Multiobjective Job Shop Scheduling using Memetic Algorithm and Shifting Bottleneck Procedure

Hsueh-Chien Cheng, Tsung-Che Chiang, and Li-Chen Fu

***Abstract*—In this work, the multiobjective job shop scheduling problem is addressed. The objectives under consideration are minimization of makespan and total tardiness. An integration of dispathing rules, Shifting Bottleneck procedure, and multiobjective memetic algorithm is proposed. The proposed approach significantly outperforms pure dispatching rule and rule-encoded memetic algorithm. In comparison with an existing benchmark approach on eight classical job shop problem instances, the proposed approach reports promising results and updates a large portion of the best known Pareto optimal solutions.**

I. Introduction

SCHEDULING in manufacturing systems is an important task for the industries faced with a large amount of resource competitions. Effective scheduling can improve the overall system performance and customer satisfaction.

A. Jobshop Scheduling Problem

One of the most intensively studied scheduling problems is the job shop scheduling problem (JSSP). JSSP is generally NP-complete, which means that nowadays there is still no polynomial-time algorithm being able to solve the problem optimally.

The classical JSSP is described briefly as follows. We have m machines and n jobs. Each job consists of a set of operations that must be processed by each machine exactly once following a predefined route. Processing time of each operation is fixed and known in advance. All machines have unit capacity, and no machine breakdown and setup time are considered. The transportation time is neglected. The release time of each job is zero.

For a job j, its completion time is denoted by C_j. Makespan is the maximum completion time among all jobs, which is denoted by C_{max}. Though traditionally the objective of JSSP was to minimize C_{max}, due date related objectives started to attract attention recently. The due date of job j is denoted by D_j. Tardiness of a job j, T_j, is calculated by $max\{0, C_j - D_j\}$. Total tardiness, $\sum_j T_j$, is the sum of tardinesses.

The two objectives we considered to minimize in this work are

1) Makespan $C_{max} = max\{C_j\}$
2) Total tardiness $\sum_j T_j$

Hsueh-Chien Cheng and Li-Chen Fu are with the Department of Computer Science and Information Engineering, National Taiwan University. Tsung-Che Chiang is with the Department of Computer Science and Information Engineering, National Taiwan Normal University. (email: {hccheng, tcchiang}@ieee.org, lichen@ntu.edu.tw).

This research is sponsored by the National Science Council of Republic of China (R.O.C.) under research grant no. NSC97-3114-E-002-002.

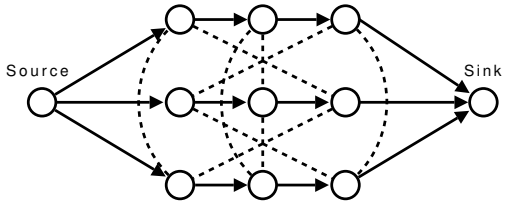


Fig. 1. Disjunctive graph representation of a JSSP with 3 jobs, each with 3 operations. The vertices connected by disjunctive arcs are being processed on the same machine.

JSSP can be represented by disjunctive graph (see Fig. 1 for an example). A disjunctive graph $D = (N, A, E)$ is defined as follows. N is the set of vertices, A is the set of conjunctive arcs, and E is the set of disjunctive arcs. Each operation corresponds to a vertex in N. Two vertices, source (0) and sink (s^*), are introduced into N for facilitating the calculation of concerned objectives. For all vertices except the source, the weights of its outgoing arcs are the processing time of its corresponding operation. The weights of outgoing arcs of source are zero. The arcs in A represent the routes of jobs. The arcs in E represent the processing order between the operations to be processed on the same machine. Construction of a schedule is achieved through the decision of the directions for all disjunctive arcs. Note that a feasible schedule is cycle-free in the disjunctive graph representation. Makespan can be calculated by the longest path from 0 to s^* after the complete schedule is generated.

Based on the disjunctive graph representation, Shifting Bottleneck (SB) procedure, first appeared in [1] and was further improved by [2] and [3], had been successively applied to JSSP. SB decomposes JSSP into m single machine subproblems. By solving these single machine subproblems in a certain sequence determined by machine criticality, a complete schedule can be built gradually. The method to solve the subproblems is refered to as the subproblem solution procedure (SSP).

