2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Memetic Algorithm for Solving Flexible Job-shop Scheduling Problems

Wenping Ma, Yi Zuo, Jiulin Zeng, Shuang Liang, Licheng Jiao

***Abstract*—The flexible Job-shop Scheduling Problem (FJSP) is an extension of the classical job-shop scheduling problem (JSP). In this paper, a memetic algorithm (MA) for the FJSP is presented. This MA is a hybrid genetic algorithm which explores the search space and two efficient local searchers to exploit information in the search region. An extensive computational study on 49 benchmark problems shows that the algorithm is effective and robust, with respect to other well-known effective algorithms.**

Keywords-flexible job-shop scheduling; memetic algorithm; tabu search; simulated annealing.

I. INTRODUCTION

Scheduling is one of the most critical issues in the planning and manufacturing processes. One of the most popular scheduling models is the job-shop scheduling problem (JSP), where a set of jobs must be processed on a set of machines. Each job is formed by a sequence of consecutive operations, and each operation requires exactly one machine at a time. JSP has been proved to be NP-hard [1]. The flexible job-shop scheduling problem (FJSP) is an extension of the classical JSP, where operations are allowed to be processed on a set of available machines. FJSP is more difficult than the classical JSP, since it introduces routing before scheduling. In recent years, several heuristic procedures such as dispatching rules [2], local search strategies and meta-heuristics including simulated annealing (SA) [3], tabu search (TS) [4]–[7] and genetic algorithm (GAs) [8]–[10] have been developed for FJSP.

Generally, the algorithms for the FJSP can be classified into two main categories: hierarchical approach and integrated approach. The hierarchical approach attempts to solve the problem by decomposing it into a sequence of sub-problems to reduce difficulty. A typical decomposition is to select an available machine for each operation first, and then the resulting scheduling problem is JSP. This approach is followed by Brandimarte [4], Paulli [11], Barnes and Chanbers [12], among the others. They all solve the assignment problem using some dispatching rules, and then solve the resulting JSP using different tabu search heuristics. Integrated approach is much more difficult to solve, but achieves better results generally, as reported in [13]–[16]. They all adopt an integrated approach, developing different tabu search to solve the problem. Among them, Mastrolilli and Gambardella presented two neighborhood functions and show their efficiency for the FJSP [13].

Recently, GAs have been successfully adopted to solve FJSP. Chen et al. [12] split the chromosome representation into two parts, the first defining the routing policy, and the second the sequence of operations on each machine. Ho and Tay [17] proposed a methodology based on a cultural evolutionary architecture for solving FJSP with recirculation. Pezzella et al. [15] proposed a GA, which integrates different strategies for generating the initial population, selecting the individuals for reproduction and reproducing new individuals. Gao et al. [14] developed an approach hybridizing genetic algorithm with variable neighborhood descent to solve the FJSP.

Memetic algorithms (MAs) , as being the combination of population-based search methods and one or more local search strategies, have been successfully applied on many complex problems [18]. In this paper, we present a memetic algorithm (MA) for the FJSP. Our MA is a hybrid GA that uses a genetic search method to explore the search space and two efficient local search methods .The local searchers efficiently exploits information in the search region. The MA has been tested on popular benchmark problems, and the experimental results show that the MA can achieve better performance for all the popular benchmark problems.

This paper is organized as follows. In Section II, we review some background, including the problem definition and the solution graph representation. The MA is discussed in detail in Section III, and experimental studies are presented in Section IV. Finally, conclusion is given in Section V.

II. BACKGROUND

A. The definition of FJSP

The FJSP is formulated as follows:

(1) Let $J = \{J_1, ..., J_n\}$ be a set of n jobs to be scheduled.

(2) Each job J_i consists of a sequence of n_i operations $J_i = \{O_{i,1}, ..., O_{i,n_i}\}$.

(3) Let $M = \{M_1, ..., M_m\}$ be a set of m machines.

(4) Each machine can process only one operation at a time.

(5) Each operation must be processed without interruption.

This work was supported by the National Natural Science Foundation of China (Grant no. 61203303), the Specialized Research Fund for the Doctoral Program of Higher Education (Grant no. 20130203110011) and the Fundamental Research Fund for the Central Universities (Grant no. K5051202053).

The authors are with Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, Xidian University, Xi'an, 710071, China.

The objective of this problem is to find a schedule that has minimum time required to complete all operations, defined as $C_{\max} = \max_{1\le i\le n}\{C_i\}$, where C_i is the completion time of J_i.

