International Journal of Computer Theory and Engineering, Vol. 2, No. 2 April, 2010
1793-8201

A hybrid particle swarm optimization and tabu search algorithm for flexible job-shop scheduling problem

Jun-qing Li, *Member*, *IEEE,* Quan-ke Pan, Sheng-xian Xie, Bao-xian Jia and Yu-ting Wang

***Abstract*—Flexible job-shop scheduling problem (FJSP) is very important in many research fields such as production management and combinatorial optimization. The FJSP problems cover two difficulties namely machine assignment problem and operation sequencing problem. In this paper, a hybrid of particle swarm optimization (PSO) algorithm and tabu search (TS) algorithm are presented to solve the FJSP with the criterion to minimize the maximum completion time (makespan). In the novel hybrid algorithm, PSO was used to produce a swarm of high quality candidate solutions, while TS was used to obtain a near optimal solution around the given good solution. The computational results have proved that the proposed hybrid algorithm is efficient and effective for solving FJSP, especially for the problems with large scale.**

***Index Terms* -Flexible job shop scheduling problem; Particle swarm optimization; Tabu search algorithm; makespan**

I. Introduction

Flexible job-shop problem (FJSP) is harder than the classical job-shop problem. In FJSP, one operation can be operated on a set of machines [1-5]. Therefore, FJSP should solve two problems: first, assign a proper machine from a set of machines to operation each operation; second, sequence each operation on every given machine. The former problem can be seen as a parallel machine problem, which is also a NP-hard problem. The latter is equal to a classical job-shop problem.

The FJSP recently captured the interests of many researchers. The first paper about FJSP was proposed by Brucker and Schlie (Brucker & Schlie, 1990) [2], which discuss a simple FJSP model with two jobs and operations performed on any machines with the same processing time. To solve more genera FJSP problems with more than two jobs and machines, many researchers proposed hierarchical approaches, i.e., decomposing the problem into two stages: machine assignment sub-problem and job shop sub-problem. The first author to use the hierarchical idea was Brandimarte (Brandimarte, 1993) [3], who solved the first stage with some existing dispatching rules and the second stage with tabu search heuristic algorithms. Mati (Mati Rezg & Xie, 2001) [4] proposed a greedy heuristic for simultaneously dealing with the two stages. Liouane (Liouane, 2007) [5] illustrated a combined ant system optimization with local search methods, including tabu search for solving the FJSP problems. Saidi-mehrabad (Saidi-mehrabad, 2007) [6] gave a detailed solution for solving FJSP with tabu search method.

In this paper, we give a new chromosome representation for the FJSP solutions, and propose some novel crossover and mutation functions for the particle swarm optimization algorithm. In each generation, we use tabu search algorithm to find near optimum solutions for the given best solution. After a detailed experiment, the result verifies that our novel method can get better solutions in very short period. The paper is organized as follows: in section 2, we introduce the problem definition; in section 3, some related algorithms such as particle swarm optimization and tabu search algorithm are given respectively; in section 4, several initial solution rules are proposed; in section 5, the framework of the hybrid algorithm is described; at last, several detailed experimental comparison are made, and the analysis of the experimental results are given.

II. Problem Definition

In the FJSP problems, there are n jobs and m machines. Consider a set of n jobs, noted $J = \{J_1, J_2, ...J_n\}$, every job in the set J has a given number operations, and should be operated on a given machine from a machine set named $M = \{M_1, M_2, ...M_m\}$. There are two kinds of FJSP, i.e., T-FJSP and P-FJSP. For the T-FJSP, each job can be operated on every machine from the set M; for the P-FJSP, there is a problem constraint for the operating process, one operation of a job must be processed by a set of machines in $M^{'} \subseteq M$. The detailed definition of the FJSP as follows [5-7]:

Manuscript received Sep 24, 2009. This research is partially supported by National Science Foundation of China under Grants 60874075, 70871065, Open Research Foundation from State Key Laboratory of Digital Manufacturing Equipment and Technology (Huazhong University of Science and Technology), Science Research and Development of Provincial Department of Public Education of Shandong under Grant J08LJ20, J09LG29, J08LJ59, Soft Science Foundation of Shandong under Grant 2009REB125.

