Multi-objective Memetic Approach for Flexible Process Sequencing Problems

Jian-Hong Chen
Department of Computer Science and Information Engineering
707 Sec. 2 Wu-Fu Road
Hsin-Chu 300, Taiwan

Jian-Hung Chen
Department of Computer Science and Information Engineering
707 Sec. 2 Wu-Fu Road
Hsin-Chu 300, Taiwan
jh.chen@ieee.org

ABSTRACT

This paper describes a multi-objective memetic approach for solving multi-objective flexible process sequencing problems in flexible manufacturing systems (FMSs). FMS can be described as an integrated manufacturing system consisting of machines, computers, robots, tools, and automated guided vehicles (AGVs).FMSs usually pose complex problems on process sequencing of operations among multiple parts. An efficient multi-objective memetic algorithm with fitness inheritance mechanism is proposed to solve flexible process problems (FPSs) with the consideration the machining time of operations and machine workload load balancing. The experimental results demonstrate that our approach can efficiently solve FPSs and fitness inheritance can speed up the convergence speed of the proposed algorithm in solving FPSs.

Categories and Subject Descriptors

J.6 [**COMPUTER-AIDED ENGINEERING**]: Computer-aided manufacturing (CAM)

General Terms

Algorithms, Design, Performance

Keywords

process planning, flexible manufacturing systems, multi-objective optimization, memetic algorithms, fitness inheritance

1. INTRODUCTION

Computer-aided process planning (CAPP) is an automated system for preparation of a plan that specifies machines, machine conditions, operations, operation sequence, and tools required to production these components. Traditionally, the process sequencing has been solved by either the experience of process planners or a fixed and static process plan consisting of an ordered sequence of operations. However, the traditional mythologies are not suitable in real flexible environment, because the techniques have a few constraints in order to cope with dynamic situations of the flexible environment [7]. Moreover, as the number of operations increase, it poses more difficulties for decision makers to plan a cost-effective process sequences for manufacturing.

In this paper, a memetic algorithm using fitness inheritance (MEFI) is proposed to solve multi-objective flexible process sequencing problems (FPSs) having three objectives: minimizing total machining time, maximum machine workload and machine workload unbalance. The proposed approach can obtain a set of non-dominated solutions for decision makers in a single run, without the necessary of problem decomposition and relative preferences. Decision makers can easily distinguish between the costs of different process sequences and choose more than one satisfactory process sequences at a time. Six benchmark problems with different complexities are used to evaluate the performance of the proposed approach. A multi-objective genetic algorithm (MOGA) without local search and fitness inheritance is used for performance comparisons. It is shown empirically that MAFI outperforms MOGA in terms of the solution quality.

This paper is organized as follows: Section 2 presents the background of process sequencing problems, multi-objective evolutionary optimization. Section 3 introduces the setup of flexible manufacturing system and the mathematical formulation of FPSs. Section 4 presents the multi-objective memetic algorithm for solving FPSs. Section 5 presents the experimental analysis of the proposed algorithm, and Section 6 summarizes our conclusions.

2. BACKGROUND

2.1 Process Sequencing Problems

Flexible process sequencing problems are well known among the combinatorial optimization problems. Previous research focused on two important key issues of process sequencing problems, described as follows. The first key issue is the objective functions of process sequencing. Several approaches [4, 1] are proposed for process sequencing with various objectives. Another key issue that arises recently is the alternative process sequences. In the view of real time scheduling, alternative process sequences provide additional capability for the decision maker (DM) to cope with unpredictable events such as machine failures or rush orders. From the view of off-line scheduling, alternative process sequences may be used to improve the schedule quality by reducing the load on bottleneck machines [1]. It is essential but also a challenge for DM to prepare a set of alternative process sequences considering the trade-off between schedule quality and the costs of process sequences. However, traditional techniques are not able to provide such flexibility for DM.

The above issues lead to flexible process sequencing problems (FPSs), which simultaneously considers alternative process plans with multiple objectives and the flexibility of process sequences. Over the past decade, a number of models have been developed to solve the process sequencing problems, but only few models [1, 7] have been reported to design the process sequencing problem considering the above issues. To date, solving the problem of flexible process sequencing with multiple objectives that are conflicting in nature is still a hard task.