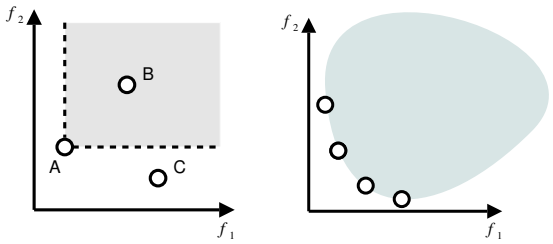
Apart from the emerging development of SB, rule-based dispatching is still regarded as the most popular approach to scheduling in the real world. This is mostly a result of the fact that rule-based dispatching is much more computationally efficient and can be implemented very easily.

B. Multiobjective Scheduling

Multiobjective scheduling has gained its importance in the last decade. For comflicting objectives, it is usually the case that the improvement of one objective is resulted from

978-1-4244-2757-4/09/$25.00 ©2009 IEEE

the degradation of other objectives. One intuitive approach to multiobjective scheduling is to define an aggregation function to combine different objectives. For example, a common aggregation function is the weighted sum function: for solution x, $f(x) = \sum w_i \cdot f_i(x)$, where f_i are objective functions. However, finding a suitable aggregation function is a challenging task for decision makers. Another approach is the Pareto-based approach, which attempts to find the set of optimal solutions based on the concept of Pareto dominance. By the use of the Pareto-based approach, decision makers are relieved from the task of defining an aggregation function. The only task for the decision maker is to select his/her favorite solution from the set of Pareto optimal solutions.



(a) (b)

Fig. 2. (a) Pareto dominance. (b) Pareto optimal solutions.

Assuming minimization of K objectives, f_i, i = 1, 2, ... K, solution x_1 is defined to dominate solution x_2 if and only if $\forall i = 1, 2, ...K, f_i(x_1) \leq f_i(x_2)$ and $\exists i = 1, 2, ..., K, f_i(x_1) < f_i(x_2)$. The Pareto dominance relation is denoted by $x_1 \prec x_2$. A solution x^* is Pareto optimal if $\nexists x \in X, x \prec x^*$ where X is the solution space. The goal of Pareto-based multiobjective optimization is to find the Pareto optimal set, which is the set consisting of Pareto optimal solutions. For example, in Fig. 2(a), A dominates B, and no dominance relation exists between A and C and between B and C. In Fig. 2(b), the four marked solutions are Pareto optimal, assuming the colored area is the search space.

Genetic Algorithm (GA), as one category of Evolutionary Algorithm, has been widely applied to a variety of multiobjective optimization problems. During the execution of GA, each individual is assigned a fitness indicating its solution quality. Individuals are selected as parents according to their fitness values. Selected parents generate offspring through crossover and mutation operator. Good individuals are kept in the population of next generation so that they can be selected to be parents again.

GA is often applied along with local search (LS) to enhance its search ability. This hybrid algorithm are referred to as the memetic algorithm (MA). In this work we address the multiobjective JSSP considering the minimization of makespan and total tardiness. A multiobjective MA is proposed to search for the Pareto optimal combinations of dispatching rules on the machines. The combined dispatching rules serve as the SSP to solve the single machine subproblems constructed by SB. Most existing approaches either use GA or SB. The proposed approach is different from those utilized GA alone in that we incorporate SB to further enhance the solution quality. In addition, most approaches related to SB considered only single objective scheduling, and the proposed approach addresses Pareto-based multiobjective scheduling problem.

The rest of this work is organized as follows. A review of literature is provided in Section II. In Section III, we present the design of the proposed approach. Experiments and results are summarized in Section IV. Finally, Conclusions and future work are highlighted in Section V.

II. Literature Review

Extensive studies on SB can be found in the literature. To determine the sequence in which the single machine subproblems are solved, various measures of machine criticality including static and dynamic ones were tested in [4]. To solve a single machine subproblem, a branch-and-bound algorithm developed in [5] has been applied widely. In [6], GA was found to be an effective alternative as the SSP.

Although SB was first intended to solve JSSP ($Jm||C_{max}$), recently researchers extended it to cope with different problems and objectives. In [7], a problem whose objective was to minimize the maximum lateness was considered. In [8], a $Jm|r_j|\sum w_j T_j$ problem was addressed. Scheduling in a complex jobshop with reentrant flow and batch machines, which is often encountered in semiconductor manufacturing scheduling, was discussed in [9]. A similar problem was addressed in [23].