J.-Q. Li is with the College of Computer Science, Liaocheng University, Liaocheng 252059, People's Republic of China, (email: lijunqing@lcu.edu.cn; lijunqing.cn@gmail.com; p2p_jql@126.com).

Q.-K. Pan is with the College of Computer Science, Liaocheng University, Liaocheng 252059, People's Republic of China.

S.-X. Xie is with the College of Computer Science, Liaocheng University, Liaocheng 252059, People's Republic of China.

B-X. Jia is with the College of Computer Science, Liaocheng University, Liaocheng 252059, People's Republic of China.

Y-T. Wang is with the College of Computer Science, Liaocheng University, Liaocheng 252059, People's Republic of China.

IACSIT
International Association of
Computer Science and Information Technology
WWW.IACSIT.ORG

- A set of J independent jobs.
- Each job J_i can be operated on a given set of machines M_i.
- The $O_{i,j}$ represents the jth operation of J_i. The machines set waiting for processing the $O_{i,j}$ noted by $M_k \subseteq M$.
- We use $p_{i,j,k}$ to represent the processing time of $O_{i,j}$ operated on the kth machine.
- There have two assumptions: a started operation can not be interrupted; each machine only can process one operation at the same time.

The objective in our paper is to find the minimum time of the whole operations..

III. The Hybrid Algorithm For FJSP

A. Particle Swarm Optimization

1) The basic PSO

The PSO (Particle Swarm Optimization) algorithm is introduced in 1995 by Kenney and Russell (Kennedy & Eberhart, 1995) [8]. The PSO algorithm is initialized with a population of particles; each of the particles represents a candidate solution. The ith particle in d-dimension solution space is denoted by $X_i = (x_{i1}, x_{i2}, ..., x_{id})$. The ith 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 ith 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:

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

$$c_2 \times rand() \times (p_{gd} - x)$$

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

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

2) Discrete PSO

In traditional PSO, each particle learns from the local best particle and the global best particle. The traditional PSO if for continuous problem, but the FJSP problem is a combinatorial discrete problem. So, we must adjust the traditional PSO algorithm and make it adapt to solving the FJSP problems. Here, we introduce two operators: crossover and mutation operator into the traditional PSO.

The crossover operator is used in GA (Genetic Algorithm) to produce better chromosomes. With crossover operator, particles in a population can exchange information with others in the same population, and find the near-optimal solution quickly. Each particle in the population can learn from two particles: the local best particle, which is the best solution it has achieved so far; the global best particle, which is the best solution the population has achieved so far. Therefore, for the population, we allocate a memory space to record the local best for each particle. We also develop a special particle which records the global best in the population. With crossover operator, the traditional PSO algorithm should be redefined as follows [9]:

$$X_j^{l+1} = c_2 \otimes g(c_1 \otimes g(w \otimes h_{i,i'}(X_j^l), pB_j^l), gB^l)$$

The formulation has three stages:

$$E_j^l = w \otimes h_{i,i'}(X_j^l) = \begin{cases} h_{i,i'}(X_j^l), & rand() < w \\ X_j^l, & rand() \ge w \end{cases} \quad (1)$$

This part considers the learning information from itself. $h_{i,i'}(X_j^l)$ represents the new particle different from the old one. The update step as follows: first, randomize two location numbers i and i', then swap the element at the position i and i'. The result is $h_{i,i'}(X_j^l)$. The new particle after the first stage equals to $h_{i,i'}(X_j^l)$ with probability w and X_j^l with probability $1-w$.

$$F_j^l = c_1 \otimes g(E_j^l, pB_j^l) = \begin{cases} g(E_j^l, pB_j^l), & rand() < c_1 \\ E_j^l, & rand() \ge c_1 \end{cases} \quad (2)$$

In this stage, we consider the learning information from the local best of X_j^l. The learning process as follows: first, random select a segment in E_j^l, and concatenate this segment before or after the local best particle pB_j^l, and then delete the repeated elements in the new particle. The result particle is F_j^l. The new particle after the second stage equals to $g(E_j^l, pB_j^l)$ with probability c_1 and E_j^l with probability $1-c_1$.