2.2 Multi-objective Evolutionary Optimization

Assume all the objective functions F_m are to be minimized. Mathematically, multi-objective optimization problems (MOOPs) can be represented as the following vector mathematical programming problems:

$$Minimize \quad F(X) = \{F_1(X), F_2(X), ..., F_m(X)\}, \tag{1}$$

where X denotes a solution and $F_m(X)$ is generally a nonlinear objective function. When the following inequalities hold between two solutions X_1 and X_2, X_2 is a *non-dominated solution* and is said to *dominate* $X_1 (X_2 \succ X_1)$:

$$\forall m : F_m(X_1) \geq F_m(X_2) \;\; and \;\; \exists n : F_n(X_1) > F_n(X_2). \tag{2}$$

When the following inequality hold between two solutions X_1 and X_2, X_2 is said to *weakly dominate* $X_1 (X_2 \succeq X_1)$:

$$\forall m : F_m(X_1) \geq F_m(X_2). \tag{3}$$

A feasible solution X^* is said to be a *Pareto-optimal solution* if and only if there does not exist a feasible solution X where X dominates X^*. The corresponding vector of Pareto-optimal solutions is called *Pareto-optimal front*.

By making use of Pareto dominance relationship, multi-objective evolutionary algorithms (MOEAs) are capable of performing the fitness assignment of multiple objectives without using relative preferences of multiple objectives. Thus, all the objective functions can be optimized simultaneously. As a result, MOEA seems to be an alternative approach to solving production planning and inspection planning problems on the assumption that no prior domain knowledge is available.

3. PROBLEM STATEMENT

3.1 The FMS Environment

An FMS consists of a set of identical and/or complementary numerically controlled machines and tool systems. All components are connected through an AGV system. Figure 1 shows the layout of a simple FMS with several machines, AGVs and a tool system.

In order to design the production planning of FMSs, the environment within which the FMS under consideration operates can be described below.

- The term *machine* is to describe a machine cell. A machine cell consists of several identical devices/machines. The types and number of machines are known. There is a sufficient input/output buffer space at each machine.

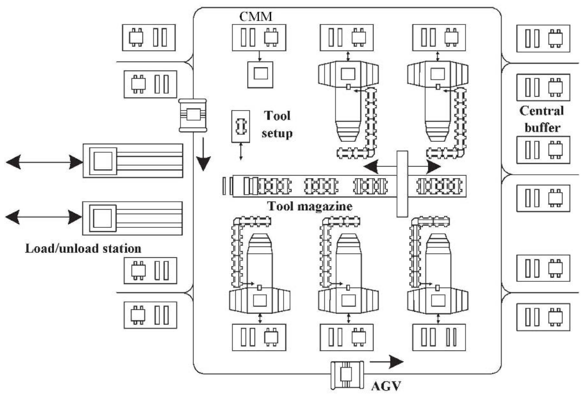

Figure 1: FMS with several machines, a coordinate measuring machine (CMM), AGVs and a central tool magazine.

- A *part* type requires a number of *operations*. A number of part types will be manufactured simultaneously in batches. Parts can choose one or more machines at each of their operation stages, and the transportation of the parts within different machines is handled by an AGV system.
- A machine can perform several types of operations, and an operation can be performed on alternative machines.
- A machine can only process an operation at one time. Operations to be performed in the machine are non-preemptive. Operation lot splitting is ignored in this paper.
- A *process sequence* is a series of machine indices corresponding to operations of all parts. Based on a process sequence, each operation is operated on its corresponding machine. An illustrative process sequence of 3 parts and 10 operations is presented in Figure 2, and the operations are operated on 3 different machines. An example of the series of machine indices to be optimized is Y=[1 1 1 3 1 2 2 2 3 3].
- Workload on each machine is contributed by those operations assigned to a machine.
- A load/unload (L/U) station serves as a distribution center for parts not yet processed and as a collection center for parts finished. All vehicles start from the L/U station initially and return to there after accomplishing all their assignments. There are sufficient input/output buffer spaces at the L/U station.
- The number of AGVs is given and the transportation time of AGVs are known. Some machines may not be linked.
- AGVs carry a limited number of products at a time. They move along predetermined paths, with the assumption of no delay because of congestion. Preemption of trips is not allowed.