For dispatching rules, a survey of conventional dispatching rules can be found in [17]. Rules Cost Over Time (COVERT) [24] and Apparent Tardiness Cost (ATC) [18] were shown to be effective to minimize total tardiness. Instead of applying a single rule, approaches to generate combined rule can be found in [25]-[27]. References [25] and [27] used a GA and [26] adopted neural network to set the weights of rules. The performance of the combined rule was found to be significantly better than that of a single rule.

Different designs of multiobjective GA have been proposed, such as NSGA-II [10], SPEA2 [11], and PESA-II [12]. The most distinguishing components between each design lie in the mechanisms they introduced to do fitness assignment and to maintain diversity during the evolution process of GA. The application of multiobjective GA to solve multiobjective scheduling problem has been found successful by many researchers. For multiobjective permutation flowshop scheduling, MOGLS [13] used a scalar function to assign fitness and to determine the search direction of LS. A MA was proposed in [14] using Pareto-ranking-based fitness assignment and multiobjective LS. Despite that a considerable number of studies are focused on multiobjective flowshop scheduling, only a few are aimed at solving multiobjective JSSP. A multiobjective GA was developed in [15] to minimize makespan and total tardiness in a JSSP. An adaptive multiobjective GA to minimize the same objectives was proposed in [16]. In [19], a multiobjective GA in search of rule combination was applied to solve JSSP

with objectives of minimizing total tardiness and maximum lateness.

Some conclusions can be drawn from the survey of existing literature. SB is a very effective approach to solve single objective JSSP; dispatching rules are computationally efficient and are suitable to serve as the SSP of SB; MA is a promising approach to deal with multiobjective optimization problems. Combining the advantages and features of the above three approaches, an integrated approach is proposed in this work.

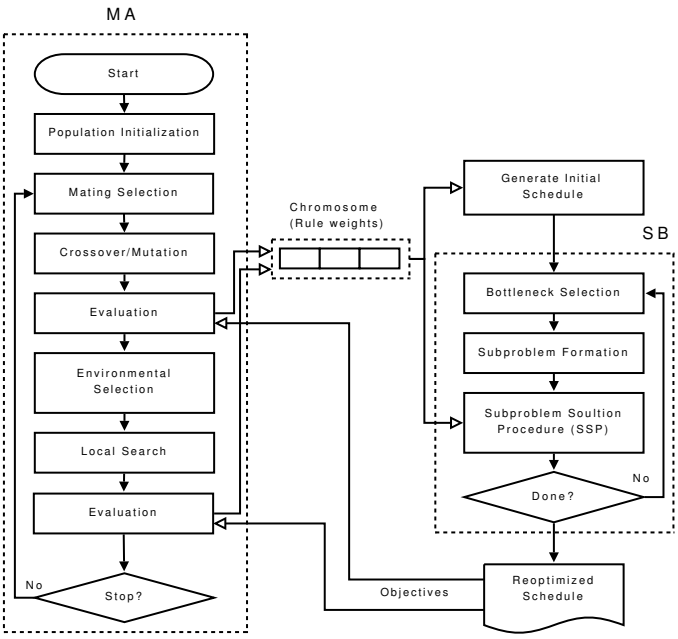


Fig. 3. The overall flow of the proposed method.

III. The Proposed Method

A. Overview

The approach proposed in this work follows the framework of MA. The search intends to find the Pareto optimal settings of SSP used in the SB. A brief description of the proposed algorithm is shown as follows. First, the MA initializes its population. After mating selection, crossover, mutation, and environmental selection, a local search is applied to one non-dominated individual in the population. The above steps repeat until the stopping criterion is satisfied. Evaluation of an individual is done by applying the SB with the combined rule recorded on the individual as the SSP. The overall flow of the MA and SB is shown in Fig. 3. Details of each component is given in the following subsections.

B. Chromosome Encoding

Each chromosome consists of $m \times R$ real numbers, where m is the number of machines and R is the number of candidate rules to be used in the SSP. Each real number stands for the weight of a certain candidate rule. Every R real numbers form a gene, which represents the weight combination of rules on a certain machine. The sum of real numbers in a gene is 1.0.

C. Population Initialization

All individuals are generated by randomly assigning each gene R real numbers that add up to 1.0.

D. Chromosome Decoding

The chromosome is decoded by the SB procedure. The SB constructs a complete schedule by iteratively solving the subproblem associated to each machine until all subproblems(machines) are solved(scheduled). In our design, the SB is invoked everytime the MA evaluates an individual. In order to shorten the computation time needed to perform the SB, only the reoptimization stage in the SB is kept. The flowchart of the SB is given in Fig. 3.