$$X_j^{l+1} = c_2 \otimes g(F_j^l, gB^l) = \begin{cases} g(F_j^l, gB^l), & rand() < c_2 \\ F_j^l, & rand() \ge c_2 \end{cases} \quad (3)$$

This stage considers the learning information from the global best gB^l. The new particle is developed following three steps: first, random select a segment from F_j^l, and join this segment before or behind gB^l, and then delete the repeated elements in the new particle. The probability to learn from the global best is c_2.

B. Framework of the Hybrid Algorithm

In this paper, we use a hybrid algorithm with TS and PSO for solving the FJSP. The main framework of the hybrid algorithm as follows.

Step 1: Initialization

Step 1.1 Set parameters for the hybrid algorithm.

Step 1.2 Produce a swarm of initial solutions using the initialization rules listed in Section 4.

Step 2: Applying PSO algorithm

Step 2.1 Get the global best and local best particle in the swarm

Step 2.2 Apply the crossover function to each particle and the global best particle, and then the particle with the local best particle of it.

Step 2.3 Apply the mutation function to each resulted particle.

Step 2.4 Update the global best particle and the local best for each particle.

Step 3: Applying TS algorithm

Step 3.1 Get the global best in the current swarm.

Step 3.2 Randomly select an operation, select another machine for the operation, and update the *AString* of the current global best.

Step 3.3 Randomly swap two operations in *BString* of the current global best, and update the *BString* of the current global best.

Step 4: Applying fitness function

Step 4.1 Apply fitness function on the global best.

Step 4.2 if the stop criterion is satisfied, then output the current global best; otherwise, go to step 2 to start a new loop.

IV. Initialization Rules

The FJSP problems involve two decision stages, i.e., machine assignment stage and operation sequence stage. Therefore, a particle consists of two parts of vectors [11-13], *AString* (machine assignment vector) and *BString* (operation sequence vector). *AString* represents the corresponding selected machine for every operation; *BString* indicates the operation sequence on every machine. The length of vector *AString* equals the total number of operations. Each element in *AString* represents the selected machine number for processing the operation in that position.

The quality of initial population often affects the solution quality or the convergence speed of the population a certain extent. So, how to produce a better quality initial population becomes a critical step in the first section. The initialization process can be divided into two stages: machine assignment initialization and operation sequence initialization.

A. Machine Selection Stage

Following are several approaches for the initialization of machine assignment:

- Random rule, noted $\boldsymbol{MS_a}$. For every operation *i*, a random selected machine from a set of candidate machines, noted M_i, will be placed in position *i* in *AString*.
- Operation Minimum processing time rule, noted $\boldsymbol{MS_b}$. For every operation *i*, the machine with minimum processing time in the capable machine set M_i will be selected and placed in position *i* in *AString*.

B. Operation sequence Stage

Once the machine assignment is fixed, we should consider how to sequence the operations on every machine, i.e., to determine the start time of every operation. In our hybrid algorithm, the operation sequence is obtained through a mix of following two different approaches:

- Random rule, noted $\boldsymbol{OS_a}$. OS_a is the most obvious approach for operation sequence. The advantage of this approach is its simplicity. The disadvantage is also obvious too, it can easily produce idle time interval and make the finding solution process more time consuming.
- Most Work Remaining (MWR), noted $\boldsymbol{OS_b}$. This approach selects the operation with the most remaining work for every machine. The operation precedence constraint of the same job must be considered at the same time.

