MOEA/D for Flowshop Scheduling Problems

Pei Chann Chang, Shih Hsin Chen, Qingfu Zhang, Jun Lin Lin

Abstract—**Many multiobjective evolutionary algorithms are based Pareto domination, among them NSGA II and SPEA 2 are two very popular ones. MOEA/D is a very recent multiobjective evolutionary algorithm using decomposition. In this paper, we implement MOEA/D for multi-objective flowshop scheduling problems. We study the replacement strategy of neighboring solutions, the determination of the reference point, and compare different decomposition methods. Experimental results demonstrate that MOEA/D outperforms NSGA II and SPEA 2 significantly for the 2-objective and 3-objective benchmark flowshop-scheduling instances.**

I. Introduction

The multiobjective optimization problem (MOP) can be stated as follows:

$$\text{Minimize } F(x) = (f_1(x),\ldots,f_m(x))^T$$
$$\text{Subject to } x \in \Omega \qquad (1)$$

In many real-life applications, no single solution can optimize all the objectives at the same time. Pareto optimal solutions, which characterizes the best trade-offs among the objectives, are of practical interest to a decision maker. Very often, there are many or even infinite Pareto optimal solutions. The set of all the optimal Pareto solutions in the objective space is called the Pareto front (PF). Multiobjective evolutionary algorithms (MOEA) attempt to find a good approximation to the PF.

Most MOEAs are based on Pareto domination to measure the qualities of the solutions generated during the search for selection, which are to find a manageable number of Pareto optimal vectors which are evenly distributed along the PF, and thus good representatives of the entire PF [2]. Among them NSGA-II [3] and SPEA2 [4] are very popular ones. It is well-known that under mild conditions, a Pareto optimal solution could be an optimal solution of a scalar optimization problem in which the objective is an aggregation of all the f_i's. This is a basic strategy behind many traditional mathematical programming methods for approximating the PF. A small number of MOEAs adopt this strategy [5-10]. MOEA/D [9-10] is a very recent one. MOEA/D has been successfully applied for continuous MOPs and knapsack problems.

In this paper, we propose an implementation of MOEA/D for multi-objective flowshop problems. We discuss the choice of decomposition methods in MOEA/D, the setting of reference point and the way of updating neighboring solutions. The MOEA/D is experimentally compared with NSGA-II and SPEA2 on multiobjective flowshop problems.

The rest of the paper is organized as follows. Section 2 introduces the decomposition methods used in this research. Section 3 presents the general framework of MOEA/D. Several issues on MOEA/D are discussed in Section 4. The experimental results are presented in Section 5. Finally, the conclusion is drawn in Section 6.

II. Decomposition of Multiobjective Optimization

There are several approaches for converting the problem of approximation of the PF into a number of scalar optimization problems and they can be found in the literature (e.g., [1]). The most popular ones among them include the weighted sum approach and Tchebycheff approach which are introduced in the following:

A. Weighted Sum Approach [1]

This approach considers a convex combination of the different objectives. Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T$ be a weight vector, i.e., $\lambda_i \geq 0$ for all $i = 1,\ldots,m$ and $\sum_{i=1}^{m} \lambda_i = 1$. Then, the optimal solution to the following scalar optimization problem:

$$\text{Minimize } g^{ws}(x \mid \lambda) = \sum_{i=1}^{m} \lambda_i f_i$$
$$\text{Subject to } x \in \Omega \qquad (2)$$

is a Pareto optimal point to (1), where we use $g^{ws}(x \mid \lambda)$ to emphasize that λ is a coefficient vector in this objective function, while x is the variables to be optimized. To generate a set of different Pareto optimal vectors, one can use different weight vectors x in the above scalar optimization problem. If PF is convex, this approach would work well. However, not every Pareto optimal vector can be obtained by this approach in the case of nonconvex PFs [1]. To overcome these shortcomings, Tchebysheff approach is suggested.

Manuscript received February 26, 2008.

