A Novel Hybrid Public Critical Block Algorithm for Solving the Job Shop Scheduling Problems

Jun-qing Li
College of Computer Science, Liaocheng University, Liaocheng, 252059, People's Republic of China
lijunqing@lcu.edu.cn

Quan-ke Pan
College of Computer Science, Liaocheng University, Liaocheng, 252059, People's Republic of China
panquanke@lcu.edu.cn

Sheng-xian Xie
College of Computer Science, Liaocheng University, Liaocheng, 252059, People's Republic of China
xsx@lcu.edu.cn

Yu-ting Wang
College of Computer Science, Liaocheng University, Liaocheng, 252059, People's Republic of China
wangyuting@lcu.edu.cn

ABSTRACT

Job-shop Scheduling Problem (JSP) is one of the classical NP-hard problems. This paper designed a fast hybrid algorithm for solving the classical job shop scheduling problems. In this novel hybrid algorithm, Particle Swarm Optimization (PSO) and Tabu Search Algorithm (TSA) are combined to find the optimal solutions. The PSO was used to quickly find a set of acceptable solutions, and the TS were used to search more suitable solution near the acceptable solutions. A new concept named Public Critical Block (PCB) structure was proposed in this paper, which was used to decrease the search space in the TS algorithm. A detailed simulation indicates that the new algorithm has good efficiency and performance.

Key Words: Job-shop Scheduling Problem; Tabu Search; Particle Swarm Optimization; public critical block structure.

1. Introduction

There are many researches on the job shop scheduling problems (JSSP). JSSP has become a popular research focus in many domains including many companies; it has been a simplified model for many other problems such as cast-iron processing. There are *n* jobs and *m* machines during an operation process, JSSP problem will make an optimization to sequence the *n* jobs on the *m* machines to produce the best fitness, that is make the whole processing time shortest. There are two constraints that must be complied: (1) time constraint. Each job has a give processing time on every machine; (2) technical constraint. Each job must be processed according to the given processing sequence on every machine. The object of the JSSP is to optimize the job processing sequence on every machine, and make the whole processing time shortest, i.e., get a shortest makespan.

There are lots of effective algorithms which have been applied to solve the classical job shop scheduling problem, such as TSA (Tabu Search Algorithm), PSO (Particle Swarm Optimization), and ACO (Ant Colony Optimization). The above methods have got better results for solving certain JSP problems. However, there still exist many problem when apply these intelligent algorithm, such as the local search domain always seems too large to make a whole search. To decrease the search space and make the search working time shorter is the main topic in this paper. First, we give a novel and public critical block hybrid optimization algorithm based on the famous TSA and PSO algorithms. To decrease the local search space deeply, we also introduce a fast public critical block structure during the evolving process. At last, we make a fine experiment; a fine comparison with other famous algorithm is also given. Through detail analysis and comparison, we can conclude that our fast hybrid algorithm is effective to solve many large scale JSP problems.

2. The Related Algorithms

In this section, we will introduce some famous solving algorithms are also discussed.

2.1. Tabu Search Algorithm

TSA is proposed by Glober in 1986, which is a famous local search algorithm to solve combined optimize problem [9]. TSA has two main features: (1) the capability to avoid local optimization. TSA uses a tabu table to memory the better local neighbors which have been searched and will be neglected; (2) the capability to find better resolution. TSA uses an aspiration rule to exploit a prohibited resolution. During a situation that all the resolution in the tabu table is prohibited, the aspiration can make the whole search processing continue.

2.2. Particle Swarm Optimization

POS is proposed by Kennedy and Ebrhart in 1995, which is based on a swarm intelligent inspired by many birds. Through detailed observation of a swarm of birds to find what they want to eat, Kennedy find that, in order to search the food, the bird will compare their location and speed with two best samples: (1) global best particle. The global best particle is the best one in the bird swarm; each particle will have the same global best value in the same swarm; (2) local best particle. The local best particle will be different with different particle, which must be memorized by every particle itself; therefore, the local best value can be seemed as the history best value of it.