V. Computational Result

In our experiment, two sets of sample problems are used to analyze this instantiation. Test sample I includes the T-FJSP instances taken from [10]. Test sample Π includes the P-FJSP instances taken from [3]. The dimensions of the problems presented in literature typically range from 4 jobs × 5 machines to 20 jobs × 15 machines. The current instantiation was implemented in C++ on a Pentium IV running at 1.6GHz with 512M memory. The experimental described below were the best and average results obtained after 30 runs

The detailed parameters as follows:

- population size: 300;
- number of generations: 1000;
- rate of $\boldsymbol{ms_a}$ is 20%, $\boldsymbol{ms_b}$ is 80%;
- rate of $\boldsymbol{os_a}$ is 20%, $\boldsymbol{os_b}$ is 80%;
- tabu table length: n*m/2;
- tabu period: n*m/4;
- crossover probability: 40%;
- sequencing assignment mutation probability: 30%;
- machine assignment mutation probability: 60%;

C. Test sample I

The first test sample comes from [10]. The comparison is made among our algorithm and the other two well known algorithms, that is, Kacem et al [10]. and Ho et al [11]. The comparison result is shown in Table 1.

The first column is the number of jobs and the number of machines of each instance; the second column shows the results from Kacem; the third column gives the results from Ho; the fourth column displays the results from our algorithm; the next column gives the relative improvement between the best result of Kacem and our algorithm for each instance (((2nd column – 4nd column)/2nd column)*100) %; the last column shows the average time of our algorithm. It can be seen from Table 2 that our algorithm outperforms the Kacem algorithm in 3 out of 4 problems, and outperforms the Ho algorithm in 4 jobs × 5 machines instance. The average computational time is also acceptable in real manufacturing process. Fig.1, Fig.2, Fig.3, and Fig.4 show the Gantt chart for the above four instances respectively.

TABLE I. COMPUTATIONAL RESULT OF TEST SAMPLE I

size	[10]	[11]	our	Im (%)	Av (s)
4×5	16	**11***	12	25.00	6.3
10×7	25	**11***	**11***	56.00	6.5
10×10	**7***	**7***	**7***	0.00	5.1

15× 10	23	**12***	**11***	52.17	10.1

-Im means Improvement

-* represents the present best makespan

D. *Test sample Π*

The second test sample comes from Brandimarte [3]. The comparison is made among our algorithm and the other two well known algorithms, that is, Brandimarte et al. and Ho et al. The comparison result is shown in Table 2.

TABLE II. COMPUTATIONAL RESULT

Name	*n*	*m*	[3]	[11]	our
MK1	10	6	42	**40***	**40***
MK2	10	6	32	29	**27***
MK3	15	8	211	N/A	**204** *
MK4	15	8	81	67	**63***
MK5	15	4	186	176	**173** *
MK6	10	15	86	67	**65***
MK7	20	5	157	147	**145** *
MK8	20	10	**523***	**523***	**523** *
MK9	20	10	369	**320***	331
MK10	20	15	296	229	**223** *

-* represents the present best makespan

-N/A represents not given by the author

The first column is the test sample name; the second column and the third column tell the number of jobs and the number of machines of each instance, respectively; the fourth column shows the results from Brandimarte; the fifth column gives the results from Ho; the last column displays the results from our algorithm. It can be seen from Table 3 that our algorithm outperforms the Brandimarte algorithm in 9 out of 10 problems. The remained instance is MK8, which has reached its minimum makespan. Our algorithm outperforms the well known algorithm from Ho in 7 out of 10 problems. The only one problem on which Ho algorithm performs slight better than our algorithm is MK9. For the other two instances, our algorithm gets the same result with the Ho algorithm.

VI. CONCLUSION

In this paper, we give a novel hybrid algorithm with TS and PSO. In the sequencing stage, we use PSO to find the best solution, and in the machine assignment stage, we use TS algorithm to find the near optimal solution around the given particle. The computational result shows that our algorithm can get better result than the GA algorithm. The next work should focus on how to decrease the computation time and make the algorithm more robust.

REFERENCES

[1] Jun-qing Li, Quan-ke Pan, Sheng-xian Xie, Yu-ting Wang, "A Fast TS Algorithm for Solving the Flexible Job-shop Scheduling Problem". In Proceeding of International Conference on Information Technology (ICIT'2009), 2009, pp:5-4.

[2] Bruker P, Schlie R, "Job-shop scheduling with multi-purpose machines". Computing, 1990, 45(4), pp:369-375.

[3] Brandimarte P, "Routing and scheduling in a flexible job shop by tabu search". Annals of Operations Research, 1993, 22, pp:158-183.

