<ARTICLE>A Genetic Algorithm for Bicriteria Scheduling Problems

A Genetic Algorithm for Bicriteria Scheduling Problems

Hiroyuki Morita

Abstract

In this paper, a genetic algorithm that produces an approximate set of all non-dominated solutions is proposed for a bicriteria permutation scheduling problem. And a measure of distance between the set of all non-dominated solutions and any approximate set is defined. In order to show its effectiveness, we shall compare its performance with exact non-dominated solutions by carrying out computational experiments for a bicriteria single machine permutation scheduling problem. Computational results demonstrate that proposed algorithm have good performance from viewpoints of both solution quality and computational efficiency for practical input size case.

1. Introduction

In real applications of industrial scheduling, decision-makers are usually faced with various multi-objective scheduling problems. As remarked by [25], a schedule that is optimum with respect to one criterion normally performs badly with respect to other criteria. Therefore a schedule with satisfactory performance on all measures may be a better alternative for the decision-maker. This consideration leads to the research on multicriteria scheduling. A number of papers (see [5], [6], [10], [20], [24], [25]) pointed out the importance of extensive research of multiobjective scheduling.

Recently, multiobjective scheduling has been studied by many researchers (see survey papers [5], [10], [24]). Most papers in the literature focused on problems on a single machine as noticed by [24]. One reason might be that even a single-objective scheduling problem on multiple machines is difficult to solve in most cases as evidenced by NP-completeness results (see [19]). Because of this, we can not enumerate all of non-dominated solutions for most multiobjective scheduling problems in reasonable computational time in practice.

On the other hand, many researchers proposed various metaheuristic approaches such as simulated annealing, tabu-search and genetic algorithms (GA) for multiobjective combinatorial optimization (MOCO) problems (see [2], [7], [8], [9], [11], [12], [15], [16], [21], [28], [29], [31], [34]). Ulungu and Teghem[35] gave an excellent survey for various metaheuris-

tic approaches for MOCO. The approaches using GA have been proposed by [7], [8], [12], [15], [16], [23], [28], [31], [34] and etc. In particular, [1] and [8] gave an extensive survey for GA approaches in multiobjective optimization. Approaches by simulated annealing have been proposed by [2], [9], [29]. Recently, the tabu search methods are proposed by [11] and [14]. Most of these algorithms are for general-purpose, and to the author's knowledge, comparative study among them has not been extensively done yet. In view of this, the authors[21] have conducted a comparative study of GA approaches and other heuristics for a bicriteria two-machine flow-shop scheduling problem that minimizes both makespan and maximum tardiness. As a result, we have observed that GA approaches are superior to the heuristics proposed by [3]. Among GA approaches, using a single strategy among those proposed by [7], [8], [12], [28] does not work well, but combining these strategies together with a new strategy, called *seeding strategy*, introduced by the authors[21] exhibits the best performance.

The aim of this paper is to improve our GA proposed in [21] and to apply to another class of bicriteria scheduling problem in order to see its effectiveness from the viewpoint of the goodness of approximation of an exact set of all non-dominated solutions. The problem dealt with in this paper is a bicriteria single machine scheduling problem minimizing both total flow time and maximum tardiness simultaneously. This problem has been studied by Wassenhove and Gelders[36], and $O(n^2 p \log n)$ time algorithm for computing an exact set of non-dominated solutions was proposed, where p is the maximum of processing times of n jobs (each processing time is assumed to take a positive integer value). Although the running time of this algorithm is pseudo-polynomial, it is efficient if p is not too large. The reason why we apply our algorithm to this problem in spite of the existence of the exact algorithm by [36] is that we can observe how closely our algorithm approximates the exact set of non-dominated solutions. In the other multiobjective scheduling problems, we can not enumerate all of non-dominated solutions for practical input size case, so we can not decide our algorithm is superior or not absolutely. As we shall see in Section 4, our algorithm can produce all non-dominated solutions for this problem of size up to 500 jobs in acceptable computational time.

The organization of this paper is as follows. Section 2 gives necessary notations and briefly reviews basic ideas the existing multiobjective GA's, and also defines a scheduling problem discussed in this paper. Section 3 presents our genetic algorithm. Section 4 reports our experimental results. Finally, we conclude this paper with discussion on a potential applicability of our method to a large class of combinatorial optimization problems.

2. Preliminaries

2.1 bicriteria combinatorial optimization problems

Suppose that we are given a finite set X of R^n, and two objective functions $f_j: X \rightarrow R$, $j=1,2$ (both are assumed to be minimization). An element $x \in X$ is called a feasible solution or simply a solution. Bicriteria optimization problems are then formulated as

$$P: \text{minimize}_{x \in X}\{f_1(x), f_2(x)\}. \tag{1}$$

Since both objectives cannot be minimized simultaneously in general, it is quite natural that the decision-maker chooses a solution from among non-dominated solutions. A solution is called non-dominated if there does not exist $y \in X$ with $y \neq x$ satisfying

$$f_1(y) \leq f_1(x) \quad \textit{and} \quad f_2(y) \leq f_2(x), \tag{2}$$

and one of the inequalities in (2) strictly holds. A set of all non-dominated solutions for a set X is denoted by $ND(X)$. $ND(X)$ is computed by solving the following single-objective problem by continuously varying the parameter from 0 to 1 (see [27], [33]).