To decode an individual back to its corresponding schedule, we first generate an initial schedule by applying the candidate rules. If a machine is available, one of the operations in the waiting queue will be selected according to the aggregated priority index, which is the weighted sum of priority values obtained by candidate rules. The corresponding weights of rules are extracted from the chromosome. The generated schedule is a non-delay one. Because some dispatching rules favor the operation with larger priority value (eg. ATC), whereas others favor smaller priority value (eg. FIFO), priority values of those rules of the first type are multiplied by -1.0, and thus all rules now favor smaller values. Different dispatching rules may have different scales of priority values, so we normalize the priority values into [0, 1]. By repeatedly selecting the operation with the smallest aggregated priority index to be processed whenever a machine is available, we can build a complete schedule.

After obtaining the initial schedule, the next step is to improve its quality using the reoptimization mechanism in the SB procedure, which involves bottleneck machine selection, subproblem formation, and SSP. Details of the reoptimization mechanism of SB can be found in [1].

We use Total Machine Load (TML) as the criticality measure to identify the bottleneck machine. TML is the sum of processing time of the operations to be processed on the machine. Intuitively, a machine with larger TML would have a larger impact on the performance of the entire schedule.

After the bottleneck machine is chosen, its corresponding subproblem is formed based on the disjunctive graph. All vertices correspond to operations that are processed by the bottleneck machine are considered. Let $L(x, y)$ denote the longest path between two vertices x and y. For an operation o of job j corresponding to vertex v, its release time is calculated by $L(0, v)$. Let k denote the last operation of job j with corresponding vertex w. The processing time of operation o and k are denoted by P_o and P_k respectively. The (internal) due date of operation o is calculated by $D_j - (L(v, w) + P_k) + P_o$, where D_j is the external due date.

After the single machine subproblem is formed, we solve it in a similar way as what we mentioned to build the initial schedule. In other words, the combined rule is taken as the SSP. Then, the schedule of the single machine subproblem

will be integrated into the schedule of the entire job shop by fixing the directions of disjunctive arcs related to the operations in the subproblem.

After the SB terminates, the objective functions (makespan and total tardiness) can be calculated, and the evaluation of an individual is completed.

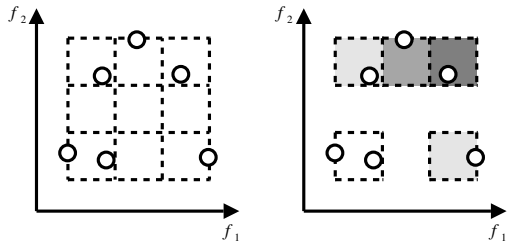


Fig. 4. An example of mating selection with $G = 3$. After non-dominated sort of the grids, the rank of $g_{1,1}$ is 1, the rank of $g_{1,3}$ and $g_{3,1}$ is 2, the rank of $g_{2,3}$ is 3, and the rank of $g_{3,3}$ is 4.

E. Mating Selection

We first apply non-dominated sorting in NSGA-II [10] to assign a rank and a crowding distance to each individual in the population. Then, the population is separated into $G \times G$ grids. (G is a parameter of our MA, and its values will be given in Section IV.) Each grid is associated with a coordinate. For example, the coordinate of the grid in the bottom left is (1, 1), and this grid is denoted by $g_{1,1}$. Only the grids with individual inside are taken into consideration in the following steps. Grid g_{x_1,y_1} is said to dominate grid g_{x_2,y_2} if and only if $(x_1 \leq x_2 \wedge y_1 \leq y_2)$ and $(x_1 < x_2 \vee y_1 < y_2)$. Another non-dominated sorting is applied to the grids. Two 2-tournament selections are performed to select two grids. The first tournament randomly chooses two non-dominated grids (or equivalently, grids with rank 1) and then picks up the one with a smaller number of individuals inside. The second tournament randomly chooses two grids and then picks up the one with a smaller rank. In case of a tie, the one with a smaller number of individuals is selected. See Fig. 4 for an example.

Inside each of the two selected grids, we apply 2-tournament selection again to select one individual. The individual with a smaller rank wins the tournament, with the crowding distance as the tie breaker. Finally, two parent individuals are selected.

F. Crossover

After a pair of parents are selected, a uniform crossover is applied with probability P_c to produce two offspring. The probability of gene exchange is 0.5. Fig. 5 shows how the operator works.