[4] Matiy RZ, Xie X, "An integrated greedy heuristic for a flexible job shop scheduling problem". IEEE International Conference on Systems, Man and Cybernetics, 2001, 4, pp:2534-2539.

[5] Liouane N, Saad I, Hammadi S, Borne P, "Ant Systems & Local Search Optimization for Flexible Job-Shop Scheduling Production". International Journal of Computers, Communications & Control, 2007, pp: 2174-184.

[6] Saidi-mehrabad M, Fattahi P, "Flexible job shop scheduling with tabu search algorithms". International Journal of Advanced Manufacturing Technology, 2007, 32, pp:563-570.

[7] Glover F., "Tabu Search: A Tutorial", Interfaces, 1990, 20(4), pp:74-94.

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

[9] Q.K. Pan, B.H. Zhao, Y.G. Qu. "Heuristics for the No-Wait Flow Shop Problem with Makespan Criterion". Chinese Journal of Computers. 2008, 31 (07), pp: 1147-1154.

[10] Kacem, I., Hammadi, S. and Borne, P, "Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems", IEEE Transactions on Systems, Man and Cybernetics, Part C, 2002, 32(1), pp:408-419.

[11] Ho, N.B., Tay, J.C., Lai, E.M.K, "An Effective Architecture for Learning and Evolving Flexible Job-Shop Schedules", European Journal of Operations Research, . 2007, 179(2), pp:316-33

Junqing Li was born in Liaocheng, China, in 1976. He received the Master degree of computer science and technology in 1997 from Shandong Economic University, Shandong, China. He is currently an associate professor in Department of Computer Science and Technology, Liaocheng University, Liaocheng, China. He is now a member of IEEE and CCF. His research is related to the evolutionary optimization methods for discrete events systems, job shop systems, operational research, computer network, peer-to-peer systems and network security. He has (co-)authored around 50 research papers published. He has been serving on editorial boards or reviewers of two international journals. He has been the member of program committees of many international conferences.

Quan-ke Pan was born in Liaocheng, China, in 1971. He received the PhD of manufacturing and automation in 2003 from Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include computational intelligent, operational research and swarm optimization algorithms. He is currently a professor in Department of Computer Science and Technology, Liaocheng University, Liaocheng, China. He has (co-)authored around 120 research papers published. He has been serving on editorial boards of several international journals and has edited special issues in international journals. He has been member of program committees of many international conferences.

Sheng-xian Xie was born in Liaocheng, China, in 1957. He is currently a professor in Department of Computer Science and Technology, Liaocheng University, Liaocheng, China, and he is the chair of the department. His research interests include network security, access control and peer-to-peer systems.