$$\text{minimize}_{x \in X} U_\alpha(f_1, f_2) \equiv \max\{\alpha f_1(x), (1-\alpha) f_2(x)\} + \beta\{\alpha f_1(x) + (1-\alpha) f_2(x)\}, \tag{3}$$

where $0 \leq \alpha \leq 1$ and β is a very small positive constant. Let $v_\alpha(X)$ denote the minimum value of the above problem. Since we are dealing with the problem for approximating $ND(X)$, we need to introduce a new measure to compare the relative degree of approximation of two approximate sets of $ND(X)$. Let S_1 and S_2 be two such sets. Let us assume that the decision maker chooses his or her utility function from among family of functions $U'_\alpha(f_1, f_2)$ defined below by randomly choosing from

the interval .

$$U'_\alpha(f_1,f_2) \equiv \max\{\alpha\frac{f_1(x)}{\pi_1},(1-\alpha)\frac{f_2(x)}{\pi_2}\}+\beta(\alpha\frac{f_1(x)}{\pi_1}+(1-\alpha)\frac{f_2(x)}{\pi_2}). \quad (4)$$

$U'_\alpha(f_1,f_2)$ is obtained from $U'_\alpha(f_1,f_2)$ by introducing range equalization factors π_1 and π_2 in order to equalize the ranges of f_1 and f_2, which was introduced by [33]. Given a set S of solutions, π_i is defined as $max_{x\in S}f_i(x)-min_{x\in S}f_i(x)$. Then the utility value for a particular α with respect to $S_i, i=1,2$ is defined by

$$v_\alpha(S_i) \equiv \min_{x\in S_i} U'_\alpha(f_1(x),f_2(x)), \qquad i=1,2 \quad (5)$$

We can compute the expected value of the ratio of the utility value $v_\alpha(S_2)$ to $v_\alpha(S_1)$ with respect to α by

$$Ex_\alpha[v_\alpha(S_2)/v_\alpha(S_1)] \equiv \int_0^1 \frac{v_\alpha(S_2)}{v_\alpha(S_1)}d\alpha \quad (6)$$

This is called the expected ratio of S_2 to S_1, denoted by $ratio(S_2|S_1)$. It seems obvious that if this value is larger than one, S_1 is a better approximate set of $ND(X)$ than S_2 Notice that the equality $ratio(S_2|S_1)=ratio(S_1|S_2)$ does not always hold, and $ratio(S_2|S_1)\cdot ratio(S_1|S_2)\geq 1$ holds in general(the proof is omitted).

2.2 Bicriteria single-machine scheduling problems

All performance measures considered in this paper are regular. Here, a measure of performance is said to be regular if it is non-decreasing function of job completion times and the scheduling objective is to minimize the performance measure. Scheduling problems treated in this paper is a bicriteria single-machine scheduling problem that minimizes both total flow time and maximum tardiness. It is denoted by $n/1//\Sigma C_j, T_{max}$ using the notation of [24] and defined as follows.

Suppose that we are given jobs that are available at time 0 and are to be sequenced on a single machine. Jobs are numbered from 1 through n. Each job j has processing time p_j and due date d_j, where p_j is assumed to be a positive integer. Since we are considering regular performance measures, scheduling is determined by a permutation. Given a schedule, completion time C_j of job is uniquely determined. Although C_j depends

on a schedule, we omit from the notation of C_j such dependency for the sake of simplicity. Total flow time ΣC_j is defined to be $\sum_{j=1}^{n} C_j$. Tardiness T_j of job j is defined to be $\max\{C_j - d_j, 0\}$. Maximum tardiness $T_{\max}$ is defined as $\max_{1 \le j \le n} T_j$.

3. Overview of multiobjective GA

Genetic algorithms have been recognized to be well suited to multiobjective optimization in nature since they are keeping multiple solutions in parallel. Genetic algorithms proceed in general by keeping a set of solutions (such set is called a *population* and a solution in the set is called an *individual* in GA) and by performing crossover, mutation, and selection operations (see [12]). The major difference between single-objective and multiobjective GA lies in the way of selecting individuals for the next generation. In a single-objective GA an individual can be evaluated according to the single objective function, while in a multiobjective GA it is desired to obtain a set of individuals that are uniformly distributed over the objective space and well approximates the set of non-dominated solutions.

We shall briefly explain the selection strategies proposed for multiobjective GA. Schaffer[28] proposed the method that divides a set of individuals into subgroups according to each objective. Selection is performed in each subgroup according to the corresponding objective, independently of other objectives. This method naturally tends to keep individuals which are good with respect to a particular objective while it seems hard to produce individuals for which all objective values are "*acceptably small*" (all objectives are assumed to be minimization).

Goldberg[12] proposed the ranking method by which all individuals in a population are ranked according to the dominance relation among individuals in a current population. Namely, the individuals that are not dominated by any other individual are ranked "*one*". After deleting rank-one individuals, the non-dominated individuals in the remaining population receives rank two, and so forth. According to the ranks of individuals, selection is performed.

Horn et al.[15] proposed the method that uses domination relationship and *Sharing*. *Sharing* is proposed by Goldberg and Richardson[13] and is a

technique which decreases fitness values of individuals when they are crowded in the objective space. In other words, when there is a lot of individuals around the individual, it makes small its fitness value. By using this, diversity of its population is hold.