Pei Chann Chang is with Department of Information Management, Yuan-Ze University. 135 Yuan-Dong Rd., Taoyuan 32026, Taiwan, R.O.C (+886 03 4638800#2305 e-mail: iepchang@saturn.yzu.edu.tw).

Shih Hsin Chen is with Department of Industrial Engineering and Management, Yuan-Ze University. (e-mail: s939506@mail.yau.edu.tw).

Qingfu Zhang is with the Department of Computing and Electronics System, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, U.K. (e-mail: qzhang@essex.ac.uk).

Jun Lin Lin is with Department of Information Management, Yuan-Ze University. 135 Yuan-Dong Rd., Taoyuan 32026, Taiwan, R.O.C (+886 03 4638800#2611 e-mail: jun@saturn.yzu.edu.tw).

978-1-4244-1823-7/08/$25.00©2008 IEEE

B. Tchebycheff Approach [1]

In this approach, the scalar optimization problem is in the form

$$\text{Minimize } g^{te}(x \mid \lambda, z^*) = \max_{1\le i\le m}(\lambda_i \mid f_i(x) - z_i^* \mid)$$
$$\text{Subject to } x \in \Omega \qquad (3)$$

where $z^* = (z_1^*, \ldots, z_m^*)^T$ is the reference point, i.e., $z^* = \min_{1\le i\le m}(f_i(x) \mid x \in \Omega)$ for each $i = 1, \ldots, m$. For each x^* Pareto optimal point there exists a weight vector λ such that x^* is the optimal solution of (3) and each optimal solution of (3) is a Pareto optimal solution of (1). Therefore, one is able to obtain different Pareto optimal solutions by altering the weight vector. One weakness with this approach is that its aggregation function is not smooth for a continuous MOP. However, since this work aims to solve scheduling problems which is a type of discrete problem, it still can be used in the EA framework in this paper.

III. INTRODUCTION OF MOEA/D

Multiobjective evolutionary algorithm based on decomposition (MOEA/D) needs to decompose the MOP under consideration. Any decomposition approaches can serve this purpose. In the following description, we suppose that the Tchebycheff approach is employed. It is very trivial to modify the following MOEA/D when other decomposition methods are used.

Let $\lambda^1, \lambda^2, \ldots, \lambda^n$ be a set of even spread weight vectors and z^* be the reference point. As shown in Section I, the problem of approximation of the PF of (1) can be decomposed into scalar optimization subproblems by using the Tchebycheff approach and the objective function of the subproblem is

$$\text{Minimize } g^{te}(x \mid \lambda^j, z^*) = \max_{1\le i\le m}\{\lambda_i^j \mid f_i(x) - z_i^* \mid\} \qquad (4)$$

where $\lambda^j = (\lambda^1, \lambda^2, \ldots, \lambda^n)^T$. MOEA/D minimizes all these objective functions simultaneously in a single run.

Note that g^{te} is continuous of $g^{te}(x \mid \lambda^i, z^*)$, the optimal solution of $g^{te}(x \mid \lambda^j, z^*)$ should be close to that of if λ^i and λ^j are close to each other. Therefore, any information about these g^{te}'s with weight vectors close to should be helpful for optimizing $g^{te}(x \mid \lambda^i, z^*)$. This is a major motivation behind MOEA/D.

In MOEA/D, a neighborhood of weight vector λ^i is defined as a set of its several closest weight vectors in $\{\lambda^1, \lambda^2, \ldots, \lambda^n\}$.

The neighborhood of the *i*th subproblem consists of all the subproblems with the weight vectors from the neighborhood of λ^i.The population is composed of the best solution found so far for each subproblem. Only the current solutions to its neighboring subproblems are exploited for optimizing a subproblem in MOEA/D.

