Computers & Industrial Engineering 60 (2011) 699–705

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

# An efficient memetic algorithm for solving the job shop scheduling problem

Liang Gao [a], Guohui Zhang [b,*], Liping Zhang [a], Xinyu Li [a]

[a] The State Key Laboratory of Digital Manufacturing Equipment and Technology, 1037 Luoyu Road, Huazhong University of Science & Technology, Wuhan 430074, China
[b] Zhengzhou Institute of Aeronautical Industry Management, Middle Daxue Road, Zhengzhou 450015, China

ARTICLE INFO

*Article history:*
Received 31 July 2009
Received in revised form 29 December 2010
Accepted 4 January 2011
Available online 26 January 2011

*Keywords:*
Job shop scheduling
Memetic algorithm
Local search
Neighborhood structure

ABSTRACT

The job shop scheduling problem (JSP) is well known as one of the most complicated combinatorial optimization problems, and it is a NP-hard problem. Memetic algorithm (MA) which combines the global search and local search is a hybrid evolutionary algorithm. In this paper, an efficient MA with a novel local search is proposed to solve the JSP. Within the local search, a systematic change of the neighborhood is carried out to avoid trapping into local optimal. And two neighborhood structures are designed by exchanging and inserting based on the critical path. The objective of minimizing makespan is considered while satisfying a number of hard constraints. The computational results obtained in experiments demonstrate that the efficiency of the proposed MA is significantly superior to the other reported approaches in the literature.



## 1. Introduction

The job shop scheduling problem is well known as one of the most complicated combinatorial optimization problems (Garey, Johnson, & Sethi, 1976), and it is also well known for its practical applications in many manufacturing industries. A classical JSP is combined with $n$ different jobs and $m$ different machines. Each job consists of a set of operations and each operation requires a different machine. All the operations of each job are processed in a fixed processing order. And each operation has a given processing time. Some assumptions and constraints on jobs and machines are made while solving the JSP. A solution is to determine the operation sequences on the machines to satisfy some constraints. And the objectives usually considered in JSP are the minimization of makespan, the minimization of tardiness, and the minimization of mean flow time, etc. In this paper, the minimization of makespan is the objective. It is defined as the total time between the starting of the first operation and the ending of the last operation in all jobs.

JSP with the objective of minimizing makespan is one of the best known and strongly NP-hard problems (Garey et al., 1976). As a matter of fact, only small size instances among the benchmark problems within the literature could be solved within a reasonable computational time by exact optimization algorithm such as shifting bottleneck procedure (Adams, Balas, & Zawack, 1988), fast taboo search algorithm (Nowicki & Smutnicki, 1996), branch and bound (Akkan & Karabati, 2004) and dynamic programming (Lorigeon, 2002), etc. However, with the problem scale increasing, the computational time of the exact methods grows exponentially. By contrast with the exact methods, the approximate and heuristic algorithms make better tradeoff between solution quality and computational time. These methods mainly include dispatching priority rules (Canbolat & Gundogar, 2004), shifting bottleneck heuristic (Huang & Yin, 2004), lagrangian relaxation (Chen & Luh, 2003). The shifting bottleneck heuristic is one of the successful heuristics for minimizing the makespan in JSP. During the processing, each unscheduled machine is considered as a separate single machine, and the machine that gives the maximum delay in a single job is identified as a bottleneck machine, then the machine is scheduled first. In recent years, much attention has been devoted to the meta-heuristics with emergence of new techniques from the field of artificial intelligence such as tabu search (TS) (Zhang, Li, Guan, & Rao, 2007), simulated annealing (SA) (Kolonko, 1999), greedy randomized adaptive search procedure (GRASP) (Binato, Hery, Loewenstern, & Resende, 2002), ant colony optimization (ACO) (Udomsakdigool & Kachitvichyanukul, 2008), artificial neural network (ANN) (Yang & Wang, 2001), genetic algorithm (GA) (Tsai, Liu, Ho, & Chou, 2008), artificial immune system (AIS) (Luh & Chueh, 2009), particle swarm optimization (PSO) (Sha & Hsu, 2006), memetic algorithm (MA) (Hasan, Sarker, Essam, & Cornforth, 2009) and so on. These meta-heuristic algorithms could be regarded as independent approaches. However, no single algorithm is suitable for solving all kinds of JSP with both a reasonably good enough solution and within a reasonable computational time (Hasan et al., 2009). GA has an efficient exploration ability to search a wide range of search space. However, it does not have an effective local search mechanism for accurately searching near a good solution. So, many researchers improve the performance of GA by incorporating different search and heuristic techniques for solving the JSP.

* Corresponding author.
*E-mail address:* linghuizhang80@163.com (G. Zhang).