Tamaki et al.[34] proposed a hybrid strategy that mixes the Schaffer's method and elitism that always keeps rank-one individuals for the next generation.

3.1 Our Genetic Algorithm

In [21], the authors have investigated several selection strategies proposed for multiobjective GA by carrying out computational experiments for a two-machine flow-shop scheduling problem that minimizes both $C_{\max}$ and $T_{\max}$ simultaneously. Considering the facts we found in [21], we shall take the following strategies in our GA.

a) All solutions that are ranked one by Goldberg's method [12] (i.e., approximate non-dominated solutions at any generation) are kept for the next generation because it is costly to rediscover them.
b) (Schaffer's strategy) When selection is performed, a few good solutions in the current population with respect to each objective are kept for the next generation without applying tournament selection[1] even if their ranks are more than one.
c) Among the solutions not selected in a) and b), solutions kept for the next generation are determined by tournament selection, hoping to keep the diversity of individuals.
d) (Seeding strategy) In addition to the set of randomly generated individuals, we initially add a few solutions to the current population that are good with respect to each objective. A certain exact or heuristic algorithm depending on the computational difficulty of the corresponding single-objective scheduling problem computes such good solutions. Using this strategy, we can obtain good solutions more quickly. Fig. 1 illustrates the idea of using seeding strategy.

1.Tournament selection is one of selection method that chooses the number of individuals (*tournament size*) randomly from a population, selects the best individual among them, and repeats until mating pool is filled. In our experiment, tournament size is two and they are compared on ranking.

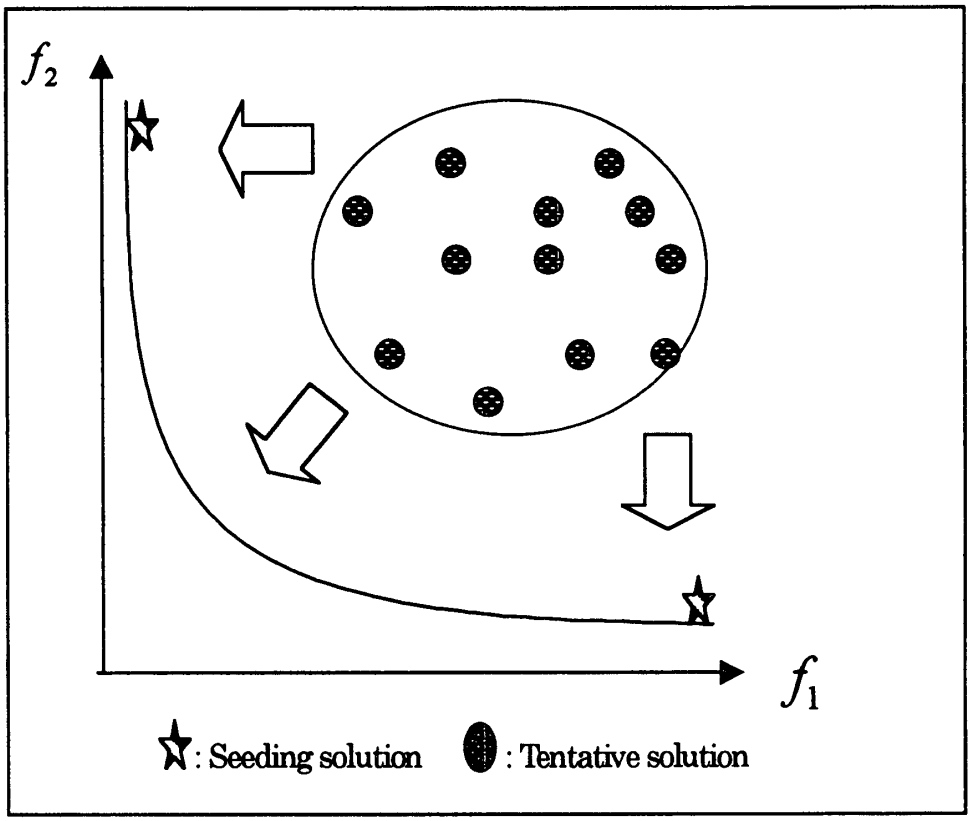

Fig. 1: Image of seeding strategy

In addition to these facts, for more large input size problem instances we found some point, which have to improve though additional experiments as follows.

a) It takes much CPU time, especially for large input size problem.
b) When we use only GA operators (i.e., crossover and mutation), it is slow to converge to non-dominated solutions generally.

To improve problem a), selection method is changed. Tournament selection is same, but its ranking method is changed. In previous version of our algorithm, we used Goldberg's ranking method. Although this is good one, it takes much time to rank each individuals at each generations. So in this version, we select individuals based on domination relationship with shearing. It was proposed by Horn et al.[15]. First, two individuals are chosen randomly, and their domination relationship is checked. When one individual dominates another, non-dominated solution (i.e., *winner*) is selected next generation. But when no individual domi-

nates each other (i.e., a tie), by using sharing winner is determined.