B. The solution graph

The schedules of FJSPs can be represented with a directed graph $G=(N, A, E)$, with node set N, precedence arc set A, disjunctive arc set E. The set A corresponds to operations, and the set E denotes immediate implementation sequence of operations to be performed on the same machine. Two dummy nodes, 0 and * are introduced, representing the start and the end of the planning period. Each node has a weight which is equal to the processing time $p_{v,\mu(\mathrm{v})}$ of the corresponding operation v, when v is processed on machine $\mu(v)$. Note that $p_0 = p_* = 0$. Let $L(i, j)$ denote the value of some longest path from node i to node j, the makespan of a solution is thus equal to the length of some longest paths from 0 to *, i.e. $L(0,*)$. This path is often referred to as the critical path. Sometimes there are several longest paths. A solution is infeasible if and only if the corresponding solution graph contains a cycle.

III. OUR MEMETIC ALGORITEM

A. Representing and coding solutions

The FJSP is a combination of machine assignment and operation scheduling decisions, so a solution can be expressed by the assignment of operations on machines and the processing sequence of operations on the machine. The chromosome is therefore composed of two parts: machine assignment vector and operation sequence vector. For the first part, we adopt Gen et al.'s representation [19]. All operations belonged to a job are denoted by the same job index. Then, they are interpreted according to the order of occurrence in the sequence of a given chromosome. Each job J_i appears in the operation sequence vector exactly n_i times to represent its n_i ordered operations. The main advantage of this representation is that each possible chromosome always represents a feasible operation sequence. In the second part, at first, we list all jobs according the order $J_1,...,J_n$, and then make an enumeration of all operations belonged to the same job according to the precedence constraints. Then we get a set of operations $\{\mathrm{O}_1, \mathrm{O}_2, ..., \mathrm{O}_N\}$, where N is the number of all operations. Each operation O_i has its unique permutation order i. Each position p_i, in the machine assignment vector corresponds to the operation whose permutation order is equal to i. The corresponding value in p_i indicates a randomly selected available machine for the operation O_i.

We decode an individual according to its chromosome to get its solution graph. Then the makespan $L(0,*)$ and the length of some longest path between an operation node v and a dummy node 0 or *, i.e. $L(0, v)$ or $L(v,*)$, can be computed using Bellman's ford algorithm [20] in $O(N)$.

B. Crossover and mutation operators

During the past decades, several crossover operators have been proposed for permutation representation, such as partial-mapped crossover, order crossover, cycle crossover, and so on [21]. In this paper, we apply the order crossover for the operation sequence vectors. The order crossover works as follows:

Step 1: Select a subsection of operation sequence from one parent at random.

Step 2: Produce a proto-child by copying the substring of operation sequence into the corresponding positions.

Step 3: Delete the operations that are already in the substring from the second parent. The resulted sequence of operations contains operations that the proto-child needs.

Step 4: Place the operations into the unfixed positions of the proto-child from left to right according to the order of the sequence in the second parent.

We use two crossover operators at equivalent probability for the machine assignment vectors: extended order crossover and uniform crossover. The extended order crossover is related to crossover for operation sequence. It copies the machine assigned for an operation from the same parent where its operation sequence comes. Uniform crossover is accomplished by taking an allele from either parental machine assignment vector to form the corresponding allele of the child.

In this study, two kinds of mutation operations are implemented: allele-based mutation and immigration mutation [21]. For machine assignment vectors, allele-based mutation randomly decides whether an allele should be selected for mutation with a certain probability. Then, another available machine will be assigned for the operation indicated by the selected allele. For operation sequence vectors, allele-based mutation randomly decides whether to mutate an allele r. If allele r is to be mutated, then another allele is randomly selected to exchange with it. Immigration mutation randomly generates a number of new members of the population from the same distribution as the initial population.

Algorithm 1: Tabu Search

Begin
Initialize tabu list TM, iter =: iter + 1;
While Stop Condition is not satisfied
  If the smallest estimated length of the new longest path containing v decrease the best makespan obtained so far
    The *best* move of v is always accepted, i.e. this is an aspiration criteria
  Else if several non-tabu moves exist
    the *best* non-tabu move of v is chosen.
  Else if only tabu moves are available
    The chosen move is the one (v, k) with the lowest value $\mathrm{TM}(v, k)$
  End if
  $\mathrm{TM}(v, k) = iter + |P| + |M_v|$
End while
End

Algorithm 2: Simulated Annealing

Begin
While Stop Condition is not satisfied
Calculate $M = 1/T$.
For $i = 1: M$
Randomly select a critical operation v from current solution S .
Calculate F_{vk} .
Performing an approximate optimal k -insertion of v .
Evaluate the new solution S' .
If $S'.makespan < S.makespan$
$S = S'$.
Else
Generate a pseudo-random value $\xi \in [0,1]$.
If $\xi < 0.01$
$S = S'$
End if
End if
Reduce temperature;
End for
End while
End

C. *Local searchers*

1) *The Neighborhood Function*

The local search method employed by our MA is based on a neighborhood function proposed by Mastrolilli and Gambardella [14].