0360-8352/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2011.01.003

In this paper, an efficient memetic algorithm (MA) is proposed to solve the JSP. MA was introduced by Moscato and Norman (1992) to describe evolutionary algorithms in which local search plays a significant part. MA combines the global search and local search by using GA (Holland, 1975) to perform exploration and a local search method to perform exploitation. MA keeps well the balance between intensification and diversification. Radcliffe and Surry (1994) gave a formal description of MA. MA has been applied on many combinatorial optimization problems successfully such as the quadratic assignment problem (QAP) (Merz & Freisleben, 2000), the traveling sales man problem (TSP) (Buriol, Franca, & Moscato, 2004), the partitioning problem of tandem AGV systems (Elmekkawy & Liu, 2009) and so on. Franca, Mendes, and Moscato (2001) proposed a MA for the total tardiness single machine scheduling (SMS) problem with due dates and sequence-dependent set-up time. Yang, Sun, Lee, Qian, and Liang (2008) proposed a clonal selection based MA for solving JSP. In the clonal selection mechanism, clonal selection, hypermutation and receptor edit theories are presented to construct an evolutionary searching mechanism. Caumond, Lacomme, and Tchernev (2008) introduced a framework based on a disjunctive graph to model the JSP and use a MA for job sequence generation on machine. Hasan et al. (2009) proposed a MA with three priority rules to improve the performance of traditional MA for solving JSP.

In this research, a novel local search is proposed and combined with GA to improve the MA for solving the JSP. As the above mentioned, GA has an efficient exploration ability to search a wide range of search space, but GA is not well suited for local optimization. Combining the GA and local search, GA is used to perform global exploration among the population, and local search is used to perform local exploitation around chromosomes. Some benchmark problems of JSP were solved by the proposed MA. The computational results show that the MA is effective and efficient through compared with the reported approaches in several literatures.

The remainder of the paper is organized as follows. Section 2 describes the job shop scheduling problem. Section 3 presents the framework based on a memetic algorithm search scheme. Section 4 deals with the computational evaluation of the framework including benchmark problems. Section 5 is the conclusion and addressed several promising research directions.

## 2. Problem representations

In this research, the JSP consists of a set of jobs Job = $\{J_1, J_2, \ldots, J_n\}$ and a set of machines Machine = $\{M_1, M_2, \ldots, M_m\}$. The objective is to minimize the makespan, i.e., the completion time of the last job being completed in the system. In the JSP, several constraints and assumptions are made as follows:

- Each machine could process at most one job at a time.
- Each job is only processed by one machine at a time.
- The sequence of machines which a job visits is completely fixed and has a linear precedence structure.
- All jobs must be processed by each machine only once and there are at most $m$ operations for a job.
- There are no precedence constraints among the operations of different jobs.
- The machines are always available at zero and never break down.
- Processing time of all operations is known.

## 3. Job shop scheduling with memetic algorithm

MA could well balance its diversification and intensification to find high quality solutions of the optimization problem. Diversification is a search of different areas of the search space to find the most promising regions. Intensification is a search of the neighborhoods of the individuals to produce better solutions (Ong & Keane, 2004). In the proposed MA, the local search procedure is applied to each child to search for a better solution. The flowchart of the proposed MA in this paper is shown in Fig. 1.

*Step 1*: Generate initial population. Set parameters of GA including population size, max iteration, mutation probability, crossover probability, etc. Then encode an initial solution into a chromosome. Repeat this step until the number of individual equals to the population size.
*Step 2*: Apply the local search procedure to improve the quality of each individual.
*Step 3*: Decode each individual of population to obtain the makespan corresponding with each individual. And compare them to obtain the best solution.
*Step 4*: Check the termination criteria. If one of the criteria is satisfied, then stop the algorithm and output the best solution; otherwise, go to step 5.
*Step 5*: Generate new population for the next generation. Genetic evolution with three operators including selection, crossover and mutation is applied to create offspring for the next population. Following this, the algorithm goes back to step 2.

In the following sub-sections, chromosome representation and decoding, initial population, genetic evolution, local search procedure, and termination criteria will be discussed.

### 3.1. Chromosome representation and decoding

Better efficiency of MA-based search could be achieved by modifying the chromosome representation and its related operators so as to generate feasible solutions and avoid repair mechanism. In order to apply the MA to the JSP, a proper chromosome represen-

