keep the diversity of the population and chaos variables are employed to perform antibody mutation. The code of an antibody is based on random keys, and a heuristic process is given to decode the antibody into a parameterized active schedule to reduce the solution space. Experimental results demonstrate the algorithm is effective for solving job shop problems.

Keywords:

Job shop scheduling; immune algorithm; optimization computation; evolutionary algorithm

1. Introduction

In this paper, an immune algorithm for solving the job shop scheduling problem is presented. In the job shop scheduling, there are *n* jobs that have to be processed on *m* machines. Each job may consist of different number of operations, and the processing orders of each job by all machines and the processing time of each operation are known and fixed. The objective is to decide how to arrange the processing orders and starting times of operations sharing the same machine, in order to optimize certain criteria.

Job-shop schedule is usually a strongly NP-complete problem of combinatorial optimization. Therefore, exact methods such as branch-and-bound and integer programming techniques require too much computation time [1]. Because of the NP-complete characteristics of job shop schedule, it is usually very hard to find its optimal solution. In the early 1980s, much interest has been devoted to search its near-optimal solutions with all kind of heuristic algorithms, which mainly includes genetic algorithm, taboo search and simulated annealing and artificial neural network methods [2].

Similar to the artificial neural network which is inspired by human's nervous system, artificial immune system (AIS) is a novel computational intelligence approach for problems solving, which inspired by theoretical immunology and observed immune functions, principles, and models [3]. The main application domains of AIS are computer and network security [4], fault and anomaly detection [5], optimization computation [6-8], data analysis and data mining [9]. The learning capability, memory, and robustness of immune system make AIS also useful for scheduling problems. In this paper, we present an immune algorithm in dealing with job-shop scheduling problems. In the algorithm, each antibody is a feasible solution and represents a priority rule that is used to construct a parameterized active schedule.

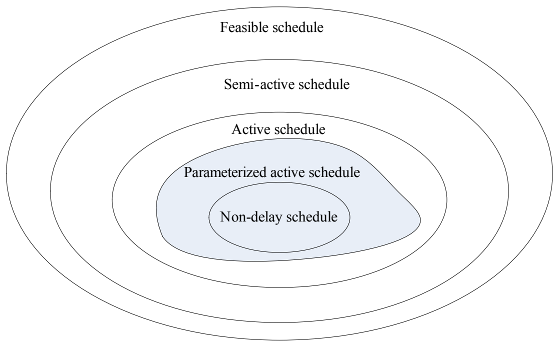

Figure 1. Parameterized active schedule

2. Parameterized active schedule

Schedule can be classified into feasible schedule, semi-active schedule, active schedule and non-delay schedule. The semi-active schedule is the subset of feasible schedule, and the active schedule is the subset of semi-active schedule (see figure 1). The optimal schedule exists in the set of active schedules. However, the set of active schedules is usually very large, and includes many schedules with large delay time, which is usually very poor quality. In order to reduce the solution space, this paper makes the solution space consist of parameterized active schedules [15]. The parameterized active schedule is a subset of active schedule. Each operation of a parameterized active schedule has a delay time that is no more than a give time. Therefore, active schedules with a large delay time are removed for the set of parameterized active schedules, and the solution space is reduced. The process of constructing a parameterized active schedule will be given in section 3.3.

3. Solving job shop problem by immune algorithm

3.1. Clonal selection and immune network theory

The immune system protects our body against bacteria and viruses, and can adapt or "learn" to recognize various antigens, even those never encountered before. The clonal selection and immune network principle reveals this learning process [3]. When an antigen invaded, the antigens are recognized by some B cells. Then the recognition stimulates the B cells to proliferate. After the proliferation, the system has a clone of cells that are copies of each other. Affinity maturation is the mutation process of the clones generated. There are two basic mutation mechanisms, namely somatic hypermutation and receptor editing. In the hypermutation process, the antibody molecules are changed randomly, and lead to generate different cells. Occasionally, such changes may lead to an increase of the affinity of the antibody. The antibodies generated which have low affinity undergo a process of receptor editing, in which the receptors of the antibodies are deleted and develop entirely new receptors.