At each generation, MOEA/D with the Tchebycheff approach maintains:

1. A population of *n* points $x^1, \ldots, x^n \in \Omega$, where x^i is the current solution of the *i*th subproblem.
2. $FV^1, \ldots, \mathrm{FV}^n$, where FV^i is the F-value of x^i, i.e., $FV^i = F(x^i)$ for each $i = 1, \ldots, n$;
3. $z = (z_1, \ldots, z_n)^T$, where z_i is the best value found so far for objective f_i;
4. An external population (EP), which is used to store nondominated solutions found during the search.

Consequently, the general framework of MOEA/D can be stated as follows:

Input:
- MOP (1);
- a stopping criterion;
- *n*: the number of the subproblems considered in MOEA/D;
- A uniform spread of weight vectors: $\lambda^1, \lambda^2, \ldots, \lambda^n$;
- *T*: the number of the weight vectors in the neighborhood of each weight vector.

Output: EP.

Step 1) Initialization:

Step 1.1) Set EP = ϕ.

Step 1.2) Compute the Euclidean distances between any two weight vectors and then work out the *T* closest weight vectors to each weight vector. For each $i = 1, \ldots, n$; set $\mathrm{B(i)} = \{\lambda_1, \ldots, \lambda_T\}$, where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the closest weight vectors to λ^i.

Step 1.3) Generate an initial population $x^1, \ldots, x^n$ randomly or by a problem-specific method. Set $FV^i = F(x^i)$.

Step 1.4) Evaluate $z = (z_1, \ldots, z_n)^T$ by a problem-specific method.

Step 2) Update:

For $i = 1, \ldots, n$, do

Step 2.1) Reproduction: Randomly select two indexes *k*, *l* from *B*(*i*), and then generate a new solution *y* from x^k and x^l by using genetic operators.

Step 2.2) Improvement: Apply a problem-specific repair/improvement heuristic on y to produce y'.

Step 2.3) Update of z: For each $j = 1, \ldots, m$, if $z_j < f_j(y')$, then set $z_j = f_j(y')$

Step 2.4) Update of Neighboring Solutions: For each index $j \in B(i)$, if $g^{te}(y' \mid \lambda^j, z) \le g^{te}(x^j \mid \lambda^j, z)$, then set $x_j = y'$ and $FV^j = f_j(y')$.

Step 2.5) Update of EP:

Remove from EP all the vectors dominated by $F(y')$.

Add $F(y')$ to EP if no vectors in EP dominate $F(y')$.

Step 3) Stopping Criteria: If stopping criteria is satisfied, then stop and output EP. Otherwise, go to **Step 2**.

In initialization, B(i) contains the indexes of the T closest vectors of λ^i. We use the Euclidean distance to measure the closeness between any two weight vectors. Therefore, λ^i's closest vector is itself, and then $i \in B(i)$. If $j \in B(i)$, the subproblem can be regarded as a neighbor of the ith subproblem.

In the ith pass of the loop in Step 2, the T neighboring subproblems of the ith subproblem are considered. Since x^k and x^l in Step 2.1 are the current best solutions to neighbors of the ith subproblem, their offspring y should hopefully be a good solution to the ith subproblem. In Step 2.2, a problem-specific heuristic is used to repair/improve y in the case when y invalidates any constraints, and/or optimize the ith g^{te}. Therefore, the resultant solution y' is feasible and very likely to have a lower function value for the neighbors of ith subproblem. Step 2.4 considers all the neighbors of the ith subproblem, it replaces x^j with y' if y' performs better than x^j with regard to the jth subproblem. FV^j is needed in computing the value of $g^{te}(x^j \mid \lambda^j, z)$ in Step 2.4.

Since it is often very time-consuming to find the exact reference point z^*, we use z, which is initialized in Step 1.4 by a problem-specific method and updated in Step 2.3, as a substitute z^* for g^{te} in Step 2.4. The external population EP, initialized in Step 1.1, is updated by the new generated solution y' in Step 2.5. In the case when the goal in (1) is to minimize $F(x)$, the inequality in Step 2.3 should be reversed.

IV. The Flowshop Problem and Implementation of MOEA/D

We introduce the MOEA/D algorithm for dealing with the flowshp problem.