- It is assumed that all the design, layout and set-up issues within FMS have already been resolved.
- Real-time issues, such as traffic control, congestion, machine failure or downtime, scraps, rework, and vehicle dispatches for battery changer are ignored here and left as issues to be considered during real-time control.

Part index	1	2	3
Operation index	1 2 3 4	1 2 3	1 2 3
Process Sequence (Machine index)	1 1 1 3	1 2 2	2 3 3

Figure 2: A process sequence of 3 parts and 10 operations, operated on 3 different machines. For example, the operation 4 of the part 1 is assigned to the machine 3.

3.2 Mathematical Formulation of FPSs

3.2.1 Notations

In order to formulate FPSs, the following notations are introduced:

- i : part index, $i = 1, 2, 3, ..., I$.
- j : operation index for part i, $j = 1, 2, 3, ..., J_i$.
- k, l : machine index $k, l = 1, 2, 3, ..., K$.
- Y : process sequence.
- pv_i : production volume (unit) for part i.
- pt_{ijk} : processing time per unit to perform operation j of part i using machine k.
- m_k : maximum workload of machine k.
- tw_k : workload in machine k, $tw_k = pt_{ijk} \times pv_i$.
- rtw_k : workload ratio in machine k, $rtw_k = \frac{tw_k}{m_k}$.
- ew : average workload of machines.
- $s_{ikl} : \begin{cases} 1, & \text{if part } i \text{ is to transfer from machine } k \text{ to } l; \\ 0, & \text{otherwise.} \end{cases}$
- $x_{ijk} : \begin{cases} 1, & \text{if machine } k \text{ is selected to perform} \\ & \text{operation } j \text{ of part } i; \\ 0, & \text{otherwise.} \end{cases}$
- abl : available capacity of AGV per trip, abl is set to 10 in this chapter.
- n_{ikl} : the number of trips between machine k and l for part i,

$$n_{ikl} = s_{ikl} \times \lceil \frac{pv_i}{abl} \rceil,$$

  where the bracket represents a ceiling operation.
- tm_{kl} : transportation time from machine k to l. If machines k and l are not linked, it is set to be a negative value for constraint handling.
- t_{ikl} : total transportation time between machines k and l for part i,

$$t_{ikl} = n_{ikl} \times tm_{kl}.$$

3.2.2 Objectives

There are three objectives to be optimized in flexible process sequencing problems, described below.

1. Minimization of total flow time. This objective is to minimize the processing time and transportation time for producing the parts. The total machine processing time (e_1) is defined as Equation 4, the transportation time (e_2) is defined as Equation 5, and the total flow time (f_1) is defined as Equation 6. Transportation between unlinked machines are penalized in e_2.

$$e_1 = \sum_{i=1}^{I} \sum_{j=1}^{J_i} \sum_{k=1}^{K} pv_i \times pt_{ijk} \times x_{ijk}, \quad (4)$$

$$e_2 = \sum_{i=1}^{I} \sum_{j=1}^{J_i - 1} \sum_{k=1}^{K} \sum_{l=1}^{K} t_{ikl} \times x_{ijk} \times x_{i(j+1)l}, \quad (5)$$

$$f_1 = e_1 + e_2. \quad (6)$$

2. Minimization of machine workload unbalance. Balancing the machine workload can avoid creating bottleneck machines. The objective function (f_2) is defined as Equation 7.

$$f_2 = \sum_{k=1}^{K} (rtw_k - ew)^2. \quad (7)$$

3. Minimization of greatest machine workload. Pursuing this objective also implies attempting to minimize the total flow time. The objective function (f_3) is defined as Equation 8.