An idiotypic network hypothesis was recently proposed, which was derived based on the clonal selection theory. The portion on the anigen recognized by the antibody is called epitope that acts as an antigen determinant. Each type of antibody has its specific antigen determinant called idiotope. In the hypothesis, an immune system includes recognizing sets, some types of which are activated by some antigen and thus produce an antibody specific to the recognizing set. The activation is propagated through entire network of recognizing sets in this manner via antigen-antibody reactions. The theory reveals that the identification of an antigen is done by a systematic recognition of recognizing sets connected by antigen-antibody reactions.

3.2. Immune algorithm for solving job shop problem

The clonal selection principle described above reveals that high affinity antibodies are generated by cumulative selection and blind variation of B cells, which can be interpreted as a microcosm of Darwin's theory of evolution. Based on this evolutionary process, an immune algorithm is presented for solving the job shop scheduling problem. In the algorithm, each antibody represents a feasible schedule and the antibody affinity is viewed as the makespan of the schedule. The main steps of the algorithm are given in the following pseudo-code:

```
Generate a population of N antibodies by chaos system.
For each generation do:
  Decode the antibodies in the population
  Determine the affinity (makespan) of the antibodies
  Adjust the affinity of the antibodies by Niche method
  Select a number of highest affinity antibodies
  Antibody crossover
  Determine the number of clones for each antibody
  Antibody clones
  For each generated clone do
    Adding chaotic disturbances to generate a new antibody
    Decode the new antibody
    If makespan(new antibody)<makespan(clone) then
      clone=new antibody
    Else
      Adding large chaotic disturbances to generate a new
      antibody
      If makespan(new antibody)<makespan(clone) then
        clone=new antibody
      Else
        clone=clone
      End if
    End if
  End for
  Some lowest affinity antibodies are replaced with new
  antibodies generated by chaos system
End for
```

3.2.1. Representation of antibody

Each antibody represents a feasible schedule scheme. The representation of an antibody is based on random keys. Each gene of an antibody is a random number in the interval [0, 1], and corresponds to an operation of the job shop schedule. Assume the job shop has n operations, denoted by (O_1, O_2, …, O_n), then an antibody can be expressed as

$$Antibody = (Priority_1\ ,Priority_2\ ,Priority_3\ ,\cdots\ ,Priority_n)$$

where $Priority_i$ is the processing priority for the ith operation.

3.2.2. Population Initialization

The chaos system has special characteristics of ergodicity and randomicity, which make it more effective in local search and avoiding local optima [12-13]. This paper uses chaos system to generate antibodies and carry out antibody mutation. The chaos system used in this paper is the well-known Logistic mappings defined by

$$z_i^{j+1} = \mu_i z_i^j \left(1 - z_i^j\right),\ z_i^j \in [0,1],\ i = 1,2,\cdots,n,\ j = 1,2,\cdots, \quad (1)$$

where i is the serial number of chaotic variables, and z_i^j is the value of the chaotic variable z_i at the jth iteration. $\mu_i = 1$ is the chaotic attractor for i chaotic variables.

The initialization population can be generated by these chaotic variables. Let $j = 0$, and given the n chaotic variables different initial values z_i^0 ($i = 1,2,\cdots,n$), then the values of the n chaotic variables z_i^1 ($i = 1,2,\cdots,n$) are produced by the Logistic equation and encoded into a real-coded antibody. Let $j = 1,2,\cdots,N-1$, and then other $N-1$ antibodies are produced by the same method. The N antibodies compose the initial population.

3.2.3. Niche method

According to the immune network theory, the affinity of an antibody is determined by the antigen, as well as by other antibodies. The affinity between the antibody and the antigen is calculated by

$$Affinity(antibody) = \frac{1}{makespan(antibody)} \quad (2)$$

The affinity between the antibody and other antibody is determined by the distance between them. If the distance is small, then the affinity of the antibody is reduced. So we use the niche method [14] to suppress the similar antibodies to maintain large diversity of the population.

Assume the ith and jth antibodies in the population are $(x_1^i, x_2^i, \cdots, x_n^i)$ and $(x_1^j, x_2^j, \cdots, x_n^j)$ respectively, the Euclidean distance between them is

$$\left\| X^i - X^j \right\| = \sqrt{\sum_{k=1}^{n} (x_k^i - x_k^j)} \quad (3)$$

If the distance is small than a given real-value L, then the affinity of the two antibodies is compared and the one with lower affinity is penalized by a penalty function

$$\text{Affinity}\left(\min(antibody_i, antibody_j)\right) = Penalty \quad (4)$$

3.2.4. Antibody Crossover

The integer($\alpha.N$) highest affinity antibodies in the population are selected, where α is a real-valued constant and $0 < \alpha < 1$. α is defined as selection rate. The selected antibodies are sorted according to their affinity. The highest affinity antibody performs crossover operation with each of other antibodies, and thus integer($\alpha.N$) -1 new antibodies are generated and their affinities are calculated. Then the integer($\alpha.N$) highest affinity antibodies are selected again from the new generated antibodies and primary antibodies.