In combinatorial domain, the neighborhood of a solution x is defined to be the set of solutions which can be reached from x by a single step of the local search algorithm. Given the initial solution graph, a neighbor is obtained by moving and inserting an operation in an allowed machine sequence. This procedure is described as follows:

Step 1: Delete v from its current machine sequence by removing all its machine arcs. Set the weight of node v equal to 0.

Step 2: Assign v to machine k and choose the position of v in the processing order of k , by adding its machine arcs and setting the weight of node v equal to p_{vk} .

Let G^{-} be the graph obtained from G at the end of step 1. A k -insertion of v is feasible if it does not create a cycle in the resulting graph. If G is acyclic, G^{-} is obviously acyclic. A k -insertion is called an optimal k -insertion if it is feasible and the makespan of the corresponding schedule is minimal. An insertion of v is called optimal if it leads to a schedule with minimal makespan within the set of all schedules resulting from optimal k -insertion of v , $k \in M_v$. Let Q_k be the set of operations processed by k in G^{-} ($v \notin Q_k$) and sorted by increasing starting time. Let R_k and L_k denote two subsequences of Q_k defined as follows:

$$R_k = \{x \in Q_k \mid L(0,x) + p_x \geqslant L(0,PJ[x]) + p_{PJ[x]}\} \quad (1)$$

$$L_k = \{x \in Q_k \mid L(x,*) + p_x \geqslant L(SJ[x],*) + p_{SJ[x]}\} \quad (2)$$

where $PJ[x](SJ(x))$ denotes the operation of the same job of x that directly precedes (follows) x .

The set F_{vk} is defined to be the set of solutions obtained by inserting v after all the operations of $L_k \setminus R_k$ and before all the operations of $R_k \setminus L_k$. Mastrolilli and Gambardella [14] proved that other k -insertions than the ones used to define F_{vk} cannot deliver a solution with a better makespan.

In order to assess the effectiveness of a given k -insertion of v we use the value of the new longest path which contains operation v can be calculated in $O(N)$ time; however, doing this for every candidate v and every machine $k \in M_v$ at every step becomes expensive. A new strategy is introduced to compute only upper bounds instead of the exact values. The k -insertion of v for which the estimated longest path is minimized is called the approximate optimal k -insertion. Experiments show that the proposed upper bound is very close to the exact length. It was on average only 0.001 percent bigger than the exact value [14].

2) *Tabu Search*

Tabu search uses a neighborhood search procedure to iteratively move from solution S to an improved solution S' in the neighborhood of S . Tabu list is a short-term set of the solutions that have been visited in recent past. In order to keep track of the actions performed, we use a $N \times m$ matrix. When an action is performed it is considered tabu for the next T iterations, where T is the tabu status length. A solution is forbidden if it is obtained by applying a tabu action to the current solution. A best move is the one with the smallest estimated length of the new longest path containing the moved operation. If several non-tabu moves exist, the next one is randomly chosen between the best two non-tabu moves. This method is useful to decrease the probability of generating cycles. In order to explore the search space in a more efficient way, tabu search is usually augmented with some aspiration criteria. Those are used to accept a move even if it has been marked tabu. Finally, when only tabu moves are available, the chosen solution is the one (v,k) with the lowest value $TM(v,k)$. The basic scheme is presented in Algorithm 1.

3) *Simulated Annealing*

The simulated annealing meta-heuristic offers an exploratory perspective in the decision space which can choose a search direction jumping out of the local optima basin. The exploration is performed an optimal k -insertion of a critical operation. We use a rule $T_m = \alpha T_{m-1}$, where $\alpha(\alpha \in (0,1))$ is the cooling coefficient, where T is the temperature. The number of iteration for each temperature T is $M(T) = 1/T$. The basic scheme is presented in Algorithm 2.

D. *Description of our MA*

Initially, the MA randomly generates a population of individuals. Then the MA starts evolving the population generation by generation. In each generation, the MA uses the genetic operators probabilistically on the individuals in the population to create new promising search points. Individuals will then undergone the local search learning procedure in the spirit of Lamarckian learning. This form of learning forces the genotype to reflect the result of improvement through the placement of locally improved individual back into the population in order to compete for reproductive opportunities. The basic scheme is presented in Algorithm 3.