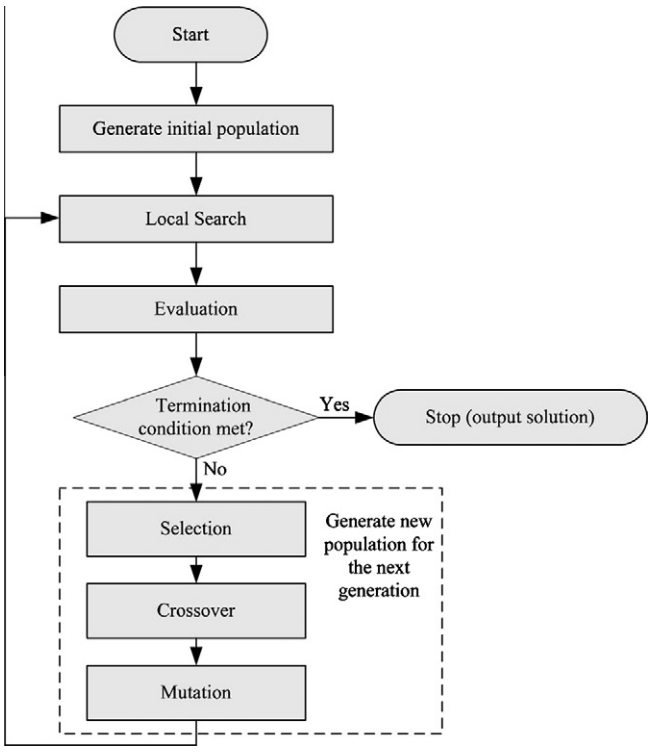


**Fig. 1.** Flowchart of the proposed MA.

tation of the solution for the JSP should be adopted. In this paper, the operation-based representation method is used to encode a schedule as a sequence of operations. For an $n$-job and $m$-machine problem, Each chromosome contains $n \times m$ genes and each gene is an integer. The number of different integers is equal to the number of jobs $n$. A job is a set of operations that has to be scheduled on $m$ machines, and each job appears $m$ times in the chromosome. By scanning the chromosome from left to right, the $j$th occurrence of a job number refers to the $j$th operation in the technological sequence of this job, then a chromosome could be decoded into a schedule indirectly. Although the schedule-to-fitness is not a one-to-one mapping, the representation method ensures that any permutation of the chromosome can be decoded to a feasible schedule. For example, a chromosome [2 3 1 2 2 1 3 1 3] is given, where {1 2 3} denotes the corresponding job $\{J_1, J_2, J_3\}$ respectively. There are three different integers, each is repeated three times. From the left to right, the first gene 2 represents the first operation of the second job to be processed first on the corresponding machine. Then, the second gene 3 represents the first operation of the third job. Therefore, the chromosome [2 3 1 2 2 1 3 1 3] is denoted as $[O_{21}\ O_{31}\ O_{11}\ O_{22}\ O_{23}\ O_{12}\ O_{32}\ O_{13}\ O_{33}]$, where $O_{ij}$ denotes the $j$th operation of $i$th job.

Decoding could obtain a schedule plan from a chromosome by scanning the gene from left to right, and the operations should be shifted to the left as compact as possible. In principles, a chromosome could be decoded into an infinite number of schedules, because superfluous idle time could be inserted between operations. Schedules are categorized into three classes: non-delay schedule, active schedule and semi-active schedule (Pinedo, 2002, chap. 2). A shift is called local left-shift if some operations could be started earlier in time without altering the operation sequence. A shift is called global left-shift if some operations can be started earlier in time without delaying any other operations even though the shift has changed the operation sequence. A schedule is active if no global left-shift exists. It has been verified and denoted in a Venn diagram in Pinedo (2002, chap. 2) that active schedule contains optimal schedule, so only the active schedule is considered in the decoding approach to reduce the search space.

### 3.2. Initial population

The initial population could be obtained by various methods such as the priority dispatching rules, the diverse insertion, the shifting bottleneck approach and random methods, etc. In general, the initial population methods have little influence on the final solution quality provided by the proposed algorithm, but they affect the running time. Therefore, according to the above chromosome representation method, the gene is randomly set in the range [1, $n$], and it must be guaranteed that each job number occurs $m$ times in every individual. Repeating this step until the number of chromosome equals the population size.

### 3.3. Genetic evolution strategy

Every initial chromosome is evaluated by the fitness function. If the termination criteria is not satisfied, the genetic evolution on the initial population is performed to generate the chromosomes in the new population for the next generation. Selection, crossover, and mutation are genetic operators to reproduce the new population. This will ensure that the best solution from the current population will be contained in the next population.

Obviously, the task of selection is choosing the individuals from the population. The chosen individuals are moved into a mating pool. In this research, the tournament approach (Park, Choi, & Kim, 2003) is adopted. Tournament approach could overcome the scaling problem of the direct fitness-based approach, and make good individuals have more "survival" opportunity. Meanwhile, it could avoid both the influence of the "super individual" and premature convergence.

In Crossover, firstly, a pair of parents (chromosomes) are selected randomly from the mating pool, and then the new offspring are created by exchanging the genetic information between the parents (Croce, Tadei, & Volta, 1995). Two parent strings are denoted as parent A and parent B, then two children strings are child A and child B. The crossover procedure could be described as follows. All jobs are divided into two sets Job Set 1 and Job Set 2 at random. Any gene in parent A which belongs to Job Set 1 will be retained in the same position in child A. And any gene in parent B which belongs to Job Set 2 will be retained in the same position in child B too. Then, the rest empty positions in child A will be filled by the genes of parent B that belong to Job Set 2 in the order presented in parent B. And the rest empty positions in child B will be filled by the genes of parent A that belong to Job Set 1 in the order presented in parent A too. The procedure is illustrated in Fig. 2.

Mutation introduces some extra variability into the population to enhance the diversity of population and to prevent the premature convergence of the population. Each child generated from the crossover operator has a probability to mutate some genes. A random number is generated and compared with the probability of mutation to determine whether the mutation operator must execute to the chromosome. This mutation procedure is demonstrated in Fig. 3. Positions 3 and 8 are randomly selected to be the mutation points. Exchanging the gene of position 3 (3) and 8 (1) respectively, then, the genes at position 3 and 8 are exchanged to generate the new offspring.