3.2.5. Antibody Clone

Each of the selected antibodies generates several clones. In immune system, the higher the affinity of an antibody, the larger number of clones generated for that antibody. So this paper uses the following method to determine the number of clones for each antibody. Assume the selected m antibodies are $v_1, v_2, \cdots, v_m$, and their affinity are $f_1, f_2, \cdots f_m$ respectively, then the probability of each antibody generating clones is

$$P_k = f_k \Big/ \sum_{i=1}^{m} f_i, \quad k = 1,2,\cdots,m \quad (5)$$

The accumulative probability is

$$q_k = \sum_{j=1}^{k} p_j, \quad k = 1,2,\cdots,m \quad (6)$$

A random number w is generated in the range [0,1], if $q_{k-1} \le w \le q_k$, then the antibody v_k generates a clone ($q_0 = 0$). Given N such random numbers, then the number of clones for each antibody can be calculated.

3.2.6. Hypermutation

Each of clones generated is submitted to a hypermutation process. In this process, the clones mutate to generate new antibodies in their small neighorhoods.

A clone $(x_1, x_2, \cdots, x_n)$ performs the hypermutation by the following equation.

$$x_i^{'} = x_i + \beta_i, \quad i = 1,2,\cdots,n \quad (7)$$

where $x_i^{'}$ is the i th gene for the new antibody generated; the chaotic disturbance β_i is given by

$$\beta_i = \alpha_1\left(2z_i^{j+1} - 1\right), \quad i = 1,2,\cdots,n \tag{8}$$

where z_i^{j+1} is the value of the i th chaotic variable at the j th iteration, and $\alpha_1 \in [0,1]$ is the scaling parameter, is used to control the mutation range. From equation (8), it is clear that β_i is a chaotic variable in the interval $[-\alpha_1, \alpha_1]$, so $x_i^{'}$ is also a chaotic variable in the interval $[x_i - \alpha_1, x_i + \alpha_1]$. If new generated antibody has higher affinity, then the clone is replaced with the new antibody, else the clone is added a larger chaotic disturbances:

$$\beta_i = \alpha_2\left(2z_i^{j+1} - 1\right), \quad i = 1,2,\cdots,n \tag{9}$$

where $\alpha_2 > \alpha_1$. If the new generated antibody has higher affinity, then the clone is replaced with the antibody, else the clone is kept unchanged.

3.2.7. Receptor Editing

In the process of receptor editing, the *d* lowest affinity antibodies are replaced with new antibodies generated by the chaos system in (1), where $d < N$. Receptor editing is used to maintain the diversity of the population.

3.3. Constructing parameterized active schedule

The antibody shown in section 3.2.1 is decoded as a parameterized active schedule. This section describes the process of constructing the parameterized active schedule.

Parameterized active schedule is an active schedule, in which the delay time of operations is no more than a given time $Delay$. Assume J is the set of all operations of the scheduling problem; S_g is the set of completed operations after gth iterates; F_g is the set of completing time for the completed operations in S_g; $A(t_k)$ is the set of all operations which are processing at time t_k, denoted by

$$A(t_k) = \{ j \in J \mid F_j - d_j \le t_k < F_j \} \tag{10}$$

where d_j is processing time of the operation j. Let $E(t_k, Delay)$ be the set of the operations which can be processed feasibly in the interval $[t_k, t_k + Delay]$, i.e., the anteceding operations of each operation in the set are all processed.

$$E(t_k, Delay) = \left\{ j \in J \setminus S_g \mid F_i \le t_k + Delay \quad (i \in P_j) \right\} \tag{11}$$

where P_j is the set of all anteceding operations for the operation j. Assume $RMC_m(t_k)$ is the occupy situation of the machine m at time t_k, where 0 denotes the machine is occupied and 1 denotes the machine can be used.

$$RMC_m(t_k) = 1 - \sum_{j \in A(t_k)} r_{j,m} \tag{12}$$

where $r_{j,m}$ =1 (0) means the machine *m* is (not) occupied by the job *j*.