$$f_3 = max\{rtw_k\}. \quad (8)$$

3.2.3 Multi-objective Mathematical Model

The overall multi-objective mathematical model of FPSs can be formulated as follows. Given the production volume pv_i, the processing time pt_{ijk}, the maximum workload m_k, the available capacity of AGV per trip abl, the transportation time tm_{kl} and the tool costs c_{ijk}, find a series of machine indices, Y, for operations of all parts such that

$$minimize \quad f_1, f_2, f_3, \quad (9)$$

subject to

$$\sum_{k=1}^{K} x_{ijk} = 1, \quad \forall (i, j), \quad (10)$$

$$tm_{kl} \geq 0, \quad \forall (k, l), \quad (11)$$

$$rtw_k \leq 1, \quad \forall i. \quad (12)$$

The constraint, Equation 10, ensures that only one machine is selected for each operation of a part. Equation 11 ensures an AGV path exists between machines k and l. Equation 12 is to ensure the machine workload tw_k is smaller or equal to its maximum machine workload m_k.

If the total number of machines is x and the total number of operations is y, then the complexity of the investigated problem is $O(x^y)$.

4. MULTI-OBJECTIVE MEMETIC ALGORITHM WITH FITNESS INHERITANCE MEFI

4.1 Schemata-Guided Local Search Strategy

Based on schema theorem and the niche hypothesis [5], a schemata-guided local search strategy is proposed to be combined with MOGA for improving the convergence speed to the Pareto-front. Extended from the niche hypothesis, it is assumed that, given a MOOP with Q Pareto-optimal solutions, Q Pareto-optimal solutions can be regarded as Q niches of the MOOP. In the worst case, to ensure MOEAs is capable of searching Q Pareto-optimal solutions, it is assumed that the population were divided into Q species (subpopulations). Thus, each species is expect to optimize its own niche (Pareto-optimal solution), as shown in Figure 3. Therefore, the optimal schemata of a species is its Pareto-optimal solution.

Let the schema of species be H_q, where the fixed positions are the maximum common string of all individuals in its species and the others are "don't care"(*). Since species are in the same population, a schemata of a species may be disrupted by schemata of the other species due to genetic operators. The disruption between species can be further classified into the following two types:

1. **Species disrupt noise**: The fixed schemata of H_{origin} are altered to "don't care" schemata by the corresponding positions of the schemata H_{other}. Thus, a species requires more time for fixing it's "don't care" schemata.
2. **Species hitchhiking noise**: The "don't care" schemata of H_{origin} are altered to fixed schema by the corresponding positions of the schema H_{other}. If the altered schemata are located in the similarity regions of their optimal schemata, the change is good for the schemata H_{origin}. On the contrary, the change is bad for the schemata H_{origin}.

Based on the foregoing inference, it is desired that a species should keep its good schemata (building blocks) while making good efforts to alter its "don't care" schemata to its ideal optimal schemata. As results, a schemata-guided local search strategy is proposed based on this guideline. Information of fixed and "don't care" schemata in species are utilized to guide local search. However, the key question of this local search strategy is that how do we classify population to different species when true Pareto-optimal solutions of MOOPs are unknown. To deal with this question, it is assumed that the best individuals in each objective functions are the *pioneers* of each species. These pioneers will be used to classify all individuals in population to different species.

Given a maximum local search times $MaxLS$ and a temporary elite set E', the procedure of the used schemata-guided local search strategy is written as follows:

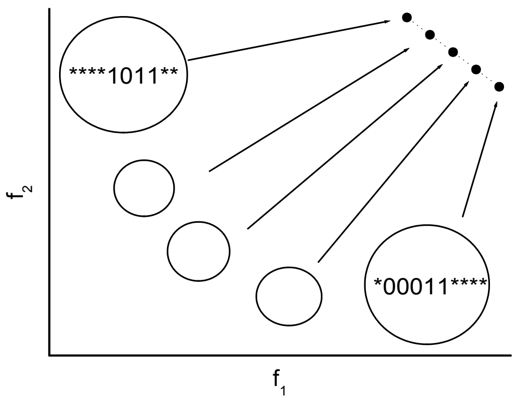

Figure 3: The population were divided into several species, and each species optimizes its own niche (Pareto-optimal solution).

Step 1: (Identification) Identify the best individuals $B_q, q = 1, 2, ..., Q$, in each objective from the current population. For FPSs, Q=3.

Step 2: (Classification) Classify the current population into Q species by the best solutions in each objective.

Step 3: (Schemata computation) For each species, compute its schemata H_q. Both fixed and "don't care" schemata are identified.