About problem b), a kind of local search methods is incorporated into our GA. Its local search method is as follows. At any generation, all elite individuals (i.e., approximate non-dominated solutions) are applied local search method except individuals that are applied at previous generation. In other words, no individual that is applied local search once are applied again. In local search part, all new solutions that are generated by local search is compared with a set of elite solutions. When a new solution dominates one or more elite solutions, a set of elite solutions is updated. Fig. 2 illustrates the image of this local search method.

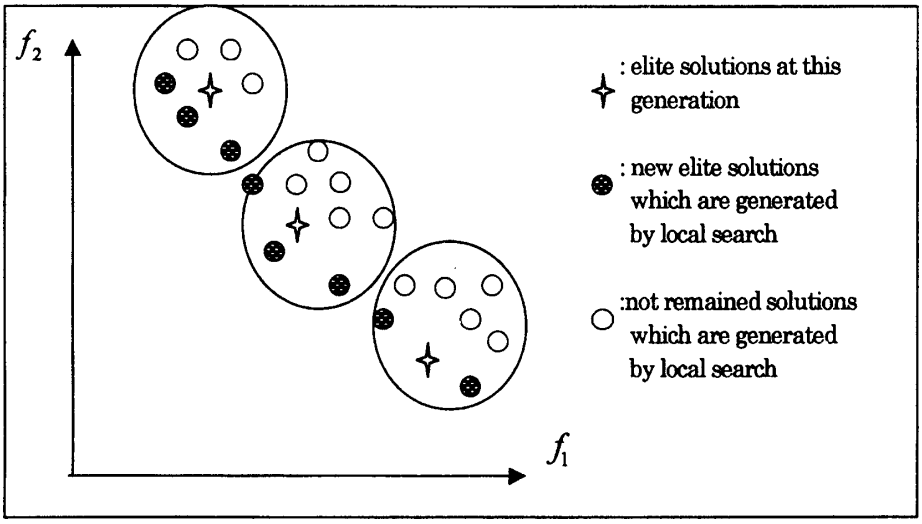

Fig. 2: New local method incorporated in our algorithm

All the other features are the same as those adopted in the conventional GA. The high-level description of our GA is given as follows:

a) Seeding solutions are generated using existing exact or approximate method for each criterion. Other solutions are generated randomly.
b) Perform crossover and mutation.
c) Local search is implemented for elite solutions.
d) Selection strategies
   d-1) Good solutions with respect to each objective are kept.
   d-2) All solutions that are ranked one are kept.
   d-3) Among the solutions not selected in d-1) and d-2), solutions kept by tournament selection.

e) Repeat b) ~d) until terminated generation is met.
f) Output non-dominated solutions that are found until GA is terminated.

4. Computational Experiments

We have implemented our multiobjective GA illustrated in the previous section. The exact set of non-dominated solutions is denoted by ND^*. We have generated ten problem instances for each of five different numbers of jobs, i.e.,$n=100, 200, 300, 400, 500$. We shall refer to the ratio that is how much non-dominated solution our GA can find in ND^* as *detection ratio*. As mentioned early, for this problem, we can compute the detection ratio, since we can compute ND^* exactly for such large n by the algorithm [36]. In addition to detection ratio, we analyze the performance by computing the expected performance ratio defined in Section 2.1. In these experiments, PA-8200 (240MHz) is used as CPU and compilation is implemented by gcc (ver.2.5.8).

4.1 Generation of problem instances

We have generated ten problem instances for each of $n=100, 200, 300, 400, 500$ according to the following scheme. For each n, problem instances are randomly generated as below. Potts and Van Wassenhove[26] have reported that the difficulty of problem is influenced by two parameters, relative range of due date (RDD) and average lateness factor (LF). Following their remark, Daniels and Chambers[3], and Yagiura and Ibaraki[37] generated instances of scheduling problems. We also did the same way. In our experiments, we have chosen ten combinations of parameters RDD and LF, from the sets {0.1, 0.2, 0.3, 0.4, 0.5} and , {0.8, 1.0} respectively. Then, for each combination of RDD and LF, we have generated ten problem instances as follows:

a) Each processing time p_j is determined by uniform distribution over the integers in the range of [1,30].
b) Then, each due date d_j is determined by uniformly distribution over the integers in the range of $[(1-LF-RDD/2)T, (1-LF+RDD/2)T]$, where $T=n\bar{p}$ and $\bar{p}$ denotes the average processing time.

4.2 Implementation details

When implementing metaheuristic algorithms, we sometimes have to be careful about details of tuning the program parameters. In order to determine such parameters in our testing for GA, we have performed preliminary computational experiments. We shall explain how we choose such parameters for GA as follows:

- Population size: From our preliminary experiments, we have observed that if we know $|ND^*|$, the size of non-dominated solutions, in advance, it is better to set population size more $|ND^*|$.
- Crossover and mutation operators: There have been proposed a number of different crossover operators that deal with permutations. Our preliminary experiments have not exhibited a significant difference among them. Thus, we have adopted OX (order crossover) proposed by [22]. As for mutation operator, we use the way to randomly choose two jobs in a current schedule and to swap their positions in the corresponding permutation.
- Seeding solution: For $n/1//\Sigma C_j, T_{max}$, we can compute an exact solution for each criterion. $n/1//\Sigma C_j$ and $n/1//T_{\max}$ can be optimally solved by means of Smith's rule [30] and by means of EDD rule [17], respectively. So these solutions are incorporated in initial solution set.
- Local search: As neighborhood solutions, two-swap method is used. To decrease computational time, it is implemented every 10 generations.