Assume that there are a null operation 0 before all operations and a null operation n+1 after all operations. The begin time of operation 0 is 0, and the begin time of operation n+1 is the maximal complete time of all jobs. The processing time of operation 0 and n+1 are all 0. The constructing process of the parameterized active schedule is shown by the following pseudo-code.

(1) Initialization: Set scheduling parameter *Delay*, and let $g = 0$, $k = 0$, $t_0 = 0$, $F_0 = \{0\}$, $S_0 = \{0\}$, $A_0(0) = \{0\}$

(2) While $\left|S_g\right| \le n+1$ do

Calculate $A(t_k)$

Let $k = k+1$

Calculate $t_k = \underset{j \in A(t_{k-1})}{Min}\{F_j\}$

Calculate $E(t_k, Delay)$

While $E(t_k, Delay) \ne \phi$ do

Select the operation with highest priority in $E(t_k, Delay)$ (The priority is determined by the antibody)

Let $j^* = \underset{j \in E(t_k, Delay)}{\arg\max}\{Priority_j\}$

Calculate the earliest completing time F_{j^*} of j^*

$$EF_{j^*} = \max_{i \in P_{j^*}}\{F_i\} + d_{j^*}$$

$$F_{j^*} = \min\left\{t \mid t \in [EF_{j^*} - d_{j^*}, \infty] \cap RMC_m(\tau) = 1, \text{ where } \tau \in [t, t + d_{j^*}],\ r_{j^*,m} = 1\right\}$$

Let $g = g+1$

Update S_g and F_g

$S_g = S_{g-1} \cup \{j^*\}$

$F_g = F_{g-1} \cup \{F_{j^*}\}$

Calculate $E(t_k, Delay)$ again

End while

End while

(3) Calculate $C_{\max}$: $F_{n+1} = \underset{l \in P_{n+1}}{Max}\{F_l\}$

If the maximal delay time $Delay = 0$, then all operations of the job shop problem are processed without delay. In this situation, the antibody is decoded into a non-delay schedule. When the delay time $Delay \to \infty$, an active schedule is constructed. The solution space of the job shop problem is controlled by the parameter *Delay*.

4. Experimental Result

To verify the effectiveness of the presented method, the immune algorithm is used to optimize the benchmark job shop problem FT06. The algorithm is programmed in VB language. The parameters of the algorithm are given by N=20, α =0.2, α_1 =0.1, α_2 =0.3, d=5, L=2, and the parameter *Delay*=10. Gantt chart of the optimal schedule obtained by the algorithm is shown in figure 2.

Given the parameter *Delay* different values, and for each given *Delay*, the immune algorithm is used to optimize the scheduling problem for 10 times. The average generations required for obtaining the optimal schedule is shown in Table 1. When *Delay* is 200, the solution space consists of active schedules and the algorithm uses 39.5 generations to find the optimal schedule. When the parameter *Delay* is 10, the algorithm can find the optimal schedule in 2.2 generations. It can be observed that when the parameter *Delay* is given properly, the solution space of scheduling problems is reduced and the algorithm can find the optimal schedule very quickly.