The PSO algorithm is initialized with a population of particles; each of the particles represents a candidate solution. The i^{th} particle in d-dimension solution space is denoted by $X_i = (x_{i1}, x_{i2}, ..., x_{id})$. The i^{th} particle is assigned a randomized velocity $V_i = (v_{i1}, v_{i2}, ..., v_{id})$ and is iteratively moved through the problem space. During the evolution phase, each particle follows two best values: $P_l = (p_{l1}, p_{l2}, ..., p_{ld})$, which is the best solution that the i^{th} particle has achieved so far and $P_g = (p_{g1}, p_{g2}, ..., p_{gd})$, which is the best solution obtained by the population so far. The update operator of each particle as follows:

$$v_{id} = w \times v_{id} + c_1 \times rand() \times (p_{ld} - x_{id}) + c_2 \times rand() \times (p_{gd} - x_{id})$$

$$x_{id} = x_{id} + v_{id}\ ,\ 0 \le w, c_1, c_2 \le 1$$

w is inertia weight, which is used to maintain the particle, and ; c_1 and c_2 are used to determine the proportion that the particle should study from personal and social history respectively.

2.3. Public Critical Block Structure

The critical problem of local search is how to define the effective neighborhood around the given solution. The promising neighborhood is based on the concept of critical path, which was firstly proposed by Adams (Adams, Balas, & Zawack, 1988) in solving JSP problems. Many researches have verified that block structure neighborhood will decrease the search space deeply. Block structure is based on the critical path. The critical path is composed by many critical operations which must be operated on the same machine. We will mark a structure $B = \{b_1, b_2, \cdots b_k\}$, here, we call b_1 and b_k the block header and block rear respectively, and $b_2, \cdots b_{k-1}$ the inner block operations. We propose a public critical block structure based on the basic critical block theory. The process of the public critical block was illustrated in Fig.1.

procedure : getPublicCriticalOperations
input: a set named M_c including all critical operations
output: all public critical operations
begin
  for every critical operation CO_i in M_c
    get the start time s_i and the end time e_i of the operation
    for every other critical operatins in M_c
      get the start time s_j and the end time e_j
      if occurs one of the case as followings:
        1) $e_i > e_j$ && $s_j < s_i$ && $s_i < e_j$
        2) $s_j < e_i$ && $e_i < e_j$ && $s_j > s_i$
        3) $s_j > s_i$ && $e_j < e_i$
        4) $s_i > s_j$ && $e_i < e_j$
    then mark the operation CO_i as a non-critical operation
  end for
 end for
 output the unmarked operations in M_c.
end

Fig.1. the process of PCB

2.4. Fast Block Insert Neighborhood

In order to describe the new algorithm, we propose some key definitions of our algorithm. First, we give some useful function definitions as follows:

- JN(b_i): This function aims to compute the job number of the operation b_i .
- MN(b_i): This function aims to compute the machine number of the operation b_i .
- $\gamma(b_i)$: This function aims to compute the start processing time of the operation b_i .
- $\varphi(b_i)$: This function aims to compute the complete processing time of the operation b_i .
- $\rho(b_i)$: This function aims to compute the processing time of the operation b_i .

Second, we give some useful neighborhood definitions as follows:

(1) *Pre-job*(b_i). We call $b_i^{'}$ the pre-job of the job b_i if and only if that b_i must be processed just after the competition of $b_i^{'}$ and the job number of $b_i^{'}$ and b_i are the same. The description of the pre-job can be illustrated as follows:

$$b_i^{'} \in \{\mathrm{JN}(b_i^{'}) = \mathrm{JN}(b_i) \wedge \varphi(b_i^{'}) \le r(b_i)\}$$

(2) *Post-job* (b_i). We call $b_i^{''}$ the post-job of the job b_i if and only if that b_i must be processed just before the competition of $b_i^{''}$ and the job number of $b_i^{''}$ and b_i are the same.

$$b_i^{''} \in \{\mathrm{JN}(b_i^{''}) = \mathrm{JN}(b_i) \wedge \varphi(b_i) \le r(b_i^{''})\}$$

(3) *Pre-machine*(b_i). We call ${}^{'}b_i$ the pre-machine of the job b_i if and only if that b_i must be processed just after the competition of ${}^{'}b_i$ and the machine number of ${}^{'}b_i$ and b_i are the same. The description of the pre-machine can be illustrated as follows:

$${}^{'}b_i \in \{\mathrm{MN}({}^{'}b_i) = \mathrm{MN}(b_i) \wedge \varphi({}^{'}b_i) \le r(b_i)\}$$