4.3 Computational results

4.3.1 non-dominated solutions

As early mentioned, it is important to compare our computational results with exact non-dominated solutions. Tab.1 illustrates the number of non-dominated solutions for each problem instances. We can see that by changing *LF* and *RDD* the number of non-dominated solutions changes. And as input size increases, the number of non-dominated solutions increases largely.

Fig.3 illustrates a state of non-dominated solutions. We can see the trade-off between two objective functions. The other problem instances are similar with this figure.

Tab.1: number of non-dominated solutions for each problem instances

	Dat0	Dat1	Dat2	Dat3	Dat4	Dat5	Dat6	Dat7	Dat8	Dat9	Avg.	min.	Max.
100 jobs	18	20	57	61	115	5	5	9	24	32	34.6	5	115
200 jobs	25	28	140	227	345	5	7	23	58	62	92.0	5	345
300 jobs	37	122	248	476	647	3	22	25	67	149	179.6	3	647
400 jobs	28	222	321	791	912	6	38	162	207	264	295.1	6	912
500 jobs	68	349	539	1180	1134	5	26	117	343	440	420.1	5	1180

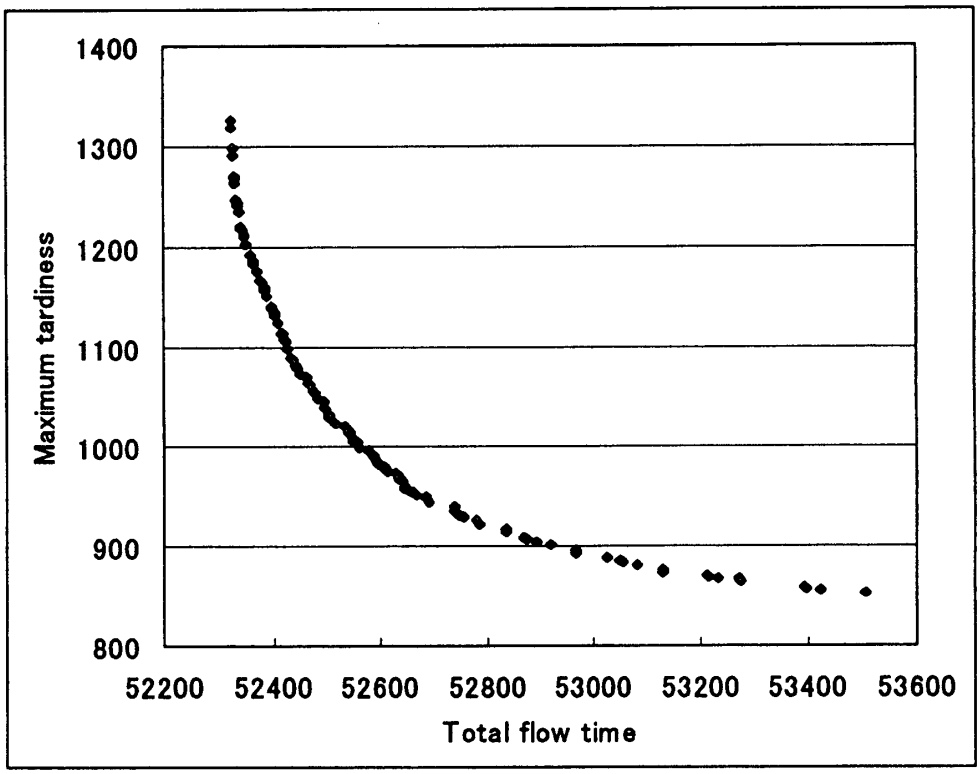

Fig.3: Example of non-dominated solutions (100 jobs: dat4)

4.3.2 Computational results for each problem instances

Of course the number of generations that need for finding all non-dominated solutions is different for each problem instances. Additionally in this experimentation, local search technique is utilized. So even if same parameter values are used, we notice that it takes different *computational* time at same generations. It depends on how much local search parts are used at any generation. In these experimentation, our algorithm can find all non-dominated solutions for all problem instances. Tab.2 illustrates entire results for all problem instances.

As an entirely, up to 500 jobs input sizes, we can find all *non-dominated* solutions in acceptable computational time. But dat3 and dat4 seems diffi-

cult to find all non-dominated solutions generally. It is a reason that has much non-dominated solutions by these *LF* and *RDD* combinations. For these problem instances of all input sizes, we need more computational time than the other problem instances. In terms of generations, it seems that we can do that in small generations. Local search has much contribution about it.

Although local search is powerful, can we find all non-dominated solutions using only local search? To research about this, the algorithm that removes GA parts computes same problem instances. In other words, only local search updates a set of elite solutions in the same way and when it can not update a set of elite solutions, algorithm is terminated. Tab.3 illustrates their results. At 400 jobs and 500 jobs problem instances, it takes too much time, so computational results are shown up to 300 jobs problem instances.