Table 1. The average generations required for finding the optimal schedule

Delay	10	30	200
Generations	2.2	19.6	39.5

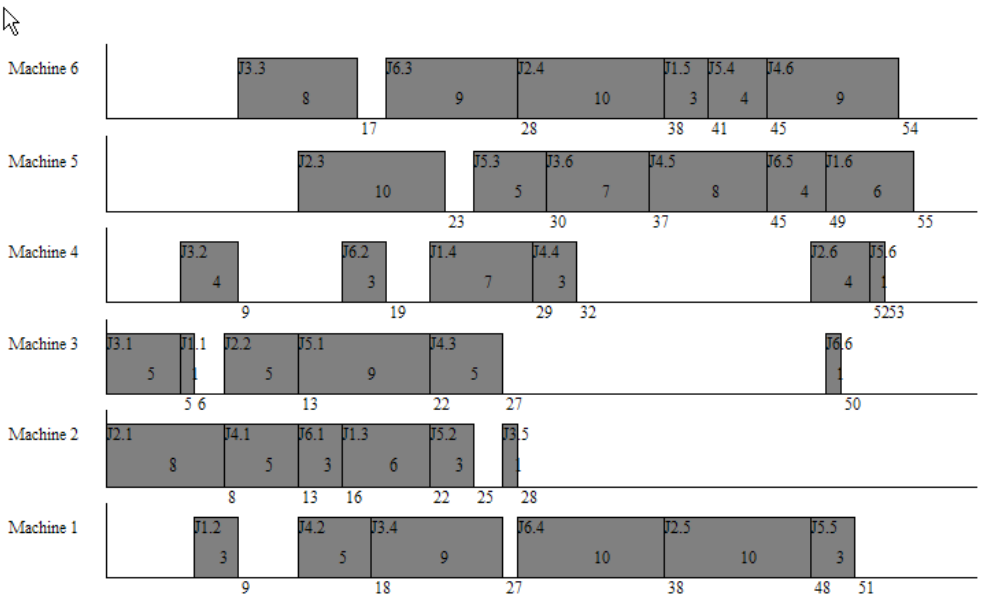

Figure 2. Gantt chart of the optimal schedule

5. Conclusions

This paper presented an immune algorithm for solving job shop schedule problems. The algorithm uses niche technology to avoid local optima and employs chaos system to improve search efficiency. A heuristic process is given to decode antibodies into parameterized active schedules to reduce the solution space. Experimental results demonstrate the effectiveness of the method for solving job shop scheduling problems.

Reference

[1] S. Hajri, N. Liouane, S. Hammadi and P. Borne. "A controlled genetic algorithm by fuzzy logic and belief functions for job-shop scheduling," IEEE Transactions

on System, Man, and Cybernetics Part B: Cybernetics, vol. 30, no. 5, pp. 812-818, 2000.

[2] S. X. Yang, D. W. Wang. "A new adaptive neural network and heuristics hybrid approach for job-shop scheduling," Computer & Operations Research, vol. 28, pp. 955-971, 2001.

[3] L. N. De Castro and J. Timmis, Artificial Immune Systems: A New Computational Intelligence Approach. Springer-verlag, 2002.

[4] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in Proc. IEEE Symposium on Research in Security and Privacy, Okaland, CA, 1994, pp. 202-212.

[5] D. Dasgupta and S. Forrest, "Novelty detection in time series data using ideas from immunology," in Proc. 5th International Conference on Intelligent systems, Reno, Nevada, USA, 1996, pp. 19-21.

[6] L. C. Jiao, and L. Wang, "A novel genetic algorithm based on immunity," IEEE Transactions on System, Man, and Cybernetics Part A: Systems and Humans, vol. 30, no. 5, pp. 552-561, 2000.

[7] L. N. De Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," IEEE Transactions on Evolutionary Computation, vol. 6, no. 3, pp. 239-251, 2002.

[8] X. Q. Zuo and S. Y. Li, "Solving function optimization problem with immune principle," Journal of Systems Engineering and Electronics, vol. 15, no. 4, pp. 702-709, 2004.

[9] X. G. Shao, Z. H. Chen, and X. Q. Lin. "Resolution of multicomponent overlapping chromatogram using an immune algorithm and genetic algorithm," Chemometrics and Intelligent Laboratory System, vol. 50, pp. 91-99, 2000.

[10] K. Mori, M. Tsukiyama, and T. Fukuda. "Adaptive scheduling system inspired by immune system," IEEE International Conference on System, Man, and Cybernetics, vol. 4, pp. 3833-3837, 1998.

[11] E. Orhan and D. Alper. "A new approach to solve hybrid flow shop scheduling problems by artificial immune algorithm," Future Generation Computer System, vol. 20, pp. 1083-1095, 2004.

[12] G. Chen and X. Dong, From chaos to order: Methodologies, perspectives and applications. Singapore: World Scientic, 1998.

[13] M. Hasegawa, T. Ikeguchi, K. Aihara, and K. Itoh, "A novel chaotic search for quadratic assignment problems," European Journal of Operational Research, vol. 139, pp. 543-556, 2002.

[14] D. E. Goldberg and J. Richardson. "Genetic algorithms with sharing for multimodal function optimization," In Proc. of 2nd Int. Conf. on Genetic Algorithms, Lawrence Erlbaum Associates, 1987, pp. 41-49.

[15] J. F. Gonçalves, J. J. M. Mendes, and M. G. C. Resende. "A hybrid genetic algorithm for the Job Shop Scheduling Problem," AT&T Labs Research Technical Report TD-5EAL6J, September, 2002.