### 3.4. Local search procedure

MA applies a separate local search process to improve quality of the solutions. Local search moves from one solution to another solution in the neighborhood until an optimal solution founded. In this research, the neighbor solutions are mainly generated based on the critical path. The features of this proposed local search method are discussed in the following sub-sections.

#### 3.4.1. Defining neighborhood

JSP could be represented with a disjunctive graph introduced by Balas (1969). The disjunctive graph $G := (N, A, E)$ is defined as follows: $N$ represents the set of nodes representing all operations $\{0, O_{11}, O_{12}, \ldots n \times m + 1\}$, where 0 and $n \times m + 1$ denotes the dummy start and end operations respectively. $A$ is the set of ordinary (conjunctive) arcs connecting consecutive operations of the same job, and $E$ is the set of disjunctive arcs connecting operations processed by the same machine. The length of an arc denotes the processing time processed on corresponding machine. Table 1 describes an example of JSP including three jobs and three machines. The disjunctive graph is shown in Fig. 4. The real arcs ($A$)

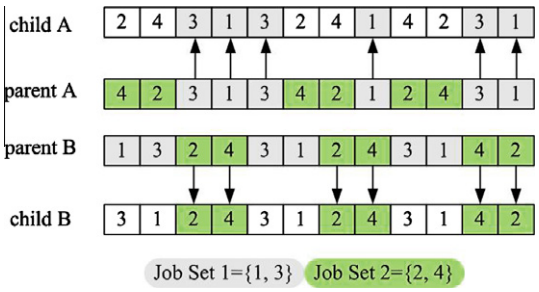


**Fig. 2.** An example of crossover procedure.

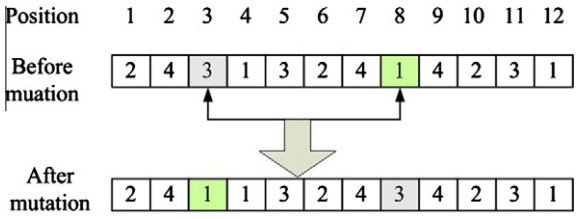


**Fig. 3.** An example of mutation procedure.

**Table 1**
An example of three jobs and three machines.

| Jobs | Machine sequence | Processing time |
|---|---|---|
| $J_1$ | 1–2–3 | 3–2–5 |
| $J_2$ | 1–3–2 | 3–5–2 |
| $J_3$ | 2–1–3 | 2–5–3 |

to precedence relations, and the dashed arcs ($E$) to pairs of operations performed on the same machine.

A key component of a feasible solution is the critical path, which is the longest path from start to end in directed graph and its length represents the makespan. Any operation on the critical path is called a critical operation. In Fig. 5 the length of the critical path is 19 and the critical path is {0–$O_{21}$–$O_{11}$–$O_{32}$–$O_{33}$–$O_{13}$–10}. The critical path is highlighted with broad-brush arcs. It is also possible to decompose the critical path into a number of blocks. A block is a maximal sequence of adjacent critical operations that is processed on the same machine. In Fig. 5, the critical operations are {$O_{21}$, $O_{11}$, $O_{32}$, $O_{33}$, $O_{13}$}, and the critical path is divided into two blocks, block 1 = {$O_{21}$, $O_{11}$, $O_{32}$} and block 2 = {$O_{33}$, $O_{13}$}.

A critical operation cannot be delayed without increasing the makespan of the schedule. There are several definitions of neighborhood. In this research, two neighborhood structures based on the critical block are adopted: exchange and insert. As the following detailed:

*Exchange*: Exchange is a function used to move around in which any two randomly selected operations based on critical block are simply swapped. If the critical block only has one operation, then make no exchange. If the critical block only has two operations, then swap them. If the critical block contains more than two operations, then randomly select two operations and swap them.

*Insert*: Insert is another effective function that removes a randomly selected gene, and inserts it in the front or back of another randomly selected gene on critical block. If the critical block only has one operation or two operations, then make no insert. If the critical block contains more than two operations, then randomly select two operations and inserted one operation in the front or back of another operation.

### 3.4.2. Local search procedure

The local search is very important for exploitation of the MA. The pseudo-code of the proposed local search in this paper is shown in Fig. 6. The basic idea of the local search procedure is changing the two neighbor structures to avoid trapping in a local optimum.

The local search for all chromosomes may cost much computation time. It might be a good idea to limit application of the local search procedure to several good chromosomes in each population. The proposed MA is applied to investigate the impact of the percentage of the best chromosomes. Our experiment results show that increasing the number of the good chromosomes used in the local search would increase the computation time, but it may not improve the quality of the final solution.