Algorithm 3: our MA

Begin
$t := 0$
Initialize population $P(t)$ of size $popsize$ with two-vector representation
 	
Evaluate all individuals in $P(t)$
Copy the elite individual into $P(t+1)$
While $\lvert P(t+1) \rvert < popsize$
Select a pair of parents using roulette wheel selection
Apply crossover to produce two children $C1$ and $C2$ with the crossover probability p_C
Apply mutation to $C1$ and $C2$ with the mutation probability p_M
End while
Apply tabu search to every individual in $P(t+1)$ with the probability p_{TS}
If the elite individual is unimproved for more than 20 generations
Apply *simulated annealing* to each of the individual in $P(t+1)$ with the probability p_{SA}
End if
$t := t+1$
End

IV. Experimental Study

A. Test problems and parameter settings

The MA was implemented on a 1.66GHz Core 2 personal computer and tested on a large number of problem instances from the literature.

(1) The first data set (BRdata) comes from Brandimarte [4]. The data were randomly generated using a uniform distribution between given limits.

(2) The second data set (DPdata) comes from Dauzére-Pérés and Paulli [6]. The set of machines capable of performing an operation was constructed by letting a machine be in that set with a probability that ranges from 0.1 to 0.5.

(3) The third data set (BCdata) comes from Barnes and Chambers [12]. The data were constructed from three of the most challenging classical job shop problem (mt10, la24, la40) by replicating machines selected according to two simple criteria: the total processing time required by a machine and the cardinality of critical operations on a machine. The processing times for operations on replicated machines are assumed to be identical to the original.

In our experiment, parameters are set as follows:

TABLE I
THE PARAMETERS OF OUR EXPERIMENTS

Population size		100
Crossover Probability		$p_C = 0.9$
Mutation Probability		$p_M = 0.05$
Tabu Search Probability		$p_{LS1} = 0.05$
Simulated Annealing Probability		$p_{LS1} = 0.05$
Tabu Search Stop Condition		N
Initial temperature		500
Final temperature		0.1
Cooling coefficient		0.8
Simulated Annealing Stop Condition		$T < T_F$
Fitness Evaluation	BRdata	500,000
	DPdata BCdata	2,000,000

B. Computational results

In Table II, we compare our MA with the algorithms proposed by Mastrolilli and Gambardella [14], Gao et al. [15] on BRdata. The first column reports the instance name; the second and third columns report the number of jobs and the number of machines for each instance, respectively. The fourth column reports the best-known lower bound and upper bound. Flex. denotes the average number of equivalent machines per operation. The fifth and sixth column reports our best makespan and average makespan over five runs of MA. The makespan marked with an asterisk is the best upper bound found to date. The remaining columns report the best results of the two algorithms we compare with. Table III and Table IV give results on DPdata and BCdata, respectively.

A comparative overview of the MA's best makespan is given in Table V. Column 4 (B:E:W) represents the number of instances for which MA's average makespan is better, equal or worse than those found by the procedure of column 3.

TABLE V
COMPARISON RESULTS BETWEEN OUR MA WITH M&G AND HGA

Dataset	Num.	Algorithms	B:E:W
BRdata	10	M&G	3:7:0
		hGA	0:8:2
DPdata	18	M&G	13:1:4
		hGA	8:0:10
BCdata	21	M&G	9:10:2
		hGA	7:11:3

Considering our best results, we found 4 better solutions in terms of best solutions found by M&G and hGA out of five runs in the 49 benchmark problems. The average makespan of our MA over five runs is better than that of M&G and hGA on 25 and 17 test instances respectively. For DPdata, although the hGA outperforms our MA with a relatively small advantage, our MA can find better upper bounds in some test instances compared with the hGA. So, our MA is also worthwhile for solving the DPdata problems. However, for BCdata, our MA is quite robust and outperforms the hGA in the most test instances. Furthermore, it should be noted that the CPU time of the hGA is much longer than that of M&G, our MA limited computational budget at a quite reasonable level. Since the number of iterations of experiments setup by M&G is limited 10^5 for BRdata and 4×10^5 for DPdata and BRdata, we limit our computational budget at the same level (each iteration evaluate only once). So our MA is as fast as M&G but can get better performance.