Step 4: (Parameter setting) Let $q = 1, counter = 0$.

Step 5: (Perturbation) Perturb B_q into a new solution B'_q. According to H_q, apply the mutation operator only on "don't care" locations of B_q with a mutation probability p_m.

Step 6: (Evaluation) Evaluate the objective functions of B'_q. Let $counter = counter + 1$.

Step 7: (Comparison) There is 3 cases in comparisons of B_q and B'_q. **Case 1**: If B_q dominates B'_q and $counter < MaxLS$, go to Step 5. **Case 2**: If B_q is dominated by B'_q, replace B_q by B'_q. **Case 3**: If B_q and B'_q doesn't dominated each other. Stored B'_q in a temporary elite set E'.

Step 8: (Termination test) Let $q = q + 1$ and counter=0, if q>Q, stop the local search strategy. Otherwise, go to Step 5.

4.2 Fitness Inheritance

An efficiency enhancement techniques called fitness inheritance [2] is used for speedup of MEFI. During the evolution of EAs, the fitness of some proportion of individuals in the subsequent population is inherited. This proportion is called the inheritance proportion, p_i.

Mathematically, for a multi-objective problem with z objective, the used fitness inheritance is defined as

$$f_z = \frac{w_1 f_{z,p1} + w_2 f_{z,p2}}{w_1 + w_2}, \tag{13}$$

where f_z is the fitness value in objective z, w_1, w_2 are the weights for the two parents p_1, p_2, and $f_(z, p_1)$, $f_(z, p_2)$ is the fitness values of p_1, p_2 in objective z, respectively. In this paper, w_1 and w_2 are set to 1.

According the literature of fitness inheritance, the population size of FIEA should be bigger than the population size used for MOGA, as shown in the following equation:

$$N_{pop,FIEA} = \frac{N_{pop,MOGA}}{1 - p_i^3} \tag{14}$$

4.3 MEFI for solving FPSs

4.3.1 Representation and Operators

A series of machine indices Y for operations of all parts is directly encoded as a integer chromosome. The range of each gene of Y is [1, K]. Each gene of Y stands for a machine index.

The selection operator of MEFI uses a binary tournament selection which works as follows. Choose two individuals randomly from the population and copy the better individual into the intermediate population. The one-point crossover is used in MEFI. A simple mutation operator is used to alter genes. For each gene, randomly generate a real value from the range [0, 1] with the probability p_m.

MEFI uses a generalized Pareto-based scale-independent fitness function GPSIFF [6] by the following function:

$$F(X) = p - q + c, \tag{15}$$

where p is the number of individuals which can be dominated by the individual X, and q is the number of individuals which can dominate the individual X in the objective space. c is the number of all participant individuals.

Based on the proposed chromosome representation, Equation 10 is always satisfied. If Equation 11 is violated, the transportation time between machines k and l, tm_{kl}, is set to be a large value, 10^7. In this way, f_2 will be penalized. For each machine k, if Equation 12 is not satisfied, one is added to r_{twk}, as follows:

$$r_{twk} = \begin{cases} \frac{tw_k}{m_k}, & \text{if } tw_k \le m_k; \\ \frac{tw_k}{m_k} + 1, & \text{otherwise.} \end{cases} \tag{16}$$

4.4 Procedure of MEFI

Since it has been recognized that the incorporation of elitism may be useful in maintaining diversity and improving the performance of multi-objective EAs [3], MEFI selects a number of elitists from an elite set E in the selection step. The elite set E with capacity E_{max} maintains the best non-dominated solutions generated so far. In addition, an external set $\overline{E}$ with no capacity is used to store all the non-dominated solutions ever generated so far. The procedure of MEFI is written as follows:

Step 1: (Initialization) Randomly generate an initial population of N_{pop} individuals and create two empty elite sets E, $\overline{E}$ and an empty temporary elite set E'.

Step 2: (Evaluation) For each individual Y in the population, excluding the inherited individuals, compute the value of objective functions $f_1(Y)$, $f_2(Y)$, and $f_3(Y)$.

Step 3: (Fitness assignment) Assign each individual a fitness value by using GPSIFF.