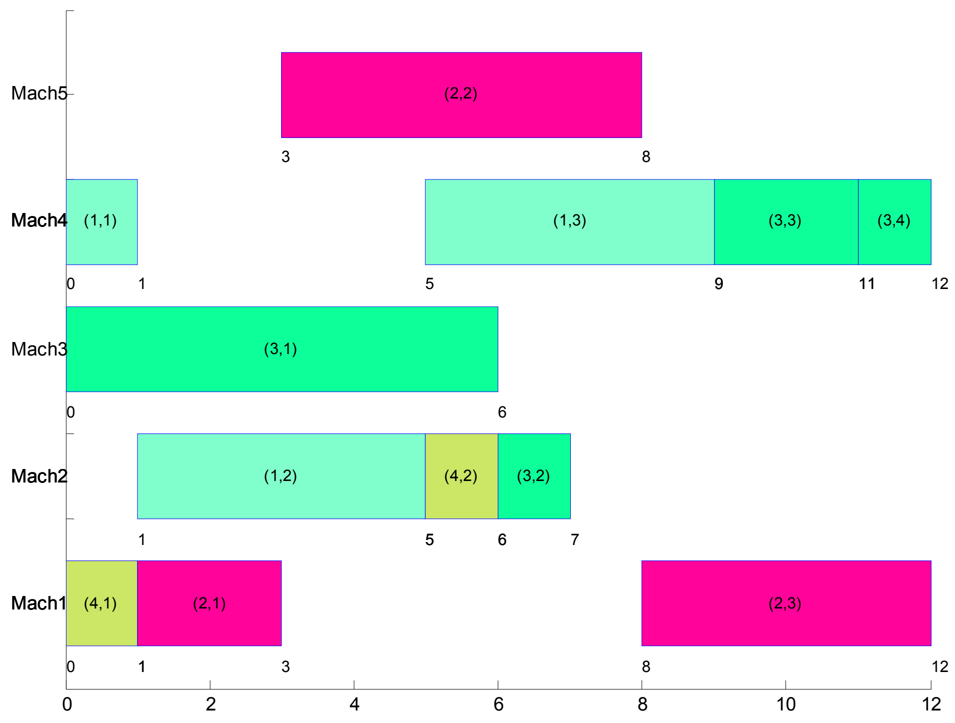

Fig.1.The Gantt chart of instance 1 (4 jobs * 5 machines)

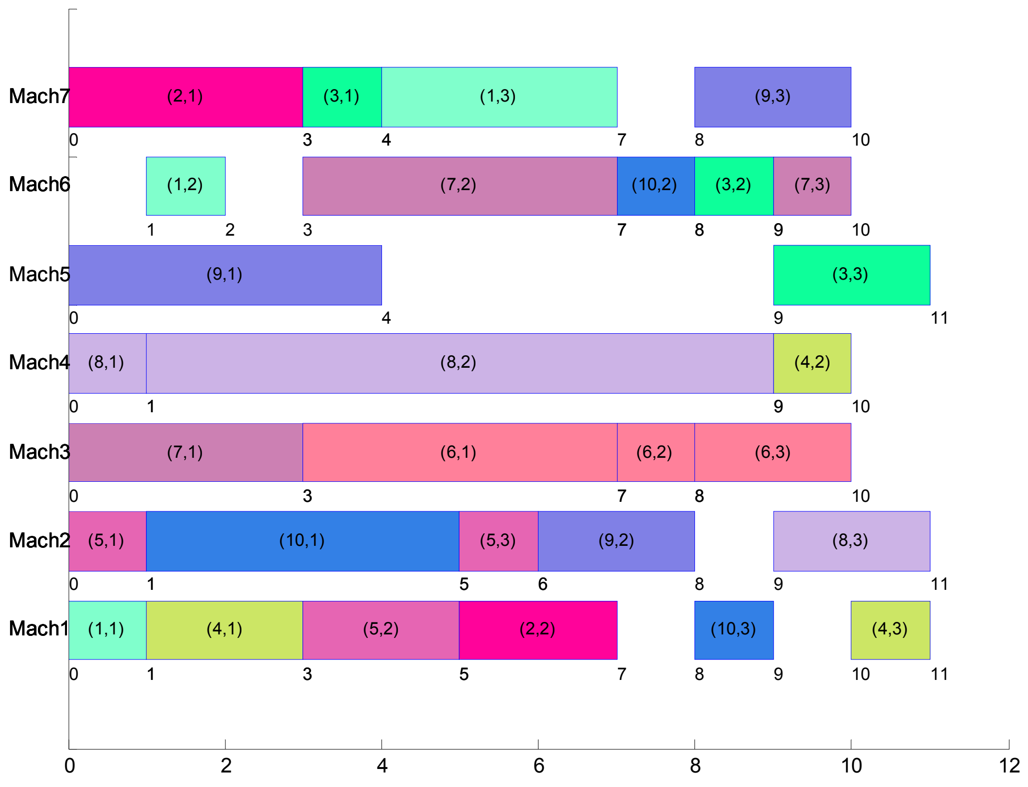

Fig.2.The Gantt chart of instance 2 (10 jobs * 7 machines)