Since we introduced the tabu search to efficiently explore and exploit the decision space and the simulated annealing to prevent an undesired premature convergence, we have made some experiments to compare the performance among the original tabu search proposed by Mastrolilli and Gambardella (2000), the standard genetic algorithm (GA) which only hybrid the tabu search, and our MA. The parameters are set in table. 1. Firstly, we picked out 12 problems and classify them into three categories——low flexibility, medium flexibility and high flexibility to represent all test problems, and then test the three algorithms on them over 30 independent runs. Table VI describes the flexibility of the test problems in detail. The statistical results are shown in boxplots in Fig. 1.

It can be found that for 16a, 06a, Mk07, Mk10, 12a, 15a test instances, our MA gets the best performance, and for 01a, 16a, 06a, only our MA can obtain the best solution. For 01a and 11a, our MA has nearly the same performance with GA+TS. However, for 07a and 18a, TS outperforms GA+TS and our MA, and for 09a and 13a, GA+TS gets the best performance. For 07a, TS and GA+TS obtain the best solution and for 12a, GA+TS obtains the best solution.

TABLE VI
THE FLEXIBILITY OF SOME REPRESENTATIVE TEST PROBLEMS.

Flex.	Test Problems	Category
1.0~2.0	01a 07a 13a 16a	Low Flexibility
2.0~4.0	06a 11a Mk07 Mk10	Medium Flexibility
4.0~6.0	09a 12a 15a 18a	High Flexibility

In summary, our MA which hybrid the SA local searcher outperforms GA+TS and TS. We can draw the conclusion that the SA local searcher can help the evolutionary algorithm prevent getting stuck at local basins and explore the search space efficiently.

V. CONCLUSION

In this paper, we studied the flexible job-shop scheduling problem. And then, a memetic algorithm is introduced which adopt a tabu search to explore and exploit the decision space efficiently and a simulated annealing strategy to prevent an undesired premature convergence. Finally, the MA was tested on 49 benchmark problems, and compared with other two well-known algorithms. Considering both the computational efforts and experimental results, our MA is a quite efficient and robust algorithm for solving flexible job-shop problems.

REFERENCES

[1] M. R. Garey, D. S. Johnson, and R. Sethi, “The complexity of flow shop and job-shop scheduling,” Math. Oper. Res., vol. 1, pp. 117–129, 1976.

[2] S. S. Panwalkar and W. Iskander, “A survey of scheduling rules,” Oper. Res., vol. 26, pp. 35-62, 1988.

[3] N. M. Najid, S. Dauzѐrѐ-Pérѐs, and A. Zaidat, “A modified simulated annealing method for flexible job shop scheduling problem,” IEEE Int. Conf. Syst., Man Cyber., vol. 5, no. 6, pp. 1–6, 2002

[4] P. Brandimarte, “Routing and scheduling in flexible job shops by tabu search,” Ann. Oper. Res., vol. 41, pp. 157–183, 1993.

[5] E. Hurink, B. Jurisch, and M. Thole, “Tabu search for the job shop scheduling problem with multi-purpose machines,” Oper. Res. Spektr., vol. 15, pp. 205–215, 1994.

[6] S. Dauzѐrѐ-Pérѐs and J. Paulli, “An integrated approach for modelling and solving the general multiprocessor job-shop scheduling problem using tabu search,” Ann. Oper. Res., vol. 70, pp. 281–306, 1997.

[7] Jun Qing li, Quan ke Pan, and P .N. Suganthan, “A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem,” Int J Adv Manuf Technol., vol. 52, pp. 683-697, 2011.

[8] H. Z. Jia, A. Y. C. Nee, J. Y. H. Fuh, and Y. F. Zhang, “A modified genetic algorithm for distributed scheduling problems,” J. Int. Man., vol. 14, pp. 351–62, Jun. 2003.

[9] H Zhang and M Gen, “Multistage-based genetic algorithm for flexible job shop scheduling problem,” J. Com. Int., vol. 48, pp. 409–425, 2005

[10] F. Pezzella G. Morganti, and G. Ciaschetti, “A genetic algorithm for the flexible job-shop scheduling problem,” Comput. Oper. Res., vol. 35, pp. 3202–3212, 2008.

[11] J. Paulli, “A hierarchical approach for the FMS scheduling problem,” Eur. J. Oper. Res. Vol. 86, pp. 32–42, 1995

[12] J. W. Barnes and J. B. Chambers, “Flexible Job Shop Scheduling by tabu search,” Graduate program in operations research and industrial engineering. Technical Report ORP 9609, University of Texas, Austin; 1996.