### 3.5. Termination criteria

Termination criteria are used to determine whether the algorithm should stop. In this research, if one of the following condi-

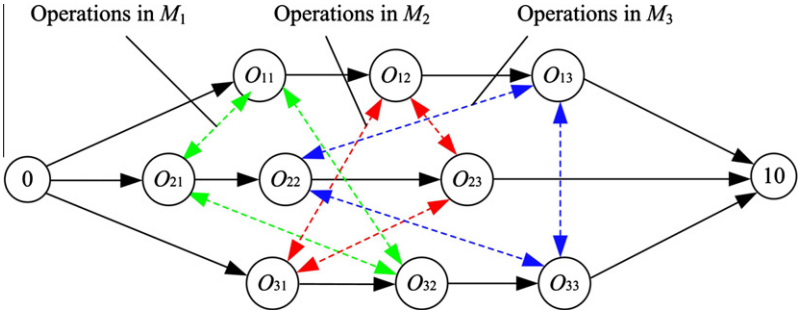


**Fig. 4.** The disjunctive graph of the example in Table 1.

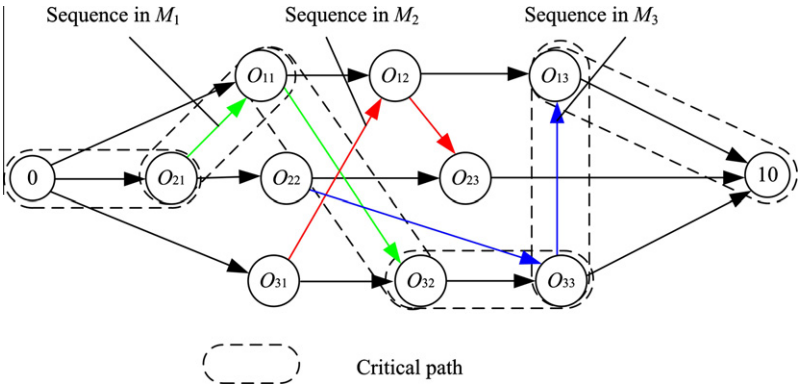


**Fig. 5.** An feasible solution for the disjunctive graph in Fig. 4.

```
procedure: the proposed local search
input: one individual s from population
begin
    while ( terminated condition not met ) do
        k = 1
        s' = Exchange(s)
        while ( k≤2 ) do
            if ( k=1 ) then
                s'' = Insert (s')
            if ( k=2 ) then
                s'' = Exchange (s')
            if ( Cm(s'')<Cm(s') ) then
                s' = s''
            else
                k = k+1
        end while
        if ( Cm(s'')<Cm(s) ) then
            s = s''
    end while
end
```

**Fig. 6.** The pseudo-code of the proposed local search.

tions is satisfied: (1) the number of iterations is equal to the maximum number of generations; (2) the makespan is equal to the known lower bound, then the algorithm terminates, and output the optimal solution.

## 4. Computational results

The proposed MA is applied to solve the JSP. The JSP benchmark problems are taken from the OR-Library (Beasley, 1990) which is well known in this field. We consider 43 instances from two classes of the JSP benchmark problems. Numerical experiments are performed in C++ language on a PC with Pentium IV 1.8 GHz processor and 512 MB memory. The proposed algorithm runs 20 times for each benchmark problem. The numerical results are compared with those reported studies using other approaches.

It is worth mentioning that there are some parameters to be determined in the proposed algorithm. However, the performance of the algorithm varies greatly for solving different instances, when some parameters are set differently. Some parameters on the proposed MA are taken as: crossover probability 0.8, mutation probability 0.01 and the max iterations 150. Then the remaining