Tab.2: Computational results for all problem instances

	100 jobs		200 jobs		300 jobs		400 jobs		500 jobs	
	Number of Generation	CPU time (sec.)	Number of generation	CPU time (sec.)	Number of Generation	CPU time (sec.)	Number of Generation	CPU time (sec.)	Number of Generation	CPU time (sec.)
Dat0	130	15.4	210	148.3	240	931.8	430	3707.5	320	3896.3
Dat1	100	8.1	290	398.3	320	2199.3	470	13978	420	18361.2
Dat2	170	48.8	280	647.6	390	7066.5	300	25372.9	580	36430.8
Dat3	140	53.3	310	2490.3	520	18471.7	640	101911.9	210	180704.1
Dat4	150	91.8	380	3654.4	540	38654.9	390	135102.3	550	288012
Dat5	80	3.4	160	51.2	280	268.9	240	751.4	330	3720
Dat6	80	3.2	160	94.5	200	576.5	300	3517.5	280	6589.6
Dat7	140	9.5	250	164.7	200	863.7	330	8476	290	8056.6
Dat8	100	10.5	220	252.6	260	1689.1	250	11358.8	360	37934
Dat9	100	13.4	290	385.1	310	3352.2	510	15683.4	340	53489.9
Avg.	119	25.74	255	828.7	326	7407.5	386	31986	368	63719.5
Min.	80	3.2	160	51.2	200	268.9	240	751.4	210	3720
Max	170	91.8	380	3654.4	540	38654.9	640	135102	580	288012

As shown Tab.3, there are some cases that can find all non-dominated solutions in 100 jobs problem instances, but in general it is difficult to find all by using only local search. Especially, as input size becomes large, it seems that it is terminated with poor performance. Fig.4 shows a example of these results, when algorithm is terminated. Although a part of non-dominated solutions are found, i.e., upper left area, there are some solutions that can not close to non-dominated solutions, i.e., right down area.

Tab.3: Computational results by using only local search

	Detection ratio(%)			CPU time(sec.)		
	100 jobs	200 jobs	300 jobs	100 jobs	200 jobs	300 jobs
Dat0	27.8	88.0	21.6	12.8	262.3	2685.3
Dat1	100.0	92.9	80.3	12.6	1108.2	9721.5
Dat2	31.6	86.4	57.3	122.5	2132.1	29349.8
Dat3	52.5	32.6	52.3	137.7	8000.6	66935.0
Dat4	65.2	47.2	31.8	260.2	11841.8	53222.6
Dat5	60.0	40.0	66.7	0.8	207.0	567.9
Dat6	100.0	85.7	31.8	6.0	286.4	1346.8
Dat7	100.0	91.3	72.0	17.2	370.6	2253.5
Dat8	100.0	81.0	67.2	23.2	542.3	5369.9
Dat9	93.8	64.5	79.9	23.3	2308.0	6849.8

Fig.5 illustrates the set of elite solutions at every ten generations. At first mainly elite solutions are updated by GA, at later generation local search improves the quality of elite solutions. Tab.4 shows detection ratio and expected ratio in same problem instances.

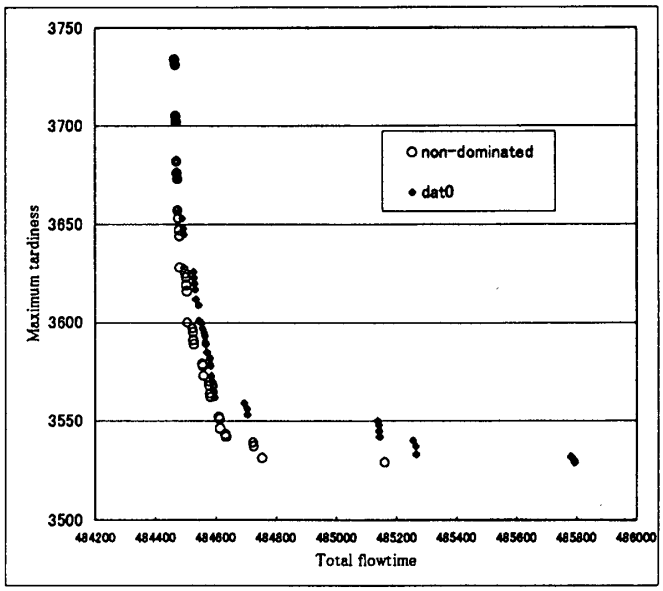

Fig.4: Computational result by only using local search (300 jobs: dat0)

5. Conclusion

In this paper, we propose a genetic algorithm for bicriteria scheduling problems. From computational experiments, we have shown that our GA has good performance both in solution quality and computation time. The problem that is slow to converge to non-dominated solutions by GA is improved by introducing local search part. Now some parameter value