[13] M. Mastrolilli and L. M. Gambardella, “Effective neighborhood functions for the flexible job shop problem,” J. Sched., vol. 3, pp. 3–20, 2000.

[14] J. Gao, L. Sun, and M. Gen, “A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems,” Comput. Oper. Res., vol. 35, pp. 2892–2907, 2008

[15] M. Yazdani, M. Amiri, and M. Zandieh,”Flexible job-shop scheduling with parallel variable neighborhood search algorithm,” Expert. Syst. Appl., vol. 37, pp. 678–687, 2010.

[16] A. Nasr, T. Y. ElMekkawy, “An efficient hybridized genetic algorithm architecture for the flexible job shop scheduling problem,” Flex. Serv. Manuf. J., vol. 23, pp. 64-85, Mar. 2011.

[17] N. B. Ho, J. C. Tay, and E. M. K. Lai, “An effective architecture for learning and evolving flexible job-shop schedules,” Eur. J. Oper. Res., vol 179, pp. 316–333, 2007.

[18] X. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation", IEEE Trans. Evol. Comput., vol. 15, no. 5, pp. 591–607, Oct 2011.

[19] M. Gen, Y. Tsujimura, and E. Kubota, “Solving job-shop scheduling problem using genetic algorithms,” In: Proc. Int. Conf. Computer. Ind. Eng., Ashikaga, Japan; pp. 576–9, 1994.

[20] Cormen, T. H., Leiserson C. E., Rivest, R. L., & Stein, C. “Introduction to algorithms”,. Possíveis Questionamentos, 2009

[21] R. Cheng, M. Gen, and Y. Tsujimura, “A tutorial survey of job-shop scheduling problems using genetic algorithms: Part 2: Hybrid genetic search strategies,” Computer. Ind. Eng., vol. 37, pp. 51–55, Oct. 1999.

TABLE II
RESULTS ON BRDATA

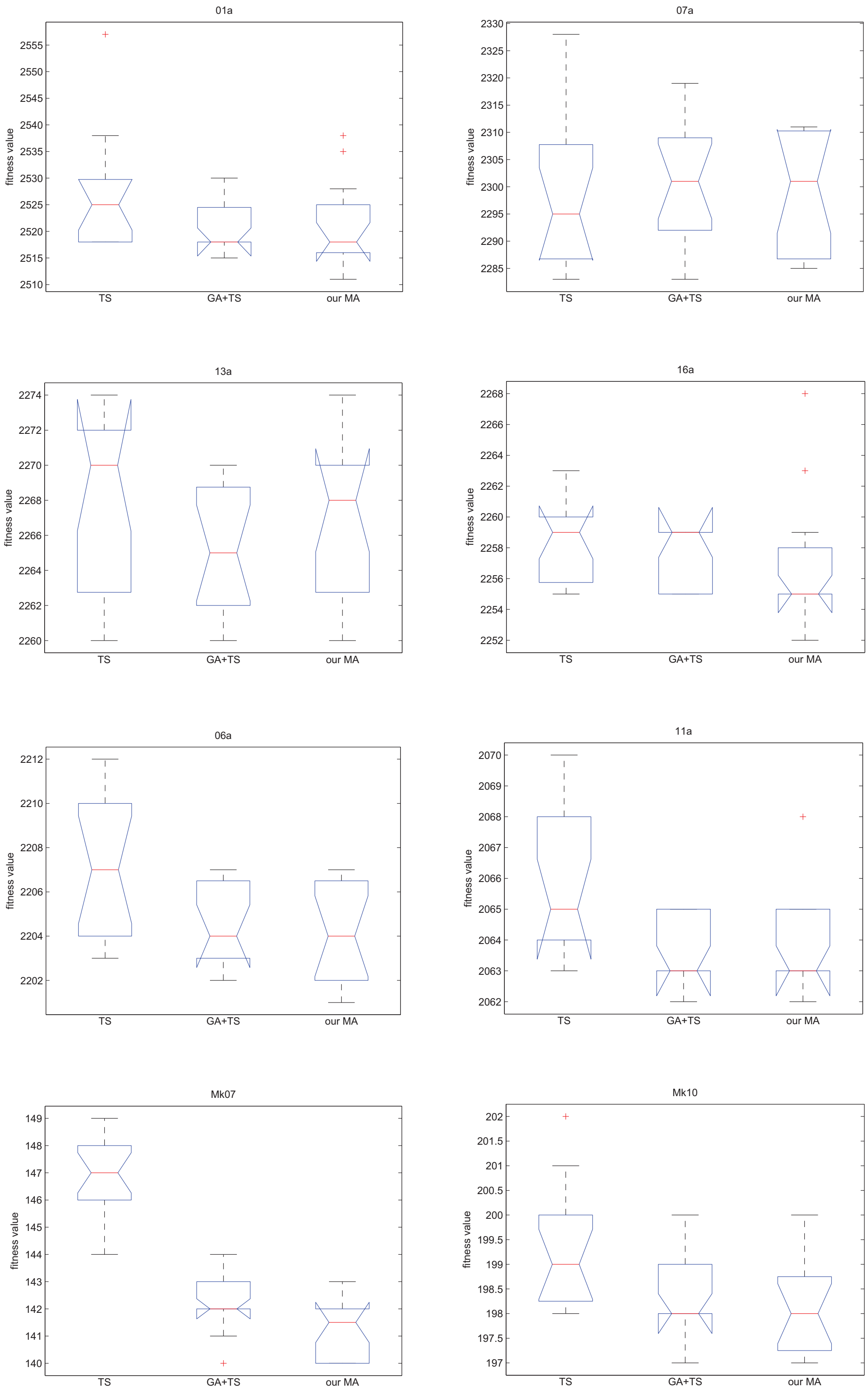
Problems	n×m	Flex.	LB,UB	MA		M&G		hGA	
				Best	Mean	Best	Mean	Best	Mean
MK01	10×6	2.09	36,42	40*	40	40*	40*	40	40*
MK02	10×6	4.01	24,32	26*	26	26*	26*	26	26*
MK03	15×8	3.01	204,211	204*	204	204*	204*	204	204*
MK04	15×8	1.91	48,81	60*	60	60*	60*	60	60*
MK05	15×4	1.71	168,186	172*	172	172*	172	172*	172
MK06	10×15	3.27	33,86	58*	58	58*	58.4	58*	58
MK07	20×5	2.83	133,157	140	141.2	144	147	139*	139
MK08	20×10	1.43	523	523*	523	523*	523	523*	523
MK09	20×10	2.53	299,369	307*	307	307*	307	307*	307
MK10	20×15	2.98	165,296	197*	198	198	199.2	197*	197