**Table 2**
Comparisons of the results between MA and other approaches.

| Problem | Size | POP | BKS | MA | PGA (Park et al., 2003) | MA(GR-RS) (Hasan et al., 2009) | HGA (Goncalves et al., 2005) | F&F (Rego & Duarte, 2009) |
|---|---|---|---|---|---|---|---|---|
| Ft06 | 6 × 6 | 50 | 55 | 55* | 55 | 55 | 55 | 55 |
| Ft10 | 10 × 10 | 100 | 930 | 930* | 936 | 930 | 930 | 930 |
| Ft20 | 20 × 5 | 100 | 1165 | 1165* | 1177 | 1165 | 1165 | 1165 |
| La01 | 10 × 5 | 100 | 666 | 666* | 666 | 666 | 666 | 666 |
| La02 | 10 × 5 | 100 | 655 | 655* | 666 | 655 | 655 | 655 |
| La03 | 10 × 5 | 100 | 597 | 597* | 597 | 597 | 597 | 597 |
| La04 | 10 × 5 | 100 | 590 | 590* | 590 | 590 | 590 | 590 |
| La05 | 10 × 5 | 100 | 593 | 593* | 593 | 593 | 593 | 593 |
| La06 | 15 × 5 | 100 | 926 | 926* | 926 | 926 | 926 | 926 |
| La07 | 15 × 5 | 100 | 890 | 890* | 890 | 890 | 890 | 890 |
| La08 | 15 × 5 | 100 | 863 | 863* | 863 | 863 | 863 | 863 |
| La09 | 15 × 5 | 100 | 951 | 951* | 951 | 951 | 951 | 951 |
| La10 | 15 × 5 | 100 | 958 | 958* | 958 | 958 | 958 | 958 |
| La11 | 20 × 5 | 100 | 1222 | 1222* | 1222 | 1222 | 1222 | 1222 |
| La12 | 20 × 5 | 100 | 1039 | 1039* | 1039 | 1039 | 1039 | 1039 |
| La13 | 20 × 5 | 100 | 1150 | 1150* | 1150 | 1150 | 1150 | 1150 |
| La14 | 20 × 5 | 100 | 1292 | 1292* | 1292 | 1292 | 1292 | 1292 |
| La15 | 20 × 5 | 100 | 1207 | 1207* | 1207 | 1207 | 1207 | 1207 |
| La16 | 10 × 10 | 200 | 945 | 945* | 977 | 945 | 945 | 947 |
| La17 | 10 × 10 | 200 | 784 | 784* | 787 | 784 | 784 | 784 |
| La18 | 10 × 10 | 200 | 848 | 848* | 848 | 848 | 848 | 848 |
| La19 | 10 × 10 | 200 | 842 | 842* | 857 | 842 | 842 | 846 |
| La20 | 10 × 10 | 200 | 902 | 902* | 910 | 907 | 907 | 907 |
| La21 | 15 × 10 | 300 | 1046 | 1055 | 1047 | 1079 | 1046 | 1052 |
| La22 | 15 × 10 | 300 | 927 | 927* | 936 | 960 | 935 | 927 |
| La23 | 15 × 10 | 300 | 1032 | 1032* | 1032 | 1032 | 1032 | 1032 |
| La24 | 15 × 10 | 300 | 935 | 940 | 955 | 959 | 953 | 941 |
| La25 | 15 × 10 | 300 | 977 | 984 | 1004 | 991 | 986 | 982 |
| La26 | 20 × 10 | 400 | 1218 | 1218* | 1218 | 1218 | 1218 | 1218 |
| La27 | 20 × 10 | 400 | 1235 | 1261 | 1260 | 1286 | 1256 | 1242 |
| La28 | 20 × 10 | 400 | 1216 | 1216* | 1241 | 1286 | 1232 | 1225 |
| La29 | 20 × 10 | 400 | 1152 | 1190 | 1190 | 1221 | 1196 | 1176 |
| La30 | 20 × 10 | 300 | 1355 | 1355* | 1356 | 1355 | 1355 | 1355 |
| La31 | 30 × 10 | 200 | 1784 | 1784* | 1784 | 1784 | 1784 | 1784 |
| La32 | 30 × 10 | 200 | 1850 | 1850* | 1850 | 1850 | 1850 | 1850 |
| La33 | 30 × 10 | 200 | 1719 | 1719* | 1719 | 1719 | 1719 | 1719 |
| La34 | 30 × 10 | 300 | 1721 | 1721* | 1730 | 1721 | 1721 | 1721 |
| La35 | 30 × 10 | 200 | 1888 | 1888* | 1888 | 1888 | 1888 | 1888 |
| La36 | 15 × 15 | 500 | 1268 | 1281 | 1305 | 1307 | 1279 | 1281 |
| La37 | 15 × 15 | 500 | 1397 | 1431 | 1441 | 1442 | 1408 | 1418 |
| La38 | 15 × 15 | 500 | 1196 | 1216 | 1248 | 1266 | 1219 | 1213 |
| La39 | 15 × 15 | 500 | 1233 | 1241 | 1264 | 1252 | 1246 | 1250 |
| La40 | 15 × 15 | 500 | 1222 | 1233 | 1252 | 1252 | 1241 | 1228 |

The solutions marked by * are optimal.