A. Problem Statement

Flowshops are useful tools in modeling manufacturing processes. A permutation fowshop is a job processing facility, which consists of several machines and several jobs to be processed on the machines. In a permutation fowshop all jobs follow the same machine or processing order. Our objectives are to find a set of compromise solutions so that the makespan and maximum tardiness are minimized.

The flowshop scheduling problem is a typical assembly line problem where m different jobs have to be processed on n different machines. All jobs are processed on all the machines in the same order. The processing times of the jobs on machines are fixed irrespective of the order in which the processing is done. The problem is characterized by a matrix $P = p(i, j), i = 1\ldots m,\ j = 1\ldots n$, of processing times. Each machine processes exactly one job at a time and each job is processed on exactly one machine at a time. The problem then is to find a sequence of jobs such that the makespan that is the completion time of the last job in the sequence on the last machine is minimized. If C_i denotes the completion time for job i, then we are trying to minimize $\max C_i$. There are many other criterions that can be considered for optimization. We refer the reader to Bagchi [11] for a detailed discussion of multi-objective scheduling using GA. For details of the flowshop and other scheduling and sequencing problems we refer the reader to Baker [12].

The flow shop scheduling can be formerly defined as follows: if $p(i, j)$ is the processing times for job i on machine j, and a job permutation $\{\pi_1, \pi_2, ..., \pi_n\}$, where there are n jobs and m machines, then the completion times $C(\pi_i, j)$ is calculated as follows:

$$C(\pi_1,1) = p(\pi_1,1)$$
$$C(\pi_i,1) = C(\pi_{i-1},1) + p(\pi_i,1) \text{ for } i = 2,...,n$$
$$C(\pi_1,1) = C(\pi_1, j-1) + p(\pi_1, j) \quad \text{for} \quad j = 2,...,m \tag{6}$$
$$C(\pi_i, j) = \max\{C(\pi_{i-1}, j), C(\pi_i, j-1)\} + p(\pi_i, j) \text{ for } i = 2,...,n;\ j = 2,...,m$$

The makespan is finally defined as

$$C_{\max}(\pi) = C(\pi_n, m). \tag{7}$$

Then, the permutation flowshop scheduling problem (PFSP) is to find a permutation π^* in the set of all permutations Π scuh that

$$C_{\max}(\pi^*) \le C_{\max}(\pi) \Pi \forall \pi \in \quad . \tag{8}$$

A more general flowshop scheduling problem can be defined by allowing the permutation of jobs to be different on each machine. However, what work has been done to show on the more general flow shop scheduling problem has tended to small improvement in solution quality over the PFSP while increasing the complexity of the problem substantially. The size of the solution space increases from $n!$ to $(n!)^m$. Other objective functions for the PFSP also received a lot of attention. For example, the mean flow-time (the time a job spends in process), or the mean tardiness (assuming some deadline for each job) are to be minimized. Other real problems from the manufacturing industries such as jobs may have non-identical release dates, there may be sequence-dependent setup times, and there may be limited buffer storage between machines and so on. These characters of the real problem will make the problem more complicated to be solved within a reasonable time frame. However, GA approaches provide a more realistic view to the problem. Since it can generate alternatives of sequences (in the evolving process each chromosome representing a feasible solution to the problem) to the decision maker, a more applicable sequence can be decided to solve the current problem with satisfactory results.

B. Implementation of MOEA/D for flowshop problem

To further improve the performance of MOEA/D, some procedures of MOEA/D are modified. First of all, it is arguable that which decomposition method can be used in the MOEA/D. Miettinen [1] aruged that the weighted-sum approach is good at convex problem while Tchebysheff approach is useful when the problem is non-convex. As a result, although our previous work showed that Tchebysheff approach outperformed the weighted sum approach, it is still not sufficient enough to conclude that Tchebysheff approach will perform better for the flowshop benchmarks. The validation of the effect of these two decomposition methods will be provided at section V.