(4) *Post-machine*(b_i). We call ${}^{''}b_i$ the post-machine of the job b_i if and only if that b_i must be processed just before the competition of ${}^{''}b_i$ and the machine number of ${}^{''}b_i$ and b_i are the same. The description of the post-machine can be illustrated as follows:

$${}^{''}b_i \in \{\mathrm{MN}({}^{''}b_i) = \mathrm{MN}(b_i) \wedge \varphi(b_i) \le r({}^{''}b_i)\}$$

(5) *impro*(X). For two feasible scheduling such as X and Y, we say Y is an improvement of X if Y is the result of swapping two neighbors operations on one machine in Y scheduling, and the complete time of Y is shorter than X.

(6) If Y is the improvement of X, then Y can be computed through two ways as follows:

First, swapping the block header with the second block operation; second, swapping the block rear with the operation just before the rear in the critical path.

3. The flow of the algorithm

The new algorithm will integrate TSA and PSO, the detail flow of the algorithm as follows:

Step 1: Initialize the system parameters, and clear the tabu table;

Step 2: Random initialize the particle swarm, optimize the particle swarm with the PSO algorithm, and get the best particle in the particle swarm;

Step 3: To change the best particle's local optimization trend, we will give a slight chaos to the best particle in step 3;

Step 4: Compute the critical operations in the best particle, and get one critical path from the last critical operation, and then divide the whole critical operation on the critical path into several critical blocks;

Step 5: Search the whole local neighborhood of the best particle using the public critical block structure method, and memory the entire resolutions in a temporary table. In the temporary table, all the resolutions will be ordered by the complete time, i.e. the best resolution will on the top of the table;

Step 6: Search the entire resolutions in the temporary table, and to see whether it has occurred in the tabu table. If so, the resolution we find will be prohibited to be searched on, if not, we will give the resolution a mark to permit it to be searched on next time;

Step 7: Search the tabu table, and find the first resolution with a mark, and continue the loop to step 4; if none, the aspiration rule will be applied to continue the search process on the first resolution in the tabu table, i.e. the best particle in the tau table.

4. The experiment result

We realize the novel hybrid algorithm PCB-PSO-TSA (Public Critical Block based Particle Swarm Optimization and Tabu Search Algorithm) with VC++6.0 development environment, the PC is PIV3.0G, 1GB RAM. Under this condition, we make several tests for eleven classical problems, and make several detailed comparison with the other famous algorithms.

4.1. The experiment parameters

The important experiment parameters in our tests given in table 1, the means of these parameters as follows:

(1) PSO parameters

Particle scale: 20;

Inertia gene: 0.9;

Accelerate gene: $c_1=c_2=2$.

(2) TSA parameters

Tabu table length: If n equals to m, then length equals to $n*m/n$; when $n>>m$, then length equals to $n*m/2$;

Tabu period equals to $n*m/4$;

Stop condition: If find the best resolution or the iterate count exceeds the given number.

Table 1 **Experiments data**

problem	length of the tabu table	tabu period	iter count	inner iter count
ft06	4	4	100	100
ft10	12	12	300	500
ft20	50	6	300	300
la01	6	6	100	100
la06	9	9	100	100
la11	12	12	100	100
la16	12	12	200	500
la21	18	10	500	600
la26	70	20	300	400
la31	150	20	100	500
la36	50	15	600	600

4.2. The experiment result

We have completed independent 20 tests for every eleven problems, the comparison results are given in table 2, 3. From the table 2 and 3, we can conclude that:

(1) PCB-PSO-TSA can find the best resolutions for most problems, only for the hardest two problems LA21 and LA36. However, we can also get nearly best resolution for the hardest two problems.

(2) The average resolutions for our PCB-PSO-TSA algorithm are better than the most famous algorithm HPSO (Hybrid Particle Swarm Optimization) in ten problems except the LA16 problem. And for the LA16 problem, we can also get a nearly best resolution in a very short complete time.

(3) The average complete times for our PCB -PSO-TSA algorithm are better than the most famous algorithm HPSO, especially for the problem with large scale.