G. Mutation

The mutation operator is applied with a small probability P_m after a pair of offspring are generated. The mutation operator randomly picks up one gene at first. Among the R real numbers in the gene, a value is subtracted from one real number r_{sub}, and is added to another real number r_{add}. An example is shown in Fig. 6. The selection of r_{sub} and r_{add} is done at random. Note that the real numbers inside a gene are positive and sum up to 1.0. The value to substract/add is a random value between zero and the value of r_{sub}.

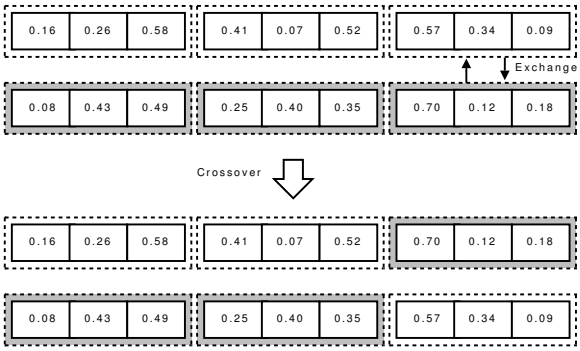


Fig. 5. Crossover operator.

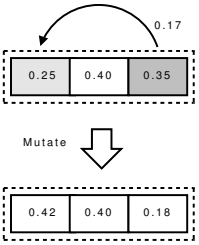


Fig. 6. Mutation operator.

H. Environmental Selection

If there exists an offspring with objectives identical to those of another individual in the population, the offspring is discarded. The non-duplicate offspring together with the population are subjected to a non-dominated sorting in order to compute their ranks and crowding distances. Among the parents and the non-duplicate offspring, two individuals with the smallest ranks and crowding distances will replace the parents in the population.

I. Local Search

After one generation is completed, a local search is invoked on one of the non-dominated individuals, S_{nd}. The local search procedure used here is similar to that in [13]. The linear weighted sum function is taken as the aggregation function, and the weights of objectives are generated at random. A 2-tournament selection is applied to select one non-dominated individual based on the aggregated scalar value. During the local search, neighbors are generated by the proposed mutation operator. After evaluation of the generated neighbor, it is accepted if the outcome of the aggregation function is not worse than 1.05 times that of the best neighbor ever found. After exploring N_{LS} neighbors, the best neighbor will replace the nearest individual it dominates in the normalized objective space if the outcome of the aggregation function is better than that of S_{nd}.

J. Stopping Criteria

The algorithm terminates after a predefined maximum generation N_G is reached.

TABLE I
PARAMETER SETTINGS

Population Size	100
Offspring Size	100
N_G	100
N_{LS}	10
P_c	1.0
P_m	0.1
Grid Size G	3, increase by 1 every 33 generations
Number of Rules R	4
Candidate Rules	First-In-First-Out
	Longest-Remaining-Processing-Time
	Apparent Tardiness Cost
	Critical Ratio
Weight Precision	Up to 0.01
Reoptimization cycles of SB	2

IV. EXPERIMENTS AND RESULTS

A. Experiment Settings

The benchmark instances are eight 10×10 instances from the existing literature. FT10 is from [21]. ABZ5 and ABZ6 are from [1]. ORB1–ORB5 are from [20]. The benchmark algorithm is proposed in [15]. For that the due date information is not provided in the original instances, due dates are set in the same manner as in [15]. Main differences between our MA and the approach in [15], CMEA, include:

- Dispatching rules are encoded and used in a machine-wise manner in our MA, whereas rules were encoded and used in an operation-wise manner in CMEA.
- We generate and use combined dispatching rules through linear weighted summation, whereas in CMEA each of the rules was used alone.
- The SB is used in the decoding step in our MA, whereas CMEA used the Giffler-Thompson (GT) algorithm as the schedule builder.

After some preliminary experiments, the parameters of the proposed method are summarized in Table. I. To determine the appropriate value of parameter k of ATC rule for each problem instance, we solve each instance by applying only ATC rule with k = 1, 1.5, 2, ..., 4.5, 5. Then, the best k value is adopted when we solve the same instance by the proposed MA. Each problem instance is solved by our MA for 20 times, and the non-dominated solutions found are collected for the purpose of performance comparison.