Secondly, the z index is applied as a substitute of z^*. Hence, the value of z is apparently larger than or equal to that of z^* in the minimization problem and z doesn't guarantee a good lower reference value when the decomposition method normalizes the objective values. To provide a good approximation of the z, a parameter α is introduced and each z_i is multiplied by α if z_i is improved. The setting of α is configured by Design-of-Experiment and the result is also shown in section V.

Finally, once a good solution is found in the MOEA/D, the algorithm will replace its neighborhood solutions immediately. When the solution is very good, this new solution inevitably replaces all neighborhood solutions. This procedure enhances the convergence of the algorithm; however, it causes the problem of degrading the diversity of the population abruptly. Therefore, the genetic operators are not able to generate different offsprings since all the solutions are identical in the *T* neighbors. Therefore, this paper sets the maximum number of replaced neighborhood solution is 1 rather than be able to replace all neighbors. Because this setting is problem specific, we examine this issue carefully at section V.A.

There are many crossover and mutation methods. We utilize the two-point crossover and moving position mutation for the Crossover procedure and the Mutation procedure, respectively, because Muruta and Ishibuchi [21] found both of them were the better approaches for these two objectives.

It is noted that there is no improvement or repair procedure applied in this multiobjective scheduling study. Finally, to evaluate the performance of the proposed algorithm, Inverted General Distance (*IGD*) is applied in our experimental studies. Let *P** be a set of uniform distributed points in the objective space along the *PF*. Let P be an approximation to the *PF*, the inverted generational distance from *P** to *P* is defined as:

$$IGD(P^*, P) = \frac{\sum_{v \in P^*} d(v, P)}{|P^*|} \quad (9)$$

Where $d(v, P)$ is the minimum Euclidean distance between v and the points in P. If $|P^*|$ is large enough to represent the PF very well, $D(P^*, P)$ could measure both the diversity and convergence of P in a sense. To have a low value of $D(P^*, P)$, P must be very close to the *PF* and can not miss any part of the whole *PF*. Consequently, the smaller IGD value, the better performance of the algorithm.

V. EXPERIMENTAL RESULTS

This paper compares the performance of MOEA/D with NSGA II and SPEA 2 on the 2-objective and 3-objective flowshop scheduling problems [7][1], whose objectives include makespan, maximum tardiness, and an extra objective average flowtime. Furthermore, there are 20, 40, 60, and 80 jobs on 20 machines in these instances. The stopping criterion is to examine 100,000 solutions in each replication and the experiments are replicated 20 times. In addition, Design-of-Experiment (DOE) is employed to select the parameters of each algorithm. Table 1 and Table 2 show the common parameter setting of the three algorithms and the parameter configurations of MOEA/D respectively.

TABLE 1
THE COMMON PARAMETERS OF THE THREE ALGORITHMS

Type	Factor	MOEA/D	NSGA II	SPEA 2
2-obj	PopSize	200	200	200
	Pc	1.0	1.0	1.0
	Pm	1.0	1.0	1.0
3-obj	PopSize	300	200	200
	Pc	0.5	0.5	0.5
	Pm	0.5	1.0	1.0

TABLE 2
PARAMETER CONFIGURATIONS OF MOEA/D

Type	Factor	MOEA/D
2-obj	Decomposition	Weighted Sum
	α	0.6
	Neighbors	10
3-obj	Decomposition	Tchebysheff
	α	0.8
	Neighbors	10

It is noticeable that the decomposition method of MOEA/D applies the weighted sum approach in 2-objective problem while the Tchebysheff method is applied in 3-objective problems. The reason is the shape of Pareto front which causes this difference; the Pareto front of the 2-objective problem is a convex problem and the 3-objective problem is a non-convex one (Stated in section 2.). Consequently, the Tchebysheff method outperforms the weighted sum method significantly in our DOE experiment.

For the setting of α, it implies no matter the weighted sum method or the Tchebysheff method, MOEA/D works better when the reference point is decreased. It means after we decrease the values of the reference point, it provides a good approximation of true ideal point. Consequently, decreasing the reference point might be useful for the class of decomposition methods.