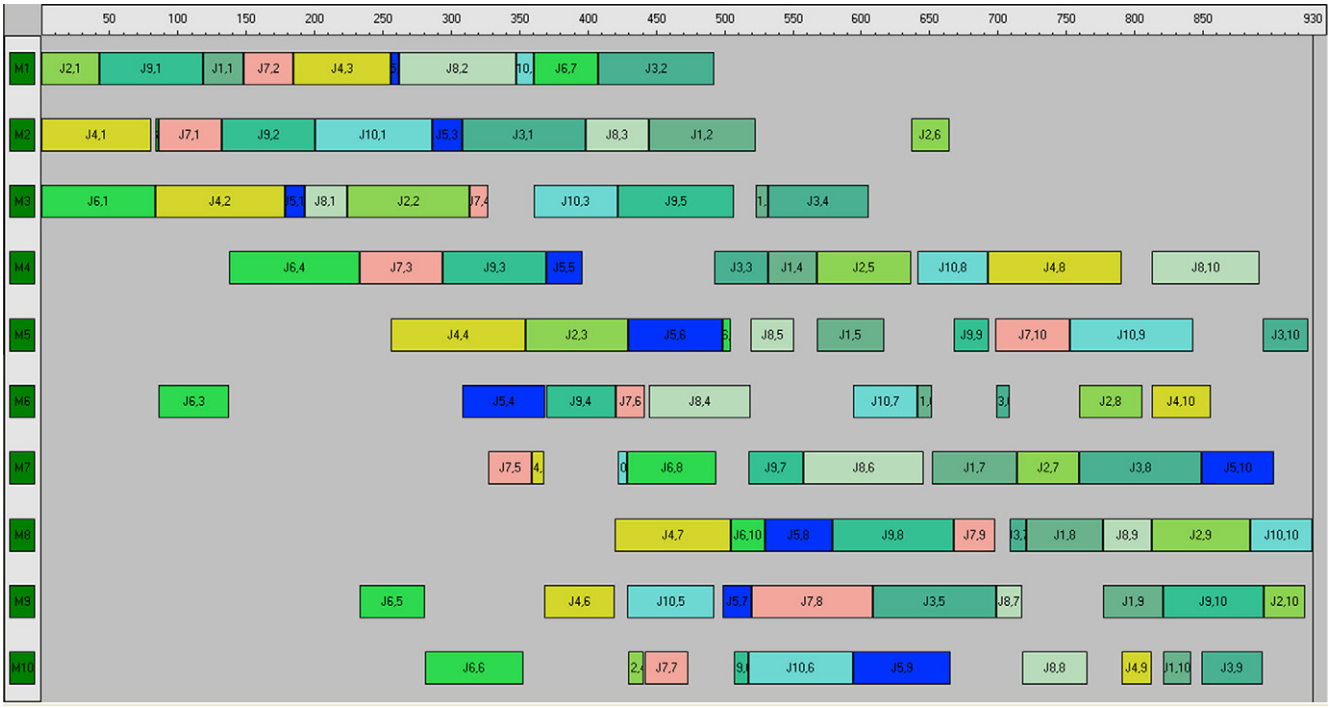


**Fig. 7.** The Gantt chart of an optimal solution of Ft10.

parameter POP in Table 2 is changed to obtain satisfactory solutions in reasonable number of population size. And in the running process, if the best solution is not update over 15 times, then the algorithm is stopped.

Table 2 summarizes the results of the experiments. The columns of the table include the name of each test benchmark problem (Problem), the size of the problem (Size), the number of population size (POP), the value of the best known solution for each problem (BKS), the value of the best solution found by using the proposed MA (MA), the solution obtained by Park et al. (2003) who proposed a hybrid genetic algorithm based on single genetic algorithm and parallel genetic algorithm (PGA); Hasan et al. (2009) who proposed a memetic algorithm with three priority rules (MA(GR-RS)); Goncalves, Mendes, and Resende (2005) who proposed a hybrid genetic algorithm(HGA), and the solution obtained by Rego and Duarte (2009) who proposed a filter-and-fan approach (F&F).

From Table 2, it could be seen that 33 best known solutions could be found among 43 experiment problems by using the proposed MA, i.e. in 79% of problem instances. There are 17 solutions better and 2 solutions worse than Park et al. (2003). There are 13 solutions better than Hasan et al. (2009). There are 9 solutions better and 4 solutions worse than Goncalves et al. (2005). And there are 6 solutions better and 7 solutions worse than Rego and Duarte (2009).

Compared with Park et al. (2003), Hasan et al. (2009), Goncalves et al. (2005) and Rego and Duarte (2009), the proposed MA shows better performance. One common problem in classical GA is premature convergence (Park et al., 2003). The proposed local search is the extension of N1 proposed by Blazewicz, Domschke, and Pesch (1996). It can improve local search performance and avoid the prematurity of the classical GA. Computational results shows that two neighborhood structures exchange and insert are good local search methods.

To illustrate the experiment results more intuitively, the problem Ft10 is described as an example. Fig. 7 shows the Gantt chart of an optimal solution obtained by the proposed MA.

## 5. Conclusions

Memetic algorithm is a hybrid evolutionary algorithm that combines the global search and local search. In this paper, an improved memetic algorithm is proposed to solve the job shop scheduling problem. In local search of the proposed algorithm, a systematic change of neighborhood is carried out to avoid trapping into local optimal and improve the quality of each solution. The two neighborhood structures based on the critical path are designed by exchanging and inserting. And the improved memetic algorithm takes advantage of the global search ability of the genetic algorithm. The computational results obtained in experiments demonstrate the efficiency of the proposed memetic algorithm, which is significantly superior to the other reported methods. In the future, the memetic algorithm can be extended to solve other practical manufacturing problems, such as flexible job shop scheduling problem, dynamic scheduling problem and so on.

## Acknowledgements

This project is supported the National Natural Science Foundation of China No. 50825503 and Program for New Century Excellent Talents in University under Grant No. NCET-08-0232. And we wish to thank the anonymous referees for their constructive and useful comments.

## References

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job-shop scheduling. *Management, 34*, 391–401.

Akkan, C., & Karabati, S. (2004). The two-machine flowshop total completion time problem: Improved lower bounds and a branch-and-bound algorithm. *European Journal of Operational Research, 159*(2), 420–429.

Balas, E. (1969). Machine sequencing via disjunctive graph: An implicit numeration algorithm. *Operations Research, 17*, 941–957.