B. Effects of SB and MA on the Performance

The performance of the proposed method (R-SB-MA) is first compared with the pure ATC rule and the MA with rules only (R-MA). The difference between R-SB-MA and R-MA is that the number of reoptimization cycles is set to zero in the latter. Due to the limitation of space, here we only show

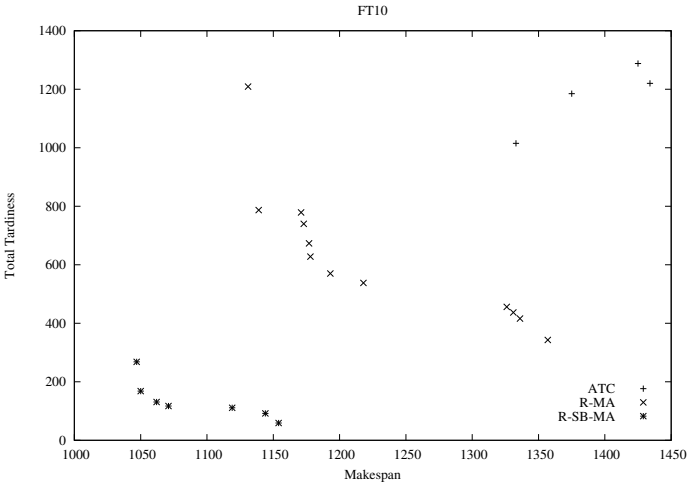


Fig. 7. Non-dominated solutions obtained for FT10 by ATC, R-MA, and R-SB-MA.

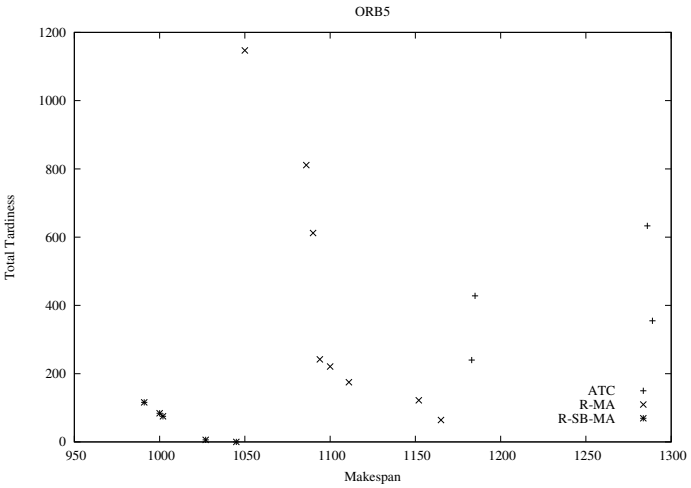


Fig. 8. Non-dominated solutions obtained for ORB5 by ATC, R-MA, and R-SB-MA.

the results of instances FT10 and ORB5 in Fig. 7 and Fig. 8, respectively. Although ATC is regarded as one of the best rules to minimize total tardiness, the combination of multiple rules enables R-MA to find solutions with much lower total tardiness and makespan simultaneously. Moreover, the integration of the SB into MA (R-SB-MA) as a schedule reoptimizer brings another significant improvement. Note that schedules generated by ATC and R-MA are non-delay ones. Through reoptimization by the SB, the reoptimized schedules may become active ones. As it is known that in some cases the optimal schedules are active but not nondelay. The experimental results confirm that R-SB-MA can provide better performance by utilizing dispatching rules and the SB to reach high quality active schedules effectively.

C. Performance Comparison

Now we compare the proposed R-SB-MA with the benchmark algorithm CMEA [15]. The non-dominated solutions obtained by R-SB-MA and CMEA are listed in Table II. For FT10 and ORB1, all non-dominated solutions found

TABLE II
NON-DOMINATED SOLUTIONS OBTAINED (SOLUTIONS MARKED * ARE FOUND BY [15])

FT10	ABZ5		ABZ6	ORB1
1047 268	1277 422*	1338 86	979 348*	1150 354.5
1050 168	1282 421	1342 85	988 155*	1183 352.5
1062 131	1283 397	1360 70	1068 91	1189 344.5
1071 117	1292 265	1363 61	1075 85	1195 313
1119 111	1308 261	1364 59	1077 83	1273 305
1144 92	1311 242	1369 50	1082 60	1296 299.5
1154 59	1313 209	1387 49	1086 32	1320 170.5
	1324 158	1392 0	1090 30	
	1336 129			