Finally, because the number of replaced neighborhood solution in step 2.4 is set to one, which may influence the diversity and convergence effect, this issue is further

[1] Test instances: http://www.ie.osakafu-u.ac.jp/~hisaoi/ci_lab_e/research/pdf_file/multiobjective/MOGLS/test_problem.html

discussed in the section A. Finally, the comparison results of the three algorithms are given in section B.

A. DOE Experiments of MOEA/D

Once a new solution is found in the modified MOEA/D, there is at most one neighborhood solution updated. It prevents from the new solution replaces all its neighbor solutions. However, it is still vague of how many number of replaced solution(s) is required. This section demonstrates the effect of the number of replaced solutions under through the instances in this paper. Therefore, we compare the MOEA/D with the number of replacement is from zero (no replacement) to all neighbor solutions (the original version of MOEA/D). Modified MOEA/D employs the DOE results and then we ran the experiments to validate the effect of the number of replaced solutions. The following figures show the result of the number of replaced solutions versus its IGD value under the 8 instances. (2/20 means it is the 2-objective problem and the number of jobs is 20; similarly, the 3/40 is the 3-objective problem while there are 40 jobs in the instances.)

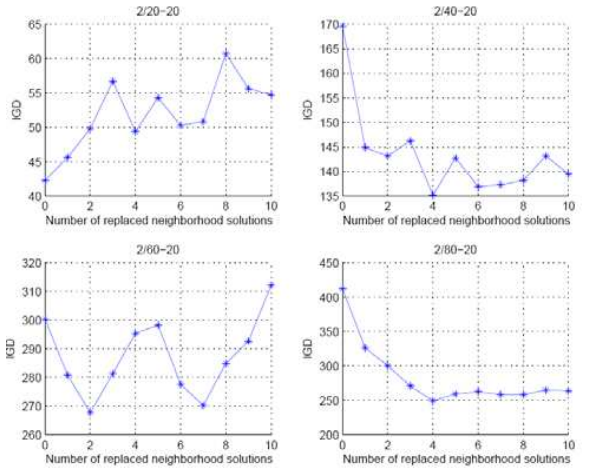

Fig 1. The number of replaced neighbor solutions vs. average IGD in the 2-objective problems (Each combination is also replicated 20 times.)

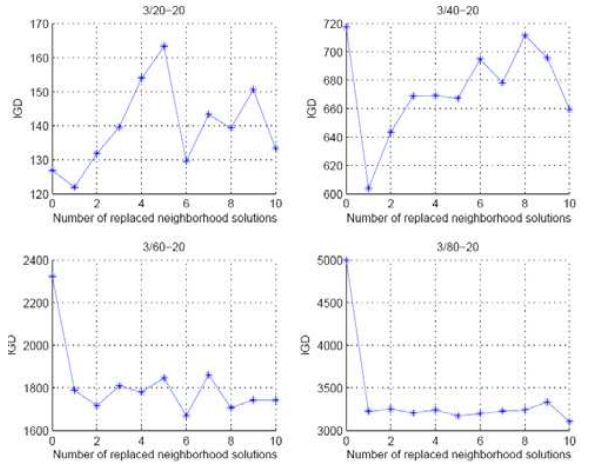

Fig 2. The number of replaced neighbor solutions vs. average IGD in the 3-objective problems (Each combination is also replicated 20 times.)

It is observed that when the problem size is smaller (e.g. 20 or 40 job), it needs less number of replaced solutions (say one replaced solution is sufficient and more robust.). On the other hand, when the problem is more complex (such as job 80), MOEA/D tends to replace more neighborhood solutions. Consequently, the implication of the number of replaced neighbor solution is that the larger problem instance requires more convergence efforts; while the diversity of Pareto solutions is more important for the smaller size of instances since each algorithm is able to generate solutions closed to Pareto front well. According to this experiment result, the suggested number of replaced solutions for two objectives flowshop problem are 0, 4, 2, and 4 and 1, 1, 6, and 10 for three objectives problems.