TABLE III
RESULTS ON DPDATA

Problems	n×m	Flex.	LB,UB	MA		M&G		hGA	
				Best	Mean	Best	Mean	Best	Mean
01a	10×5	1.13	2505,2530	2511*	2515.6	2518	2528	2518	2518
02a	10×5	1.69	2228,2244	2231*	2233	2231*	2234	2231*	2231
03a	10×5	2.56	2228,2235	2229*	2229	2229*	2229.6	2229*	2229.3
04a	10×5	1.13	2503,2565	2503*	2507.2	2503*	2516.2	2515	2518
05a	10×5	1.69	2189,2229	2218	2219	2216*	2220	2217	2218
06a	10×5	2.56	2162,2216	2201	2203.4	2203	2206.4	2196*	2198
07a	15×8	1.24	2187,2408	2285	2300	2283*	2297.6	2307	2309.8
08a	15×8	2.42	2061,2093	2068*	2070.4	2069	2071.4	2073	2076
09a	15×8	4.03	2061,2074	2066*	2068.2	2066*	2067.4	2066*	2067
10a	15×8	1.24	2178,2362	2293	2301.4	2291*	2305.6	2315	2315.2
11a	15×8	2.42	2017,2078	2062*	2063.8	2063	2065.6	2071	2072
12a	15×8	4.03	1969,2047	2031	2034	2034	2038	2030*	2030.6
13a	20×10	1.34	2161,2302	2260	2265.4	2260	2266.2	2257*	2260
14a	20×10	2.99	2161,2183	2168	2168	2167*	2168	2167*	2167.6
15a	20×10	5.02	2161,2171	2166	2167	2167	2167.2	2165*	2165.4
16a	20×10	1.34	2148,2301	2252*	2254.8	2255	2258.8	2256	2258
17a	20×10	2.99	2088,2168	2141	2143.4	2141	2144	2140*	2142
18a	20×10	5.02	2057,2139	2137	2140.4	2137	2140.2	2127*	2130.7