ORB2	ORB3	ORB4		ORB5
941 36*	1137 1246	1100 1185.5	1305 89	988 23*
956 31*	1146 1073	1138 314.5*	1437 78	994 18*
980 0	1150 994	1163 283.5	1439 70	1027 6
	1166 819	1168 275.5	1448 69	1045 0
	1167 367*	1170 212.5	1452 45	
	1173 364*	1185 211		
	1187 276	1188 183.5		
	1257 219	1257 152		
	1268 161	1264 123		
	1310 128	1265 120		
	1317 111	1302 90		

TABLE III
HYPERVOLUME METRIC

	CMEA	The Proposed Method
FT10	4030.86	6958.49
ABZ5	1226.46	5951.16
ABZ6	5577.09	4565.98
ORB1	6256	7817.11
ORB2	8913.43	6183.46
ORB3	6474.89	7193.93
ORB4	6812.55	7730.61
ORB5	8402.9	4641.56
avg.	5961.77	6417.79

by CMEA are dominated by those found by R-SB-MA. In four instances (ABZ5, ABZ6, ORB3, and ORB4), at least 75% non-dominated solutions are found by R-SB-MA. The non-dominated fronts obtained by R-SB-MA spread widely and evenly on the objective space. Instance ORB2 is the only instance of which CMEA finds more non-dominated solutions than R-SB-MA does. Solutions found by R-SB-MA tend to have lower total tardiness, whereas solutions found by CMEA tend to have smaller makespan. A possible reason is that we are lack of an effective dispatching rule to reduce makespan. The operation-wise encoding scheme in CMEA enables it the possibility to generate schedules with lower makespan than the machine-wise encoding in R-SB-MA can do.

In the following, we compare the two approaches more formally by the metric in [22], which measures the hypervolume that is dominated by the obtained set of non-dominated solutions. This metric takes into account not only the proximity of the non-dominated solutions but also the distribution of the solutions. A larger value of hypervolume metric indicates a better performance. Since our problem is a minimization problem, the reference point is chosen to be the point with both objectives as the maximum values among all non-dominated solutions found by R-SB-MA over 20 runs and the non-dominated solutions found by CMEA. The objectives are normalized into [0, 100] to remove the effect of scale difference. The result is summarized in Table III, the proposed R-SB-MA has an increased average hypervolume by 7% compared to that of CMEA.

V. CONCLUSIONS AND FUTURE WORK

In this work, a multiobjective job shop scheduling problem is addressed. The goal is to find a set of Pareto optimal schedules in terms of two objective functions - makespan and total tardiness. Although many multiobjective evolutionary algorithms have been proposed and showed successful applications on many multiobjective optimization problems, multiobjective scheduling still in need of the incorporation of domain-specific knowledge to enhance the solution quality. The SB procedure and dispatching rules are two popular approaches for solving job shop scheduling problems, but most of the relevant works considered only a single objective. The main contribution of this work is a successful integration of dispatching rules, the SB procedure, and a multiobjective MA. The SB procedure serves as a schedule reoptimizer in the evaluation stage of MA, and dispatching rules play the role of an efficient subproblem solution procedure in the SB. The proposed algorithm R-SB-MA demonstrates very good performance when comparing with the benchmark algorithm on eight widely used benchmark instances. The update of best known non-dominated solutions for these instances, which facilitates performance assessment of future approaches for multiobjective job shop scheduling, is another contribution of this work. In our future work, we will extend the algorithm to solve scheduling problems in flexible jobshop and complex jobshop like wafer fabrication facilities. Different SSP such as GA-based one will also be developed and examined.

REFERENCES

[1] Adams, J., Balas, E. and Zawack, D., "The shifting bottleneck procedure for job-shop scheduling," *Management Science*, vol. 34, no. 3, pp. 391–401, 1988.

[2] Dauzere-Peres, S. and Lasserre, J. B., "A Modified Shifting Bottleneck Procedure for Job Shop Scheduling," *International Journal of Production Research*, vol. 31, no. 4, pp. 923–932, 1993.

[3] Balas, E., Lenstra, J. K. and Vazacopoulos, A., "The one machine scheduling with delayed precedence constraints," *Management Science*, vol. 41, no. 1, pp. 94–109, 1995.

[4] Aytug, H., Kempf, K. and Uzsoy, R., "Measures of subproblem criticality in decomposition algorithms for shop scheduling," *International Journal of Production Research*, vol. 41, no. 5, pp. 865–882, 2003.

[5] Carlier, J., "The one-machine sequencing problem," *European Journal of Operational Research*, vol. 11, no. 1, pp. 42–47, 1982.