5. Conclusion

In order to decrease the local search space, we give a detailed analysis of the problem and propose a fast hybrid algorithm to solve the large scale problem. We propose six block swapping or inserting function in the evolve process, and the experiment result verify that our novel algorithm can decrease the local search space deeply, and the average complete time can save to a certain extent. The future work will focus on how to make our algorithm more robust and how to decrease the search space farther.

Table 2 **Comparisons of the makespans of many algorithms (Time unit: sec)**

problem	n,m	C*	PSO [7]	HPSO [5]	GASA [6]	TSAB [4]	SA [8]	PCB -PSO-TSA
ft06	6,6	55	55	55	55	55	55	55
ft10	10,10	930	930	930	930	930	930	930
ft20	20,5	1165	1165	1178	1165	1165	1165	1165
la01	10,5	666	666	666	666	666	666	666
la06	15,5	926	926	926	926	926	926	926
la11	20,5	1222	1222	1222	1222	1222	1222	1222
la16	10,10	945	945	945	945	945	956	945
la21	15,10	1046	1046	1047	1058	1047	1063	1048
la26	20,10	1218	1218	1218	1218	1218	1218	1218
la31	30,10	1784	1784	1784	1784	1784	1784	1784
la36	15,15	1268	1269	1269	1292	1268	1293	1270

Table 3 **Comparison of the PCB -PSO-TSA and the Hybrid PSO (Time unit: sec)**

	HPSO				PCB -PSO-TSA			
problem	avg	best	worst	avg time	avg	best	worst	avg time
ft06	55	55	55	7.0	55	55	55	0.02
ft10	947.1	930	959	164.7	937.6	930	945	10.35
ft20	1178.2	1165	1200	310.0	1171.30	1165	1180	23.5
la01	666	666	666	17.8	666	666	666	0.015
la06	926	926	926	61.1	926	926	926	0.05
la11	1222	1222	1222	145.6	1222	1222	1222	0.05
la16	945.4	945	946	148.3	945.70	945	946	4.51
la21	1064.3	1046	1088	733.7	1050.9	1048	1053	50.6
la26	1218	1218	1218	2449.5	1218	1218	1218	20.2
la31	1784	1784	1784	3883.3	1784	1784	1784	1.00
la36	1283.5	1269	1297	3541.7	1275.1	1270	1283	76.0

6. Acknowledgement

This research was partially supported by grants 2006AA01Z455 from the National High-Tech Research and Development Plan of China, 60874075, 70871065 from National Science Found from China, 2004ZX17 and 2004ZX14 from the Natural Science Foundation of Shandong Province, J08LJ20 from Science Research and Development of Provincial Department of Public Education of Shandong and X061015 from the Science Foundation of Liaocheng University.

References

[1] Wang Ling, Liu Bo. Particle swarm optimization and scheduling algorithms [M]. Beijing: Tsinghua University Press, 2008.

[2] Wang Ling. Shop Scheduling with Genetic Algorithms [M]. Beijing: Tsinghua University Press, 2003.
[3] Xing Wen-xun, Xie Jin-xing. Modern optimization algorithm [M]. Beijing: Tsinghua University Press, 2005.
[4] Nowicki E, Smutnicki C. A fast taboo search algorithm for the job-shop problem, Management Science, 1996, 42:797-813.
[5] Xia WJ, Wu ZM. A hybrid particle swarm optimization approach for the job-shop scheduling problem. International Journal of Advanced Manufacturing Technology, 2006, 29:360-366.
[6] Wang L, Zheng DZ. An effective optimization strategy for job-shop scheduling problems. Computer and Operations Research, 2001, 28:585-596.
[7] Sha DY, Hsu CY. A hybrid particle swarm for job shop scheduling problem. Computer and Industrial Engineering, 2006, 51:791-808.
[8] Van Laarhoven P J M, Aarts E H L, Lenstra J K. Job shop scheduling by simulated annealing. Operations Research, 1992,40:113-125
[9] Glover F., "Tabu Search: A Tutorial", Interfaces, 1990,20(4):74-94.
[10] Wang Ling. Intelligent Optimization Algorithms with Applications [M]. Beijing: Tsinghua University Press, 2001.