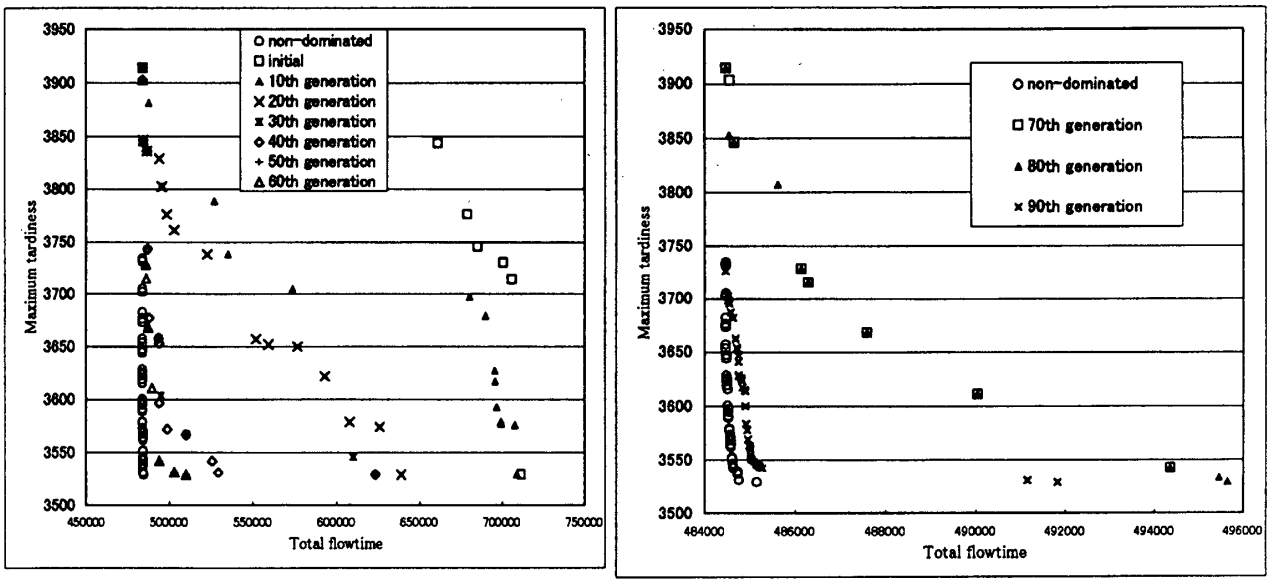

Fig.5: A set of elite solutions and each generation

Tab.4: Detection ratio and Expected ratio (300 jobs: dat0)

Generation	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200
Detection ratio	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	5.4%	10.8%	35.1%	89.2%	97.3%	97.3%	97.3%	97.3%	97.3%	97.3%	97.3%	97.3%	100.0%
Expected ratio	132.5	91.46	82.06	82.06	15.37	15.37	15.13	8.38	6.81	6.06	3.15	2.41	1.94	1.63	1.45	1.32	1.23	1.16	1.12	1.00	1.00

may not be appropriate, so I guess that there is a chance to decrease computational time more. About these problems, we have to analyze computational results more in the future.

Up to the present, we applied our GA to two kinds of bicriteria scheduling problem and confirmed good performance from some computational results. As future researches, we would like to apply our GA to other scheduling problems as well as the other MOCO problems in order to see its effectiveness.

References

[1] C. A. Coello Coello, An Updated Survey of Evolutionary Multiobjective Optimization Techniques: State of the Art and Future Trends, *In 1999 Congress on Evolutionary Computation*, pp. 3-13, Washington, D.C., July (1999).

[2] P. Czyzak and A. Jaszkiewicz, Pareto simulated annealing - a metaheuristic technique for multiple objective combinatorial optimization (in preparation).

[3] R. L. Daniels and R. J. Chambers, Multiobjective flow-shop scheduling, *Naval Research Logistics*, Vol.37, pp.981-995 (1990).

[4] L. Davis, Applying adaptive algorithms to epistatic domains, *Proc. of the 9th IJCAI*, A. Joshi (ed.), Morgan Kaufumann, pp.162 (1985).

[5] P. Dileepan and T. Sen, Bicriterion static scheduling research for a single machine; *OMEGA Int. Journal of Management Science*, Vol.16, No.1, pp.53-59 (1988).

[6] R. A. Dudek, S. S. Panwalkar and M. L. Smith, The lessons of flow-shop scheduling research, *Operations Research*, Vol.40, pp.7-13 (1992).

[7] C. M. Fonseca and P. J. Fleming, Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization; *Proc. of 5th Int. Conf. on Genetic Algorithms*, pp.416-423 (1993).

[8] C. M. Fonseca and P. J. Fleming, An overview of evolutionary algorithms in multiobjective optimization, *Evolutionary Computation*,Vol.3 No.1, pp.1-16 (1995).

[9] P. Fortemps, J. Teghem, and B. Ulungu, Heuristics for multiobjective combinatorial optimization by simulated annealing, *Proc. of 11th Int. Conf. on MCDM*, Coimbra, Portugal, August 1-6 (1994).

[10] T. D. Fry, R. D. Armstrong, and H. Lewis, A framework for single machine multiple objective sequencing research, *OMEGA Int. Journal of Management Science*, Vol.17, No.6 pp.595-607 (1989).

[11] X. Gandibleux, N. Mezdaoui and A. Freville, A tabu search procedure to solve multiobjective combinatorial optimization problems, to appear in *Proc.* volume of *MOPGP'96*, R. Caballero and R. Steuer (eds.), Springer-Verlag.

[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley (1989).

[13] D. E. Goldberg and J. J. Richardson, Genetic algorithms with sharing for multimodal function optimization. *Genetic Algorithms and Their Applications: Proc. of the 2nd ICGA*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp.41-49 (1987).

[14] M. P. Hansen, Tabu search for Multiobjective Optimization: MOTS, *Proc. of 13th Int. Conf. On MCDM, Cape Town, South Africa, January 6-10* (1997).