B. Comparison of MOEA/D with NSGA II and SPEA 2

After DOE experiments are done and the properties of MOEA/D are well discussed, we compared the MOEA/D with NSGA II and SPEA 2 which are the well-known multi-objective algorithms. NSGA II and SPEA 2 algorithms are coded based on the platform PISA[2]. Table 3 demonstrates the experiment results of the three algorithms in the eight flowshop instances evaluated by IGD metric.

TABLE 3
EXPERIMENTAL RESULTS OF THE THREE ALGORITHMS ON THE EIGHT INSTANCES (IGD). THE LOWEST AVERAGE IGD VALUE IS MARKED IN BOLD.

Problem	MOEA/D			NSGAII			SPEA2		
	Min	Avg.	Max	Min	Avg.	Max	Min	Avg.	Max
2/20	28.25	42.3	82	28.25	40.23	81.74	19.02	**37.54**	84.08
2/40	106.42	**135.2**	156.26	106.42	144.6	198	106.8	139.7	194.3
2/60	214.33	**267.7**	415.25	214.33	325.7	464	214.3	292.4	430.7
2/80	182.16	**249.5**	326.4	182.16	424.9	682.7	282.2	398.4	598.9
3/20	80.26	122	168.47	83.18	117.3	161.6	56.43	**116**	171.9
3/40	414.22	604.0	795.36	424.6	771.8	1384	380.3	**581.2**	738
3/60	1208.29	**1670.8**	2119.89	1539	2354	3275	918.4	1902	2571
3/80	2192.13	**3106.6**	4624.58	3451	4408	5635	2837	3740	5191

From the experimental results, though SPEA 2 performs well in some small instances (say 20 jobs for 2 and 3-objective problem and 40 job in 2-objective problem), MOEA/D is only worse than SPEA2 for 4.5%. However, when it goes to problems with other size, MOEA/D outperforms the SPEA2 for 19.2%. In the general case, MOEA/D performs better than NSGAII and SPEA2, which are 38.5% and 16.3%, respectively.

The following ANOVA table shows there is significant difference among the three algorithms. Consequently, Duncan pair-wise comparison (table 5) is used to test the difference between each of them. From the Duncan analysis, MOEA/D is the best, SPEA 2 is the second one and the

[2] http://www.tik.ee.ethz.ch/sop/pisa/

NSGA II is the worst. (Because these algorithms don't share the same alphabet that means they are not in the same group, they are significant to each other and MOEA/D is better than others significantly.) Thus, MOEA/D is indeed better than Pareto based algorithms. Finally, the following two figures are the selected Pareto plots of 20-job and 80-job instances in 2-objective and 3-obejective problems.

TABLE 4
ANOVA RESULTS OF THE THREE ALGORITHMS

Source	DF	Seq SS	Adj SS	Adj MS	F	P
ObjType	1	2.4E+08	2.4E+08	2.4E+08	3091.52	0
Algorithm	2	7189331	7189331	3594665	46.17	0
size	3	2.8E+08	2.8E+08	9.3E+07	1189.27	0
ObjType *Algorithm	2	4689637	4689637	2344819	30.12	0
ObjType*size	3	2E+08	2E+08	6.6E+07	846.03	0
Algorithm*size	6	6796419	6796419	1132736	14.55	0
ObjType *Algorithm*size	6	3923777	3923777	653963	8.4	0
Error	456	3.6E+07	3.6E+07	77860		
Total	479	7.7E+08				

TABLE 5
DUNCAN POST-HOC ANALYSIS OF THE ANOVA TEST

Duncan	Grouping	Mean	N	Method
	A	975.2	160	NSGAII
	B	736.2	160	SPEA2
	C	688.6	160	MOEA/D