Beasley, J. E. (1990). OR-library: Distributing test problems by electronic mail. *Journal of Operational Research Society, 41*(11), 1069–1072.

Binato, S., Hery, W. J., Loewenstern, D. M., & Resende, M. G. C. (2002). A GRASP for job shop scheduling. In C. Ribiero & P. Hansen (Eds.), *Essays and surveys on metaheuristics* (pp. 59–80). Kluwer.

Blazewicz, J., Domschke, W., & Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research, 93*, 1–33.

Buriol, L., Franca, P. M., & Moscato, P. (2004). A new memetic algorithm for the asymmetric traveling sales man problem. *Journal of Heuristics, 10*, 483–506.

Canbolat, Y. B., & Gundogar, E. (2004). Fuzzy priority rule for job shop scheduling. *Journal of Intelligent Manufacturing, 15*(4), 527–533.

Caumond, A., Lacomme, P., & Tchernev, N. (2008). A memetic algorithm for the job-shop with time-lags. *Computers & Operations Research, 35*, 2331–2356.

Chen, H. X., & Luh, P. B. (2003). An alternative framework to Lagrangian relaxation approach for job shop scheduling. *European Journal of Operational Research, 149*(3), 499–512.

Croce, F. D., Tadei, R., & Volta, G. (1995). A genetic algorithm for the job shop problem. *Computer and Operations Research, 22*(1), 15–24.

Elmekkawy, T. Y., & Liu, S. (2009). A new memtic algorithm for optimizing the partitioning problem of tandem AGV systems. *International Journal of Production Economics, 118*, 508–520.

Franca, P. M., Mendes, A., & Moscato, P. (2001). A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research, 132*, 224–242.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research, 1*, 117–129.

Goncalves, J. F., Mendes, J. M., & Resende, M. C. G. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research, 167*, 77–95.

Hasan, S. M. K., Sarker, R., Essam, D., & Cornforth, D. (2009). Memetic algorithm for solving job-shop scheduling problems. *Memetic Computing, 1*, 69–83.

Holland, J. H. (1975). *Adaption in natural and artificial systems*. Ann Arbor: The University of Michigan Press.

Huang, W. Q., & Yin, A. H. (2004). An improved shifting bottleneck procedure for the job shop scheduling problem. *Computers & Operations Research, 31*(12), 2093–2110.

Kolonko, M. (1999). Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operational Research, 113*(1), 123–136.

Lorigeon, T. (2002). A dynamic programming algorithm for scheduling jobs in a two-machine open shop with an availability constraint. *Journal of the Operational Research Society, 53*(11), 1239–1246.

Luh, G. C., & Chueh, C. H. (2009). A multi-model immune algorithm for the job shop scheduling problem. *Information Sciences, 179*, 1516–1532.

Merz, P., & Freisleben, B. (2000). Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Transaction on Evolutionary Computation, 4*(4), 337–352.

Moscato, P., & Norman, M. G. (1992). A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. *Parallel Computing and Transputer Applications, 1*, 177–186.

Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management Science, 42*(6), 797–813.

Ong, Y. S., & Keane, A. J. (2004). Meta-Lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation, 8*, 99–110.

Park, B. J., Choi, H. R., & Kim, H. S. (2003). A hybrid genetic algorithm for the job shop scheduling problems. *Computers & Industrial Engineering, 45*(4), 597–613.

Pinedo, M. (2002). *Scheduling theory, algorithms, and systems*. Englewood Cliffs, NJ: Prentice-Hall.

Radcliffe, N. J., Surry, P. D. (1994). Formal memetic algorithms. In T. Fogarty (Ed.), *AISB workshop, evolutionary computing, lecture notes in computer science, selected papers* (pp. 1–16).

Rego, C., & Duarte, R. (2009). A filter-and-fan approach to the job shop scheduling problem. *European Journal of Operational Research, 194*, 650–662.

Sha, D. Y., & Hsu, C. Y. (2006). A hybrid particle swarm optimization for job shop scheduling problem. *Computers & Industrial Engineering, 51*, 791–808.

Tsai, J. T., Liu, T. K., Ho, W. H., & Chou, J. H. (2008). An improved genetic algorithm for job-shop scheduling problems using Taguchi-based crossover. *International Journal of Advanced Manufacture Technology, 38*, 987–994.

Udomsakdigool, A., & Kachitvichyanukul, V. (2008). Multiple colony ant algorithm for job-shop scheduling problem. *International Journal of Production Research, 46*(15), 4155–4175.

Yang, J. H., Sun, L., Lee, H. P., Qian, Y., & Liang, Y. C. (2008). Clonal selection based memetic algorithm for job shop scheduling problems. *Journal of Bionic Engineering, 5*, 111–119.

Yang, S. X., & Wang, D. W. (2001). A new adaptive neural network and heuristics hybrid approach for job-shop scheduling. *Computers & Operations Research, 28*, 955–971.

Zhang, C. Y., Li, P. G., Guan, Z. L., & Rao, Y. Q. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research, 34*, 3229–3242.