[15] J. Horn, J. Nafpliotis and D. E. Goldberg: A niched pareto genetic algorithm for multi-

objective optimization, *Proc. of 1st IEEE Conf. on Evolutionary Computation*, pp.82-87 (1994).

[16] H. Ishibuchi and T. Murata, Multi-objective genetic local search algorithm; *Proc. of 3rd IEEE International Conference on Evolutionary Computation*, pp.119-124 (1996).

[17] J. R. Jackson, Scheduling *a production line to minimize maximum tardiness*, Research Report 43, Management Science Research Project, UCLA (1955).

[18] S. M. Johnson, Optimal two- and three-stage production scheduling with setup times included, *Naval Res. Logist. Quart.* 1, pp.61-68 (1954).

[19] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, Sequencing and scheduling: Algorithms and complexity; *Handbooks in OR & MS, Vol.4, Logistics of Production and Inventory* (S. C. Graves et al., eds.) Chap.9, pp.445-522, North-Holland (1993).

[20] S. A. Melnyk, S. K. Vickery and R. L. Carter, Scheduling, sequencing and dispatching: alternative perspectives, *J. Prodn Inv. Mgmt*, Vol.27, No.2, pp.58-68 (1986).

[21] H. Morita and N. Katoh, A Genetic algorithm for a bicriteria flow-shop scheduling problem, *Transactions of the Institute of Systems, Control and Information Engineers*, Vol.10,No.3, pp.127-136 (1997) (in Japanese).

[22] H. Mühlenbein,M. Gorges-Schleuter and O. Krämer, Evolution algorithms in combinatorial optimization, *Parallel Computing*, Vol.7, pp.65 (1988).

[23] T. Murata and H. Ishibuchi, MOGA: Multi-objective genetic algorithms, *Proc. of 2nd IEEE International Conference on Evolutionary Computation*, pp.289-294 (1995).

[24] A. Nagar, J. Haddock and S. Heragu, Multiple and bicriterion scheduling: A literature survey, *European Journal of Operational Research*, Vol.81, pp.88-104 (1995).

[25] S. S. Panwalkar, R. A. Dudek and M. L. Smith, Sequencing research and the industrial scheduling problem, *In Symposium on the Theory of Scheduling and Its Application (Ed. by SE Elmaghraby)*, Springer, New York (1973).

[26] C. N. Potts and L. N. Van Wassenhove, A decomposition algorithm for the single machine total tardiness problem, *Operations Research Letters*, Vol.1, pp.177-181 (1982).

[27] Y. Sawaragi, H. Nakayama and T. Tanino, *Theory of Multiobjective Optimization*, Academic Press (1985).

[28] J. D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, *Proc. of 1st Int. Conf. on Genetic Algorithms and Their Applications*, pp.93-100 (1985).

[29] P. Serafini, Simulated annealing for multi objective optimization problems, *Proc. of the Xth International Conference on MCDM*, Taipei, pp.87-96 (1992).

[30] W. E. Smith, Various optimizers for single-stage production, *Naval Res. Logist. Quart.* 3,

pp.59-66 (1956)

[31] N. Srinivas and K. Deb, Multiobjective optimization using non-dominated sorting in genetic algorithms, *Evolutionary Computation*, Vol.2, No.3, pp.221-248 (1995).

[32] T. Starkweather, S. McDaniel and C. Whitley, A comparison of genetic sequencing operators, *Proc. of 4th International Conference on Genetic Algorithms*, pp.69-76 (1991).

[33] R. E. Steuer, *Multiple criteria optimization: Theory, computation, and application*, John Wiley & Sons (1986).

[34] H. Tamaki, M. Mori and M. Araki, Multi-criteria optimization by genetic algorithms, *Abstracts of the 3rd Conf. on the Association of Asian-Pacific Operational Research Societies within IFORS*, p.51 (1994).

[35] E. L. Ulungu and J. Teghem, Multi-objective Combinatorial Optimization Problems, A Survey; *Journal of Multi-Criteria Decision Analysis*, Vol.3, pp.83-104 (1994).

[36] L. N. Van Wassenhove and L. F. Gelders, Solving a bicriterion scheduling problem, *European Journal of Operational Research*, Vol. 4, pp.42- 48 (1980).

[37] M. Yagiura and T. Ibaraki, Genetic and local search algorithms as robust and simple optimization tools, *Meta-Heuristics: Theory & Applications*, I. H. Osman and J. P. Kelly (eds.), Kluwer Academic Publishers, pp.63-82 (1996).

[38] E. Falkenauer and S. Bouffouix, A Genetic algorithm for job shop, *Proc. of the 1991 IEEE International Conference on Robotics and Automation*, pp.824-829 (1991).

[39] F. Glover, Genetic algorithms and scatter search: unsuspected potentials, *Statistics and Computing*, vol.4, pp.131-140 (1991).

[40] A. Hertz, B. Jaumard, C. C. Ribeiro and W. P. Formosinho Filho, A multi-criteria tabu search approach to cell formation problems in group technology with multiple objectives, *Recherche operationnelle/Operations Research*, 28/3, pp.303-328 (1994).

[41] E. D. Taillard, L. M. Gambardella, Adaptive Memories for the quadratic assignment problem, *IDSIA-87-97 TECHNICAL REPORT*, pp.1-18.