Table 1: The parameter settings of MEFI and MOGA.

Parameters	MEFI	MOGA
N_{pop}	115	100
E_{max}	115	100
p_s	0.25	0.25
p_i	0.5	N/A
p_c	0.6	0.6
p_m	0.05	0.05
$MaxLS$	3	N/A

Step 4: (Local search) Apply the proposed schemata-guided local search strategy. Non-dominated solutions obtained by the local search strategy will be stored in temporary elite set E'.

Step 5: (Update elite sets) Add the non-dominated individuals in both the population and E' to E, and empty E'. Considering all individuals in E, remove the dominated ones in E. Add E to $\overline{E}$, remove the dominated ones in $\overline{E}$. If the number of non-dominated individuals in E is larger than E_{max}, randomly discard excess individuals.

Step 6: (Selection) Select $N_{pop} - N_{ps}$ individuals from the population using the binary tournament selection and randomly select N_{ps} individuals from E to form a new population, where $N_{ps} = N_{pop} \times p_s$ and p_s is a selection proportion. If N_{ps} is greater than the number N_E of individuals in E, let $N_{ps} = N_E$.

Step 7: (Recombination) Perform the one-point crossover operation with a recombination probability p_c.

Step 8: (Fitness inheritance) Perform fitness inheritance on the selected $N_{pop} \times p_i$ individuals. The inherited objective values are calculated according to Equation 13.

Step 9: (Mutation) Apply the mutation operator to each gene in the individuals with a mutation probability p_m.

Step 10: (Termination test) If a stopping condition is satisfied, stop the algorithm and output $\overline{E}$. Otherwise, go to Step 2.

5. RESULTS AND DISCUSSION

Six benchmark problems: *m3o10*, *m4o20*, *m5o100*, *m5o200*, *m10o100* and *m10o200*, where *mxoy* stands for the x machine and y operation problem. A MOGA, MEFI without the local search strategy and fitness inheritance, is implemented to solve FPSs as the baseline performance. The parameter settings of MEFI and MOGA are given in Table 1. Thirty independent runs with the same number of function evaluations $100xy$ were performed per test problems.

The coverage metric $C(A, B)$ of two solution sets A and B [8] used to compare the performance of two corresponding algorithms considering the six objectives:

$$C(A, B) = \frac{|\{a \in A, b \in B, a \succeq b\}|}{|B|}, \tag{17}$$

Fig. 4 depicts the coverage metrics of $C(MEFI, MOGA)$ and $C(MOGA, MEFI)$ from 30 runs. In solving the small problem *m3o10*, Fig. 4 shows that the performance of MEFI

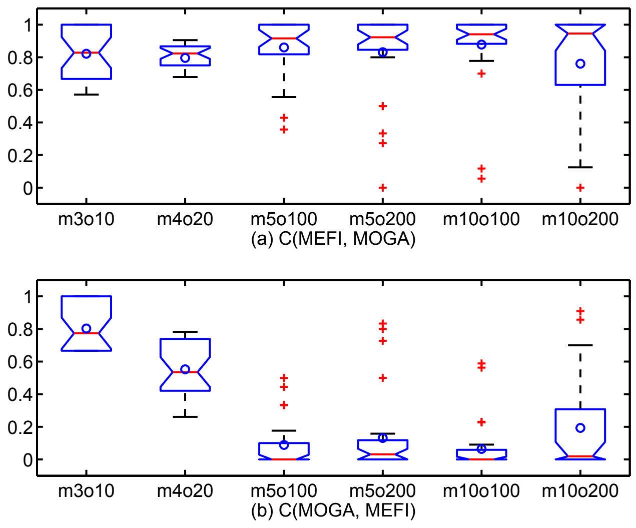

Figure 4: Box plots based on the cover metric. (a) C(MEFI, MOGA), (b) C(MOGA, MEFI).