TABLE IV
RESULTS ON BCDATA

Problems	n×m	Flex.	LB,UB	MA		M&G		hGA	
				Best	Mean	Best	Mean	Best	Mean
mt10c1	10×11	1.10	655,927	927*	927	928	928	927*	927.2
mt10cc	10×12	1.20	655,914	910*	910	910*	910	910*	910
mt10x	10×11	1.10	655,929	918*	918	918*	918	918*	918
mt10xx	10×12	1.20	655,929	918*	918	918*	918	918*	918
mt10xxx	10×13	1.30	655,936	918*	918	918*	918	918*	918
mt10xy	10×12	1.20	655,913	906	906	906	906	905*	905
mt10xyz	10×13	1.30	655,849	847*	850.0	847*	850.0	849	849
setb4c9	15×11	1.10	857,924	914*	914	919	919.2	914*	914
setb4cc	15×12	1.20	857,909	909*	909	909*	911.6	914	914
setb4x	15×11	1.10	846,937	925*	925	925*	925	925*	931
setb4xx	15×12	1.20	847,930	925*	925	925*	926.4	925*	925
setb4xxx	15×13	1.30	846,925	925*	925	925*	925	925*	925
setb4xy	15×12	1.20	845,924	916*	916	916*	916	916*	916
setb4xyz	15×13	1.30	838,914	905*	905	905*	908.2	905*	905
seti5c12	15×16	1.07	1027,1185	1174*	1174.2	1174*	1174.2	1175	1175
seti5cc	15×17	1.13	955,1136	1136*	1136	1136*	1136.4	1138	1138
seti5x	15×16	1.07	955,1218	1204	1204	1201*	1203.6	1204	1204
seti5xx	15×17	1.13	955,1204	1199*	1200	1199*	1200.6	1202	1203
seti5xxx	15×18	1.20	955,1213	1199	1200.2	1197*	1198.4	1204	1204
seti5xy	15×17	1.13	955,1148	1136*	1136.2	1136*	1136.4	1136*	1136.5
seti5xyz	15×18	1.20	955,1127	1125*	1126.6	1125*	1126.6	1126	1126


01a
07a
13a
16a
06a
11a
Mk07
Mk10
fitness value
TS
GA+TS
our MA

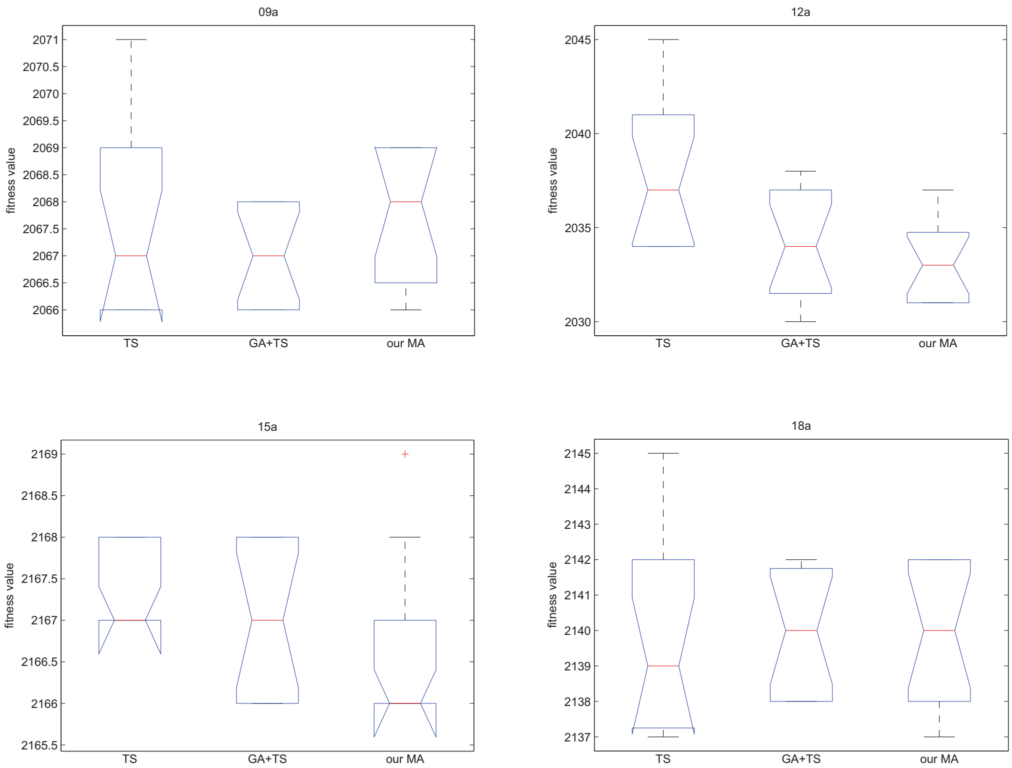


Fig. 1 Statistical results of TS, GA+TS and our MA on 12 typical test problems.