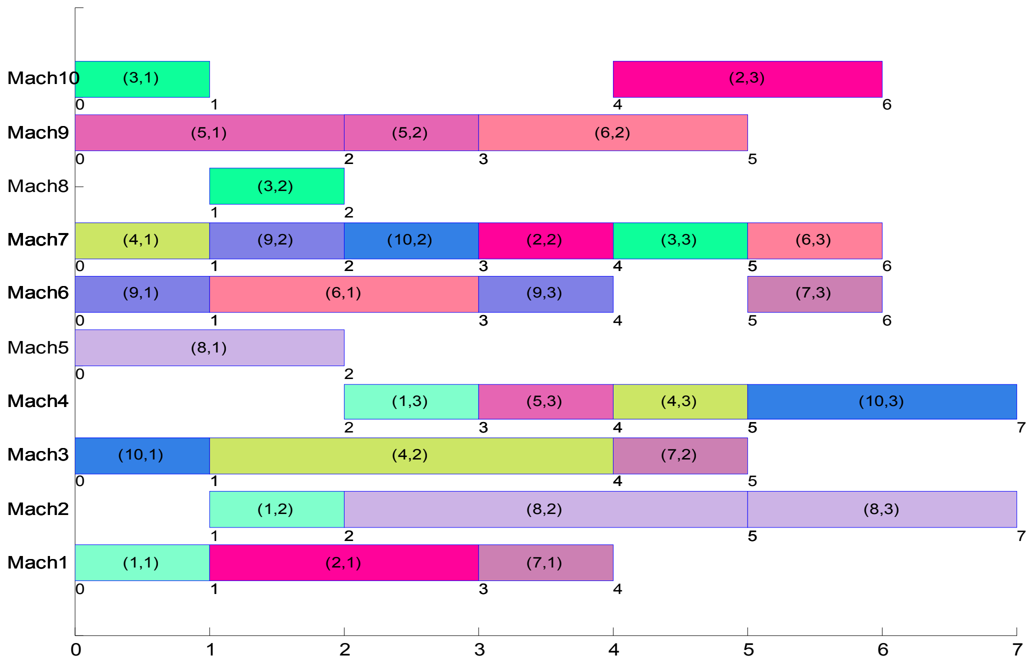

Fig.3.The Gantt chart of instance 3 (10 jobs * 10 machines)

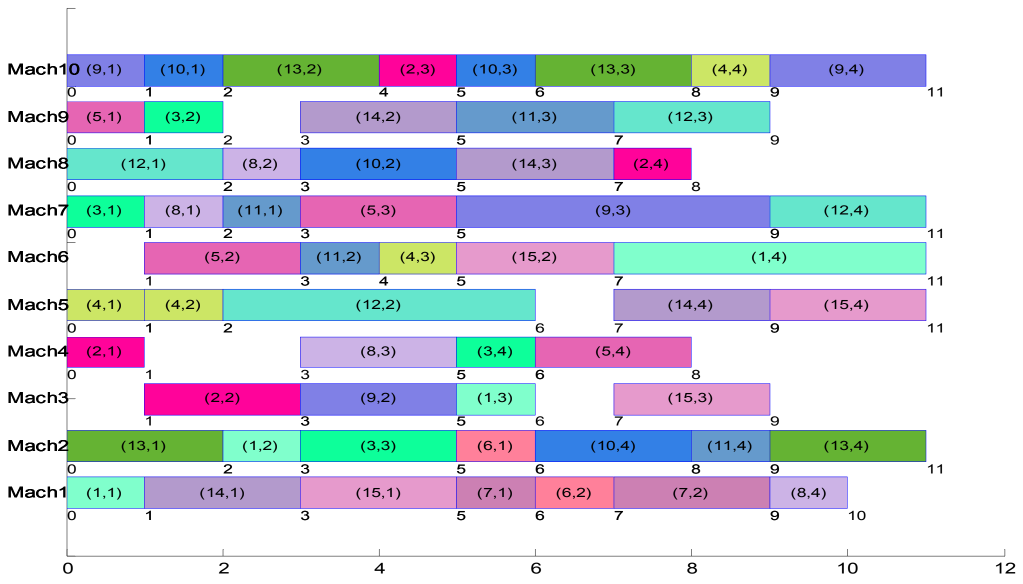

Fig.4.The Gantt chart of instance 4 (15 jobs * 10 machines)