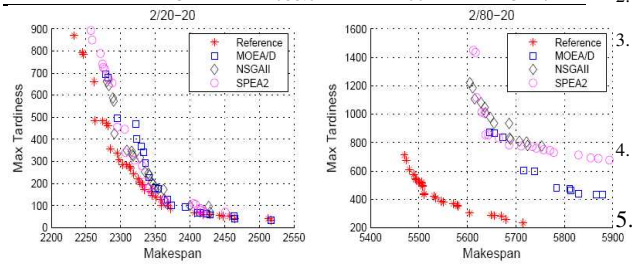

Fig 3. The 2-objective problems with 20 and 80 jobs

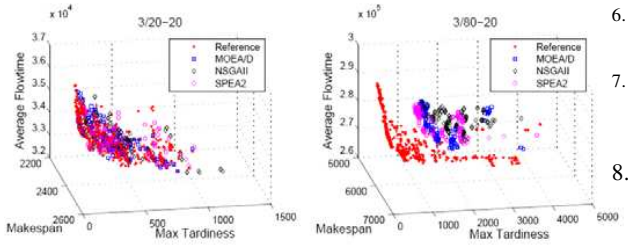

Fig 4. The 3-objective problems with 20 and 80 jobs

VI. CONCLUSIONS

Even though there are several multi-objective algorithms which work well, most of them are Pareto based algorithms while the decomposition algorithms remain little. MOEA/D is one of the decomposition algorithms, which is able to apply some decomposition methods to solve different types of problems. This paper discussed the properties of MOEA/D in the multi-objective flowshop scheduling problems. According to the experimental results, the decomposition methods using weighted-sum approach and Tchebysheff approach are good for the 2-objective and 3-objective problem because the previous one is convex and the latter one is non-convex. Consequently, an appropriate decomposition method is selected only if the problem type (convex or non-convex) is studied in advance. Moreover, no matter for the weighted-sum or the Tchebysheff approach is used in MOEA/D, it is beneficial for obtaining better solution quality when the reference point should be decreased. Moreover, the requirement of the number of replaced neighbors may influence the solution quality. The results shown for smaller problem, MOEA/D needs less replaced neighborhood solutions. While solving the larger problem instances, MOEA/D attends to apply more replaced neighborhood solutions. The reason is that if the problem size is small, the diversity is more important so that fewer numbers of replaced neighbors is employed. On the other hand, the more replaced neighbor solutions are suggested for the larger size problems. Finally, MOEA/D outperforms NSGAII and SPEA2 in 2-objective and 3-objective flowshop scheduling instances. Consequently, decomposition algorithm provides a new direction to be explored for the researchers interested in tackling the multi-objective problems.

REFERENCES

1. K. Miettinen, *Nonlinear Multiobjective Optimization*. Norwell, MA: Kluwer, 1999.
2. K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. New York: Wiley, 2001.
3. E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, Athens, Greece, pp. 95–100, 2001.
4. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
5. A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem – A comparative experiment," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 402–412, Aug. 2002.
6. P. C. Chang, S. H. Chen, and K. L. Lin, "Two-phase sub population genetic algorithm for parallel machine-scheduling problem," *Expert Systems with Applications,* vol. *29*, no. 3, pp. 705-712, 2005.
7. H. Ishibuchi, T. Yoshida, and T. Murata, "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.
8. T. Muruta, & H. Ishibuchi, Performance evolution of genetic algorithms for flowshop scheduling problems. Proceedings of first IEEE international conference on evolutionary computation, 812–817, 1994.
9. Q. Zhang and H. Li, MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition, IEEE Trans. on Evolutionary Computation, vol.11, no. 6, pp. 712-731, 2007.
10. H. Li and Q. Zhang, Comparison Between NSGA-II and MOEA/D on a Set of Multiobjective Optimization Problems with Complicated Pareto Sets, Working Report CES-476, Dept of CES, University of Essex.
11. T. P. Bagchi, Multiobjective scheduling by genetic algorithms. Kluwer academic publishers, 1999.
12. K.R. Baker, Introduction to sequencing and scheduling. New York: Wiley, 1974.