[6] Mönch, L., Schabacker, R., Pabst, D. and Fowler, J. W., "Genetic algorithm-based subproblem solution procedures for a modified shifting bottleneck heuristic for complex job shops," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2100–2118, 2007.

[7] Demirkol, E., Mehta, S. and Uzsoy, R., "A Computational Study of Shifting Bottleneck Procedures for Shop Scheduling Problems," *Journal of Heuristics*, vol. 3, no. 2, pp. 111–137, 1997.

[8] Pinedo, M. and Singer, M., "A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop," *Naval Research Logistics*, vol. 46, no. 1, pp. 1–17, 1999.

[9] Mason, S. J., Fowler, J. W. and Carlyle, W. M., "A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops," *Journal of Scheduling*, vol. 5, pp. 247–262, 2002.

[10] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[11] Zitzler, E., Laumanns, M. and Thiele, L., "SPEA2: Improving the strength Pareto evolutionary algorithm," Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

[12] Corne, D. W., Jerram, N. R., Knowles, J. D. and Oates, M.J., "PESA-II: Region-based selection in evolutionary multiobjective optimization," *Proc. of the Genetic and Evolutionary Computation Conference*, pp. 283–290, 2001.

[13] Ishibuchi, H., Yoshida, T. and Murata, T., "Balance Between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.

[14] Arroyo, J. E. C. and Armentano, V. A., "Genetic local search for multi-objective flowshop scheduling problems," *European Journal of Operational Research*, vol. 167, no. 3, pp. 717–738, 2005.

[15] Lei, D. and Wu, Z., "Crowding-measure-based multiobjective evolutionary algorithm for job shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 30, no. 1, pp. 112–117, 2006.

[16] Chang, P. C., Hsieh, J. C. and Wang, C. Y., "Adaptive multi-objective genetic algorithms for scheduling of drilling operation in printed circuit board industry," *Applied Soft Computing*, vol. 7, no. 3, pp. 800–806, 2007.

[17] Blackstone, J. H., Phillips, D. T. and Hogg, G. L., "A state-of-the-art survey of dispatching rules for manufacturing job shop operations," *International Journal of Production Research*, vol. 20, no. 1, pp. 27–45, 1982.

[18] Vepsalainen, A. P. J. and Morton, T. E., "Priority rules for job shops with weighted tardiness costs," *Management Science*, vol. 33, no. 8, pp.1035–1047, 1987.

[19] Chiang, T. C. and Fu, L. C., "Multiobjective Job Shop Scheduling using Genetic Algorithm with Cyclic Fitness Assignment," *Proc. of IEEE Congress on Evolutionary Computation*, pp. 3266–3273, 2006.

[20] Applegate, D. and Cook, W., "A computational study of the job shop scheduling problem," *ORSA Journal on Computing*, vol. 3, no. 2, pp. 149–156, 1991.

[21] Fisher, H. and Thompson, G. L., "Probabilistic learning combinations of local job-shop scheduling rules," *Industrial Scheduling*, Prentice Hall, pp. 225–251, 1963.

[22] Zitzler, E. and Thiele, L., "Multiobjective Optimization Using Evolutionary Algorithms – A Comparative Study," *Parallel Problem Solving from Nature V*, Springer, pp. 292–301, 1998.

[23] Sourirajan, K. and Uzsoy, R., "Hybrid decomposition heuristics for solving large-scale scheduling problems in semiconductor wafer fabrication," *Journal of Scheduling*, vol. 10, no. 1, pp. 41–65, 2007.

[24] Carroll, D. C., "Heuristic sequencing and multiple component jobs," Unpublished Ph.D. dissertation, Massachusetts Institute of Technology.

[25] Chen, J. H., Fu, L. C., Lin, M. H. and Huang, A. C., "Petri-net and GA-based approach to modeling, scheduling, and performance evaluation for wafer fabrication," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 619–636, 2001.

[26] Min, H. S. and Yih, Y., "Selection of dispatching rules on multiple dispatching decision points in real-time scheduling of a semiconductor wafer fabrication system," *International Journal of Production Research*, vol. 41, no. 16, pp. 3921–3941, 2003.

[27] Chiang, T. C., Shen, Y. S. and Fu, L. C., "A new paradigm for rule-based scheduling in the wafer probe centre," *International Journal of Production Research*, vol. 46, no. 15, pp. 4111–4133, 2008.