and MOGA are almost the same. For another small problem *m4o20*, the non-dominated solutions obtained by MEFI dominates 80% of the solutions obtained by MOGA in average, while the non-dominated solutions obtained by MOGA only dominates 60% of the non-dominated solutions obtained by MEFI in average. As the complexity of problems increases, Fig. 4 shows that 80%-90% of the non-dominated solutions obtained by MOGA are weakly dominated by the non-dominated solutions obtained by MEFI in solving the problems *m4o20*, *m5o100*, *m5o200*, *m10o100* and *m10o200*. On the contrast, the non-dominated solutions of MOGA dominate nearly 3-10% of the non-dominated solutions obtained by MEFI. Fig. 5 shows the non-dominated solutions obtained by thirty runs of MEFI and MOGA in solving the *m10o200* problem. The results indicate that MEFI can converge to better solutions more quickly than MOGA. It reveals that the proposed schemata-guided local search strategy and fitness inheritance plays an important role in obtaining good solutions and accelerating the convergence speed.

6. CONCLUSION

In this paper, a novel approach to solve flexible process sequencing problems using an multi-objective memetic algorithm MEFI is proposed. A schemata-guided local search strategy and fitness inheritance are integrated in the proposed algorithm for enhancing the performance. Experimental results demonstrated that the quality of non-dominated solutions obtained by MEFI is better than that of MOGA in terms of convergence speed and accuracy using the same number of function evaluations. While prior domain knowledge for the decomposition of problems or relative preferences of multiple objectives are not available, the proposed approach is an expedient method to solve flexible process sequencing problems. Moreover, the proposed approach can obtain a set of non-dominated solutions for decision makers in a single run. Decision makers can easily distinguish between the costs of different process sequences and choose more than one satisfactory process sequences at a time.

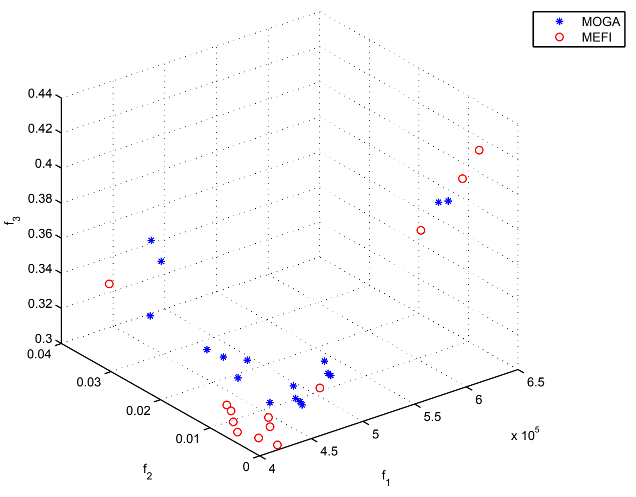

Figure 5: The non-dominated solutions obtained by MEFI and MOGA in solving the *m10o200* problem, merged from 30 runs.

7. ACKNOWLEDGMENTS

This work was supported by the National Science Council of Taiwan, R.O.C. under Contract NSC-96-2221-E-216-037-MY2 and NSC-095-SAF-I-564-616-TMS, and Chung-Hua University under Contract CHU-96-2221-E-216-037-MY2.

8. REFERENCES

[1] P. Brandimarte. Exploiting process plan flexibility in production scheduling: A multi-objective approach. *European Journal of Operational Research*, (114):59–71, 1999.

[2] J.-H. Chen, D. E. Goldberg, S.-Y. Ho, and K. Sastry. Fitness inheritance in multi-objective optimization. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 319–326, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[3] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley-Interscience series in systems and optimization. John Wiley & Sons, 2001.

[4] M. Gen and R. Cheng. *Genetic algorithms and engineering design*. John Wiley, New York, 1997. 1944- Mitsuo Gen, Runwei Cheng. ill. ; 24 cm.

[5] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Pub. Co., 1989.

[6] S.-Y. Ho, L.-S. Shu, and J.-H. Chen. Intelligent evolutionary algorithms for large parameter optimization problems. *IEEE Transaction on Evolutionary Computation*, 8(6):522–541, Dec. 2004.

[7] C. Moon, Y.-Z. Li, and M. Gen. Evolutionary algorithm for flexible process sequencing with multiple objectives. In *Proceeding of IEEE International Conference on Computational Intelligence*, pages 27–32, 1998.

[8] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strengthen Pareto approach. *IEEE Transaction on Evolutionary Computation*, 4(3):257–271, 1999.